

**ESP32 klok – wekkerradio – internetradio/mp3 speler -
bediening via webpagina.**



10:42

VoWiFi LTE1 51%



192.168.1.178



ESP32 wekkerradio webinterface

Alarm	Dag	Tijd	Mode	Wake_up
1	Dagelijks	06:58	radio	1
2	Woensdag	12:34	mp3	0
3	Weekdagen	00:00	mp3	0

Aktief **Dag** **Tijd** **Mode** **Wake_up**

Radio bediening

Radio 1 Classics

EQ -40 <-> 6 Volume 0 <-> 21

L: M: H: V:

Instellen zender en url : 10

Instellen tijdzone zomeruur

Vrijdag 10:42

uren **minuten** **zomertijd**

Klok helderheid (0 <-> 15)

Klok

Groot en duidelijk display.

Interne klok 2 x dag gesynchroniseerd met NTP server.

Tijdzone en winter- / zomertijd correctie manueel via webpagina.

Instelbare helderheid klokdisplay.

Wekkerradio

3 programmeerbare alarmen

- * bepaalde dag

- * dagelijks

- * weekdays

- * weekend

mp3 speler of internetradio als alarmbron.

Mogelijkheid om gewekt te worden door vrij te kiezen wake-up call.

Valt automatisch uit na 1 uur.

Internetradio

10 voorgeprogrammeerde internetradio's, kunnen gewijzigd worden via webpagina.

Toon en Volume regeling.

Continue bedrijf of als alarm.

Random mp3 speler 5000+ mp3 bestanden zonder probleem.

Enkele nuttige url's.

Veel nuttige info over ESP32 vind je hier :

De ESP32 wordt geprogrammeerd met de Arduino IDE hoe je de IDE en de benodigde ESP32 software op je PC moet installeren vind je hier, zowel voor Windows als voor Linux.

<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/>

Internet streaming adressen:

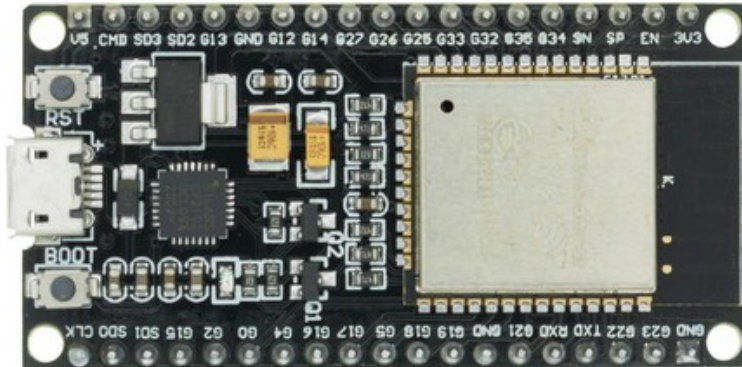
<https://www.hendrikjansen.nl/henk/streaming.html#cz>

Audio library

<https://github.com/schreibfaul1/ESP32-audioI2S>

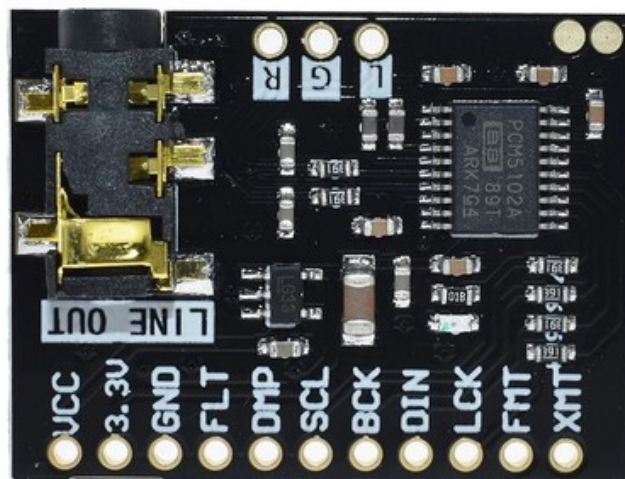
Wat hebben we nodig

ESP32-WROOM (38 pin versie)



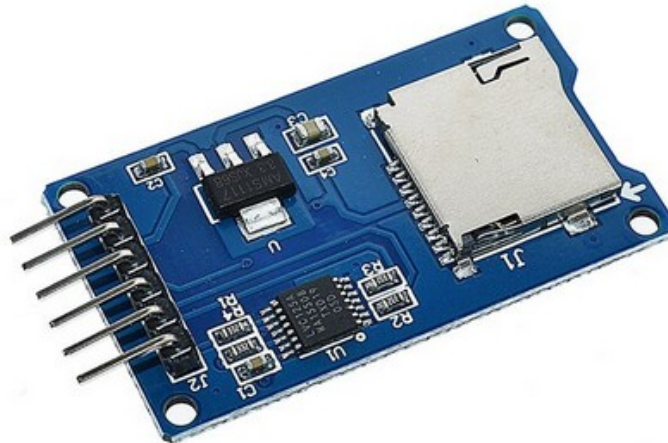
https://nl.aliexpress.com/item/32864722159.html?spm=a2g0o.productlist.main.1.61942b27Eg45Rj&algo_pvid=1e38ae4b-13a1-4111-bf98-932a2d50efc6&algo_exp_id=1e38ae4b-13a1-4111-bf98-932a2d50efc6-0&pdp_ext_f=%7B%22sku_id%22%3A%2212000028744589167%22%7D&pdp_npi=2%40dis%21EUR%213.71%213.71%21%21%21%21%40210212c016740610529416210d0669%2112000028744589167%21sea&curPageLogUid=wgT9nNbq322I

PCM5102A



https://nl.aliexpress.com/item/33004561102.html?spm=a2g0o.productlist.main.7.6693j1VuJ1VuNK&algo_pvid=9dfdb51f-aaf3-48d1-b5fe-64b830e67c62&algo_exp_id=9dfdb51f-aaf3-48d1-b5fe-64b830e67c62-3&pdp_ext_f=%7B%22sku_id%22%3A%2267118680566%22%7D&pdp_npi=2%40dis%21EUR%215.76%211.44%21%21%21%21%40212244c416740698615196556d0685%2167118680566%21sea&curPageLogUid=zWKmpHWXxf05

SD kaart module

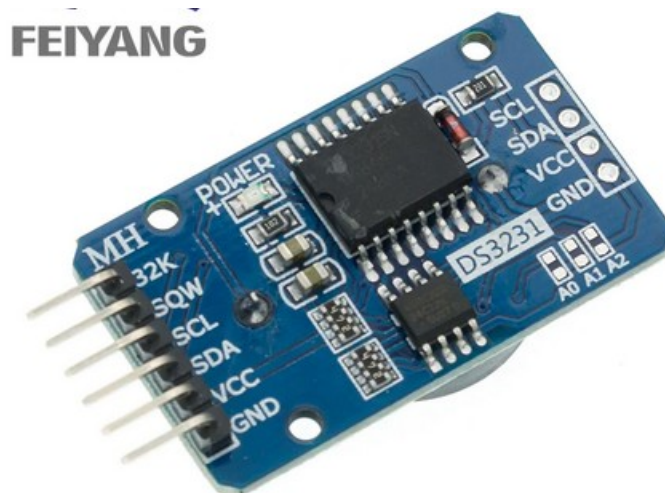


https://nl.aliexpress.com/item/32523546123.html?spm=a2g0o.productlist.main.5.2df62586ZnbmrS&algo_pvid=e69357b3-aeab-4073-85b6-9062215c8cc1&algo_exp_id=e69357b3-aeab-4073-85b6-9062215c8cc1-2&pdp_ext_f=%7B%22sku_id%22%3A%2210000002486114694%22%7D&pdp_npi=2%40dis%21EUR%210.64%210.52%21%21%21%21%40212240a316740612452115676d0665%2110000002486114694%21sea&curPageLogUid=oi29kmat1NDI

SD kaart FAT32 geformatteerd

Afhankelijk van aantal mp3 files, anders minimale grootte voldoende.

DS3231 RTC



https://nl.aliexpress.com/item/32833136577.html?spm=a2g0o.productlist.main.1.5c2077d2GrMsYN&algo_pvid=03a30c30-a94b-4443-9f02-dbd6dec845fc&algo_exp_id=03a30c30-a94b-4443-9f02-dbd6dec845fc-0&pdp_ext_f=%7B%22sku_id%22%3A%2267454834461%22%7D&pdp_npi=2%40dis%21EUR%212.1%212.1%21%21%21%21%4021227a0f16740613309012224d06b3%2167454834461%21sea&curPageLogUid=kFy3BkroDI2w

MAX7219 module



https://nl.aliexpress.com/item/4000587266290.html?spm=a2g0o.productlist.main.3.533e6eba6edqN9&algo_pvid=f41d132e-4c7c-4e1a-b8c5-77b4bc1ea871&algo_exp_id=f41d132e-4c7c-4e1a-b8c5-77b4bc1ea871-1&pdp_ext_f=%7B%22sku_id%22%3A%2210000003439612437%22%7D&pdp_npi=2%40dis%21EUR%214.49%213.69%21%21%21%21%21%402102111816740614473856175d0688%2110000003439612437%21sea&curPageLogUId=RUunHGplwHcQ

Glasfolie wit mat

dimmen helderheid MAX7219 module

<https://www.gamma.be/nl/assortiment/decoratiefolie-mat-wit-346-0001-45x200cm/p/B389981>

5V stereo versterker



https://nl.aliexpress.com/item/32791319633.html?spm=a2g0o.productlist.main.23.656633a7pQVaRS&algo_pvid=140d2c11-5b6b-4201-b402-c7aa658dc2ab&algo_exp_id=140d2c11-5b6b-4201-b402-c7aa658dc2ab-11&pdp_ext_f=%7B%22sku_id%22%3A%2263650640578%22%7D&pdp_npi=2%40dis%21EUR%212.17%211.41%21%21%21%21%21%4021021aa216740618720378241d0701%2163650640578%21sea&curPageLogUId=elbg0VoJ1Wxb

2 x luidspreker



<https://www.allekabels.be/luidspreker-zelfbouw/237/1370099/full-range-speaker-58-cm-23-8-10-w.html>

Mosfet IRL540 / IRF540

IRL540L Vgs(th)	1.0V	2.0V
IRF540F Vgs(th)	2.0V	4.0V

IRF540 doet wat hij moet doen, maar indien op voorraad IRL540 is aan te raden

R 1K

R 4K7

Drukknop

5V (USB)voeding

1 x smartphone voor bediening radio / wekkerradio.

Internetverbinding is noodzakelijk

Verbindingen

5V voeding

+5V	5V	ESP32-WROOM	
	VCC	PCM5102A	
	VCC	MAX7219	
	DC+	versterker	
GND	GND	ESP32-WROOM	3X
	GND	PCM5102A	
	GND	SD module	
	GND	RTC DS3231	
	GND	MAX7219	
	Source	IRF(IRL)540	versterker aan/uit

ESP32-WROOM

+5V	<<	+5V	voeding
GND	<<	GND	voeding
GND	<<	GND	voeding
GND	<<	GND	voeding
GND	>>	R4K7	
+3.3V	>>	+3V3	SD module
	>>	VCC	DS3231RTC
	>>	drukknop	alarm uit
GPIO25	>>	DIN	PCM5102A
GPIO26	>>	LCK	PCM5102A
GPIO27	>>	BCK	PCM5102A
GPIO5	>>	CS	SD module
GPIO23	>>	MOSI	SD module
GPIO19	>>	MISO	SD module
GPIO18	>>	SCK	SD module
GPIO21	>>	SDA	DS3231
GPIO22	>>	SCL	DS3231
GPIO32	>>	SQW	DS3231
GPIO15	>>	CS	MAX7219
GPIO13	>>	DIN	MAX7219
GPIO14	>>	CLK	MAX7219
GPIO4	>>	Drukknop	Alarm uit
GPIO33	>>	Gate	IRF(IRL)540 versterker aan/uit

PCM5102A

5V	<<	+5V	Voeding
GND	<<	GND	Voeding
FLT	<<	GND	PCM5102A
DMP	<<	+3.3V	PCM5102A
SCL	<<	GND	PCM5102A
BCK	<<	GPIO27	ESP32-WROOM
DIN	<<	GPIO25	ESP32-WROOM
LCK	<<	GPIO26	ESP32-WROOM
FMT	<<	GND	
XMT	<<	+3.3V	PCM5102A

Audio

Links uit	>>	Links in versterker
Rechts uit	>>	Rechts in versterker

GND-uit niet verbinden met GND-in versterker, versterker wordt door IRF540 gestuurd via de GND

SD module

+3V3	<<	+3.3V	ESP32-WROOM
GND	<<	GND	Voeding
CS	<<	GPIO5	ESP32-WROOM
MOSI	<<	GPIO23	ESP32-WROOM
MISO	<<	GPIO19	ESP32-WROOM
SCK	<<	GPIO18	ESP32-WROOM

DS3231 RTC

VCC	<<	+3.3V	ESP32-WROOM
GND	<<	GND	ESP32-WROOM
SDA	<<	GPIO21	ESP32-WROOM
SCL	<<	GPIO22	ESP32-WROOM
SQW	<<	GPIO32	ESP32-WROOM

MAX7219 module

VCC	<<	+5V	Voeding
GND	<<	GND	Voeding
CS	<<	GPIO15	ESP32-WROOM
DIN	<<	GPIO13	ESP32-WROOM
CLK	<<	GPIO14	ESP32-WROOM

Drukknop alarm uit

GPIO4	<<	drukknop N.O.	<< + 3.3V ESP32-WROOM
GPIO4	<<	R 4K7	<< GND

Versterker

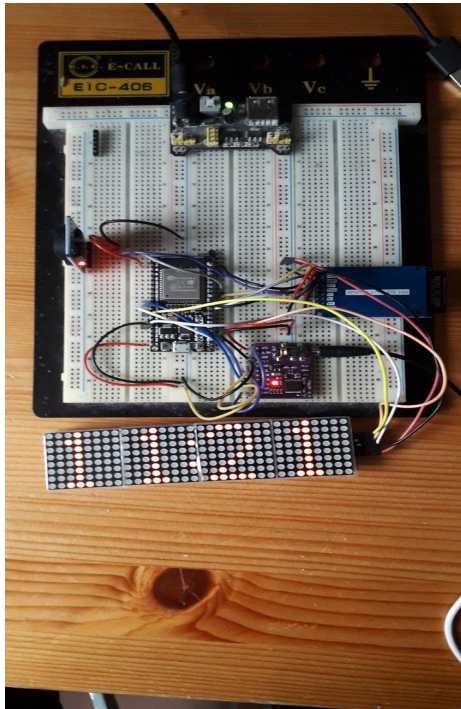
DC+	<<	+5V	voeding
DC-	<<	Drain	IRF540 (IRL540)
Links in	<<	Links uit PCM5102A	
Rechts in	<<	Rechts uit PCM5102A	

GND-in NIET VERBINDEN

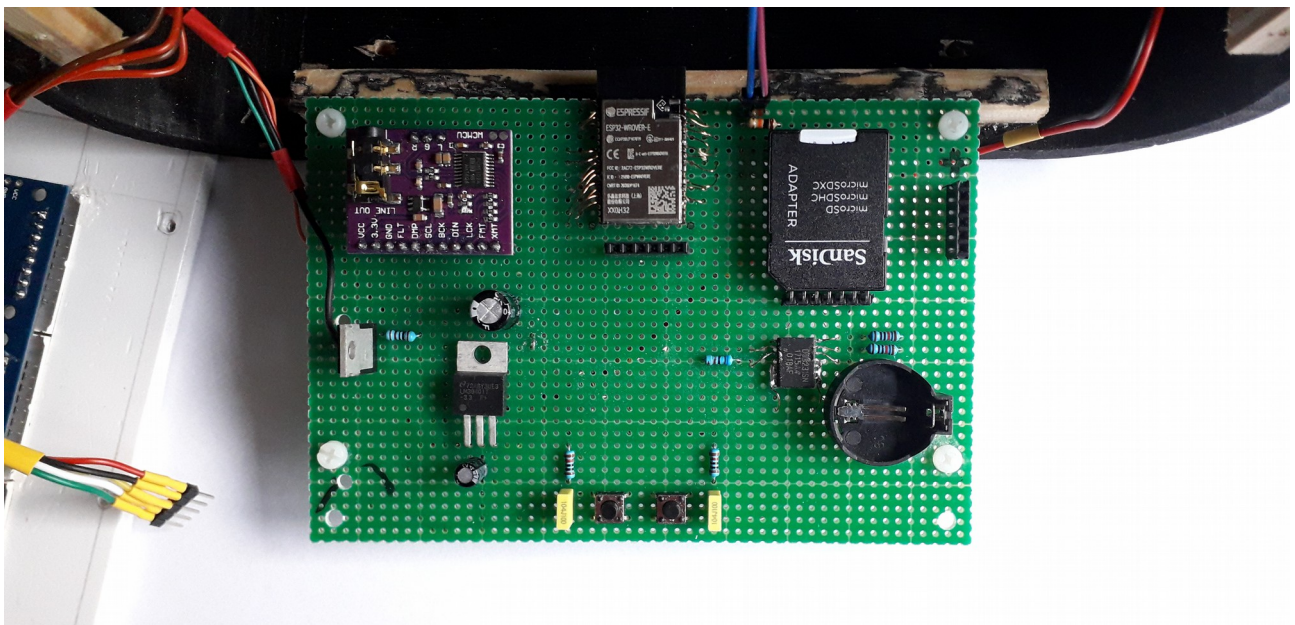
IRF540(IRL540)

Source	<<	GND	Voeding
Drain	>>	GND	Versterker
Gate	<< R 1K	<< GPIO33	ESP32-WROOM

Breadboard



Hobby versie



Kleef indien gewenst 1 of meerdere lagen folie (zie ‘*wat hebben we nodig*’ hierboven) op de MAX7219 module om de lichtintensiteit te minderen.

Ga naar <https://github.com/thieu-b55/ESP32-clock-radio-with-web-interface>

download het zipbestand : SD card files.zip en unzip.

download het programma : ESP32_webradio_WROOM_wekker.ino

Kopieer de bestanden die je vindt in de folder SD card files (totaal, pswd, ssid en zender_data.csv) naar de SD kaart en plaats SD kaart in de SD kaart houder.

Open het programma ESP32_webradio_WROOM_wekker.ino met de Arduino IDE.

Settings van de Arduino IDE zie screenprint. Poort setting is afhankelijk van jouw configuratie.



Upload programma in ESP32-WROOM

Bij 1ste gebruik : verbinden met

netwerk : **ESP32klok**
paswoord : **ESP32pswd**

Open de webpagina op adres 192.168.4.1

11:31 192.168.4.1 67%

Stop mp3 speler

- + OK

EQ -40 <-> 6 Volume 0 <-> 21

L: 2 M: 0 H: 2 V: 16

OK

ESP32 Netwerk instellingen

ssid :

pswd :

Gewenst IP address (default 192.168.1.178)

192 168 1 178

Bevestig

Onder de titel **ESP Netwerk instellingen**

ssid

pswd

van thuisnetwerk invullen.

Onder **Gewenst IP address ...**

indien gewenst kan hier een ander IP address worden ingegeven. Hou rekening met de range van de modem.

Na <Bevestig> herstart de ESP32 automatisch, als alles correct is ingevuld is het netwerk

ESP32klok niet meer beschikbaar.

Maak terug verbinding met thuisnetwerk en ga naar 192.168.1.178, of naar het zelf gekozen IP-address.

10:42

VoWiFi
LTE1 51%

192.168.1.178



ESP32 wekkerradio webinterface

Alarm	Dag	Tijd	Mode	Wake_up
1	Dagelijks	06:58	radio	1
2	Woensdag	12:34	mp3	0
3	Weekdagen	00:00	mp3	0

+	Aktief	Dag	Tijd	Mode	Wake_up
2	1	3	12:34	0	0
-					
OK					

Radio bediening

-	Alarm	+
---	-------	---

Radio 1 Classics

Radio 1 Classics
- + OK

EQ -40 <-> 6 Volume 0 <-> 21

L :	2	M :	0	H :	2	V :	16
OK							

Instellen zender en url : 10

- + OK

Instellen tijdzone zomertijd

Vrijdag 10:42		
uren	minuten	zomertijd
1	0	0
OK		

Klok helderheid (0 <-> 15)

0
OK

Hoe werkt het :

Wekkerradio

3 actieve alarmen

Alarm	Dag	Tijd	Mode	Wake_up
1	Dagelijks	06:58	radio	1
2	Woensdag	12:34	mp3	0
3	Weekdagen	00:00	mp3	0

1 actief alarm

Alarm	Dag	Tijd	Mode	Wake_up
1	Dagelijks	06:58	radio	1

In dit vak op de webpagina zien we welke alarmen er actief zijn en welke de instellingen zijn. Alleen de actieve alarmen worden getoond.

Instellen alarmen

<input data-bbox="451 1196 531 1238" type="button" value="+"/>	Aktief	Dag	Tijd	Mode	Wake_up
<input data-bbox="448 1267 534 1308" type="text" value="1"/>	<input data-bbox="571 1267 657 1308" type="text" value="1"/>	<input data-bbox="694 1267 780 1308" type="text" value="8"/>	<input data-bbox="810 1267 906 1308" type="text" value="06:58"/>	<input data-bbox="938 1267 1024 1308" type="text" value="1"/>	<input data-bbox="1056 1267 1142 1308" type="text" value="1"/>
<input data-bbox="451 1339 531 1382" type="button" value="-"/>					
<input data-bbox="767 1395 831 1435" type="button" value="OK"/>					

In dit vak kunnen we de 3 alarmen instellen.

+ gaat naar het volgende alarm

– naar het vorige alarm

Aktief

0 alarm niet actief

1 alarm actief

Dag

Welke dag(en) moet er gealarmeerd worden

1: maandag 2: dinsdag 3: woensdag 4: donderdag 5: vrijdag 6: zaterdag 7: zondag

8: dagelijks 9: weekdagen 10: weekend

Tijd

Uur en minuten van alarm begin

Mode

- 0 mp3 speler. Zorg ervoor dat de mp3lijst is aangemaakt, zie verder in handleiding.
- 1 internetradio

Wake_up

- 0 geen wake_up
- 1 indien er zich een file wake_up.mp3 in de root van de SD kaart bevindt worden deze eerst geopend alvorens over te gaan naar mp3 speler of internetradio

Accepteren met OK

Radio/mp3 alarm of aan



+/- kiezen Alarm of Aan

Alarm radio/mp3 speler spelen als er een alarm is

Aan radio/mp3 speler spelen zolang Aan gekozen is. Alarmen werken nu niet.

Alarm afduwen met behulp van drukknop verbonden met GPIO4 van ESP32

Alarm schakelt zichzelf uit na 1 uur

Om alarm definitief uit te schakelen , alarm op niet actief zetten.

Instellen tijdzone zomer/winteruur



uren uren afwijking ten opzichte UTC tijd + of –

minuten minuten afwijking ten opzichte van UTC tijd (0 of 30)

zomertijd 1 = zomertijd 0 = wintertijd

Accepteren met OK

Licht intensiteit

Klok helderheid (0 <-> 15)

0

OK

Vul hier de gewenste lichtsterkte van de display in. Waarde van 0 tot en met 15.

Accepteren met OK

Hoe werkt het :

Zenderkeuze

Een andere zender kiezen wanneer de radio in Alarm stand staat lukt niet altijd, best is de radio in de Aan stand te zetten alvorens van zender te willen wisselen.



Radio 1 Classics is radio die momenteel speelt.

Indien titel en uitvoerder beschikbaar zijn worden deze hieronder getoond.

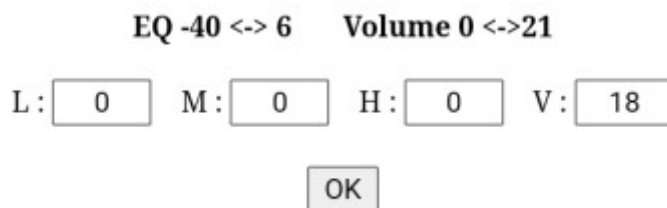
Keuzevak

Radio 1 Classics staat nu in het keuzevak, met behulp van de + en – toets kan men een ander station selecteren.

Accepteren met OK

In de keuzelijst is er ook nog de keuze **mp3 lijst maken** en **mp3 speler** hierover meer verder in de handleiding.

Volume en EQ



L : laag

M: midden

H: hoog

V: volume

L M H kan je instellen tussen -40 en 6

V kan je instellen tussen 0 en 21

bevestigen met <OK>

Zenders instellen



Instellen zender en url : 6

Radio 10 Non-Stop

http://playerservices.streamtheworld.com/api/livestrea...

- + OK

Reeds ingestelde zenders of nog lege posities kan je veranderen naar keuze. Maximum in te stellen zenders is 10.

In het eerste vak onder <Instellen zender en url : ..> kan je een willekeurige naam invullen voor de zender.

In het tweede vak moet het webadres van de zender ingevuld worden.

Bevestigen met <OK>, naar een volgende of vorige positie gaan doe je met de <-><+> toetsen.

mp3 speler.

Het beste is de radio in de Aan stand te zetten en niet in de Alarm stand alvorens de mp3 bestanden in te lezen.

Alvorens mp3 bestanden te kunnen beluisteren dienen er eerst songlijsten te worden aangemaakt. Bij een groot aantal mp3 bestanden is het aan te raden om deze gelijkmatig te verdelen in verschillende folders.

Beginnende bij mp3_0 en zo verder mp3_1, mp3_2.

Deze nummers dienen elkaar op te volgen. Programma stopt met zoeken als de volgende mp3_X folder niet bestaat.

Het beste is om in elke mp3_X folder ongeveer evenveel bestanden te hebben als er mp3_X folders zijn.

Dus bij 900 mp3's geeft dit 30 mp3_X folders met elk 30 mp3 files.

SD programma leest sequentieel en op deze manier zijn er het minst aantal lees acties nodig om het laatste mp3 bestand te vinden.

Belangrijk:

Als dit niet de 1ste maal is dat je een mp3 lijst maakt moet je eerst alle *songlijstx* folders van de SD kaart verwijderen.

Schermafdruck van SD kaart met mp3_ folders die nog niet ingelezen zijn.



mp3_0	32,8 kB map
mp3_1	32,8 kB map
mp3_2	32,8 kB map
mp3_3	32,8 kB map
mp3_4	32,8 kB map
mp3_5	32,8 kB map
mp3_6	32,8 kB map
mp3_7	32,8 kB map
mp3_8	32,8 kB map
mp3_9	32,8 kB map
mp3_10	16,4 kB map
pswd	20 byte plattetekst-document
ssid	14 byte plattetekst-document
totaal	4 byte plattetekst-document
zender_data.csv	4,9 kB CSV-document

Als je al je mp3's naar de SD kaart gekopieerd hebt en alle eventuele songlijstx folders verwijderd hebt, plaats dan de SD kaart terug in de houder.

In het gedeelte van de zenderkeuze ga naar <mp3 lijst maken> en druk <OK>



Het inlezen begint.



























Het programma heeft ongeveer 2 minuten per 1000 mp3 bestanden nodig indien deze gelijkmatig verdeeld zijn over verschillende mp3_X folders zoals hierboven beschreven.

De voortgang van het inlezen is te volgen door rechtsboven in de webpagina op het refresh cirkeltje te drukken.

Na het inlezen van alle mp3 bestanden gaat het programma over naar mp3 speler



Schermafdruck van de SD kaart na het inlezen van de mp3_.. folders

 mp3_0	32,8 kB map
 mp3_1	32,8 kB map
 mp3_2	32,8 kB map
 mp3_3	32,8 kB map
 mp3_4	32,8 kB map
 mp3_5	32,8 kB map
 mp3_6	32,8 kB map
 mp3_7	32,8 kB map
 mp3_8	32,8 kB map
 mp3_9	32,8 kB map
 mp3_10	16,4 kB map
 songlijst0	16,4 kB map
 songlijst1	16,4 kB map
 songlijst2	16,4 kB map
 songlijst3	16,4 kB map
 songlijst4	16,4 kB map
 songlijst5	16,4 kB map
 songlijst6	16,4 kB map
 songlijst7	16,4 kB map
 songlijst8	16,4 kB map
 songlijst9	16,4 kB map
 songlijst10	16,4 kB map
 pswd	20 byte plattetekst-document
 ssid	14 byte plattetekst-document
 totaal	4 byte plattetekst-document
 zender_data.csv	4,9 kB CSV-document

Belangrijk:

om een eventueel eindeloze loop te vermijden bij een fout tijdens het mp3 spelen wordt er na een reset of spanningsuitval steeds opgestart als webradio.

Zo dat was het zo een beetje,
groeten en veel luistergenot,
thieu-b55

```

/*
 * MIT License
 *
 * Copyright (c) 2023 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.
 */
/* ESP32-WROVER
 * Board :          ESP32 WROVER Module
 * Partition Scheme : Minimal SPIFFS
 *
 *
 * PCM5102A
 * FLT >> GND
 * DMP >> 3.3V
 * SCL >> GND
 * BCK >> GPIO27
 * DIN  >> GPIO25
 * LCK  >> GPIO26
 * FMT >> GND
 * XMT  >> 3.3V
 * VCC  >> 5V
 * GND  >> GND
 *
 * SD Kaart
 * CS   >> GPIO05
 * MOSI >> GPIO23
 * MISO >> GPIO19
 * SCK  >> GPIO18
 *
 * 4 digit LED matrix display (MAX7219)
 * CS   >> GPIO15
 * MOSI >> GPIO13
 * MISO >> GPIO02 niet gebruikt
 * CLK  >> GPIO14
 *
 * DS3231SN
 * SDA  >> GPIO21
 * SCL  >> GPIO22
 * INT  >> GPIO32
 *
 * Alarm uit
 *      >> GPIO4
 *
 * Audio librarie
 * https://github.com/schreibfaul1/ESP32-audioI2S
 *
 *
 * zender_data.csv
 * geen header
 * kolom 1 >> zendernaam
 * kolom2 >> zender url
 */

#include "Arduino.h"
#include "WiFi.h"
#include "Audio.h"
#include <SPI.h>
#include <Preferences.h>

```

```

#include "FS.h"
#include "SD.h"
#include "SPIFFS.h"
#include <CSV_Parser.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "Wire.h"
#include "time.h"

SPIClass SPI2(VSPI);
SPISettings spiSettings = SPISettings(15000, SPI_MSBFIRST, SPI_MODE2);

Audio audio;
Preferences pref;
AsyncWebServer server(80);

#define DS3231SN      0x68

#define I2C_SDA      21
#define I2C_SCL      22

//PCM1502A
#define I2S_DOUT      25
#define I2S_BCLK      27
#define I2S_LRC       26

#define mySPI_CS       15
#define mySPI_MOSI     13
#define mySPI_MISO     02
#define mySPI_SCK      14

#define VERSTERKER     33
#define MINUUT_PULS    32
#define ALARM_UIT      4

#define MAX_AANTAL_KANALEN 11

bool kiezen = false;
bool lijst_maken = false;
bool speel_mp3 = false;
bool webradio = false;
bool schrijf_csv = false;
bool netwerk;
bool nog_mp3;
bool mp3_ok;
bool mp3_lijt_maken = false;
bool ssid_ingevoeld = false;
bool pswd_ingevoeld = false;
bool songlijsten = false;
bool songlijst_bestaat_bool;
bool tijd_init_bool = false;
bool tijd_update_bool = false;
bool alarm_1_set_bool = false;
bool alarm_2_set_bool = false;
bool alarm_3_set_bool = false;
bool wake_up_bool;
bool wake_up_end_bool;
bool alarm_1_afgewerkt_bool;
bool alarm_2_afgewerkt_bool;
bool alarm_3_afgewerkt_bool;
bool minuut_interrupt_bool = false;
char ip_char[20];
char songfile[200];
char mp3file[200];
char song[200];
char datastring[200];
char password[40];
char ssid[40];
char zendernaam[40];
char url[100];
char mp3_dir[10];
char folder_mp3[10];
char aantal_mp3[10];
char songlijst_dir[12];
char totaal_mp3[15];
char mp3_lijt_folder[10];

```

```

char mp3_lijst_aantal[5];
char mp3_in_folder[5];
char leeg[0];
char zenderarray[MAX_AANTAL_KANALEN][40];
char urlarray[MAX_AANTAL_KANALEN][100];
char dag_uren_char[25];
char lees_char[20];
char file_char[20];
char data_char[20];
char alarm_tijd_char[6];
char alarm_tijd_1_char[6];
char alarm_tijd_2_char[6];
char alarm_tijd_3_char[6];
char tijd_char[6];
char hoeveel_mp3[5];
const char* IP_1_KEUZE = "ip_1_keuze";
const char* IP_2_KEUZE = "ip_2_keuze";
const char* IP_3_KEUZE = "ip_3_keuze";
const char* IP_4_KEUZE = "ip_4_keuze";
const char* KEUZEMIN_INPUT = "minKeuze";
const char* KEUZEPLUS_INPUT = "plusKeuze";
const char* BEVESTIGKEUZE_INPUT = "bevestigKeuze";
const char* LAAG = "laag_keuze";
const char* MIDDEN = "midden_keuze";
const char* HOOG = "hoog_keuze";
const char* VOLUME = "volume_keuze";
const char* VOLUME_BEVESTIG = "bevestig_volume";
const char* APssid = "ESP32klok";
const char* APpswd = "ESP32pswd";
const char* STA_SSID = "ssid";
const char* STA_PSWD = "pswd";
const char* ZENDER = "zender";
const char* URL = "url";
const char* ARRAY_MIN = "array_index_min";
const char* ARRAY_PLUS = "array_index_plus";
const char* BEVESTIG_ZENDER = "bevestig_zender";
const char* MIN_INPUT = "min";
const char* PLUS_INPUT = "plus";
const char* BEVESTIG_MP3 = "bevestig_mp3";
const char* ALARM_PLUS = "alarm_plus";
const char* ALARM_MIN = "alarm_min";
const char* AKTIEF = "aktief";
const char* DAG = "dag";
const char* TIJD = "tijd";
const char* MODE = "mode";
const char* WAKEUP = "wakeuper";
const char* ALARM_BEVESTIG = "alarm_bevestig";
const char* RADIO_BEDIENING_MIN = "radio_bediening_min";
const char* RADIO_BEDIENING_PLUS = "radio_bediening_plus";
const char* H_CHAR = "h";
const char* TZ_UUR = "tz_uur";
const char* TZ_MINUUT = "tz_minuut";
const char* ZOMER = "zomer";
const char* TZ_BEVESTIG = "tz_bevestig";
const char* INTENSITEIT_GEWENST = "intensiteit_gewenst";
const char* INTENSITEIT_BEVESTIG = "intensiteit_bevestig";
const char* week_dagen_char[] = {"dummy", "Maandag", "Dinsdag", "Woensdag", "Donderdag", "Vrijdag", "Zaterdag", "Zondag", "Dagelijks", "Weekdagen", "Weekend"};
const char* mode_char[] = {"mp3", "radio"};
const char* radio_bediening_char[] = {"Alarm", "Aan"};
int gekozen = 1;
int keuze = 1;
int volgend;
int totaalmp3;
int eerste;
int tweede;
int songindex;
int row;
int volume_keuze;
int volume_gekozen;
int laag_keuze;
int laag_gekozen;
int midden_keuze;
int midden_gekozen;
int hoog_gekozen;
int hoog_keuze;

```

```

int mp3_per_songlijst;
int array_index = MAX_AANTAL_KANALEN - 1;
int songlijst_index_vorig;
int songlijst_index;
int mp3_folder_teller;
int teller = 0;
int mp3_aantal;
int gn_keuze = 0;
int ip_1_int = 192;
int ip_2_int = 168;
int ip_3_int = 1;
int ip_4_int = 178;
int dag_int;
int uren_int;
int minuten_int;
int seconden_int;
int local_uren_int;
int local_minuten_int;
int local_seconden_int;
int local_dag_int;
int utc_uren_int;
int utc_minuten_int;
int utc_dag_int;
int utc_offset_uren_int = 0;
int utc_offset_minuten_int = 0;
int alarm_teller_int = 0;
int dag_teller_int = 1;
int alarm_aktief_int;
int alarm_dag_int;
int alarm_uur_int;
int alarm_minuut_int;
int alarm_mode_int;
int alarm_wake_up_int;
int twaalf_uur_int = 0;
int winter_zomer_int = 0; // winter = 0 <> zomer = 1
int intensiteit_int = 0;;
int radio_bediening_int = 0;
int alarm_aktief_array[] = {0, 0, 0};
int alarm_dag_array[] = {8, 8, 8};
int alarm_uur_array[] = {0, 0, 0};
int alarm_minuut_array[] = {0, 0, 0};
int alarm_mode_array[] = {0, 0, 0};
int wake_up_array[] = {0, 0, 0};
uint8_t cijfers[][11] = {{0x38, 0x44, 0x4c, 0x54, 0x64, 0x44, 0x44, 0x38}, //0
                          {0x10, 0x30, 0x10, 0x10, 0x10, 0x10, 0x10, 0x38}, //1
                          {0x38, 0x44, 0x04, 0x04, 0x08, 0x10, 0x20, 0x7c}, //2
                          {0x7c, 0x08, 0x10, 0x08, 0x04, 0x04, 0x44, 0x38}, //3
                          {0x08, 0x18, 0x28, 0x48, 0x48, 0x7c, 0x08, 0x08}, //4
                          {0x7c, 0x40, 0x78, 0x04, 0x04, 0x04, 0x44, 0x38}, //5
                          {0x18, 0x20, 0x40, 0x78, 0x44, 0x44, 0x44, 0x38}, //6
                          {0x7c, 0x04, 0x04, 0x08, 0x10, 0x20, 0x20, 0x20}, //7
                          {0x38, 0x44, 0x44, 0x38, 0x44, 0x44, 0x44, 0x38}, //8
                          {0x38, 0x44, 0x44, 0x44, 0x3c, 0x04, 0x08, 0x30}, //9
                          {0x3c, 0x42, 0xa5, 0x81, 0xa5, 0x99, 0x42, 0x3c}}; //smiley

uint8_t tiental_uur_int;
uint8_t eenheden_uur_int;
uint8_t tiental_minuut_int;
uint8_t eenheden_minuut_int;
unsigned long wacht_op_netwerk;
unsigned long inlezen_begin;
unsigned long inlezen_nu;
unsigned long wachttijd;
unsigned long alarm_aanwezig_long;
String songstring = "
";
String inputString = "
";
String mp3titel = "
";
String zenderFile = "
";
String urlFile = "
";
String maxurl = "
";
String totaal = "
";
String streamsong = "
";
String mp3_folder = "
";
String songlijst_folder = "
";

```

```

String mp3test = "mp3";
String ip_1_string = " ";
String ip_2_string = " ";
String ip_3_string = " ";
String ip_4_string = " ";
String ip_string = " ";
String lees_string = " ";
String aktief_string = "/aktief";
String dag_string = "/dag";
String uur_string = "/uur";
String minuut_string = "/minuut";
String mode_string = "/mode";
String wake_up_string = "/wup";
String file_string = " ";
String data_string = " ";
String alarm_tijd_string = " ";
String alarm_aktief_string = " ";
String alarm_dag_string = " ";
String alarm_mode_string = " ";
String alarm_wake_up_string = " ";
String mp3_per_folder = " ";

void readFile(fs::FS &fs, const char * path){
    File file = fs.open(path);
    if(!file){
        Serial.println("Kan file niet openen om te lezen : ");
        Serial.println(path);
        return;
    }
    teller = 0;
    inputString = "";
    while(file.available()){
        inputString += char(file.read());
        teller++;
    }
    file.close();
}

void readIP(fs::FS &fs, const char * path){
    int temp;
    int temp1;
    File file = fs.open(path);
    if(!file){
        return;
    }
    teller = 0;
    inputString = "";
    while(file.available()){
        inputString += char(file.read());
        teller++;
    }
    temp = inputString.indexOf('.');
    ip_1_int = (inputString.substring(0, temp - 1)).toInt();
    temp1 = inputString.indexOf('.', temp + 1);
    ip_2_int = (inputString.substring(temp + 1, temp1 - 1)).toInt();
    temp = inputString.indexOf('.', temp1 + 1);
    ip_3_int = (inputString.substring(temp1 + 1, temp - 1)).toInt();
    ip_4_int = (inputString.substring(temp + 1, inputString.length() - 1)).toInt();
    file.close();
}

void deleteFile(fs::FS &fs, const char * path){
    fs.remove(path);
}

void appendFile(fs::FS &fs, const char * path, const char * message){
    File file = fs.open(path, FILE_APPEND);
    file.print(message);
    file.close();
}

void writeFile(fs::FS &fs, const char * path, const char * message){
    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Kan file niet openen om te schrijven : ");
        Serial.println(path);
    }
}

```

```

        return;
    }
    file.print(message);
    file.close();
}

void testDir(fs::FS &fs, const char * path){
    File root = fs.open(path);
    if(root){
        songlijst_bestaat_bool = true;
    }
}

void createDir(fs::FS &fs, const char * path){
    File root = fs.open(path);
    if(root){
        songlijsten = true;
        lijst_maken = false;
        while(1){
            yield();
        }
    }
    else{
        fs.mkdir(path);
    }
}

void audio_showstreamtitle(const char *info){
    if(kiezen == false){
        streamsong = info;
    }
}

void audio_eof_mp3(const char *info){
    if(wake_up_bool == true){
        wake_up_end_bool = true;
        return;
    }
    mp3_volgend();
    streamsong = mp3titel.substring(0, (mp3titel.length() - 4));
}

void files_in_mp3_0(fs::FS &fs, const char * dirname, uint8_t levels){
    File root = fs.open(dirname);
    if(!root){
        Serial.println("Geen mp3_0 folder");
        return;
    }
    File file = root.openNextFile();
    mp3_per_songlijst = 0;
    while(file){
        file = root.openNextFile();
        mp3_per_songlijst ++;
    }
    String(mp3_per_songlijst).toCharArray(hoeveel_mp3, (String(mp3_per_songlijst).length() + 1));
    writeFile(SD, "/files", hoeveel_mp3);
}

void maak_lijst(fs::FS &fs, const char * dirname){
    File root = fs.open(dirname);
    if(!root){
        nog_mp3 = false;
        return;
    }
    File file = root.openNextFile();
    while(file){
        songlijst_index = mp3_aantal / mp3_per_songlijst;
        if(songlijst_index != songlijst_index_vorig){
            songlijst_index_vorig = songlijst_index;
            songlijst_folder = "/songlijst" + String(songlijst_index);
            songlijst_folder.toCharArray(songlijst_dir, (songlijst_folder.length() + 1));
            createDir(SD, songlijst_dir);
        }
        songstring = file.name();
        songlijst_folder = "/songlijst" + String(songlijst_index) + "/" + String(mp3_aantal);
        songlijst_folder.toCharArray(songlijst_dir, (songlijst_folder.length() + 1));
    }
}

```



```

    songstring = file.name();
    songstring.toCharArray(song, (songstring.length() + 1));
    writeFile(SD, songlijst_dir, song);
    file = root.openNextFile();
    mp3_aantal ++;
}
}

void mp3_lijt_maken_gekozen(){
    inlezen_begin = millis();
    files_in_mp3_0(SD, "/mp3_0", 1);
    mp3_aantal = 0;
    nog_mp3 = true;
    mp3_folder_teller = 0;
    songlijst_index_vorig = -1;
    while(nog_mp3){
        mp3_folder = "/mp3_" + String(mp3_folder_teller);
        inlezen_nu = millis() - inlezen_begin;
        Serial.println(mp3_aantal);
        Serial.println(inlezen_nu);
        mp3_folder.toCharArray(mp3_dir, (mp3_folder.length() + 1));
        maak_lijt(SD, mp3_dir);
        mp3_folder_teller ++;
    }
    String(mp3_aantal).toCharArray(totaal_mp3, (String(mp3_aantal - 1).length() + 1));
    writeFile(SD, "/totaal", totaal_mp3);
    int verstreken_tijd = (millis() - inlezen_begin) / 1000;
    Serial.println("verstreken tijd");
    Serial.println(verstreken_tijd);
    lijst_maken = false;
    keuze = -1;
    gekozen = -1;
    mp3_gekozen();
}

void mp3_gekozen(){
    readFile(SD, "/totaal");
    totaal = inputString.substring(0, teller);
    totaalmp3 = totaal.toInt();
    readFile(SD, "/files");
    mp3_per_folder = inputString.substring(0, teller);
    mp3_per_songlijst = mp3_per_folder.toInt();
    mp3_volgend();
}

void mp3_volgend(){
    mp3_ok = false;
    while(mp3_ok == false){
        volgend = random(totaalmp3);
        songindex = volgend / mp3_per_songlijst;
        songstring = "/songlijst" + String(songindex) + "/" + String(volgend);
        songstring.toCharArray(songfile, (songstring.length() + 1));
        readFile(SD, songfile);
        inputString.toCharArray(mp3file, inputString.length() + 1);
        mp3_ok = audio.connecttoFS(SD, mp3file);
    }
    if((radio_bediening_int == 0) && (alarm_1_set_bool == false) && (alarm_2_set_bool == false) && (alarm_3_set_bool == false)){
        audio.stopSong();
    }
    songstring = String(mp3file);
    eerste = songstring.indexOf("/");
    tweede = songstring.indexOf("/", eerste + 1);
    mp3titel = songstring.substring(tweede + 1);
    streamsong = mp3titel.substring(0, (mp3titel.length() - 4));
}

void radio_gekozen(){
    memset(url, 0, sizeof(url));
    strcpy(url, urlarray[keuze]);
    memset(zendernaam, 0, sizeof(zendernaam));
    strcpy(zendernaam, zenderarray[keuze]);
    audio.stopSong();
    delay(100);
    audio.connecttohost(url);
    kiezen = false;
}

```

```

if((radio_bediening_int == 0) && (alarm_1_set_bool == false) && (alarm_2_set_bool == false) && (alarm_3_set_bool == false)){
    audio.stopSong();
}
}

void schrijf_naar_csv(){
    char terminator = char(0x0a);
    String datastring = "
";
    char datastr[150];
    deleteFile(SD, "/zender_data.csv");
    for(int x = 0; x < MAX_AANTAL_KANALEN; x++){
        datastring = String(zenderarray[x]) + "," + String(urlarray[x]) + String(terminator);
        datastring.toCharArray(datastr, (datastring.length() + 1));
        appendFile(SD, "/zender_data.csv", datastr);
    }
    lees_CSV();
}

void lees_CSV(){
    CSV_Parser cp("ss", false, ',');
    if(cp.readSDfile("/zender_data.csv")){
        char **station_naam = (char**)cp[0];
        char **station_url = (char**)cp[1];
        for(row = 0; row < cp.getRowsCount(); row++){
            memset(zenderarray[row], 0, sizeof(zenderarray[row]));
            strcpy(zenderarray[row], station_naam[row]);
            memset(urlarray[row], 0, sizeof(urlarray[row]));
            strcpy(urlarray[row], station_url[row]);
        }
    }
}

void display_setup(uint8_t adres, uint8_t waarde){
    SPI2.beginTransaction(spiSettings);
    digitalWrite(mySPI_CS, LOW);
    SPI2.transfer(adres);
    SPI2.transfer(waarde);
    SPI2.transfer(adres);
    SPI2.transfer(waarde);
    SPI2.transfer(adres);
    SPI2.transfer(waarde);
    SPI2.transfer(adres);
    SPI2.transfer(waarde);
    digitalWrite(mySPI_CS, HIGH);
    SPI2.endTransaction();
}

void display_digits(uint8_t adres, uint8_t digit_1, uint8_t digit_2, uint8_t digit_3, uint8_t digit_4){
    SPI2.beginTransaction(spiSettings);
    digitalWrite(mySPI_CS, LOW);
    SPI2.transfer(adres);
    SPI2.transfer(digit_1);
    SPI2.transfer(adres);
    SPI2.transfer(digit_2);
    SPI2.transfer(adres);
    SPI2.transfer(digit_3);
    SPI2.transfer(adres);
    SPI2.transfer(digit_4);
    digitalWrite(mySPI_CS, HIGH);
    SPI2.endTransaction();
}

void initTime(){
    struct tm timeinfo;
    configTime(0, 0, "pool.ntp.org");
    if(!getLocalTime(&timeinfo)){
        return;
    }
    tijd_init_bool = true;
}

byte dec_naar_bcd(byte waarde){
    return (((waarde / 10) << 4) + (waarde % 10));
}

byte bcd_naar_dec(byte waarde){

```

```

    return (((waarde >> 4) * 10) + (waarde % 16));
}

```

```

void tijdzone_correctie(){
if(tijd_update_bool == true){
    utc_offset_uren_int = pref.getShort("uur_of");
    utc_offset_minuten_int = pref.getShort("min_of");
    winter_zomer_int = pref.getShort("w_z");
    local_uren_int = utc_uren_int + utc_offset_uren_int + winter_zomer_int;
    if(utc_offset_uren_int > -1){
        local_minuten_int = utc_minuten_int + utc_offset_minuten_int;
        if(local_minuten_int > 59){
            local_minuten_int -= 60;
            local_uren_int += 1;
        }
        if(local_uren_int > 23){
            local_uren_int -= 24;
            local_dag_int += 1;
            if(local_dag_int == 8){
                local_dag_int = 1;
            }
        }
    }
    if(utc_offset_uren_int < 0){
        local_minuten_int = utc_minuten_int - utc_offset_minuten_int;
        if(local_minuten_int < 0){
            local_minuten_int += 60;
            local_uren_int -= 1;
        }
        if(local_uren_int < 0){
            local_uren_int += 24;
            local_dag_int -= 1;
            if(local_dag_int == 0){
                local_dag_int = 7;
            }
        }
    }
}
Serial.print(week_dagen_char[local_dag_int]);
Serial.print(" : ");
Serial.print(local_uren_int);
Serial.print(" : ");
Serial.println(local_minuten_int);
Wire.beginTransmission(DS3231SN);
Wire.write(0x00);
Wire.write(dec_naar_bcd(local_seconden_int));
Wire.write(dec_naar_bcd(local_minuten_int));
Wire.write(dec_naar_bcd(local_uren_int) & 0x3f);
Wire.write(local_dag_int);
Wire.endTransmission();
}
}

```

```

void tijd_update(){
    tijd_update_bool = false;
    struct tm timeinfo;
    if(getLocalTime(&timeinfo)){
        tijd_update_bool = true;
        Serial.println("update ok");
        utc_dag_int = timeinfo.tm_wday;
        utc_uren_int = timeinfo.tm_hour;
        utc_minuten_int = timeinfo.tm_min;
        local_seconden_int = timeinfo.tm_sec;
        local_dag_int = utc_dag_int + 7;
        if(local_dag_int > 7){
            local_dag_int -= 7;
        }
        tijdzone_correctie();
        return;
    }
    Serial.println("update nok");
}

```

```

void smiley(){
    display_digits(0x01, cijfers[10][0], cijfers[10][0], cijfers[10][0], cijfers[10][0]);
    display_digits(0x02, cijfers[10][1], cijfers[10][1], cijfers[10][1], cijfers[10][1]);
    display_digits(0x03, cijfers[10][2], cijfers[10][2], cijfers[10][2], cijfers[10][2]);
}

```

```

display_digits(0X04, cijfers[10][3], cijfers[10][3], cijfers[10][3], cijfers[10][3]);
display_digits(0X05, cijfers[10][4], cijfers[10][4], cijfers[10][4], cijfers[10][4]);
display_digits(0X06, cijfers[10][5], cijfers[10][5], cijfers[10][5], cijfers[10][5]);
display_digits(0X07, cijfers[10][6], cijfers[10][6], cijfers[10][6], cijfers[10][6]);
display_digits(0X08, cijfers[10][7], cijfers[10][7], cijfers[10][7], cijfers[10][7]);
}

void tijd_naar_led(){
  Wire.beginTransmission(DS3231SN);
  Wire.write(0x01);
  Wire.endTransmission();
  Wire.requestFrom(DS3231SN, 3);
  minuten_int = (bcd_naar_dec(Wire.read()));
  uren_int = (bcd_naar_dec(Wire.read()));
  dag_int = Wire.read();
  tiental_uur_int = uren_int / 10;
  eenheden_uur_int = uren_int - (tiental_uur_int * 10);
  tiental_minuut_int = minuten_int / 10;
  eenheden_minuut_int = minuten_int - (tiental_minuut_int * 10);
  display_digits(0x01, cijfers[tiental_uur_int][0], cijfers[eenheden_uur_int][0], cijfers[tiental_minuut_int][0], cijfers[eenheden_minuut_int][0]);
  display_digits(0X02, cijfers[tiental_uur_int][1], cijfers[eenheden_uur_int][1], cijfers[tiental_minuut_int][1], cijfers[eenheden_minuut_int][1]);
  display_digits(0X03, cijfers[tiental_uur_int][2], cijfers[eenheden_uur_int][2] + 1, cijfers[tiental_minuut_int][2], cijfers[eenheden_minuut_int][2]);
  display_digits(0X04, cijfers[tiental_uur_int][3], cijfers[eenheden_uur_int][3], cijfers[tiental_minuut_int][3], cijfers[eenheden_minuut_int][3]);
  display_digits(0X05, cijfers[tiental_uur_int][4], cijfers[eenheden_uur_int][4], cijfers[tiental_minuut_int][4], cijfers[eenheden_minuut_int][4]);
  display_digits(0X06, cijfers[tiental_uur_int][5], cijfers[eenheden_uur_int][5], cijfers[tiental_minuut_int][5], cijfers[eenheden_minuut_int][5]);
  display_digits(0X07, cijfers[tiental_uur_int][6], cijfers[eenheden_uur_int][6], cijfers[tiental_minuut_int][6], cijfers[eenheden_minuut_int][6]);
  display_digits(0X08, cijfers[tiental_uur_int][7], cijfers[eenheden_uur_int][7], cijfers[tiental_minuut_int][7], cijfers[eenheden_minuut_int][7]);
}

void write_char(fs::FS &fs, const char * path, const char * message){
  File file = fs.open(path, FILE_WRITE);
  if(!file){
    return;
  }
  if(file.print(message)){

  }
  file.close();
}

void read_string(fs::FS &fs, const char * path){
  File file = fs.open(path);
  if(!file){
    return;
  }
  int teller = 0;
  memset(lees_char, 0, sizeof(lees_char));
  while(file.available()){
    lees_char[teller] = file.read();
    teller++;
  }
  file.close();
  lees_string = String(lees_char);
}

void schrijf_spiffs(){
  for(teller = 0; teller < 3; teller++){
    file_string = aktief_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    data_string = String(alarm_aktief_array[teller]);
    data_string.toCharArray(data_char, data_string.length() + 1);
    write_char(SPIFFS, file_char, data_char);
    file_string = dag_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    data_string = String(alarm_dag_array[teller]);
    data_string.toCharArray(data_char, data_string.length() + 1);
    write_char(SPIFFS, file_char, data_char);
    file_string = uur_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    data_string = String(alarm_uur_array[teller]);
    data_string.toCharArray(data_char, data_string.length() + 1);
    write_char(SPIFFS, file_char, data_char);
    file_string = minuut_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    data_string = String(alarm_minuut_array[teller]);
    data_string.toCharArray(data_char, data_string.length() + 1);
  }
}

```

```

write_char(SPIFFS, file_char, data_char);
file_string = mode_string + String(teller);
file_string.toCharArray(file_char, file_string.length() + 1);
data_string = String(alarm_mode_array[teller]);
data_string.toCharArray(data_char, data_string.length() + 1);
write_char(SPIFFS, file_char, data_char);
file_string = wake_up_string + String(teller);
file_string.toCharArray(file_char, file_string.length() + 1);
data_string = String(wake_up_array[teller]);
data_string.toCharArray(data_char, data_string.length() + 1);
write_char(SPIFFS, file_char, data_char);
}
}

void lees_spiffs(){
  for(teller = 0; teller < 3; teller++){
    file_string = aktief_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    alarm_aktief_array[teller] = lees_string.toInt();
    file_string = dag_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    alarm_dag_array[teller] = lees_string.toInt();
    file_string = uur_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    alarm_uur_array[teller] = lees_string.toInt();
    file_string = minuut_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    alarm_minuut_array[teller] = lees_string.toInt();
    file_string = mode_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    alarm_mode_array[teller] = lees_string.toInt();
    file_string = wake_up_string + String(teller);
    file_string.toCharArray(file_char, file_string.length() + 1);
    read_string(SPIFFS, file_char);
    wake_up_array[teller] = lees_string.toInt();
  }
}

void IRAM_ATTR minuut_voorbij() {
  minuut_interrupt_bool = true;
}

const char index_html[] = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head>
  <iframe style="display:none" name="hidden-form"></iframe>
  <title>Internetradio bediening</title>
  <meta name="viewport" content="width=device-width, initial-scale=.85">
  <style>
    div.kader {
      position: relative;
      left: 0px;
      width: 400px;
    }
    div.kader_1 {
      position: absolute;
      left : 0px;
      width: 80px;
    }
    div.kader_2 {
      position: absolute;
      left : 80px;
      width: 80px;
    }
    div.kader_3 {
      position: absolute;
      left : 160px;
      width: 80px;
    }
    div.kader_4 {

```

```

    position: absolute;
    left : 240px;
    width: 80px;
}
div.kader_5 {
    position: absolute;
    left : 320px;
    width: 80px;
}
div.vak_1 {
    position: absolute;
    left : 0px;
    width: 67px;
}
div.vak_2 {
    position: absolute;
    left : 67px;
    width: 67px;
    height: 8x;
}
div.vak_3 {
    position: absolute;
    left : 134px;
    width: 67px;
}
div.vak_4 {
    position: absolute;
    left : 201px;
    width: 67px;
}
div.vak_5 {
    position: absolute;
    left : 268px;
    width: 67px;
}
div.vak_6 {
    position: absolute;
    left : 335px;
    width: 67px;
}
div.blanco_20{
    width: auto;
    height: 20px;
}
div.blanco_30{
    width: auto;
    height: 30px;
}
div.blanco_40{
    width: auto;
    height: 40px;
}
div.blanco_60{
    width: auto;
    height: 60px;
}
div.blanco_80{
    width: auto;
    height: 80px;
}
}
</style>
</head>
<body>
<h3><center> ESP32 wekkerradio webinterface </center></h3>
<small>
<b>
<div class="kader">
    <div class="kader_1">
        <center>Alarm</center>
    </div>
    <div class="kader_2">
        <center>Dag</center>
    </div>
    <div class="kader_3">
        <center>Tijd</center>
    </div>

```

```

<div class="kader_4">
  <center>Mode</center>
</div>
<div class="kader_5">
  <center>Wake_up</center>
</div>
</div>
</b>
<div class="blanco_20">&nbsp;</div>
<div class="kader">
  <div class="kader_1">
    <center><b>%alarm_1%</b></center>
  </div>
  <div class="kader_2">
    <center>%dag_1%</center>
  </div>
  <div class="kader_3">
    <center>%tijd_1%</center>
  </div>
  <div class="kader_4">
    <center>%mode_1%</center>
  </div>
  <div class="kader_5">
    <center>%wake_up_1%</center>
  </div>
</div>
<div class="blanco_20">&nbsp;</div>
<div class="kader">
  <div class="kader_1">
    <center><b>%alarm_2%</b></center>
  </div>
  <div class="kader_2">
    <center>%dag_2%</center>
  </div>
  <div class="kader_3">
    <center>%tijd_2%</center>
  </div>
  <div class="kader_4">
    <center>%mode_2%</center>
  </div>
  <div class="kader_5">
    <center>%wake_up_2%</center>
  </div>
</div>
<div class="blanco_20">&nbsp;</div>
<div class="kader">
  <div class="kader_1">
    <center><b>%alarm_3%</b></center>
  </div>
  <div class="kader_2">
    <center>%dag_3%</center>
  </div>
  <div class="kader_3">
    <center>%tijd_3%</center>
  </div>
  <div class="kader_4">
    <center>%mode_3%</center>
  </div>
  <div class="kader_5">
    <center>%wake_up_3%</center>
  </div>
</div>
</small>
<div class="blanco_40">&nbsp;</div>
<small>
  <b>
<form action="/get" target="hidden-form">
  <div class="kader">
    <div class="vak_1">
      <center><input type="submit" value=" + " name="alarm_plus" onclick="bevestig()"></center>
    </div>
    <div class="vak_2">
      <center>Aktief</center>
    </div>
    <div class="vak_3">
      <center>Dag</center>
    </div>
  </div>

```



```

</div>
<div class="vak_4">
    <center>Tijd</center>
</div>
<div class="vak_5">
    <center>Mode</center>
</div>
<div class="vak_6">
    <center>Wake_up</center>
</div>
</div>
<b>
<div class="blanco_40">&nbsp;</div>
<div class="kader">
    <div class="vak_1">
        <center><input type="text" value=%alarm_nr% style="text-align:center;" size=1></center>
    </div>
    <div class="vak_2">
        <center><input type="text" value=%alarm_aktief% name="aktief" style="text-align:center;" size=1></center>
    </div>
    <div class="vak_3">
        <center><input type="text" value=%alarm_dag% name="dag" style="text-align:center;" size=1></center>
    </div>
    <div class="vak_4">
        <center><input type="text" value=%alarm_tijd% name="tijd" style="text-align:center;" size=2></center>
    </div>
    <div class="vak_5">
        <center><input type="text" value=%alarm_mode% name="mode" style="text-align:center;" size=1></center>
    </div>
    <div class="vak_6">
        <center><input type="text" value=%wake_up% name="wakeup" style="text-align:center;" size=1></center>
    </div>
</div>
<div class="blanco_40">&nbsp;</div>
<div class="kader">
    <div class="vak_1">
        <center><input type="submit" value=" - " name="alarm_min" onclick="bevestig()"></center>
    </div>
</div>
<div class="blanco_30">&nbsp;</div>
    <center><input type="submit" value="OK" name="alarm_bevestig" onclick="bevestig()"></center>
</form>
</small>
<h5><center> Radio bediening </center></h5>
<small>
<form action="/get" target="hidden-form">
<div class="kader">
    <div class="kader_2">
        <center><input type="submit" name="radio_bediening_min" value=" - " onclick="bevestig()"></center>
    </div>
    <div class="kader_3">
        <center><input type="text" value="%radio_bediening%" style="text-align:center;" size=1></center>
    </div>
    <div class="kader_4">
        <center><input type="submit" name="radio_bediening_plus" value=" + " onclick="bevestig()"></center>
    </div>
</div>
<div class="blanco_30">&nbsp;</div>
</form>
</small>
<h5><center> %zenderNu% </center></h5>
<p><small><center>%song%</center></small></p>
<center>
    <input type="text" style="text-align:center;" value="%selecteren%" name="keuze" size=30>
</center>
<br>
<form action="/get" target="hidden-form">
<center>
    <input type="submit" name="minKeuze" value=" - " onclick="bevestig()">
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <input type="submit" name="plusKeuze" value=" + " onclick="bevestig()">
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
    <input type="submit" name="bevestigKeuze" value="OK" onclick="bevestig()">
</center>
</form>
<br>

```

[illegible]

```

        <center><input type= "text" style="text-align:center;" value=%tijd_zone_uur% name="tz_uur" size = 1></center>
    </div>
    <div class="kader_3">
        <center><input type= "text" style="text-align:center;" value=%tijd_zone_minuut% name="tz_minuut" size = 1></center>
    </div>
    <div class="kader_4">
        <center><input type= "text" style="text-align:center;" value=%zomertijd% name="zomer" size = 1></center>
    </div>
</div>
<div class="blanco_40">&nbsp;</div>
<div class="kader">
    <div class="kader_3">
        <center><input type="submit" value="OK" name="tz_bevestig" onclick="bevestig()"></center>
    </div>
</div>
</form>
</small>
<div class="blanco_40">&nbsp;</div>
<h5><center> Klok helderheid (0 <-> 15)</center></h5>
<small>
<form action="/get" target="hidden-form">
<div class="kader">
    <div class="kader_3">
        <center><input type= "text" style="text-align:center;" value=%intensiteit% name="intensiteit_gewenst" size = 1></center>
    </div>
</div>
<div class="blanco_40">&nbsp;</div>
<div class="kader">
    <div class="kader_3">
        <center><input type="submit" value="OK" name="intensiteit_bevestig" onclick="bevestig()"></center>
    </div>
</div>
</form>
</small>
<br>
<br>
<h6>thieu-b55 januari 2023</h6>
<script>
    function bevestig(){
        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>Internetradio bediening</title>
    <meta name="viewport" content="width=device-width, initial-scale=.85">
    <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.blanco_20{
            width: auto;
            height: 20px;
        }
        div.blanco_40{

```

```
width: auto;
height: 40px;
}
</style>
</head>
<body>
<p><small><center>%song%</center></small></p>
<center>

</center>
</form>
<br>
<p><small><b><center>%tekst1%</center></b></small></p>
<p><small><center>%tekst2%</center></small></p>
<p><small><b><center>%tekst3%</center></b></small></p>
<p><small><center>%tekst4%</center></small></p>
<p><small><b><center>%tekst5%</center></b></small></p>
<p><small><center>%tekst6%</center></small></p>
<p><small><center><b>EQ -40 <-> 6 &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Volume 0 <->21</b></center></small></p>
<form action="/get" target="hidden-form">
<small>
<center>
<labelfor="dummy">L :</label>

</center>
</form>
<br><br>
<h5><center><strong>ESP32 Netwerk instellingen</strong></center></h5>
<form action="/get">
<small>
<div class="kader">
<div class="links"><b>ssid :</b></div>
<div class="links_midden"><input type="text" style="text-align:center;" name="ssid"></div>
</div>
<div class="blanco_40">&nbsp;&nbsp;&nbsp;&~</div>
<div class="kader">
<div class="links"><b>pswd :</b></div>
<div class="links_midden"><input type="text" style="text-align:center;" name="pswd"></div>
</div>
<div class="blanco_20">&nbsp;&nbsp;&~</div>
</small>
<h5><center>Gewenst IP address (default 192.168.1.178)</center></h5>
<div class="kader">
<center>

```

```

<div class="blanco_20">&nbsp;  </div>
</small>
<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){
    if(var == "alarm_1"){
        if(alarm_aktief_array[0] == 1){
            return("1");
        }
        else{
            return(leeg);
        }
    }
    if(var == "dag_1"){
        if(alarm_aktief_array[0] == 1){
            return(week_dagen_char[alarm_dag_array[0]]);
        }
        else{
            return(leeg);
        }
    }
    if(var == "tijd_1"){
        if(alarm_aktief_array[0] == 1){
            sprintf(alarm_tijd_1_char, "%02d:%02d", alarm_uur_array[0], alarm_minuut_array[0]);
            return(alarm_tijd_1_char);
        }
        else{
            return(leeg);
        }
    }
    if(var == "mode_1"){
        if(alarm_aktief_array[0] == 1){
            return(mode_char[alarm_mode_array[0]]);
        }
        else{
            return(leeg);
        }
    }
    if(var == "wake_up_1"){
        if(alarm_aktief_array[0] == 1){
            return(String(wake_up_array[0]));
        }
        else{
            return(leeg);
        }
    }
    if(var == "alarm_2"){
        if(alarm_aktief_array[1] == 1){
            return("2");
        }
        else{
            return(leeg);
        }
    }
    if(var == "dag_2"){
        if(alarm_aktief_array[1] == 1){
            return(week_dagen_char[alarm_dag_array[1]]);
        }
        else{
            return(leeg);
        }
    }
    if(var == "tijd_2"){
        if(alarm_aktief_array[1] == 1){
            sprintf(alarm_tijd_2_char, "%02d:%02d", alarm_uur_array[1], alarm_minuut_array[1]);
            return(alarm_tijd_2_char);
        }
    }
}

```

```

    }
    else{
        return(leeg);
    }
}
if(var == "mode_2"){
    if(alarm_aktief_array[1] == 1){
        return(mode_char[alarm_mode_array[1]]);
    }
    else{
        return(leeg);
    }
}
if(var == "wake_up_2"){
    if(alarm_aktief_array[1] == 1){
        return(String(wake_up_array[1]));
    }
    else{
        return(leeg);
    }
}
if(var == "alarm_3"){
    if(alarm_aktief_array[2] == 1){
        return("3");
    }
    else{
        return(leeg);
    }
}
if(var == "dag_3"){
    if(alarm_aktief_array[2] == 1){
        return(week_dagen_char[alarm_dag_array[2]]);
    }
    else{
        return(leeg);
    }
}
if(var == "tijd_3"){
    if(alarm_aktief_array[2] == 1){
        sprintf(alarm_tijd_3_char, "%02d:%02d", alarm_uur_array[2], alarm_minuut_array[2]);
        return(alarm_tijd_3_char);
    }
    else{
        return(leeg);
    }
}
if(var == "mode_3"){
    if(alarm_aktief_array[2] == 1){
        return(mode_char[alarm_mode_array[2]]);
    }
    else{
        return(leeg);
    }
}
if(var == "wake_up_3"){
    if(alarm_aktief_array[2] == 1){
        return(String(wake_up_array[2]));
    }
    else{
        return(leeg);
    }
}
if(var == "alarm_nr"){
    return(String(alarm_teller_int + 1));
}
if(var == "alarm_aktief"){
    return(alarm_aktief_string);
}
if(var == "alarm_dag"){
    return(alarm_dag_string);
}
if(var == "alarm_tijd"){
    return(alarm_tijd_char);
}
if(var == "alarm_mode"){
    return(alarm_mode_string);
}

```

```

}
if(var == "wake_up"){
    return(alarm_wake_up_string);
}
if(var == "radio_bediening"){
    return(radio_bediening_char[radio_bediening_int]);
}
if(var == "zenderNu"){
    if(gekozen == -2){
        return("mp3 lijst maken");
    }
    else if(gekozen == -1){
        return("mp3 speler");
    }
    else{
        return(zenderarray[gekozen]);
    }
}
if(var == "song"){
    return(streamsong);
}
if(var == "selectie"){
    if(gn_keuze == 0){
        return("Stop mp3 speler");
    }
    if(gn_keuze == 1){
        return("mp3 speler");
    }
    if(gn_keuze == 2){
        return("Maak mp3 lijst");
    }
}
if(var == "selecteren"){
    if(keuze == - 2){
        return("mp3 lijst maken");
    }
    else if((keuze == -1) || (gn_keuze == 1)){
        return("mp3 speler");
    }
    else{
        return(zenderarray[keuze]);
    }
}
if(var == "tekst1"){
    if(!(!lijst_maken) && (!songlijsten)){
        return(leeg);
    }
    if(lijst_maken){
        return("inlezen van : ");
    }
    if(songlijsten){
        return("EERST ALLE SONGLIJSTXX VERWIJDEREN");
    }
}
if(var == "tekst2"){
    if(!lijst_maken){
        return(leeg);
    }
    else{
        return(mp3_folder);
    }
}
if(var == "tekst3"){
    if(!lijst_maken){
        return(leeg);
    }
    else{
        return("aantal mp3's ingelezen : ");
    }
}
if(var == "tekst4"){
    if(!lijst_maken){
        return(leeg);
    }
    else{
        return(String(mp3_aantal));
    }
}

```

```

    }
}
if(var == "tekst5"){
    if(!lijst_maken){
        return(leeg);
    }
    else{
        return("seconden reeds bezig : ");
    }
}

if(var == "tekst6"){
    if(!lijst_maken){
        return(leeg);
    }
    else{
        int seconden = (millis() - inlezen_begin) / 1000;
        return(String(seconden));
    }
}
if(var == "laag_kiezen"){
    return(String(laag_gekozen));
}
if(var == "midden_kiezen"){
    return(String(midden_gekozen));
}
if(var == "hoog_kiezen"){
    return(String(hoog_gekozen));
}
if(var == "volume_kiezen"){
    return(String(volume_gekozen));
}
if(var == "array_index"){
    return(String(array_index));
}
if(var == "zender"){
    return(zenderarray[array_index]);
}
if(var == "url"){
    return(urlarray[array_index]);
}
if(var == "folder"){
    String folder = mp3_folder;
    folder.toCharArray(mp3_lijt_folder, (folder.length() + 1));
    return(mp3_lijt_folder);
}
if(var == "mp3"){
    String aantal = String(mp3_aantal);
    aantal.toCharArray(mp3_lijt_aantal, (aantal.length() + 1));
    return(mp3_lijt_folder);
}
if(var == "ip_address_1"){
    return(String(ip_1_int));
}
if(var == "ip_address_2"){
    return(String(ip_2_int));
}
if(var == "ip_address_3"){
    return(String(ip_3_int));
}
if(var == "ip_address_4"){
    return(String(ip_4_int));
}
if(var == "dag"){
    return(week_dagen_char[dag_int]);
}
if(var == "tijd"){
    sprintf(tijd_char, "%02d:%02d", uren_int, minuten_int);
    return(tijd_char);
}
if(var == "tijd_zone_uur"){
    return(String(utc_offset_uren_int));
}
if(var == "tijd_zone_minuut"){
    return(String(utc_offset_minuten_int));
}
}

```



```

if(var == "zomertijd"){
    return(String(winter_zomer_int));
}
if(var == "intensiteit"){
    return(String(intensiteit_int));
}
return String();
}

void html_input(){
    server.begin();
    Serial.println(WiFi.localIP());
    Serial.println(WiFi.softAPIP());
    if(network){
        server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
            request->send_P(200, "text/html", index_html, processor);
        });
        server.on("/get", HTTP_GET, [](AsyncWebServerRequest *request){
            String http_zender = "                ";
            String http_url = "                ";
            char terminator = char(0x0a);
            int temp;
            if(request->hasParam(ALARM_PLUS)){
                alarm_teller_int ++;
                if(alarm_teller_int > 2){
                    alarm_teller_int = 2;
                }
                alarm_aktief_string = String(alarm_aktief_array[alarm_teller_int]);
                alarm_dag_string = String(alarm_dag_array[alarm_teller_int]);
                sprintf(alarm_tijd_char, "%02d:%02d", alarm_uur_array[alarm_teller_int], alarm_minuut_array[alarm_teller_int]);
                alarm_mode_string = (alarm_mode_array[alarm_teller_int]);
                alarm_wake_up_string = String(wake_up_array[alarm_teller_int]);
            }
            if(request->hasParam(ALARM_MIN)){
                alarm_teller_int --;
                if(alarm_teller_int < 0){
                    alarm_teller_int = 0;
                }
                alarm_aktief_string = String(alarm_aktief_array[alarm_teller_int]);
                alarm_dag_string = String(alarm_dag_array[alarm_teller_int]);
                sprintf(alarm_tijd_char, "%02d:%02d", alarm_uur_array[alarm_teller_int], alarm_minuut_array[alarm_teller_int]);
                alarm_mode_string = (alarm_mode_array[alarm_teller_int]);
                alarm_wake_up_string = String(wake_up_array[alarm_teller_int]);
            }
            if(request->hasParam(AKTIEF)){
                alarm_aktief_int = ((request->getParam(AKTIEF)->value()) + String(terminator)).toInt();
            }
            if(request->hasParam(DAG)){
                alarm_dag_int = ((request->getParam(DAG)->value()) + String(terminator)).toInt();
            }
            if(request->hasParam(TIJD)){
                alarm_tijd_string = (request->getParam(TIJD)->value()) + String(terminator);
            }
            if(request->hasParam(MODE)){
                alarm_mode_int = ((request->getParam(MODE)->value()) + String(terminator)).toInt();
            }
            if(request->hasParam(WAKEUP)){
                alarm_wake_up_int = ((request->getParam(WAKEUP)->value()) + String(terminator)).toInt();
            }
            if(request->hasParam(ALARM_BEVESTIG)){
                if((alarm_aktief_int == 0) || (alarm_aktief_int == 1)){
                    alarm_aktief_array[alarm_teller_int] = alarm_aktief_int;
                }
                if((alarm_dag_int > 0) && (alarm_dag_int < 11)){
                    alarm_dag_array[alarm_teller_int] = alarm_dag_int;
                }
                temp = alarm_tijd_string.indexOf(":");
                if(temp != -1){
                    alarm_uur_int = alarm_tijd_string.substring(0, temp).toInt();
                    alarm_minuut_int = alarm_tijd_string.substring(temp + 1).toInt();
                    if((alarm_uur_int > -1) && (alarm_uur_int < 24) && (alarm_minuut_int > -1) && (alarm_minuut_int < 60)){
                        alarm_uur_array[alarm_teller_int] = alarm_uur_int;
                        alarm_minuut_array[alarm_teller_int] = alarm_minuut_int;
                    }
                }
            }
            if((alarm_mode_int == 0) || alarm_mode_int == 1){

```

```

    alarm_mode_array[alarm_teller_int] = alarm_mode_int;
}
if((alarm_wake_up_int == 0) || alarm_wake_up_int == 1){
    wake_up_array[alarm_teller_int] = alarm_wake_up_int;
}
schrijf_spiffs();
lees_spiffs();
alarm_aktief_string = String(alarm_aktief_array[alarm_teller_int]);
alarm_dag_string = String(alarm_dag_array[alarm_teller_int]);
sprintf(alarm_tijd_char, "%02d:%02d", alarm_uur_array[alarm_teller_int], alarm_minuut_array[alarm_teller_int]);
alarm_mode_string = (alarm_mode_array[alarm_teller_int]);
alarm_wake_up_string = String(wake_up_array[alarm_teller_int]);
}
if(request->hasParam(RADIO_BEDIENING_MIN)){
    radio_bediening_int--;
    if(radio_bediening_int < 0){
        radio_bediening_int = 0;
    }
    pref.putShort("radio", radio_bediening_int);
    radio_bediening_int = pref.getShort("radio");
    audio.stopSong();
    digitalWrite(VERSTERKER, LOW);
}
if(request->hasParam(RADIO_BEDIENING_PLUS)){
    radio_bediening_int++;
    if(radio_bediening_int > 1){
        radio_bediening_int = 1;
    }
    pref.putShort("radio", radio_bediening_int);
    radio_bediening_int = pref.getShort("radio");
    gekozen = pref.getShort("station");
    keuze = gekozen;
    radio_gekozen();
    digitalWrite(VERSTERKER, HIGH);
    if(gekozen == -1){
        mp3_gekozen();
    }
}
if(request->hasParam(KEUZEMIN_INPUT)){
    wachttijd = millis();
    kiezen = true;
    keuze--;
    while((urlarray[keuze][0] != *H_CHAR) && (keuze > 0)){
        keuze--;
    }
    if(keuze < -2){
        keuze = MAX_AANTAL_KANALEN - 1;
        while((urlarray[keuze][0] != *H_CHAR) && (keuze > 0)){
            keuze--;
        }
    }
}
if(request->hasParam(KEUZEPLUS_INPUT)){
    wachttijd = millis();
    kiezen = true;
    keuze++;
    if(keuze > MAX_AANTAL_KANALEN + 1){
        keuze = 0;
    }
    if((keuze > 0) && (keuze < MAX_AANTAL_KANALEN)){
        while((urlarray[keuze][0] != *H_CHAR) && (keuze < MAX_AANTAL_KANALEN)){
            keuze++;
        }
    }
    if(keuze == MAX_AANTAL_KANALEN){
        keuze = -2;
    }
    if(keuze == MAX_AANTAL_KANALEN + 1){
        keuze = -1;
    }
}
if((request->hasParam(BEVESTIGKEUZE_INPUT)) && (kiezen == true)){
    kiezen = false;
    if(keuze == -2){
        songlijst_bestaat_bool = false;
        testDir(SD, "songlijst0");
    }
}

```

```

if(songlijst_bestaat_bool == false){
    lijst_maken = true;
}
else{
    keuze = pref.getShort("station");
    webradio = true;
}
}
else if(keuze == -1){
    gekozen = keuze;
    speel_mp3 = true;
}
else{
    gekozen = keuze;
    pref.putShort("station", gekozen);
    webradio = true;
}
}
if(request->hasParam(LAAG)){
    laag_keuze = ((request->getParam(LAAG)->value()) + String(terminator)).toInt();
}
if(request->hasParam(MIDDEN)){
    midden_keuze = ((request->getParam(MIDDEN)->value()) + String(terminator)).toInt();
}
if(request->hasParam(HOOG)){
    hoog_keuze = ((request->getParam(HOOG)->value()) + String(terminator)).toInt();
}
if(request->hasParam(VOLUME)){
    volume_keuze = ((request->getParam(VOLUME)->value()) + String(terminator)).toInt();
}
if(request->hasParam(VOLUME_BEVESTIG)){
    if((laag_keuze > -41) && (laag_keuze < 7)){
        laag_gekozen = laag_keuze;
    }
    if((midden_keuze > -41) && (midden_keuze < 7)){
        midden_gekozen = midden_keuze;
    }
    if((hoog_keuze > -41) && (hoog_keuze < 7)){
        hoog_gekozen = hoog_keuze;
    }
    if((volume_keuze > -1) && (volume_keuze < 22)){
        volume_gekozen = volume_keuze;
    }
    audio.setVolume(volume_gekozen);
    audio.setTone(laag_gekozen, midden_gekozen, hoog_gekozen);
    pref.putShort("laag", laag_gekozen);
    pref.putShort("midden", midden_gekozen);
    pref.putShort("hoog", hoog_gekozen);
    pref.putShort("volume", volume_gekozen);
}
if(request->hasParam(ARRAY_MIN)){
    array_index -= 1;
    if(array_index < 0){
        array_index = MAX_AANTAL_KANALEN - 1;
    }
}
if(request->hasParam(ARRAY_PLUS)){
    array_index += 1;
    if(array_index > MAX_AANTAL_KANALEN - 1){
        array_index = 0;
    }
}
if(request->hasParam(ZENDER)){
    http_zender = (request->getParam(ZENDER)->value());
}
if(request->hasParam(URL)){
    http_url = (request->getParam(URL)->value());
}
if(request->hasParam(BEVESTIG_ZENDER)){
    memset(zenderarray[array_index], 0, sizeof(zenderarray[array_index]));
    http_zender.toCharArray(zenderarray[array_index], http_zender.length() + 1);
    memset(urlarray[array_index], 0, sizeof(urlarray[array_index]));
    http_url.toCharArray(urlarray[array_index], http_url.length() + 1);
    schrijf_csv = true;
}
if(request->hasParam(TZ_UUR)){

```

```

    utc_offset_uren_int = ((request->getParam(TZ_UUR)->value()) + String(terminator)).toInt();
}
if(request->hasParam(TZ_MINUUT)){
    utc_offset_minuten_int = ((request->getParam(TZ_MINUUT)->value()) + String(terminator)).toInt();
    if((utc_offset_minuten_int != 0) && (utc_offset_minuten_int != 30)){
        utc_offset_minuten_int = 0;
    }
}
if(request->hasParam(ZOMER)){
    winter_zomer_int = ((request->getParam(ZOMER)->value()) + String(terminator)).toInt();
    if((winter_zomer_int != 0) && (winter_zomer_int != 1)){
        winter_zomer_int = 0;
    }
}
if(request->hasParam(TZ_BEVESTIG)){
    pref.putShort("uur_of", utc_offset_uren_int);
    pref.putShort("min_of", utc_offset_minuten_int);
    pref.putShort("w_z", winter_zomer_int);
    tijd_update();
    tijd_naar_led();
}
if(request->hasParam(INTENSITEIT_GEWENST)){
    intensiteit_int = ((request->getParam(INTENSITEIT_GEWENST)->value()) + String(terminator)).toInt();
    if((intensiteit_int < 0) || (intensiteit_int > 15)){
        intensiteit_int = pref.getShort("intens");
    }
}
if(request->hasParam(INTENSITEIT_BEVESTIG)){
    pref.putShort("intens", intensiteit_int);
    intensiteit_int = pref.getShort("intens");
    display_setup(0x0A, intensiteit_int);
}
});
}
if(!network){
    Serial.println("geen network");
    server.on("/", HTTP_GET, [] (AsyncWebServerRequest *request){
        request->send_P(200, "text/html", netwerk_html, processor);
    });
    server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request){
        String netwerk = "          ";
        String paswoord = "          ";
        char terminator = char(0x0a);

        if(request->hasParam(MIN_INPUT)){
            wachttijd = millis();
            kiezen = true;
            gn_keuze --;
            if(gn_keuze < 0){
                gn_keuze = 2;
            }
        }
        if(request->hasParam(PLUS_INPUT)){
            wachttijd = millis();
            kiezen = true;
            gn_keuze ++;
            if(gn_keuze > 2){
                gn_keuze = 0;
            }
        }
        if(request->hasParam(BEVESTIG_MP3)){
            kiezen = false;
            if(gn_keuze == 0){
                audio.stopSong();
            }
            if(gn_keuze == 1){
                speel_mp3 = true;
            }
            if(gn_keuze == 2){
                songlijst_bestaat_bool = false;
                testDir(SD, "/songlijst0");
                if(songlijst_bestaat_bool == false){
                    lijst_maken = true;
                }
            }
        }
    });
}

```

```

if(request->hasParam(LAAG)){
    laag_keuze = ((request->getParam(LAAG)->value()) + String(terminator)).toInt();
}
if(request->hasParam(MIDDEN)){
    midden_keuze = ((request->getParam(MIDDEN)->value()) + String(terminator)).toInt();
}
if(request->hasParam(HOOG)){
    hoog_keuze = ((request->getParam(HOOG)->value()) + String(terminator)).toInt();
}
if(request->hasParam(VOLUME)){
    volume_keuze = ((request->getParam(VOLUME)->value()) + String(terminator)).toInt();
}
if(request->hasParam(VOLUME_BEVESTIG)){
    if((laag_keuze > -41) && (laag_keuze < 7)){
        laag_gekozen = laag_keuze;
    }
    if((midden_keuze > -41) && (midden_keuze < 7)){
        midden_gekozen = midden_keuze;
    }
    if((laag_keuze > -41) && (laag_keuze < 7)){
        hoog_gekozen = hoog_keuze;
    }
    if((volume_keuze > -1) && (laag_keuze < 22)){
        volume_gekozen = volume_keuze;
    }
    audio.setVolume(volume_gekozen);
    audio.setTone(laag_gekozen, midden_gekozen, hoog_gekozen);
    pref.putShort("laag", laag_gekozen);
    pref.putShort("midden", midden_gekozen);
    pref.putShort("hoog", hoog_gekozen);
    pref.putShort("volume", volume_gekozen);
}
if(request->hasParam(STA_SSID)){
    netwerk = (request->getParam(STA_SSID)->value());
    netwerk = netwerk + String(terminator);
    netwerk.toCharArray(ssid, (netwerk.length() + 1));
    writeFile(SD, "/ssid", ssid);
    ssid_ingevoerd = true;
}
if(request->hasParam(STA_PSWD)){
    paswoord = (request->getParam(STA_PSWD)->value());
    paswoord = paswoord + String(terminator);
    paswoord.toCharArray(password, (paswoord.length() + 1));
    writeFile(SD, "/pswd", password);
    pswd_ingevoerd = true;
}
if(request->hasParam(IP_1_KEUZE)){
    ip_1_string = (request->getParam(IP_1_KEUZE)->value()) + String(terminator);
}
if(request->hasParam(IP_2_KEUZE)){
    ip_2_string = (request->getParam(IP_2_KEUZE)->value()) + String(terminator);
}
if(request->hasParam(IP_3_KEUZE)){
    ip_3_string = (request->getParam(IP_3_KEUZE)->value()) + String(terminator);
}
if(request->hasParam(IP_4_KEUZE)){
    ip_4_string = (request->getParam(IP_4_KEUZE)->value()) + String(terminator);
}
if((ssid_ingevoerd) && (pswd_ingevoerd)){
    ssid_ingevoerd = false;
    pswd_ingevoerd = false;
    ip_string = ip_1_string + "." + ip_2_string + "." + ip_3_string + "." + ip_4_string;
    ip_string.toCharArray(ip_char, (ip_string.length() + 1));
    writeFile(SD, "/ip", ip_char);
    Serial.println("Restart over 5 seconden");
    delay(5000);
    ESP.restart();
}
});
}
}

```

```

void setup(){
    Serial.begin(115200);
    pinMode(mySPI_CS, OUTPUT);
    digitalWrite(mySPI_CS, HIGH);
}

```

```

pinMode(VERSTERKER, OUTPUT);
digitalWrite(VERSTERKER, LOW);
pinMode(MINUUT_PULS, INPUT_PULLDOWN);
pinMode(ALARM_UT, INPUT_PULLDOWN);
SPIFFS.begin();
SPI2.begin(mySPI_SCK, mySPI_MISO, mySPI_MOSI);
SPI.begin();
if(!SD.begin()){
  Serial.println("check SD kaart");
}
display_setup(0x0F, 0x00);
delay(10);
display_setup(0x0C, 0x00);
display_setup(0x0C, 0x01);
display_setup(0x09, 0x00);
display_setup(0x0A, 0x00);
display_setup(0x0B, 0x07);
smiley();
Wire.begin(I2C_SDA, I2C_SCL);
Wire.beginTransmission(DS3231SN);
Wire.write(0x0B);
Wire.write(0x80);
Wire.write(0x80);
Wire.write(0x80);
Wire.write(0x46);
Wire.write(0x00);
Wire.endTransmission();
pref.begin("WebRadio", false);
//pref.clear();
if(pref.getString("controle") != "dummy geladen"){
  Serial.println("SPIFFS wordt geformatteerd");
  SPIFFS.format();
  Serial.println("SPIFFS is geformatteerd");
  schrijf_spiffs();
  pref.putShort("station", 0);
  pref.putShort("volume", 10);
  pref.putShort("laag", 0);
  pref.putShort("midden", 0);
  pref.putShort("hoog", 0);
  pref.putShort("uur_of", utc_offset_uren_int);
  pref.putShort("min_of", utc_offset_minuten_int);
  pref.putShort("w_z", winter_zomer_int);
  pref.putShort("radio", radio_bediening_int);
  pref.putShort("intens", intensiteit_int);
  pref.putString("controle", "dummy geladen");
}
lees_spiffs();
gekozen = pref.getShort("station");
volume_gekozen = pref.getShort("volume");
volume_keuze = volume_gekozen;
laag_gekozen = pref.getShort("laag");
laag_keuze = laag_gekozen;
midden_gekozen = pref.getShort("midden");
midden_keuze = midden_gekozen;
hoog_gekozen = pref.getShort("hoog");
hoog_keuze = hoog_gekozen;
utc_offset_uren_int = pref.getShort("uur_of");
utc_offset_minuten_int = pref.getShort("min_of");
winter_zomer_int = pref.getShort("w_z");
radio_bediening_int = pref.getShort("radio");
intensiteit_int = pref.getShort("intens");
display_setup(0x0A, intensiteit_int);
alarm_aktief_string = String(alarm_aktief_array[alarm_teller_int]);
alarm_dag_string = String(alarm_dag_array[alarm_teller_int]);
sprintf(alarm_tijd_char, "%02d:%02d", alarm_uur_array[alarm_teller_int], alarm_minuut_array[alarm_teller_int]);
alarm_mode_string = (alarm_mode_array[alarm_teller_int]);
alarm_wake_up_string = String(wake_up_array[alarm_teller_int]);
if(radio_bediening_int == 1){
  audio.pauseResume();
  digitalWrite(VERSTERKER, HIGH);
}
else{
  audio.stopSong();
  digitalWrite(VERSTERKER, LOW);
}
audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);

```

```

audio.setVolume(volume_gekozen);
audio.setTone(laag_gekozen, midden_gekozen, hoog_gekozen);
lees_CSV();
readFile(SD, "/ssid");
inputString.toCharArray(ssid, teller);
readFile(SD, "/pswd");
inputString.toCharArray(password, teller);
readIP(SD, "/ip");
WiFi.disconnect();
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
network = true;
wacht_op_network = millis();
while(WiFi.status() != WL_CONNECTED){
  delay(500);
  if(millis() - wacht_op_network > 15000){
    network = false;
    break;
  }
}
if(network == true){
  IPAddress subnet(WiFi.subnetMask());
  IPAddress gateway(WiFi.gatewayIP());
  IPAddress dns(WiFi.dnsIP(0));
  IPAddress static_ip(ip_1_int, ip_2_int, ip_3_int, ip_4_int);
  WiFi.disconnect();
  if (WiFi.config(static_ip, gateway, subnet, dns, dns) == false) {
    Serial.println("Configuration failed.");
  }
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  wacht_op_network = millis();
  while(WiFi.status() != WL_CONNECTED){
    delay(500);
    if(millis() - wacht_op_network > 15000){
      network = false;
      break;
    }
  }
  delay(1000);
  tijd_init_bool = false;
  initTime();
  if(tijd_init_bool == true){
    tijd_update();
  }
  attachInterrupt(digitalPinToInterrupt(MINUUT_PULS), minuut_voorbij, FALLING);
  tijd_naar_led();
  keuze = gekozen;
  radio_gekozen();
  html_input();
}
else{
  WiFi.mode(WIFI_AP);
  WiFi.softAP(APssid, APPswd);
  Wire.beginTransmission(DS3231SN);
  attachInterrupt(digitalPinToInterrupt(MINUUT_PULS), minuut_voorbij, FALLING);
  tijd_naar_led();
  html_input();
}
}

void loop(){
  if(minuut_interrupt_bool == true){
    Wire.beginTransmission(DS3231SN);
    Wire.write(0x0F);
    Wire.write(0x00);
    Wire.endTransmission();
    minuut_interrupt_bool = false;
    twaalf_uur_int ++;
    if(twaalf_uur_int == 720){
      twaalf_uur_int = 0;
      if(tijd_init_bool == false){
        initTime();
      }
    }
    if(tijd_init_bool == true){
      tijd_update();
    }
  }
}

```

```

    }
}
tijd_naar_led();
sprintf(dag_uren_char, "%s %02d:%02d", week_dagen_char[dag_int], uren_int, minuten_int);
Serial.println(dag_uren_char);
}

/*
 * Alarm 1
 */
if(radio_bediening_int == 0){
    if((alarm_aktief_array[0] == 1) && (alarm_1_set_bool == false)){
        if((alarm_dag_array[0] == 8) || ((alarm_dag_array[0] == 9) && (dag_int < 6))
            || ((alarm_dag_array[0] == 10) && ((dag_int == 6) || dag_int == 7))
            || (alarm_dag_array[0] == dag_int)){
            if(alarm_uur_array[0] == uren_int){
                if(alarm_minuut_array[0] == minuten_int){
                    alarm_1_set_bool = true;
                    wake_up_end_bool = true;
                    alarm_1_afgewerkt_bool = false;
                    alarm_aanwezig_long = millis();
                    digitalWrite(VERSTERKER, HIGH);
                    if(wake_up_array[0] == 1){
                        wake_up_bool = true;
                        wake_up_end_bool = false;
                        audio.connecttoFS(SD, "/wake_up.mp3");
                    }
                }
            }
        }
    }
    if((alarm_1_set_bool == true) && (wake_up_end_bool == true) && (alarm_1_afgewerkt_bool == false)){
        alarm_1_afgewerkt_bool = true;
        wake_up_bool = false;
        if(alarm_mode_array[0] == 0){
            mp3_gekozen();
        }
        else{
            gekozen = pref.getShort("station");
            keuze = gekozen;
            radio_gekozen();
        }
    }
}

/*
 * Alarm 2
 */
if(radio_bediening_int == 0){
    if((alarm_aktief_array[1] == 1) && (alarm_2_set_bool == false)){
        if((alarm_dag_array[1] == 8) || ((alarm_dag_array[1] == 9) && (dag_int < 6))
            || ((alarm_dag_array[1] == 10) && ((dag_int == 6) || dag_int == 7))
            || (alarm_dag_array[1] == dag_int)){
            if(alarm_uur_array[1] == uren_int){
                if(alarm_minuut_array[1] == minuten_int){
                    alarm_2_set_bool = true;
                    wake_up_end_bool = true;
                    alarm_2_afgewerkt_bool = false;
                    alarm_aanwezig_long = millis();
                    digitalWrite(VERSTERKER, HIGH);
                    if(wake_up_array[1] == 1){
                        wake_up_bool = true;
                        wake_up_end_bool = false;
                        audio.connecttoFS(SD, "/wake_up.mp3");
                    }
                }
            }
        }
    }
    if((alarm_2_set_bool == true) && (wake_up_end_bool == true) && (alarm_2_afgewerkt_bool == false)){
        alarm_2_afgewerkt_bool = true;
        wake_up_bool = false;
        if(alarm_mode_array[1] == 0){
            mp3_gekozen();
        }
        else{

```



```

    gekozen = pref.getShort("station");
    keuze = gekozen;
    radio_gekozen();
}
}
}
/*
* Alarm 3
*/
if(radio_bediening_int == 0){
    if((alarm_aktief_array[2] == 1) && (alarm_3_set_bool == false)){
        if((alarm_dag_array[2] == 8) || ((alarm_dag_array[2] == 9) && (dag_int < 6))
            || ((alarm_dag_array[2] == 10) && ((dag_int == 6) || dag_int == 7))
            || (alarm_dag_array[2] == dag_int)){
            if(alarm_uur_array[2] == uren_int){
                if(alarm_minuut_array[2] == minuten_int){
                    alarm_3_set_bool = true;
                    wake_up_end_bool = true;
                    alarm_3_afgewerkt_bool = false;
                    alarm_aanwezig_long = millis();
                    digitalWrite(VERSTERKER, HIGH);
                    if(wake_up_array[2] == 1){
                        wake_up_bool = true;
                        wake_up_end_bool = false;
                        audio.connecttoFS(SD, "/wake_up.mp3");
                    }
                }
            }
        }
    }
    if((alarm_3_set_bool == true) && (wake_up_end_bool == true) && (alarm_3_afgewerkt_bool == false)){
        alarm_3_afgewerkt_bool = true;
        wake_up_bool = false;
        if(alarm_mode_array[2] == 0){
            mp3_gekozen();
        }
        else{
            gekozen = pref.getShort("station");
            keuze = gekozen;
            radio_gekozen();
        }
    }
}
if((((millis() - alarm_aanwezig_long) > 3600000) || digitalRead(ALARM_UIT)) && (alarm_1_set_bool == true)){
    alarm_1_set_bool = false;
    alarm_1_afgewerkt_bool = false;
    audio.stopSong();
    digitalWrite(VERSTERKER, LOW);
}
if((((millis() - alarm_aanwezig_long) > 3600000) || digitalRead(ALARM_UIT)) && (alarm_2_set_bool == true)){
    alarm_2_set_bool = false;
    alarm_2_afgewerkt_bool = false;
    audio.stopSong();
    digitalWrite(VERSTERKER, LOW);
}
if((((millis() - alarm_aanwezig_long) > 3600000) || digitalRead(ALARM_UIT)) && (alarm_3_set_bool == true)){
    alarm_3_set_bool = false;
    alarm_3_afgewerkt_bool = false;
    audio.stopSong();
    digitalWrite(VERSTERKER, LOW);
}
if(schrijf_csv == true){
    schrijf_csv = false;
    schrijf_naar_csv();
}
if(lijst_maken == true){
    mp3_lijst_maken_gekozen();
}
if(speel_mp3 == true){
    speel_mp3 = false;
    mp3_gekozen();
}
if(webradio == true){
    webradio = false;
    gekozen = keuze;
    pref.putShort("station", gekozen);
}

```

```
    radio_gekozen();  
  }  
  if(((millis() - wachttijd) > 5000) && (kiezen == true)){  
    kiezen = false;  
    keuze = gekozen;  
  }  
  audio.loop();  
}
```