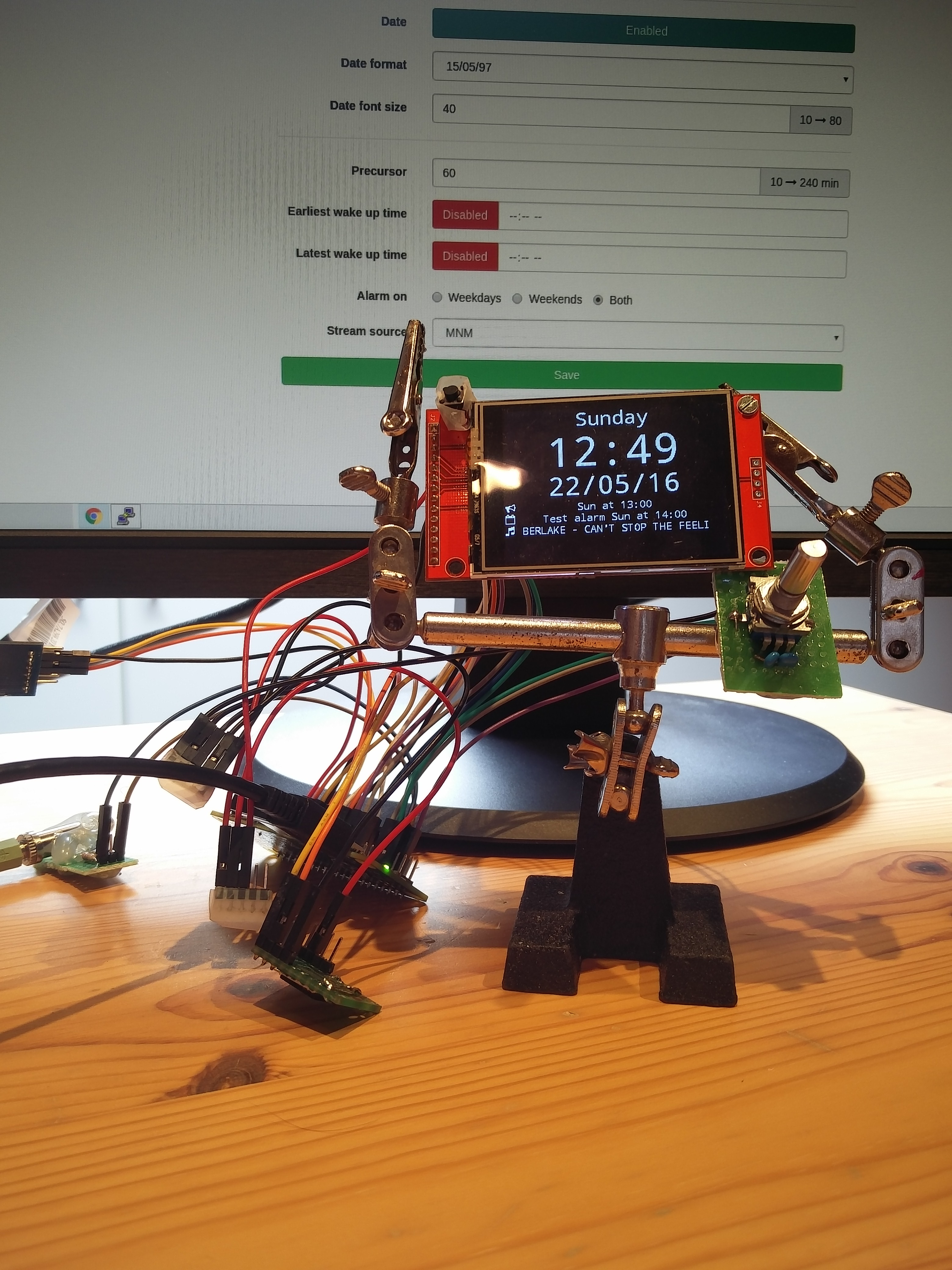
Dries Kennes & Michiel Bellekens | Project II | 2015-2016

SmartAlarmClock

Project II – ICT-Elektronica



Voorwoord

Wij zijn Dries Kennes en Michiel Bellekens, studenten ICT-Elektronica, Embedded ICT Fase 2 aan Thomas More Mechelen op Campus De Nayer.

Dit project heeft als doel het maken van een SmartAlarmClock . Zoals de naam reeds doet vermoeden is het een klok/wekker met enkele slimme extra functies die nuttig kunnen zijn voor de gebruiker. Zo is de belangrijkste functionaliteit dat de gebruiker kan worden gewekt aan de hand van zijn/haar Google-Calender .

De gebruiker kan via een web interface de klok configureren naar wens. Ten eerste kan de gebruiker instellen hoe lang voor de eerste afspraak de wekker moet afgaan. Ten tweede kan hij kiezen uit een lijst van muziek streams om gewekt te worden bv. MNM of Studio Brussel. Ten derde kan de gebruiker ook een minimum en maximum wek-tijd instellen. Dit betekent dat, onafhankelijk van de afspraken, de wekker nooit vroeger dan het minimum en later dan het maximum mag afgaan. Tot slot zijn er nog enkele instellingen voor de lay-out op het scherm zoals het formaat en de grootte van het lettertype van de tijd en de datum.

Ons contacteren, kan via email, [dries.kennes@student.thomasmore.be](mailto:dries.kennes@student.thomasmore.be), michiel.bellekens@student.thomasmore.be.

Inhoudsopgave

[Voorwoord 2](#_Toc451701953)

[Inhoudsopgave 3](#_Toc451701954)

[1 Hasrdware 4](#_Toc451701955)

[1.1 Raspberry Pi Zero Essentials kit 4](#_Toc451701956)

[1.2 Rotary encoder 4](#_Toc451701957)

[1.3 I2C RTC (3.3 V DS3231) 5](#_Toc451701958)

[1.4 Spanningsregelaar (MCP1703) 5](#_Toc451701959)

[1.5 2.4” 240x320 spi tft LCD (3.3 V) 6](#_Toc451701960)

[1.6 Buzzer 6](#_Toc451701961)

[1.7 Levelshifter (AN10441) 7](#_Toc451701962)

[1.8 Audio stereo DAC (PCM 5102A 3.3 V) 8](#_Toc451701963)

[1.8.1 Intermezzo I2S 8](#_Toc451701964)

[1.9 Audio amplifier (TPA2016D2 5V) 8](#_Toc451701965)

[1.10 NOR gate flipflop 9](#_Toc451701966)

[1.11 Supercap 9](#_Toc451701967)

[1.12 WS2812 Leds 10](#_Toc451701968)

[1.13 Het schema & PCB 10](#_Toc451701969)

[2 Software 11](#_Toc451701970)

[2.1 De nodige drivers en initialisatiecode 11](#_Toc451701971)

[2.1.1 Kernel-parameters en –modules 11](#_Toc451701972)

[2.1.2 “.bash\_profile” 11](#_Toc451701973)

[2.1.3 Nginx 12](#_Toc451701974)

[2.2 Het hoofdprogramma (app.py) 13](#_Toc451701975)

[2.2.1 LCD aansturen 13](#_Toc451701976)

[2.2.2 RTC aansturen 13](#_Toc451701977)

[2.2.3 Google Calendar 15](#_Toc451701978)

[2.2.3.1 Registreren van SmartAlarmClock 15](#_Toc451701979)

[2.2.3.2 Toestemming krijgen van de gebruiken 15](#_Toc451701980)

[2.2.3.3 Kalender afspraken opvragen 16](#_Toc451701981)

[2.2.3.4 Een vervallen “Access token” vervangen 16](#_Toc451701982)

[2.2.4 Flask (Web interface API) 16](#_Toc451701983)

[2.2.5 Netwerk en acces point 16](#_Toc451701984)

[2.2.6 Rotary encoder 17](#_Toc451701985)

[2.2.7 Alarm/muziek 17](#_Toc451701986)

[2.2.8 Web interface 18](#_Toc451701987)

[2.2.8.1 Status tab 18](#_Toc451701988)

[2.2.8.2 Wifi settings tab 18](#_Toc451701989)

[2.2.8.3 Clock settings tab 18](#_Toc451701990)

[2.2.8.4 Google Calendar tab 18](#_Toc451701991)

[3 Budget/kostenraming 19](#_Toc451701992)

[4 Besluit 20](#_Toc451701993)

[5 Bijlagen 21](#_Toc451701994)

[5.1 Schema’s & Lay-out printplaat 21](#_Toc451701995)

[5.1.1 De schema’s 21](#_Toc451701996)

[5.1.2 De printplaat 23](#_Toc451701997)

# Hasrdware

## Raspberry Pi Zero Essentials kit

Het platform dat wordt gebruikt in dit project is de Raspberry Pi Zero. Dit is gekozen om een aantal verschillende redenen. Ten eerste biedt de Zero een zeer goede ondersteuning voor verschillende communicatie interfaces zoals SPI, I2S en I2C maar ook voor het opzetten van draadloze netwerken of webservers. Ten tweede is de kostprijs ook zeer aantrekkelijk aangezien de Raspberry Pi Zero Essentials kit slechts € 8 kost en alle basis benodigdheden bevat.



Figuur 1‑1: De inhoud van de Raspberry Pi Zero Essentials kit.

## Rotary encoder

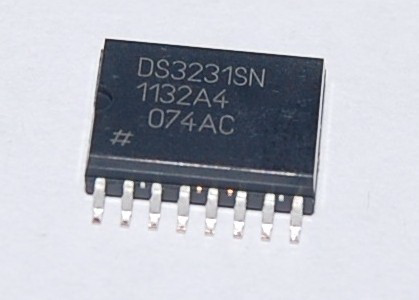
Als input voor bv. het volume van het geluid of de helderheid van het scherm, wordt gebruik gemaakt van een rotary encoder. De gebruikte rotary encoder heeft naast een rotatie functie ook een drukknop functie.



Figuur 1‑2: Een rotary encoder

## I2C RTC (3.3 V DS3231)

Een andere component van de SmartAlarmClock is de DS3231 Real Time Clock. Deze chip werd gekozen voor zijn hoge nauwkeurigheid en zijn instelbaarheid via de I2C interface die ook wordt ondersteund in de Linux kernel. Zijn hoge nauwkeurigheid dankt de chip aan de ingebouwde temperatuur gecompenseerde kristal oscillator. Een extra reden voor de keuze van deze chip is de mogelijkheid om een batterij toe te voegen als back-up voeding. Het overschakelen naar de batterijspanning gebeurt automatisch wanneer de chip detecteert dat zijn voedingspanning wegvalt. Dit is handig om te kunnen garanderen dat een backup alarm toch nog minstens 1 keer kan afgaan nadat de hoofdvoeding is uitgevallen. Naast de tijd in uren, minuten en seconden (24 uurs of 12 uurs met AM/PM) wordt ook de datum bijgehouden. Deze datum wordt automatisch aangepast, rekening houdend met het aantal dagen in elke maand en ook is er een correctie voor schrikkeljaren. De klok geeft toegang tot 2 alarmen die bij een alarmconditie de INT pin gaan aansturen (actief laag).



Figuur 1‑3: de DS3231 chip

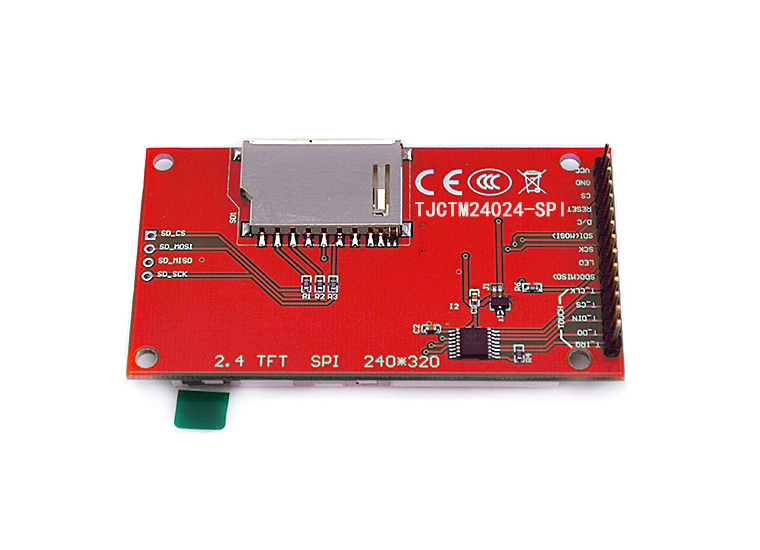
## Spanningsregelaar (MCP1703)

Aangezien onze voeding 5V bedraagt en een deel van onze componenten zoals de lcd op 3.3 V werken moeten we de spanning naar 3.3 V kunnen regelen. De MCP1703 is een CMOS low drop-out (max 650mV) spanningsregelaar die 250 mA kan leveren en zelf slechts 2 µA verbruiken.

## 2.4” 240x320 spi tft LCD (3.3 V)

Om de klok en eventueel andere informatie te kunnen tonen aan de gebruiker hebben we een 2.4” 240x320 SPI tft scherm gebruikt. Correcte datasheets voor deze module zijn moeilijk te vinden. Hierdoor moesten we ons baseren op de weinige informatie die we wel konden vinden en voor de rest wachten tot de displays toekwamen.

De SD kaartlezer is in dit project overbodig, en dus ook niet aangesloten.



Figuur 1‑4 & 2-5: De SPI 2.4" 240x320 tft lcd display voor -en achteraanzicht.

## Buzzer

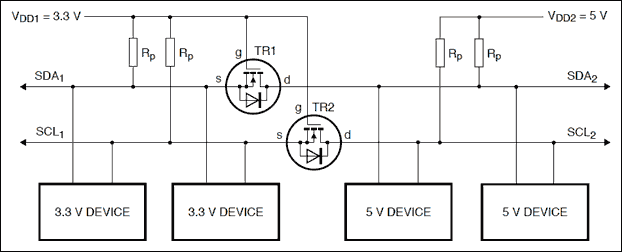
De buzzer dient hoofdzakelijk als back-up alarm. Indien de voedingspanning wegvalt, zal door de back-up voeding van de RTC deze toch nog blijven werken. Tijdens normale werking wordt het alarm in de RTC afgezet vooraleer het zijn interrupt kan geven en zullen de speakers voor het alarm zorgen. Indien de Raspberry pi niet gevoed wordt, zal dit niet gebeuren waardoor de interrupt wel wordt gegeven, en er een set plaats vindt van de flipflop. Dit zorgt er op zijn beurt voor dat de buzzer zal afgaan tot de gebruiker de flipflop reset via een aparte knop. De Raspberry pi kan de buzzer ook setten en resetten indien nodig.



Figuur 1‑5: De buzzer

## Levelshifter (AN10441)

De Raspberry Pi gebruikt 0 V en 3.3 V als GPIO level. Om te kunnen communiceren met de andere componenten op 5 V zoals de RTC hebben we levelshifters nodig. Aangezien de communicatie over bv. I2C bi-directioneel is, moeten de levelshifters ook bi-directioneel zijn en bovendien snel kunnen werken. De makkelijkste manier is om MOSFET’s op elke lijn te plaatsen. De AN10441’s werken bi-directioneel door zijn 3 mogelijke states. State 1 is wanneer het 3.3 V gedeelte hoog wordt/is. In dit geval is de MOSFET niet in geleiding aangezien de drempelspanning tussen de gate en de source niet is bereikt. Doordat de MOSFET niet in geleiding is wordt de 5 V kant op zijn beurt ook hoog getrokken door zijn eigen pull-up weerstand. Beide kanten zijn dus hoog maar op een ander spanningsniveau. De tweede state is wanneer de 3.3 V kant wordt laag getrokken. In dit geval wordt de drempelspanning tussen de gate en de source wel overschreden waardoor de MOSFET in geleiding gaat. Hierdoor wordt het 5 V gedeelte ook laag getrokken. De derde state is wanneer de 5 V kant laag wordt getrokken. In dit geval zal de diode, ingebouwd in de MOSFET ervoor zorgen dat de 3.3 V kant laag wordt getrokken tot een level waarbij de drempelspanning wordt overschreden. Wanneer dit gebeurt, zal de MOSFET in geleiding gaan, waardoor het 3.3 V gedeelte nog verder wordt laag getrokken.



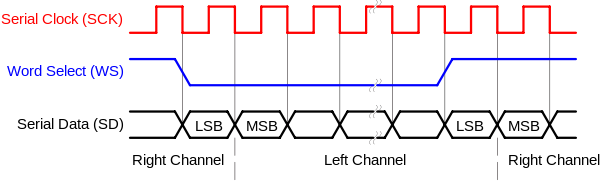
Figuur 1‑6: Shiften tussen 3.3 V en 5 V op een I2C bus.

## Audio stereo DAC (PCM 5102A 3.3 V)

Om ook muziek te kunnen spelen wanneer de wekker afgaat, hebben we een DAC (digital to analog converter) nodig. Deze chip ondersteunt de I2S serial bus interface standaard die dient voor digitale audio (zie intermezzo I2S). Dit is handig aangezien de Raspberry Pi Zero deze standaard ondersteunt. De PCM 5102A chip heeft een stereo output, wat wil zeggen dat er een L (links) en R (rechts) kanaal is voor de audio.

### Intermezzo I2S

I2S staat voor Inter-IC Sound en is een seriële bus interface standaard die wordt gebruikt om verschillende digitale audio devices te verbinden. De bus heeft minimum 3 lijnen: bit clock lijn, word clock lijn (WS of LRCLK) en een data lijn. De bit clock wordt gepulst voor elke bit op de datalijnen. De word clock laat het device weten voor welk kanaal (1 of 2 ) de huidige data is bedoeld. Wanneer de word clock laag is, is de data bedoeld voor het linker kanaal, anders voor het rechter kanaal.



Figuur 1‑7: De timing van een I2S bus interface.

## Audio amplifier (TPA2016D2 5V)

De TPA2016D2 is een stereo audio versterker die tot 2.8 W/kanaal kan leveren afhankelijk van de weerstand van de speakers en de voedingsspanning. De chip bevat ook een Dynamic Range Compression (DRC) en Automatic Gain Control (AGC) functie. De DRC functie gaat dynamisch de range van het geluid beperken. Dit wil zeggen dat de harde geluiden boven een bepaalde waarde worden afgezwakt terwijl de waardes onder deze drempel ongewijzigd blijven. De belangrijkste functie van de DRC is het opvangen van te grote niveauverschillen in het geluid. De AGC gaat op zijn beurt ervoor zorgen dat de versterking automatisch wordt aangepast aan het ingangssignaal. Zwakkere signalen zullen dus harder worden versterkt als de sterkere signalen. De versterker kan tussen -28dB en 30dB versterken op beide kanalen van het stereo signaal. Het instellen van de versterker kan door via I2C, 7 registers in te stellen. In deze registers kan bv. De versterking en de versterkingssnelheid worden ingesteld.

## NOR gate flipflop

We hebben een flipflop gemaakt met NOR gates (SR NOR latch). Dit wordt gebruikt in het back-up alarm aangestuurd door zowel de RTC als de Pi. Dit is ook rechtstreeks verbonden met de alarm drukknop zodat het alarm ook kan worden uitgeschakeld zonder netspanning.

## Supercap

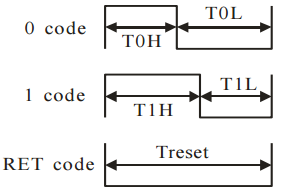
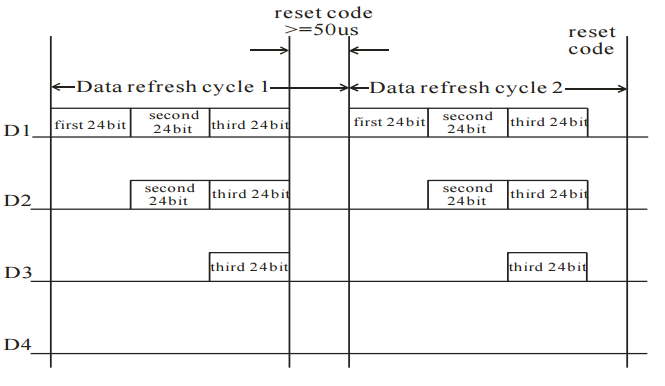
Als back-up voeding voor de basisfunctionaliteit van de klok wordt een condensator van 1.5 farad (“supercap”) gebruikt. De RTC schakelt over naar de voeding van de condensator indien de voedingsspanning wegvalt. Aangezien de RTC en flipflop slechts enkele µA gebruiken zou de backup functionaliteit zeer lang moeten meegaan, tenzij het alarm afgaat. Dan was de backup succesvol en is het dus niet erg is als de wekker uitvalt.



Figuur 1‑8:Een 'supercap' condensator

## WS2812 Leds

De WS2812 leds werken als een lange serie schakeling, waarbij elke led de eerste 24 bits gebruikt om zijn kleur in te stellen. De andere bits worden doorgegeven, zie onderstaande figuren. De volgorde van de bits is niet RGB, maar GRB, met hoogste bit eerst.



Figuur 1‑9:Timing diagram ws2812 leds.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Code | Betekenis | Tijd | Min | Typ | Max |
| T0H | 0 code - high voltage time | 0,35 µs ±150 ns | 200 ns | 350 ns | 500 ns |
| T1H | 1 code - high voltage time | 0,70 µs ±150 ns | 550 ns | 700 ns | 850 ns |
| T0L | 0 code - low voltage time | 0,80 µs ±150 ns | 650 ns | 800 ns | 950 ns |
| T1L | 1 code - low voltage time | 0,60 µs ±150 ns | 450 ns | 600 ns | 750 ns |
| Reset | low voltage time | > 50µs | 50 µs |  |  |
| TH+TL | Cyclus time | 1,25µs ±600ns | 660 ns | 1250 ns | 1850 ns |

## Het schema & PCB

Zie bijlage 6.1.

# Software

Hieronder is een korte uitleg te vinden over verschillende onderdelen uit de software. De volledige code met commentaar is te vinden in de bijlage.

## De nodige drivers en initialisatiecode

Wij gebruiken voor dit project een Linux distributie genaamd “Arch Linux ARM” omdat deze weinig onnodige toeters en bellen heeft. Er is echter wel een zeer ruime aanbieding software en help beschikbaar.

### Kernel-parameters en –modules

De Linux kernel heeft de mogelijkheid om via “serial console” te worden bestuurd, hiervoor moeten in “/boot/cmdline.txt” het loglevel naar 5 worden veranderd. Om dan ook een login console te krijgen en niet enkel kernel debug informatie wordt de “getty@ttyAMA0” service geactiveerd (met automatische root login, zie “.bash\_profile”).

Om I²C, SPI en I²S kernel modules te kunnen gebruiken moeten deze worden geactiveerd in “/boot/config.txt” en “/etc/modules-load.d/raspberrypi.conf”.

Om de LCD aan te sturen via een framebuffer, in plaats van rechtstreeks SPI te gebruiken, laden we via “/etc/modules-load.d/raspberrypi.conf” de “fbtft\_device” kernel module. Deze module heeft parameters nodig aangezien er meerdere LCD modules worden ondersteund. De parameters worden in “/etc/modprobe.d/fbtft.conf” beschreven (op één regel):

“options fbtft\_device custom name=fb\_ili9341 gpios=reset:23,dc:22 fps=23 speed=42000000 rotate=90”

* “name=fb\_ili9341” is de naam van de gebruikte LCD chip.  
  Dit is nodig voor de initialisatie code en de SPI data frames.
* “gpios= reset:23,dc:22” laat de driver weten waar de relevante pinnen zijn aangesloten. De SPI pinnen moeten niet worden beschreven.
* “fps=23” bepaald de maximale vernieuwingsfrequentie.
* “speed=42000000” legt de snelheid van de SPI bus vast op 42MHz.  
  (Experimenteel bepaalde maximale frequentie voor foutloze communicatie)
* “rotate=90” draait de framebuffer 90° zodat de LCD in landschap modus kan worden gebruikt zonder extra werk in de applicatiecode.

De achtergrondverlichting wordt niet via deze module geregeld omdat die geen ondersteuning bied voor dimmen. Later meer hierover.

### “.bash\_profile”

Door een bug in pygame is het niet mogelijk het hoofdprogramma uit te voeren als service. Om rond deze beperking te werken wordt gebruik gemaakt van een automatische login op de serial console. Deze voert dan “.bash\_profile” uit.

Dit script print een kleine hoeveelheid debug informatie, zet enkele omgevingsvariabelen juist, registreert de RTC en start het hoofdprogramma.

### Nginx

Om het Python programma niet onnodig te belasten met de web interface, worden alle statische files (HTML, CSS, JavaScript, Lettertypes) via het webserverprogramma Nginx naar de gebruiker gestuurd. Nginx luistert naar poort 80 (de standaard HTTP poort) en stuurt, indien het verkeer voor Python bestemd is, intern het verkeer door naar poort 5000. Dit is de poort waarop normaal Flask draait (zie verder). Nginx kan die onderscheiding makkelijk maken omdat al onze API calls naar python via een virtuele sub-directory “/api/” gaan.

Deze manier van werken heeft nog als voordeel dat Nginx veel sneller start dan Flask, en dus is de web-interface altijd gereed wanneer de gebruiker naar het IP adres surft. Als de web-interface wordt geopend voor de Flask server klaar is falen de API calls vanuit JavaScript maar dit kan worden opgevangen met een boodschap (“Even geduld a.u.b., het programma is nog aan het opstarten”) waarna opnieuw wordt geprobeerd.

## Het hoofdprogramma (app.py)

Voor dit programma is Python 3 gebruikt aangezien er veel ondersteuning is voor Python op de Raspberry Pi. Ook omdat Python in ons lessenpakket zit en omdat er (bijna) alle delen van het project in gemaakt kunnen worden.

Omdat CPython (de standaard Python implementatie) een “Global Interpreter Lock” gebruikt is het eenvoudig om veilig globale variabelen te gebruiken als gedeelde status tussen het aansturen van de LCD, de web interface en het alarm.

De code word hier gegroepeerd per functionaliteit.

### LCD aansturen

Om tekst op het scherm te krijgen wordt gebruik gemaakt van de Python module pygame. Deze module is bedoeld om via Python spelletjes te ontwikkelen en is veruit de makkelijkste manier on vanuit Python een framebuffer aan te sturen. Om aan te geven welke framebuffer SDL (de achterliggende grafische bibliotheek van pygame) moet gebruiken is in “.bash\_profile” de omgevingsvariabele “SDL\_FBDEV” op “/dev/fb1” gezet.

De achtergrondverlichting van de LCD module is niet verbonden via een van de kernel module opties, maar met PWM0 (pin 12). Het “gpio” commando wordt gebruikt om deze pin aan te sturen omdat dit minder CPU gebruikt dan de Python GPIO module.

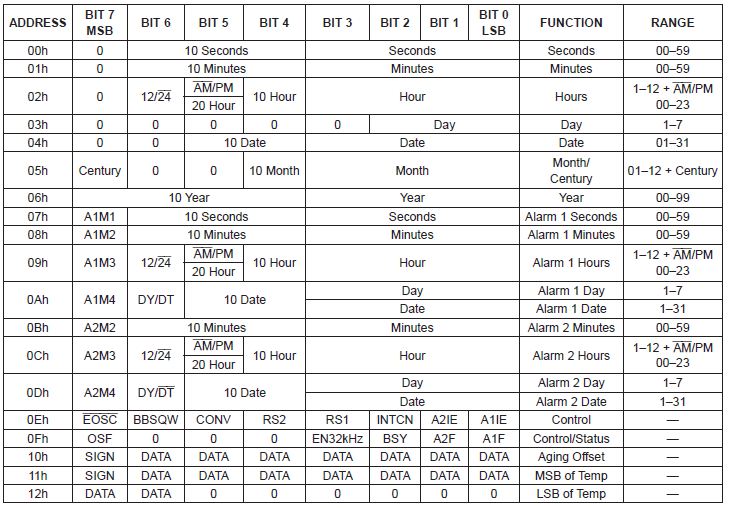
### RTC aansturen

Door het toevoegen van de nodige modules in de kernel parameters en het instellen van de RTC in “.bash\_profile” kunnen de basisfuncties van de RTC worden aangesproken zonder manueel I²C commando’s uit te voeren. Het ingebouwde commando “hwclock” kan nu worden opgeroepen via een “subprocess.call” met parameters

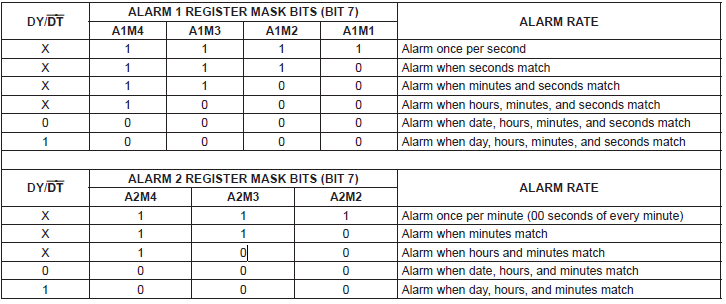
* “-w“ (write) voor schrijven van de systeemklok naar de RTC klok
* “-r” (read) voor het weergeven van de RTC klok (handig voor debug)
* “-s” (sync) voor het synchroniseren van de systeemklok naar de RTC klok

Om de alarmfuncties van de RTC te gebruiken is wel kennis van de registers en adressen nodig, aangezien die niet zijn ondersteund door de kernel. Hiervoor worden de volgende commando’s van het pakket “i2c-tools” gebruikt.

* **i2cset** [-f] [-y] [-m mask] [-r] i²cbus chip-address data-address [value] ... [mode]
* **i2cget** [-f] [-y] i²cbus chip-address [data-address [mode]]



Figuur 2‑1:De register tabel voor de RTC. Geeft weer welke registers moeten worden ingesteld voor het alarm.



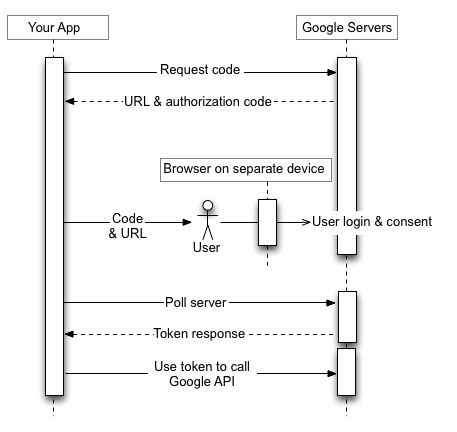
Figuur 2‑2:Deze waardes bepalen bij welke overeenkomst de interrupt pin moet worden geactiveerd.

### Google Calendar

De SmartAlarmClock gebruikt de Google Calendar API om toegang te kunnen krijgen tot de kalender van de gebruiker.

#### Registreren van SmartAlarmClock

Om als ontwikkelaar toegang te krijgen tot de Google APIs moet de toepassing worden geregistreerd via de Google ontwikkelaarsconsole. Tijdens het registeren moet de ontwikkelaar aanduiden tot welke onderdelen van Google de toepassing de toegang wilt. Na de registratie geeft Google de ontwikkelaar een “client ID” en “client secret”, deze zijn zeer belangrijk verder in het authenticatie proces. De “client ID” en “client secret” zijn hetzelfde voor alle SmartAlarmClocks en dienen dus uitsluitend om de applicatie te onderscheiden.



Figuur 2‑3:Schematische voorstelling van het stappenplan om toegang te krijgen tot de Google Calendar van de gebruiker. Dit is oAuth 2.0 voor embedded devices.

#### Toestemming krijgen van de gebruiken

Dit is de eerste stap die moet worden uitgevoerd om de gebruiker om toestemming te vragen. Er moet een HTTP POST request worden gestuurd naar Google met de “client ID” en een lijst van “scopes”. De “scopes” geven weer tot wat de applicatie toegang wil in dit geval is het alleen lezen toegang tot de kalender. Het antwoord van deze request is een JSON object dat 5 items bevat. De “user code” en “verificatie URL” moeten aan de gebruiker worden getoond. Het “interval”, “interval device code” en “expire time” zijn nodig voor de toepassing maar moeten niet aan de gebruiker worden getoond.

De gebruiker moet naar de “verificatie URL” surfen en vervolgens zijn “user code” ingeven en op volgende klikken. Nu zal een nieuwe pagina laden waarin staat beschreven welke toepassing tot welke delen toegang vraagt. De laatste stap voor de gebruiker is het klikken op “toestaan”.

Tegelijkertijd met het tonen van de “user code” en de “verificatie URL”, kan de toepassing beginnen met het pollen van het Google API OAuth endpoint voor een “access” en “refresh token”. Het pollen bestaat uit een POST request die de “device code”, “client ID” en “client secret” bevat. De tijd tussen requests wordt gespecifieerd door het “interval” uit het eerste request. Zolang de gebruiker geen toegang heeft verleend, zal het antwoord op de request een JSON object zijn dat een error bevat: “authorization\_pending”. Deze error kan ook informatie bevatten zoals “slow down” indien de requests te snel op elkaar volgen. Indien de gebruiker wel toegang heeft verleend zal het antwoord een JSON object zijn dat een “access token”, “refresh token”, “token type” en “expire” bevat. De “access token” wordt gebruikt tijdens het opvragen van informatie uit de kalender. De “refresh token” is nodig bij het verkrijgen van een nieuwe “access token” na het verlopen van de vorige en moet dus worden opgeslagen.

#### Kalender afspraken opvragen

De volgende stap is het opvragen van de informatie uit de Google Calendar. Hiervoor is een GET request nodig naar het “/calendars/<calendar id >/events” endpoint met een geldige “access token”. De “calendar id” is standaard “primary”, maar kan indien gewenst worden aangepast om informatie uit een ander “kalenderbestand” te gebruiken. Dit is een eenvoudige manier om de afspraken te filteren. Extra parameters kunnen worden toegevoegd aan de request om de hoeveelheid nutteloze informatie te beperken. Zo zijn enkel de nabije toekomstige afspraken nuttig en dus geven we als “timeMin” parameter de huidige tijd mee, en als “timeMax” de huidige tijd plus 7 dagen. Het antwoord is een JSON object dat kan bewaard en gebruikt worden in de rest van dit programma als “status[‘items’]”.

#### Een vervallen “Access token” vervangen

Indien de access token is vervallen moet met de “refresh token” een nieuwe worden opgevraagd. Een POST request met de “client ID”, “client secret” en “refresh token” zal als antwoord een nieuwe access token geven.

De volledige handleiding over de Google OAuth 2.0 for devices API met voorbeelden is te vinden op: <https://developers.google.com/identity/protocols/OAuth2ForDevices>

### Flask (Web interface API)

Flask is een micro framework voor Python waarmee webpagina’s en Python code met elkaar kunnen worden verweven. Eerst moet een instantie van de Flask klasse worden gemaakt. Dit object heeft een “run” functie die met enkele parameters kan worden opgeroepen. De 2 belangrijke parameters zijn “host” en “port”. De “host” parameter geeft weer op welk IP-adres Flask moet luisteren. De “port” parameter geeft op zijn beurt weer op welke poort Flask moet luisteren. Met de “route” annotatie wordt ingesteld op welke URL een bepaalde functie moet worden uitgevoerd. Op deze manier kunnen AJAX calls vanuit de web interface een functie in Python oproepen. De opgeroepen Python functie wordt uitgevoerd en de return waarde wordt als response teruggestuurd naar de web interface. De response informatie kan dan worden opgenomen in de web interface. Op deze manier worden bijvoorbeeld de huidige “settings” opgevraagd die in de “settings” tab van de web interface kunnen worden weergegeven.

### Netwerk en acces point

De eerste stap in verband met het netwerk is het controleren van het bestaan van de “wlan0” adapter. Dit kan door te controleren of het pad “/sys/class/net/wlan0” bestaat. Indien deze niet bestaat wordt de errorboodschap “no wifi interface” op het scherm weergegeven. De tweede stap is controleren of er al een geldig wifi profiel is ingesteld. Indien dit het geval is wordt de functie “attempt\_connect” opgeroepen.

Deze functie zal eerst alle actieve netwerkverbindingen op “wlan0” verbreken. Vervolgens wordt via “subproces.call” het “netctl” commando uitgevoerd om naar het ingestelde wifi profiel te wisselen. Indien dit lukt, zal de “status[‘network’]” op “True” worden gezet en zal het IP adres op de display worden getoond. Nu de SmartAlarmClock een netwerk heeft wordt meteen ook via “ntp” (network time protocol) de tijd juist gezet. Indien de synchronisatie met “ntp” mislukt wordt dit via een foutboodschap op het scherm aan de gebruiker getoond. Indien de synchronisatie lukt wordt de RTC tijd ook worden ge-update en wordt “status[‘draw’][‘clock’]” op “True” gezet zodat de tijd op het scherm kan worden getoond. Indien het wisselen naar het netwerk profiel mislukt, wordt dit aan de gebruiker getoond.

Wanneer de “status[‘network’]” op “True” staat kunnen we er zeker van zijn dat we verbonden zijn met een geldig netwerk. Indien er een “refresh token” is opgeslagen wordt die gebruikt om een nieuwe “acces token” te vragen. Anders wordt de aanvraagprocedure gestart. Zolang er geen netwerk verbinding is zal er een eigen access point worden gemaakt om de gebruiker in staat te stellen om een netwerk te selecteren.

Het maken van een wifi profiel gebeurt via de tab “wifi settings” in de web interface. Een lijst met beschikbare WiFi netwerken wordt geladen via JavaScript. Als het formulier is ingevuld en is verzonden, wordt er een nieuw bestand aangemaakt in “/etc/netclt” met het juiste formaat en de gegevens over het gekozen WiFi netwerk. Via een “attempt\_connect” wordt er dan geprobeerd dit profiel te laden.

Via het extern programma “iwlist” wordt de scan naar wifi netwerken uitgevoerd. De output van dit wordt het eerst met RegEx omgezet naar een JSON vriendelijk formaat.

### Rotary encoder

Om de rotary encoder aan te sturen wordt gebruik gemaakt van de “RPi.GPIO” module in Python. Eerst wordt “GPIO” in de BCM mode gezet, dit wil zeggen dat de pinnen kunnen worden aangesproken via de BCM pinnummering. Vervolgens worden pin A, pin B en pin S (switch) via “GPIO.setup” als input gezet en worden de inwendige pull-up weerstanden geactiveerd. Als laatste wordt een callback toegevoegd aan pin A en pin S. Deze callback functies reageren op een falling edge en roepen respectievelijk de functie “int\_rot” en “int\_btn\_ok” op. Voor pin A (rotatie) wordt een bouncetime van 25ms toegevoegd en voor de pin S (switch) 250ms. De callback functies navigeren door het menu aan de hand van het aantal klikken en/of rotaties door waardes in de “status” dictionary te veranderen. De mogelijke waardes zijn “None” en elke waarde die in de enum Menu zit. Elk element uit deze enum bevat een naam voor het menu veld en eventueel namen voor welke settings het menu item kan aanpassen. De @unique annotatie wordt toegevoegd aan de enum om zeker te zijn dat er geen waardes dubbel worden opgenomen. Dit vermijdt domme typfoutjes en lang debug werk.

### Alarm/muziek

Voor het afspelen van het geluid gingen we oorspronkelijk de I2S interface gebruiken. Aangezien de printen echter niet tijdig werden geleverd moesten we een PWM pin opofferen om op deze manier muziek af te spelen. Hierdoor kunnen we geen hardware PWM meer gebruiken voor de WS2812 leds. Zolang de muziek niet speelt wordt de software PWM gebruikt om de LCD te dimmen. Wanneer de muziek speelt werkt dit echter niet meer aangezien de software en de muziek beide DMA nodig hebben. Om dit op te lossen opteerde we om de helderheid van het scherm tijdens het spelen van muziek op het maximum te zetten. Voor het spelen van muziek wordt gekozen uit een lijst van MP3 streams (VRT radiozenders). Deze worden afgespeeld via de commandline mp3 speler: “mpg123”.

### Web interface

Voor de web interface is de HTML, CSS en javascript framework bootstrap gebruikt. Dit stelt ons in staat om een mooie website te maken die schaalbaar is voor verschillende toestellen. De web interface bestaat uit 1 webpagina met 4 verschillende tabs. Het nut van elke tab wordt hieronder kort uitgelegd. De html en javascript met commentaar is in de bijlage te vinden.

#### Status tab

De status tab geeft de gebruiker wat informatie over de status van het toestel.

#### Wifi settings tab

Deze tab geeft een lijst van alle beschikbare netwerken. De gebruiker kan hier zijn netwerk kiezen en instellen.

#### Clock settings tab

De settings tab is de belangrijkste tab na het in gebruik nemen van het toestel. Hier kan de gebruiker de SmartAlarmClock instellen naar zijn eigen wensen. Ten eerste kan het formaat en de grootte van de tijd en de datum worden ingesteld. Ten tweede kan de gebruiker instellen hoelang voor de eerste afspraak de wekker moet afgaan. Ten derde kan ook optioneel een minimum en maximum wek tijd worden ingesteld. Deze waarden willen zeggen dat de wekker nooit vroeger dan het minimum en later dan het maximum mag afgaan. Een vierde instelling bepaalt of de wekker enkel in de week, weekend of elke dag mag afgaan. Ten vijfde kan de gebruiker kiezen of de dag wordt weergegeven of niet en indien deze wordt getoond in welke grootte dit moet. De laatste instelling bepaalt welk type van alarm er moet worden gebruikt tijdens het wekken. De gebruiker heeft hierbij de keuze uit verschillende muziekstreams.

#### Google Calendar tab

In de Google Calendar tab kan de gebruiker kiezen welke kalender moet worden gebruikt. Ook het resetten van de Google Calendar link, indien deze is vervallen kan in deze tab. Op deze pagina kan ook de user code en de verification URL worden getoond wanneer de gebruiker toegang moet geven aan de SmartAlarmClock .

# Budget/kostenraming

Dit is een kostenraming van de componenten voor dit project (voor 1 product):

|  |  |
| --- | --- |
| Naam | Kostprijs |
| Wifi dongle | € 2,00 |
| Buzzer | € 3,00 |
| USB adapter | € 1,00 |
| RTC | € 0,50 |
| Supercap | € 2,00 |
| Levelshifters | € 0,20 |
| 2 x speaker | € 2,50 |
| 3.3V spanningsregulator | € 0,50 |
| Micro SD | € 8,00 |
| Leds | € 0,50 |
| Power adapter | € 2,00 |
| Amplifier | € 2,50 |
| Rotary encoders | € 0,50 |
| PCB | € 3,50 |
| Raspberry Pi Zero essentials kit | € 8,00 |
| LCD scherm | € 7,00 |
| DAC | € 2,50 |
| 8 x WS2812 leds | € 0,80 |
| **Totaal** | **€ 47,00** |

# Besluit

Dit project was zeer leerrijk aangezien vele onderdelen met elkaar moesten worden verbonden worden om een werkend resultaat te verkrijgen. De grootste tegenslag tijdens het project was de levertijd van de LCD module. Aangezien er voor de bestelde display niet echt eenduidige datasheets (pin lay-out, grootte, afstand van de gaten) te vinden waren, moesten de printen wachten tot de display werd geleverd. Dit zou op zich geen probleem geven, maar door het overschrijden van de maximum levertijd zorgde dit voor een extra tijdsdruk.

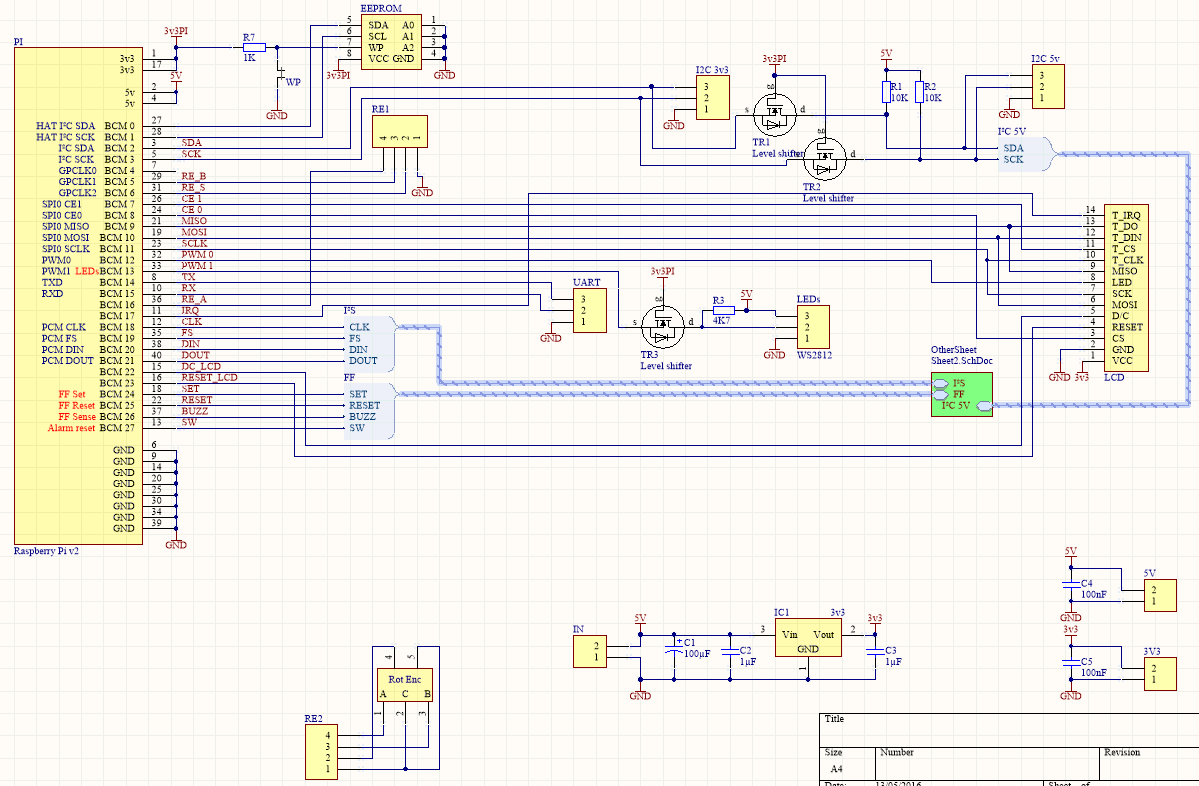
Voor de muziek werd initieel gedacht om de I2S interface te gebruiken. Aangezien de printen niet tijdig werden geleverd ging dit echter niet meer. Om rond dit probleem te werken hebben we dan enkel zaken aangepast. Voor het geluid besloten we om via PWM te werken waardoor de ws2812 leds moesten worden weggelaten. Zolang er geen muziek speelt zal de achtergrondverlichting van de LCD via software PWM werken. Wanneer de muziek speelt werkt dit echter niet meer aangezien de software en de muziek beide DMA nodig hebben. Als oplossing voor dit probleem kozen we om de helderheid van het scherm tijdens het spelen van de muziek op 100% te zetten.

Al de weggelaten functionaliteiten zijn wel onderzocht en ontworpen, daarom zijn er nog sporen van te vinden in de software en is de beschrijving in dit document gebleven.

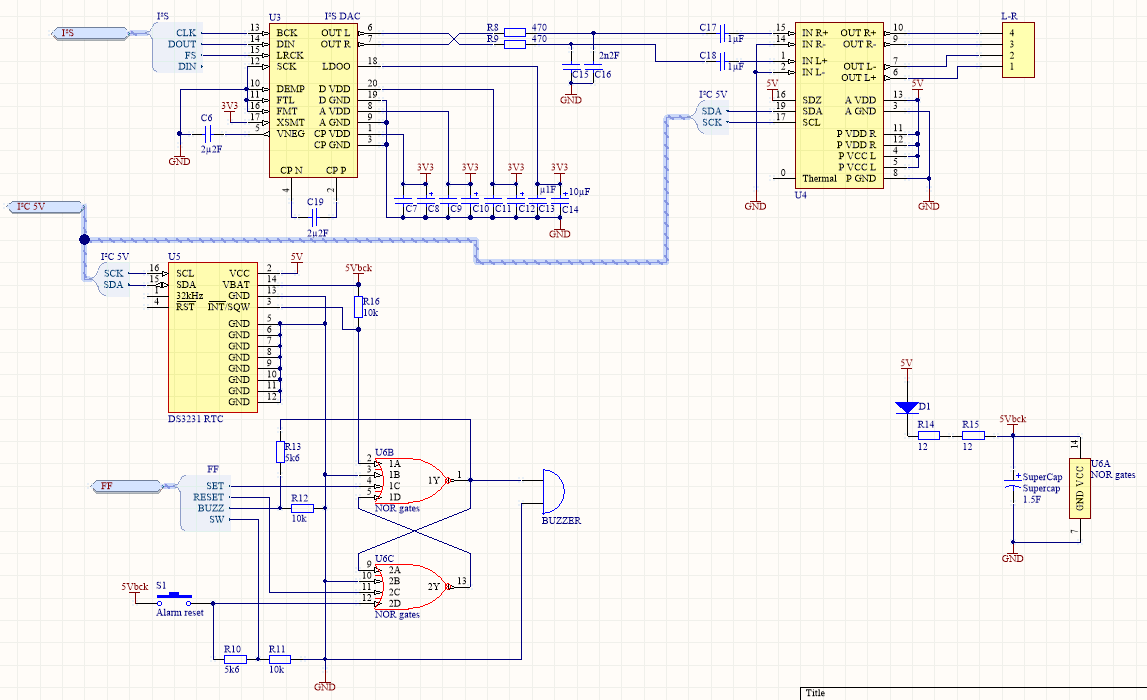
# Bijlagen

## Schema’s & Lay-out printplaat

### De schema’s

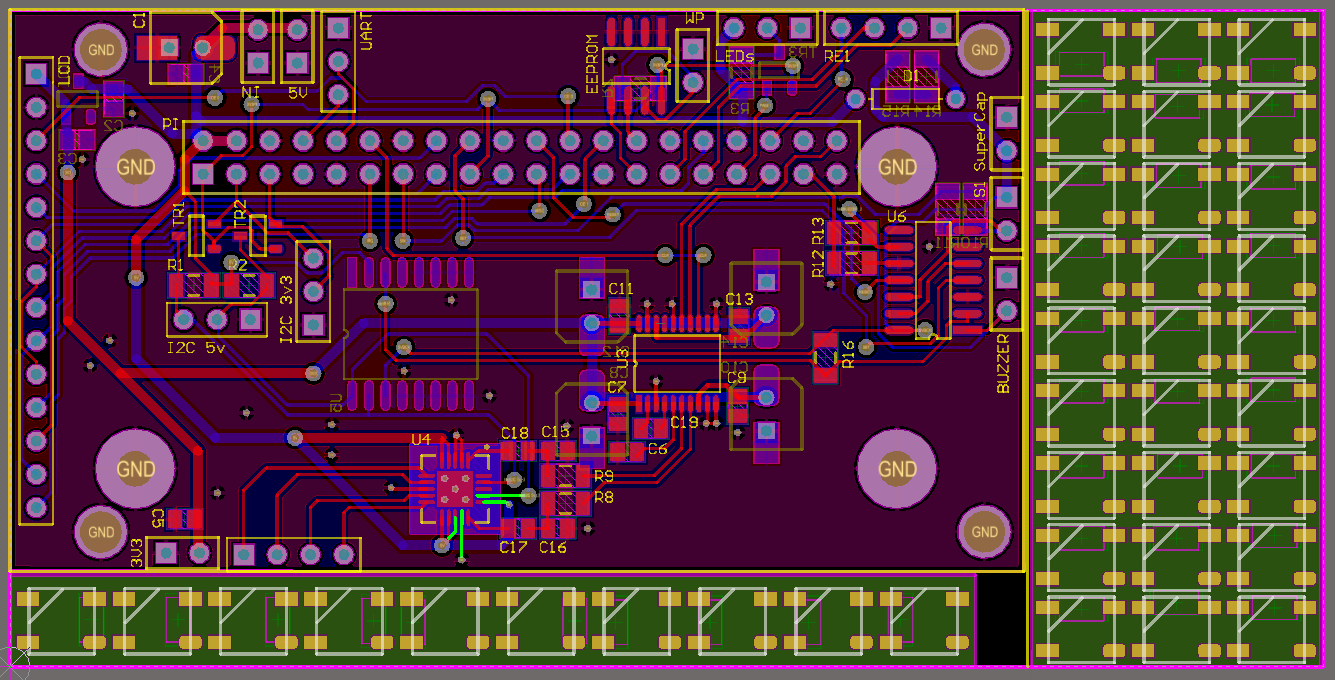


Figuur 5‑1: Het schema, sheet 1.



Figuur 5‑2: Het schema, sheet 2.

### De printplaat



Figuur 5‑3:Het roze gedeelte is de PCB lay-out voor dit project. Om de resterende ruimte van de print niet verloren te laten gaan werden hier pads voor ws2812 leds voorzien.

## Het hoofdprogramma (app.py)

#!/bin/env pythons

# ####################################### BOOT SEQ, PART 1 - Pre

**import** datetime

**print(**'Boot sequence part 1 - Pre (%s)' **%** datetime**.**datetime**.**now**())**

# ############################## Imports

**import** os

**import** subprocess

**import** threading

**import** time

**import** json

**import** sys

**from** enum **import** Enum

**from** enum **import** unique

# ############################## Definitions

VERSION **=** '0.1'

SETTINGS\_FILE **=** "/root/www/settings.json"

# Main Menu enum. If setting is None, its a toggle

@unique

**class** **Menu(**Enum**):**

Exit **=** **{**'name'**:** 'Exit'**,** 'setting'**:** **None}**

Show\_IP **=** **{**'name'**:** 'Show IP'**,** 'setting'**:** **None}**

Set\_Volume **=** **{**'name'**:** 'Set Volume'**,** 'setting'**:** **(**'sound'**,** 'volume'**)}**

Set\_Brightness **=** **{**'name'**:** 'Set Brightness'**,** 'setting'**:** **(**'brightness'**,** 'preference'**)}**

# Days enum.

@unique

**class** **Days(**Enum**):**

Weekdays **=** **[**1**,** 2**,** 3**,** 4**,** 5**]**

Weekends **=** **[**6**,** 7**]**

Both **=** **[**1**,** 2**,** 3**,** 4**,** 5**,** 6**,** 7**]**

# Volatile storage

status **=** **{**'booting'**:** **True,** 'network'**:** **False,** 'skipped'**:** **False,**

'draw'**:** **{**'clock'**:** **False,** 'option'**:** **None},** 'menu'**:** **None,** 'clock'**:** **False,**

'pulsing'**:** **False,** 'streaming'**:** **False,** 'gcal'**:** **{**

**}}**

# Permanent storage

settings **=** **{**'day'**:** **{**'enabled'**:** **True,** 'size'**:** 40**},**

'clock'**:** **{**'format'**:** '%H:%M:%S'**,** 'size'**:** 60**},**

'date'**:** **{**'enabled'**:** **True,** 'format'**:** '%d-%m-%y'**,** 'size'**:** 36**},**

'alarm'**:** **{**'offset'**:** 60**,** 'min'**:** 6 **\*** 60**,** 'max'**:** 12 **\*** 60**,** 'days'**:** Days**.**Weekdays**,**

'stream'**:** 'MNM Hits'**},** 'sound'**:** **{**'volume'**:** 50**,** 'min'**:** 15**,** 'step'**:** 1**,** 'max'**:** 100**},**

'brightness'**:** **{**'preference'**:** 50**,** 'now'**:** 100**,** 'target'**:** 100**,** 'step'**:** 1**,** 'min'**:** 15**,**

'max'**:** 100**}}**

# Google Calender API app specific data

gcal **=** **{**'client\_id'**:** os**.**getenv**(**'APP\_GCAL\_ID'**),**

'client\_secret'**:** os**.**getenv**(**'APP\_GCAL\_SECRET'**),**

'scope'**:** 'https://www.googleapis.com/auth/calendar.readonly'**}**

# List of pre-defined streams

streams **=** **{**'MNM'**:** 'http://mp3.streampower.be/mnm-high.mp3'**,**

'MNM Hits'**:** 'http://mp3.streampower.be/mnm\_hits-high.mp3'**,**

'Studio Brussel'**:** 'http://mp3.streampower.be/stubru-high.mp3'**,**

'Klara'**:** 'http://mp3.streampower.be/klara-high.mp3'**,**

'Radio 1'**:** 'http://mp3.streampower.be/radio1-high.mp3'**,**

'Radio 2 Antwerpen'**:** 'http://mp3.streampower.be/ra2ant-high.mp3'**}**

# Misc globals

music\_process **=** **None**

**if** gcal**[**'client\_id'**]** **is** **None** **or** gcal**[**'client\_secret'**]** **is** **None:**

**print(**'APP\_GCAL\_ID and or APP\_GCAL\_SECRET not set.'**)**

sys**.**exit**(**1**)**

# Required for json encoding Enums

**class** **EnumEncoder(**json**.**JSONEncoder**):**

**def** default**(**self**,** obj**):**

**if** isinstance**(**obj**,** Enum**):**

**return** str**(**obj**)**

**return** json**.**JSONEncoder**.**default**(**self**,** obj**)**

# Send volume to alsa

**def** set\_volume**():**

vol **=** **(**50 **+** **(**settings**[**'sound'**][**'volume'**]** **/** 2**))**

subprocess**.**call**([**'amixer'**,** 'sset'**,** 'PCM,0'**,** '%.0f%%' **%** vol**,** '-M'**])**

# Save settings

**def** save**():**

json**.**dump**(**settings**,** open**(**SETTINGS\_FILE**,** 'w'**),** indent**=**2**,** cls**=**EnumEncoder**)**

# Clamp value between 0 and 100 by default

**def** clamp**(**n**,** minn**=**0**,** maxn**=**100**):**

**return** max**(**min**(**maxn**,** n**),** minn**)**

# Go back from "<Class>.<Name>" to actual enum instance

**def** as\_enum**(**full**):**

**if** full **is** **None:**

**return** **None**

name**,** member **=** full**.**split**(**"."**)**

**return** getattr**(**globals**()[**name**],** member**)**

**def** stream\_start**():**

**global** music\_process

**if** music\_process **is** **not** **None:**

**return**

stream **=** streams**[**settings**[**'alarm'**][**'stream'**]]** **if** settings**[**'alarm'**][**

'stream'**]** **in** streams **else** \

settings**[**'alarm'**][**'stream'**]**

status**[**'streaming'**]** **=** **True**

music\_process **=** subprocess**.**Popen**([**'mpg123'**,** '-T'**,** stream**],** universal\_newlines**=True,**

bufsize**=**1**,** stderr**=**subprocess**.**PIPE**)**

**def** stream\_parser**():**

re\_title **=** re**.**compile**(**r"ICY-META: StreamTitle='(.\*)';"**)**

**while** music\_process **is** **not** **None** **and** music\_process**.**poll**()** **is** **None:**

line **=** music\_process**.**stderr**.**readline**()**

matcher **=** re\_title**.**search**(**line**)**

**if** matcher**:**

status**[**'draw'**][**'title'**]** **=** matcher**.**group**(**1**)**

**if** 'title' **in** status**[**'draw'**]:**

**del** status**[**'draw'**][**'title'**]**

threading**.**Thread**(**target**=**stream\_parser**,** name**=**'StreamParser'**,** daemon**=True).**start**()**

**def** stream\_stop**():**

**global** music\_process

**if** music\_process **is** **None:**

**return**

**if** music\_process**.**poll**()** **is** **None:**

music\_process**.**terminate**()**

music\_process**.**wait**()**

status**[**'streaming'**]** **=** **False**

music\_process **=** **None**

# ############################## Sequential code

# set pwm pin

# subprocess.call(['gpio', '-g', 'mode', '12', 'pwm'])

# set pwm pin sound

subprocess**.**call**([**'gpio\_alt'**,** '-p'**,** '13'**,** '-f'**,** '0'**])**

# Kill any existing ap

subprocess**.**call**([**'create\_ap'**,** '--stop'**,** 'wlan0'**])**

# Sometimes the above isn't enough

subprocess**.**call**([**'killall'**,** 'hostapd'**])**

**if** os**.**getenv**(**'SDL\_FBDEV'**)** **is** **None:**

**print(**'SDL\_FBDEV not set.'**)**

sys**.**exit**(**1**)**

# Write to SDL\_FBDEV (normally /dev/fb1)

**with** open**(**os**.**getenv**(**'SDL\_FBDEV'**),** 'wb'**)** **as** outfile**:**

# Raw image data of 'Booting..' centered on the lcd. Made by a little custom C program

frame **=** bytearray**([**0x00**]** **\*** 44446 **+** **[**0xff**]** **\*** 6 **+** **[**0x00**]** **\*** …………………………………………………………………………**)**

outfile**.**write**(**frame**)**

outfile**.**close**()**

# ####################################### BOOT SEQ, PART 2 - GPIO

**print(**'Boot sequence part 2 - GPIO (%s)' **%** datetime**.**datetime**.**now**())**

# ############################## Imports

# noinspection PyUnresolvedReferences

**import** RPi**.**GPIO **as** GPIO

# ############################## Definitions

RE\_A **=** 16

RE\_B **=** 5

RE\_S **=** 6

ALARM\_S **=** 27

PWM\_BG **=** 12

# Set PWM of LCD background LED via gpio program because the GPIO python module

**def** set\_brightness**(**percent**=**100**):**

p**.**ChangeDutyCycle**(**percent**)**

# subprocess.call(['gpio', '-g', 'pwm', '12', '%.0f' % (clamp(percent) \* 10.23)])

# While booting, ramp up the LCD backlight from 0 to 100% over 2 seconds

**def** pre\_boot\_pwm**():**

i **=** 0

**while** status**[**'booting'**]** **and** i **<** 100**:**

set\_brightness**(**i**)**

time**.**sleep**(**0.2**)**

i **+=** 1

set\_brightness**(**100**)**

# 'Interrupt' handler of alarm button

**def** int\_btn\_alarm**(**chan**):**

**print(**'BTN: ALARM'**)**

**if** 'alarm' **in** status **and** **not** status**[**'skipped'**]:**

ts\_alarm **=** status**[**'alarm'**].**replace**(**second**=**0**,** microsecond**=**0**).**timestamp**()**

ts\_now **=** datetime**.**datetime**.**now**().**replace**(**second**=**0**,** microsecond**=**0**).**timestamp**()**

**if** int**(**ts\_alarm **-** ts\_now**)** **<** 600**:**

status**[**'skipped'**]** **=** **True**

**if** 'alarm' **in** status**[**'draw'**]:**

**del** status**[**'draw'**][**'alarm'**]**

**if** status**[**'streaming'**]:**

stream\_stop**()**

**del** status**[**'alarm'**]**

next\_event**()**

**return**

**if** status**[**'streaming'**]:**

stream\_stop**()**

**else:**

stream\_start**()**

# 'Interrupt' handler button of rot encoder

**def** int\_btn\_ok**(**chan**):**

**print(**'BTN: OK'**)**

**if** **not** status**[**'draw'**][**'clock'**]:**

**return**

**if** status**[**'menu'**]:**

status**[**'draw'**][**'option'**]** **=** **None** **if** status**[**'menu'**]** **==** Menu**.**Exit **else** status**[**'menu'**]**

status**[**'menu'**]** **=** **None**

**elif** status**[**'draw'**][**'option'**]:**

status**[**'draw'**][**'option'**]** **=** **None**

save**()**

**else:**

status**[**'menu'**]** **=** Menu**.**Show\_IP

# 'Interrupt' handler for A of rotary encoder

**def** int\_rot**(**chan**):**

a **=** GPIO**.**input**(**RE\_A**)**

# Since A triggers on falling edge, it should be 0, if not, debounce

**if** a**:**

**return**

# To get the direction, pull B

b **=** GPIO**.**input**(**RE\_B**)**

**print(**'Rotate: %s' **%** **(**'Right,+' **if** b **else** 'Left,-'**))**

# If we are in menu, go left or right

**if** status**[**'menu'**]:**

items **=** list**(**Menu**)**

i **=** items**.**index**(**status**[**'menu'**])**

status**[**'menu'**]** **=** items**[(**i **+** **(**1 **if** b **else** **-**1**))** **%** len**(**items**)]**

# If we are doing a setting

**elif** status**[**'draw'**][**'option'**]:**

**if** status**[**'draw'**][**'option'**].**value**[**'setting'**]:**

setting **=** status**[**'draw'**][**'option'**].**value**[**'setting'**]**

current **=** settings

# Go down the the sencond to last object and property name, so it can be set

**for** key **in** setting**[:-**1**]:**

current **=** current**[**key**]**

current**[**setting**[-**1**]]** **=** clamp**(**current**[**setting**[-**1**]]** **+** **(**1 **if** b **else** **-**1**))**

set\_volume**()**

# ############################## Sequential code

GPIO**.**setmode**(**GPIO**.**BCM**)**

GPIO**.**setwarnings**(False)**

GPIO**.**setup**([**RE\_A**,** RE\_B**,** RE\_S**,** ALARM\_S**],** GPIO**.**IN**,** pull\_up\_down**=**GPIO**.**PUD\_UP**)**

GPIO**.**setup**(**PWM\_BG**,** GPIO**.**OUT**)**

p **=** GPIO**.**PWM**(**PWM\_BG**,** 250**)**

p**.**start**(**0**)**

GPIO**.**add\_event\_detect**(**RE\_A**,** GPIO**.**FALLING**,** callback**=**int\_rot**,** bouncetime**=**25**)**

GPIO**.**add\_event\_detect**(**RE\_S**,** GPIO**.**FALLING**,** callback**=**int\_btn\_ok**,** bouncetime**=**500**)**

GPIO**.**add\_event\_detect**(**ALARM\_S**,** GPIO**.**FALLING**,** callback**=**int\_btn\_alarm**,** bouncetime**=**500**)**

# Start the preboot LCD backlight ramp up

threading**.**Thread**(**target**=**pre\_boot\_pwm**,** name**=**'PreBootPWM'**,** daemon**=True).**start**()**

# ####################################### BOOT SEQ, PART 3 - Pygame

**print(**'Boot sequence part 3 - Pygame (%s)' **%** datetime**.**datetime**.**now**())**

# ############################## Imports

# noinspection PyUnresolvedReferences

**import** pygame

**import** requests

**import** socket

**import** re

**import** urllib

**import** sched

**import** dateutil**.**parser

**import** signal

# ############################## Definitions

# Required to avoid pygame.display.init hanging on a second boot

**def** signal\_handler**(**signal**,** frame**):**

**print(**'EXIT: SIGTERM or SIGINT (%s)' **%** datetime**.**datetime**.**now**())**

time**.**sleep**(**1**)**

pygame**.**quit**()**

p**.**stop**()**

# last, because its possible it may throw up, if GPIO hasn't imported yet. That is

# fine, if it happens after pygame.quit

GPIO**.**cleanup**()**

sys**.**exit**(**0**)**

# ############################## Sequential code

# Handle kill command

signal**.**signal**(**signal**.**SIGTERM**,** signal\_handler**)**

# Handle Control-C

signal**.**signal**(**signal**.**SIGINT**,** signal\_handler**)**

BLACK **=** **(**0**,** 0**,** 0**)**

WHITE **=** **(**255**,** 255**,** 255**)**

RED **=** **(**255**,** 0**,** 0**)**

**print(**'Init pygame... (%s)' **%** datetime**.**datetime**.**now**())**

# This call takes a while, it can also hang if pygame wasn't quited last time,

# hence the kill/^C handling

pygame**.**display**.**init**()**

# Instead of initing all pygame subsystems, we only do display and font to cut the

# loading time by a lot

pygame**.**font**.**init**()**

pygame**.**mouse**.**set\_visible**(False)**

# Monospace fonts

FONT\_XL **=** pygame**.**font**.**SysFont**(**'notomono'**,** 60**)**

FONT\_L **=** pygame**.**font**.**SysFont**(**'notomono'**,** 36**)**

FONT\_M **=** pygame**.**font**.**SysFont**(**'notomono'**,** 26**)**

FONT\_S **=** pygame**.**font**.**SysFont**(**'notomono'**,** 15**)**

FONT\_ICO **=** pygame**.**font**.**SysFont**(**'fontawesome'**,** 15**)**

# Screen size

SIZE **=** **(**pygame**.**display**.**Info**().**current\_w**,** pygame**.**display**.**Info**().**current\_h**)**

# Screen surface

SCREEN **=** pygame**.**display**.**set\_mode**(**SIZE**,** pygame**.**FULLSCREEN**)**

# Clear screen

SCREEN**.**fill**(**BLACK**)**

**print(**'Done init pygame & clear ... (%s)' **%** datetime**.**datetime**.**now**())**

**if** os**.**path**.**isfile**(**SETTINGS\_FILE**):**

**try:**

settings**.**update**(**json**.**load**(**open**(**SETTINGS\_FILE**)))**

settings**[**'alarm'**][**'days'**]** **=** as\_enum**(**settings**[**'alarm'**][**'days'**])**

set\_volume**()**

**except** json**.**decoder**.**JSONDecodeError**:**

**print(**'Config file unreadable, lets just throw it away and start fresh'**)**

**else:**

save**()**

# ####################################### BOOT SEQ, PART 4 - Scheduler

**print(**'Boot sequence part 4 - Scheduler (%s)' **%** datetime**.**datetime**.**now**())**

# ############################## Definitions

# Draw text on screen, below other height, in font & color, centered by default

**def** draw\_text**(**message**,** font**=**FONT\_S**,** color**=**WHITE**,** height**=**0**,** center**=True):**

# if height is 0, assume the screen needs clearing

**if** height **==** 0**:**

SCREEN**.**fill**(**BLACK**)**

# Render the text on a new surface

text **=** font**.**render**(**message**,** **False,** color**)**

x **=** 0

# Center the text

**if** center**:**

x **=** **(**SCREEN**.**get\_width**()** **-** text**.**get\_width**())** **/** 2

# Draw the text surface on the screen

SCREEN**.**blit**(**text**,** **(**x**,** height**))**

# 'Commit' the changes

pygame**.**display**.**update**()**

# Return the new height

**return** height **+** text**.**get\_height**()**

# Special message display, used for 'fatal crashes'

**def** error**(**message**):**

SCREEN**.**fill**(**BLACK**)**

height **=** draw\_text**(**'SmartAlarmClock (%s)' **%** VERSION**,** FONT\_M**)**

height **=** draw\_text**(**'Fatal Error'**,** color**=**RED**,** height**=**height**)**

draw\_text**(**message**,** height**=**height**)**

# Goodbye cruel world. We can't exit because it would clear the screen.

**while** **True:**

time**.**sleep**(**1**)**

# Task to periodically update the IP to be displayed

**def** task\_update\_ip**():**

threadLocal**.**ip **=** socket**.**gethostbyname**(**socket**.**gethostname**())**

CLOCK**.**enter**(**10**,** 10**,** task\_update\_ip**)**

# To make the font objects updatable, but not waste resources

**def** task\_update\_font**():**

threadLocal**.**font\_day **=** pygame**.**font**.**SysFont**(**'notomono'**,** settings**[**'day'**][**'size'**])**

threadLocal**.**font\_clock **=** pygame**.**font**.**SysFont**(**'notomono'**,** settings**[**'clock'**][**'size'**])**

threadLocal**.**date\_clock **=** pygame**.**font**.**SysFont**(**'notomono'**,** settings**[**'date'**][**'size'**])**

# Task to do the background brightness, handles pulsing effect if required

**def** task\_update\_pwm**():**

**if** status**[**'streaming'**]:**

set\_brightness**(**100**)**

**else:**

bgt **=** settings**[**'brightness'**]**

bgt**[**'target'**]** **=** clamp**(**bgt**[**'target'**],** bgt**[**'min'**],** bgt**[**'max'**])**

# If we are drawing the brightness slider, give instant feedback

**if** status**[**'draw'**][**'option'**]** **==** Menu**.**Set\_Brightness**:**

set\_brightness**(**bgt**[**'preference'**])**

# If not drawing the brightness slider

**else:**

# If we are not at target level brightness

**if** bgt**[**'target'**]** **!=** bgt**[**'now'**]:**

# Deviate predetermined step size from the current brightness towards the

**if** bgt**[**'target'**]** **<** bgt**[**'now'**]:**

bgt**[**'now'**]** **=** max**(**bgt**[**'target'**],** bgt**[**'now'**]** **-** bgt**[**'step'**],** bgt**[**'min'**],** 0**)**

**else:**

bgt**[**'now'**]** **=** min**(**bgt**[**'target'**],** bgt**[**'now'**]** **+** bgt**[**'step'**],** bgt**[**'max'**],** 100**)**

set\_brightness**(**bgt**[**'now'**])**

# if we are at target brightness

**else:**

# if pulsing

**if** status**[**'pulsing'**]:**

**if** bgt**[**'target'**]** **<=** bgt**[**'min'**]:**

bgt**[**'target'**]** **=** bgt**[**'max'**]**

**else:**

bgt**[**'target'**]** **=** bgt**[**'min'**]**

# if not pulsing

**else:**

bgt**[**'target'**]** **=** bgt**[**'preference'**]**

CLOCK**.**enter**(**0.1**,** 2**,** task\_update\_pwm**)**

**def** truncate\_scroll\_text**(**string**,** length**=**30**):**

**if** len**(**string**)** **<=** length**:**

**return** string

td **=** int**(**datetime**.**datetime**.**now**().**timestamp**()** **\*** 2**)** **%** **(**len**(**string**)** **-** length **+** 1**)**

**return** string**[**td**:**td **+** length**]**

# Task that draws to the LCD

**def** task\_draw\_clock**():**

# To make adding code easy, leftover/unnecessary height assignments are left!

height **=** 0

# draw the main clock

**if** status**[**'draw'**][**'clock'**]:**

**if** settings**[**'day'**][**'enabled'**]:**

height **=** draw\_text**(**datetime**.**datetime**.**now**().**strftime**(**'%A'**),** height**=**height**,**

font**=**threadLocal**.**font\_day**)**

height **-=** **(**threadLocal**.**font\_day**.**get\_height**()** **\*** 0.1**)**

height **=** draw\_text**(**datetime**.**datetime**.**now**().**strftime**(**settings**[**'clock'**][**'format'**]),**

height**=**height**,** font**=**threadLocal**.**font\_clock**)**

height **-=** **(**threadLocal**.**font\_clock**.**get\_height**()** **\*** 0.1**)**

**if** settings**[**'date'**][**'enabled'**]:**

height **=** draw\_text**(**datetime**.**datetime**.**now**().**strftime**(**settings**[**'date'**][**'format'**]),**

height**=**height**,** font**=**threadLocal**.**date\_clock**)**

# if we need to register the devie with gcal

**if** 'user\_code' **in** status**[**'gcal'**]:**

height **=** draw\_text**(**status**[**'gcal'**][**'verification\_url'**],** height**=**height**,** font**=**FONT\_S**)**

height **=** draw\_text**(**'Code: ' **+** status**[**'gcal'**][**'user\_code'**],** height**=**height**,** font**=**FONT\_S**)**

# If the menu needs drawing

**if** status**[**'menu'**]:**

height **=** draw\_text**(**'Menu'**,** height**=**height**,** font**=**FONT\_S**)**

height **=** draw\_text**(**status**[**'menu'**].**value**[**'name'**],** height**=**height**,** font**=**FONT\_S**)**

# if we have a menu option selected

**elif** status**[**'draw'**][**'option'**]:**

**if** status**[**'draw'**][**'option'**]** **==** Menu**.**Show\_IP**:**

height **=** draw\_text**(**threadLocal**.**ip**,** height**=**height**,** font**=**FONT\_S**)**

# If the option is tied to a setting

**elif** status**[**'draw'**][**'option'**].**value**[**'setting'**]:**

setting **=** status**[**'draw'**][**'option'**].**value**[**'setting'**]**

current **=** settings

# Move down the list of keys: ('foo', 'bar') => settings['foo']['bar]

**for** key **in** setting**:**

current **=** current**[**key**]**

height **=** draw\_text**(**status**[**'draw'**][**'option'**].**value**[**'name'**]** **+** ': %d%%' **%** current**,**

height**=**height**,** font**=**FONT\_S**)**

# Draw the rectangle below the text and % value

pygame**.**draw**.**rect**(**SCREEN**,** WHITE**,** **(**0**,** height**,** SIZE**[**0**]** **\*** **(**current **/** 100**),** 5**))**

# Need to move 5 px down manually

height **+=** 5

**else:**

**if** 'alarm' **in** status**[**'draw'**]:**

draw\_text**(**'\uf0a1'**,** height**=**height**,** font**=**FONT\_ICO**,** center**=False)**

height **=** draw\_text**(**status**[**'draw'**][**'alarm'**],** height**=**height**,** font**=**FONT\_S**)**

**if** 'next' **in** status**[**'draw'**]:**

draw\_text**(**'\uf133'**,** height**=**height**,** font**=**FONT\_ICO**,** center**=False)**

height **=** draw\_text**(**truncate\_scroll\_text**(**status**[**'draw'**][**'next'**]),** height**=**height**,**

font**=**FONT\_S**)**

**if** 'title' **in** status**[**'draw'**]:**

draw\_text**(**'\uf001'**,** height**=**height**,** font**=**FONT\_ICO**,** center**=False)**

height **=** draw\_text**(**truncate\_scroll\_text**(**status**[**'draw'**][**'title'**]),** height**=**height**,**

font**=**FONT\_S**)**

# Actually commit the LCD

pygame**.**display**.**update**()**

CLOCK**.**enter**(**0.5**,** 1**,** task\_draw\_clock**)**

**def** task\_check\_gcal**():**

**print(**'Checking gcal (%s)' **%** datetime**.**datetime**.**now**())**

gcal\_get\_events**()**

CLOCK**.**enter**(**60 **\*** 30**,** 2**,** task\_check\_gcal**)**

# Helper method, to run the task once manually before passing it off to the scheduler

**def** run\_clock\_thread**():**

# We are now out of booting

status**[**'booting'**]** **=** **False**

task\_update\_font**()**

task\_update\_ip**()**

task\_update\_pwm**()**

task\_draw\_clock**()**

task\_check\_gcal**()**

task\_alarm\_check**()**

CLOCK**.**run**()**

**def** task\_alarm\_check**():**

**if** 'alarm' **in** status**:**

**print(**'Alarm poll'**)**

ts\_alarm **=** status**[**'alarm'**].**replace**(**second**=**0**,** microsecond**=**0**).**timestamp**()**

ts\_now **=** datetime**.**datetime**.**now**().**replace**(**second**=**0**,** microsecond**=**0**).**timestamp**()**

**if** int**(**ts\_alarm **-** ts\_now**)** **==** 600**:**

**print(**'10 minute mark'**)**

status**[**'pulsing'**]** **=** **True**

**if** int**(**ts\_alarm **-** ts\_now**)** **==** 0**:**

**print(**'alarm time'**)**

**if** status**[**'skipped'**]:**

status**[**'skipped'**]** **=** **False**

**del** status**[**'alarm'**]**

next\_event**()**

**else:**

stream\_start**()**

**if** 'alarm' **in** status**[**'draw'**]:**

**del** status**[**'draw'**][**'alarm'**]**

CLOCK**.**enter**(**30**,** 1**,** task\_alarm\_check**)**

# Try to connect to the wifi network set via settings['wifiProfile']

**def** attempt\_connect**(**height**=**0**):**

# Just to be sure

subprocess**.**call**([**'killall'**,** 'hostapd'**])**

subprocess**.**call**([**'create\_ap'**,** '--stop'**,** 'wlan0'**])**

height **=** draw\_text**(**'Connecting to %s' **%** settings**[**'wifiProfile'**],** height**=**height**)**

# Use switch to make sure no other network is using wlan0

**if** subprocess**.**call**([**'netctl'**,** 'switch-to'**,** settings**[**'wifiProfile'**]])** **==** 0**:**

# Congratulations, we have liftoff

status**[**'network'**]** **=** **True**

# Draw the IP in screen to make accessing the web interface easy

ip **=** socket**.**gethostbyname**(**socket**.**gethostname**())**

height **=** draw\_text**(**ip**,** height**=**height**,** font**=**FONT\_M**)**

height **=** draw\_text**(**'Syncing time & date'**,** height**=**height**)**

# Use NTP to get internet time

subprocess**.**call**([**'systemctl'**,** 'restart'**,** 'ntpd'**])**

# Make sure we actually have NTP time

**if** subprocess**.**call**([**'ntp-wait'**,** '-n'**,** '5'**])** **!=** 0**:**

error**(**'NTP sync failed.'**)**

**else:**

# Write to RTC

subprocess**.**call**([**'hwclock'**,** '-w'**])**

# Now we can start rendering the clock

status**[**'draw'**][**'clock'**]** **=** **True**

# Connecting to wifi failed

**else:**

height **=** draw\_text**(**'Failed...'**,** height**=**height**)**

status**[**'network'**]** **=** **False**

**return** height

# Refresh our token

**def** gcal\_refresh**():**

**if** **not** status**[**'network'**]** **or** 'refresh\_token' **not** **in** settings**[**'gcal'**]:**

**return**

out **=** json**.**loads**(**requests**.**post**(**'https://www.googleapis.com/oauth2/v4/token'**,**

data**={**'refresh\_token'**:** settings**[**'gcal'**][**'refresh\_token'**],**

'client\_id'**:** gcal**[**'client\_id'**],**

'client\_secret'**:** gcal**[**'client\_secret'**],**

'grant\_type'**:** 'refresh\_token'**}).**text**)**

status**[**'gcal'**]** **=** out

**if** 'expires\_in' **in** out**:**

status**[**'gcal'**][**'expires'**]** **=** datetime**.**datetime**.**now**()** **+** \

datetime**.**timedelta**(**seconds**=**out**[**'expires\_in'**]** **-** 10**)**

# Poll with device token to see if user has granted us permission yet

**def** gcal\_poll**():**

out **=** json**.**loads**(**requests**.**post**(**'https://www.googleapis.com/oauth2/v4/token'**,**

data**={**'client\_id'**:** gcal**[**'client\_id'**],**

'client\_secret'**:** gcal**[**'client\_secret'**],**

'code'**:** status**[**'gcal'**][**'device\_code'**],**

'grant\_type'**:**

'http://oauth.net/grant\_type/device/1.0'**}).**text**)**

# Some information, 'authorization\_pending' is normal

**if** "error" **in** out**:**

**print(**out**[**'error'**])**

# No error = good

**else:**

status**[**'gcal'**]** **=** out

# 10 sec for safety

status**[**'gcal'**][**'expires'**]** **=** datetime**.**datetime**.**now**()** **+** \

datetime**.**timedelta**(**seconds**=**out**[**'expires\_in'**]** **-** 10**)**

# Store the token, and make the primary calendar default

settings**[**'gcal'**]** **=** **{**'refresh\_token'**:** out**[**'refresh\_token'**],** 'calendar\_id'**:** 'primary'**}**

# Save the new settings

save**()**

# Fetch the upcoming events

gcal\_get\_events**()**

# to prevent re-registering

**return**

# if the token request is expired, drop it

**if** datetime**.**datetime**.**now**()** **>** status**[**'gcal'**][**'expires'**]:**

**print(**'Token request expired. (%s)' **%** datetime**.**datetime**.**now**())**

status**[**'gcal'**]** **=** **{}**

# Poll at the rate requested by Google

**else:**

CLOCK**.**enter**(**status**[**'gcal'**][**'interval'**],** 3**,** gcal\_poll**)**

# Do a new token request, overrides all old gcal data, except for appointments

**def** gcal\_request\_token**():**

out **=** json**.**loads**(**requests**.**post**(**'https://accounts.google.com/o/oauth2/device/code'**,**

data**={**'client\_id'**:** gcal**[**'client\_id'**],**

'scope'**:** gcal**[**'scope'**]}).**text**)**

status**[**'gcal'**]** **=** **{}**

status**[**'gcal'**][**'device\_code'**]** **=** out**[**'device\_code'**]**

status**[**'gcal'**][**'interval'**]** **=** out**[**'interval'**]**

status**[**'gcal'**][**'user\_code'**]** **=** out**[**'user\_code'**]**

status**[**'gcal'**][**'verification\_url'**]** **=** out**[**'verification\_url'**]**

# 10 sec for safety

status**[**'gcal'**][**'expires'**]** **=** datetime**.**datetime**.**now**()** **+** \

datetime**.**timedelta**(**seconds**=**out**[**'expires\_in'**]** **-** 10**)**

**if** 'gcal' **in** settings**:**

**del** settings**[**'gcal'**]**

save**()**

CLOCK**.**enter**(**out**[**'interval'**],** 3**,** gcal\_poll**)**

# Pull in new events, if token is expired / missing it will request a new one.

**def** gcal\_get\_events**():**

**if** **not** status**[**'network'**]** **or** 'gcal' **not** **in** settings**:**

**return**

**if** 'calendar\_id' **not** **in** status**[**'gcal'**]** **or** \

datetime**.**datetime**.**now**()** **>** status**[**'gcal'**][**'expires'**]:**

gcal\_refresh**()**

**if** 'access\_token' **not** **in** status**[**'gcal'**]:**

# Something exploded.

**return**

out **=** json**.**loads**(**requests**.**get**(**

'https://www.googleapis.com/calendar/v3/calendars/%s/events' **%** settings**[**'gcal'**][**

'calendar\_id'**],** headers**={**

'Authorization'**:** status**[**'gcal'**][**'token\_type'**]** **+** ' ' **+** status**[**'gcal'**][**

'access\_token'**]},** params**={**

'timeMin'**:** datetime**.**datetime**.**now**(**datetime**.**timezone**.**utc**).**astimezone**().**isoformat**(**'T'**),**

'timeMax'**:** **(**datetime**.**datetime**.**now**(**datetime**.**timezone**.**utc**)** **+** datetime**.**timedelta**(**

days**=**7**)).**astimezone**().**isoformat**(**'T'**),** 'singleEvents'**:** **True,**

'orderBy'**:** 'startTime'**}).**text**)**

**if** 'items' **not** **in** out**:**

**print(**'No items in response gcal\_get\_events'**)**

**print(**out**)**

status**[**'items'**]** **=** **None**

**return** **False**

status**[**'items'**]** **=** out**[**'items'**]**

next\_event**()**

**def** next\_event**():**

**for** item **in** status**[**'items'**]:**

alarm **=** dateutil**.**parser**.**parse**(**item**[**'start'**][**'dateTime'**])**

status**[**'draw'**][**'next'**]** **=** item**[**'summary'**]** **+** ' ' **+** \

dateutil**.**parser**.**parse**(**item**[**'start'**][**'dateTime'**]).**strftime**(**'%a at %H:%M'**)**

**if** alarm**.**isoweekday**()** **not** **in** settings**[**'alarm'**][**'days'**].**value**:**

**print(**'Event not in target days, skipping. %s' **%** item**[**'summary'**])**

**continue**

alarm **-=** datetime**.**timedelta**(**minutes**=**settings**[**'alarm'**][**'offset'**])**

**if** settings**[**'alarm'**][**'min'**]** **!=** **-**1 **and** \

**(**alarm**.**hour **\*** 60**)** **+** alarm**.**minute **<** settings**[**'alarm'**][**'min'**]:**

alarm **=** alarm**.**replace**(**hour**=**settings**[**'alarm'**][**'min'**]** **//** 60**,**

minute**=**settings**[**'alarm'**][**'min'**]** **%** 60**)**

**if** settings**[**'alarm'**][**'max'**]** **!=** **-**1 **and** \

**(**alarm**.**hour **\*** 60**)** **+** alarm**.**minute **>** settings**[**'alarm'**][**'max'**]:**

alarm **=** alarm**.**replace**(**hour**=**settings**[**'alarm'**][**'max'**]** **//** 60**,**

minute**=**settings**[**'alarm'**][**'max'**]** **%** 60**)**

**if** alarm**.**timestamp**()** **<** datetime**.**datetime**.**now**().**timestamp**():**

**print(**'Alarm time passed, skipping. %s' **%** item**[**'summary'**])**

**continue**

status**[**'alarm'**]** **=** alarm

status**[**'draw'**][**'alarm'**]** **=** alarm**.**strftime**(**'%a at %H:%M'**)**

**return**

# Only get here if no (correct) items

**if** 'alarm' **in** status**:**

**del** status**[**'alarm'**]**

**if** 'alarm' **in** status**[**'draw'**]:**

**del** status**[**'draw'**][**'alarm'**]**

# ############################## Sequential code

# Network adapter not plugged in => panic

**if** **not** os**.**path**.**exists**(**'/sys/class/net/wlan0'**):**

error**(**'No wifi interface'**)**

# For the IP & fonts

threadLocal **=** threading**.**local**()**

# Make the scheguler

CLOCK **=** sched**.**scheduler**(**time**.**time**,** time**.**sleep**)**

# Start the 'main' thread

threading**.**Thread**(**target**=**run\_clock\_thread**,** name**=**'ClockThread'**,** daemon**=True).**start**()**

# Draw name & version

h **=** draw\_text**(**'SmartClock (%s)' **%** VERSION**,** font**=**FONT\_M**)**

# If we have a wifi network saved, try to connect

**if** 'wifiProfile' **in** settings **and** settings**[**'wifiProfile'**]** **!=** ''**:**

h **=** attempt\_connect**(**h**)**

# if we have connected

**if** status**[**'network'**]:**

**if** 'gcal' **in** settings**:**

# if we have some gcal stuff saved, update the event list

gcal\_get\_events**()**

**else:**

# Start the token procedure

gcal\_request\_token**()**

# If we don't have network

**else:**

# Pull in the clock/date from the RTC, if it got reset, it'll be at 1 jan 2000

subprocess**.**call**([**'hwclock'**,** '-s'**])**

h **=** draw\_text**(**'Starting AP'**,** height**=**h**)**

subprocess**.**call**([**'create\_ap'**,** '-n'**,** '--daemon'**,** '--redirect-to-localhost'**,** 'wlan0'**,**

'SmartAlarmClock'**])**

# Give the adapter some time

time**.**sleep**(**5**)**

h **=** draw\_text**(**'Connect to wifi network for setup:'**,** height**=**h**)**

h **=** draw\_text**(**'SmartAlarmClock'**,** height**=**h**,** font**=**FONT\_M**)**

h **=** draw\_text**(**'And browse to:'**,** height**=**h**)**

# Display IP

h **=** draw\_text**(**socket**.**gethostbyname**(**socket**.**gethostname**()),** height**=**h**,** font**=**FONT\_M**)**

# ####################################### BOOT SEQ, PART 5 - Flask

**print(**'Boot sequence part 5 - Flask (%s)' **%** datetime**.**datetime**.**now**())**

# ############################## Imports

**from** flask **import** Flask

**from** flask **import** json

**from** flask **import** Response

**from** flask **import** request

**from** flask **import** url\_for

# ############################## Definitions

app **=** Flask**(**\_\_name\_\_**)**

# List of possible access points

@app.route**(**'/'**)**

**def** api**():**

output **=** **[]**

**for** rule **in** app**.**url\_map**.**iter\_rules**():**

options **=** **{}**

**for** arg **in** rule**.**arguments**:**

options**[**arg**]** **=** '[%s]' **%** arg

output**.**append**({**'name'**:** rule**.**endpoint**,** 'methods'**:** ','**.**join**(**rule**.**methods**),**

'url'**:** '/api' **+** urllib**.**parse**.**unquote**(**

url\_for**(**rule**.**endpoint**,** **\*\***options**))})**

**return** Response**(**json**.**dumps**(**output**,** cls**=**EnumEncoder**),** mimetype**=**'text/javascript'**)**

# Get the list of wifi networks OR set the wifi profile settings (ssid & pass) (via POST)

@app.route**(**'/wifi'**,** methods**=[**'GET'**,** 'POST'**])**

**def** api\_wifi**():**

# Set wifi settings by making a profile file for it, so we can (ab)use netctl

**if** request**.**method **==** 'POST'**:**

# stop drawing the clock for a second

status**[**'draw'**][**'clock'**]** **=** **False**

text **=** "Description='Automatically generated profile by python'\n"

text **+=** 'Interface=wlan0\n'

text **+=** 'Connection=wireless\n'

text **+=** 'IP=dhcp\n'

text **+=** "ESSID='%s'\n" **%** request**.**form**[**'ssid'**]**

**if** 'pass' **in** request**.**form **and** request**.**form**[**'pass'**]** **!=** ''**:**

text **+=** "Security=wpa\nKey='%s'" **%** request**.**form**[**'pass'**]**

**else:**

text **+=** 'Security=none'

settings**[**'wifiProfile'**]** **=** 'GEN-wlan0-%s' **%** request**.**form**[**'ssid'**]**

# Write the file

f **=** open**(**'/etc/netctl/%s' **%** settings**[**'wifiProfile'**],** 'w'**)**

f**.**write**(**text**)**

f**.**close**()**

save**()**

attempt\_connect**()**

**if** 'gcal' **in** settings**:**

# if we have some gcal stuff saved, update the event list

gcal\_get\_events**()**

**else:**

# Start the token procedure

gcal\_request\_token**()**

# Return a readable value

**if** status**[**'network'**]:**

**return** 'OK'

**return** 'ERROR'

# GET

**else:**

# Regex yey

re\_cell **=** re**.**compile**(**r'Cell \d+'**)**

re\_mac **=** re**.**compile**(**r'Address: (?P<Address>.\*)'**)**

re\_ssid **=** re**.**compile**(**r'ESSID:"(?P<SSID>.\*)"'**)**

re\_quality **=** re**.**compile**(**r'Quality=(?P<Quality>\d+)/100'**)**

re\_signal **=** re**.**compile**(**r'Signal level=(?P<SignalLevel>\d+)/100'**)**

re\_encrypted **=** re**.**compile**(**r'Encryption key:(?P<Protected>on|off)'**)**

re\_encryption **=** re**.**compile**(**r'IE: (?P<Protection>.\*)'**)**

re\_authentication **=** re**.**compile**(**

r'Authentication Suites \(1\) : (?P<Authentication>.\*)'**)**

proc **=** subprocess**.**Popen**([**'iwlist'**,** 'wlan0'**,** 'scan'**],** stdout**=**subprocess**.**PIPE**,**

universal\_newlines**=True)**

out**,** err **=** proc**.**communicate**()**

# data array

data **=** **[]**

# Used to remeber the last object we are working on

cell **=** **None**

**for** line **in** out**.**split**(**'\n'**):**

# strip whitespace

line **=** line**.**strip**()**

matcher **=** re\_cell**.**search**(**line**)**

# if this is the start of a new network

**if** matcher**:**

# if cell wasn't None, add it to the data list

**if** cell**:**

data**.**append**(**cell**)**

cell **=** **{}**

**for** regex **in** **[**re\_mac**,** re\_ssid**,** re\_quality**,** re\_signal**,** re\_encrypted**,** re\_encryption**,**

re\_authentication**]:**

matcher **=** regex**.**search**(**line**)**

# if this regex matched

**if** matcher**:**

# add the match to the list of properties in the dict

cell**.**update**(**matcher**.**groupdict**())**

**return** Response**(**json**.**dumps**(**data**,** cls**=**EnumEncoder**),** mimetype**=**'text/javascript'**)**

# GET: Dump the settings POST: Set settings

@app.route**(**'/settings'**,** methods**=[**'GET'**,** 'POST'**])**

**def** api\_settings**():**

**if** request**.**method **==** 'GET'**:**

**return** Response**(**json**.**dumps**(**settings**,** cls**=**EnumEncoder**),** mimetype**=**'text/javascript'**)**

**else:**

settings**.**update**(**request**.**get\_json**())**

settings**[**'alarm'**][**'days'**]** **=** as\_enum**(**settings**[**'alarm'**][**'days'**])**

CLOCK**.**enter**(**0**,** 1**,** task\_update\_font**)**

set\_volume**()**

next\_event**()**

save**()**

**return** 'OK'

# Reset (and restart) the gcal linking process

@app.route**(**'/pollgcal'**,** methods**=[**'POST'**])**

**def** api\_pollgcal**():**

gcal\_get\_events**()**

**return** 'OK'

# Reset (and restart) the gcal linking process

@app.route**(**'/resetgcal'**,** methods**=[**'POST'**])**

**def** api\_resetgcal**():**

gcal\_request\_token**()**

**return** 'OK'

# GET: Dump the status POST: Set status

@app.route**(**'/status'**,** methods**=[**'GET'**,** 'POST'**])**

**def** api\_status**():**

**if** request**.**method **==** 'GET'**:**

**return** Response**(**json**.**dumps**(**status**,** cls**=**EnumEncoder**),** mimetype**=**'text/javascript'**)**

**else:**

status**.**update**(**request**.**get\_json**())**

status**[**'draw'**][**'option'**]** **=** as\_enum**(**status**[**'draw'**][**'option'**])**

status**[**'menu'**]** **=** as\_enum**(**status**[**'menu'**])**

save**()**

**return** 'OK'

# ############################## Sequential code

**print(**'Starting webserver... (%s)' **%** datetime**.**datetime**.**now**())**

# Blocking call!

app**.**run**(**host**=**'127.0.0.1'**,** port**=**5000**,** use\_reloader**=False)**

**print(**'EXIT: Flask died (%s)' **%** datetime**.**datetime**.**now**())**

pygame**.**quit**()**

GPIO**.**cleanup**()**

## Web interface

### Index.html

<!doctype html>

<html lang=**"en"**>

<head>

<meta charset=**"UTF-8"**>

<meta http-equiv=**"X-UA-Compatible"** content=**"IE=Edge"**/>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1, shrink-to-fit=no"**>

<meta name=**"description"** content=**"SmartAlarmClock Web-interface"**>

<meta name=**"author"** content=**"Dries K. & Michiel B."**>

<link rel=**"stylesheet"** href=**"css/bootstrap.min.css"**>

<link rel=**"stylesheet"** href=**"css/font-awesome.css"**>

<style rel=**"stylesheet"**>

**.container {**

**max-width: 730px;**

**}**

**h1 {**

**text-align: center;**

**}**

**.tab-pane {**

**padding-top: 1em;**

**}**

**a.disabled {**

**cursor: not-allowed;**

**}**

**.pad-bottom {**

**padding-bottom: 1rem;**

**}**

</style>

<title>**SmartClock**</title>

</head>

<body>

<div class=**"container"**>

<h1>**SmartClock**</h1>

<div>

<!-- Nav tabs -->

<ul class=**"nav nav-tabs"** role=**"tablist"**>

<li role=**"presentation"** class=**"active"**><a href=**"#status"** role=**"tab"** data-toggle=**"tab"**>**Status**</a></li>

<li role=**"presentation"**><a href=**"#wifi"** role=**"tab"** data-toggle=**"tab"**>**WiFi Settings**</a></li>

<li role=**"presentation"**><a href=**"#clock"** role=**"tab"** data-toggle=**"tab"**>**Clock Settings**</a></li>

<li role=**"presentation"**><a href=**"#gcal"** role=**"tab"** data-toggle=**"tab"**>**Google Calender**</a></li>

</ul>

<!-- Tab panes -->

<div class=**"tab-content"**>

<!-- Status panel -->

<div role=**"tabpanel"** class=**"tab-pane fade in active"** id=**"status"**>

<p>**Welcome to the SmartAlarmClock web interface. You can go to the right tab once connection has been made with the backend.**</p>

<p>**Some operations (like switching wifi networks) may require you to reconnect to the right IP address (displayed on the LCD module of the clock).**</p>

</div>

<!-- Wifi panel -->

<div role=**"tabpanel"** class=**"tab-pane fade"** id=**"wifi"**>

<form class=**"form-horizontal"** id=**"wifi-form"**>

<div class=**"form-group"**>

<label for=**"wifi-dropdown"** class=**"col-sm-3 control-label"**>**Select network**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<select class=**"form-control"** id=**"wifi-dropdown"**>

<option value=**"-1"** disabled>**Scanning...**</option>

</select>

<span class=**"input-group-btn"**>

<button id=**"wifi-reload"** type=**"button"** class=**"btn btn-default"** disabled>**Scan again**</button>

</span>

</div>

</div>

</div>

<div class=**"form-group"**>

<label for=**"wifi-ssid"** class=**"col-sm-3 control-label"**>**Wifi SSID**</label>

<div class=**"col-sm-9"**>

<input type=**"text"** id=**"wifi-ssid"** class=**"form-control"** placeholder=**"SSID"** readonly>

</div>

</div>

<div class=**"form-group"**>

<label for=**"wifi-pass"** class=**"col-sm-3 control-label"**>**Wifi Password**</label>

<div class=**"col-sm-9"**>

<input type=**"password"** class=**"form-control"** id=**"wifi-pass"** placeholder=**"Wifi Password, Leave blank for open network."** readonly>

</div>

</div>

<button type=**"submit"** class=**"btn btn-block btn-success"**>**Connect**</button>

</form>

</div>

<!-- Clock panel -->

<div role=**"tabpanel"** class=**"tab-pane fade"** id=**"clock"**>

<form class=**"form-horizontal"** id=**"clock-form"**>

<div class=**"form-group"**>

<label for=**"clock-weekday-enable"** class=**"col-sm-3 control-label"**>**Weekday**</label>

<div class=**"col-sm-9"**>

<button id=**"clock-weekday-enable"** type=**"button"** class=**"btn btn-block btn-success"**>**Enabled**</button>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-weekday-size"** class=**"col-sm-3 control-label"**>**Weekday font size**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<input id=**"clock-weekday-size"** class=**"form-control"** type=**"number"** name=**"clock-weekday-enable"** min=**"10"** max=**"50"** value=**"-1"**>

<span class=**"input-group-addon"**>**10**&nbsp;<i class=**"fa fa-long-arrow-right"** aria-hidden=**"true"**></i>&nbsp;**50**</span>

</div>

</div>

</div>

<hr/><!-- ----------------------------------------------------- -->

<div class=**"form-group"**>

<label for=**"clock-format1"** class=**"col-sm-3 control-label"**>**Clock format**</label>

<div class=**"col-sm-9"**>

<select class=**"form-control"** id=**"clock-format1"**>

<option value=**"%I:%M:%S"**>**05:20:30 (12h clock)**</option>

<option value=**"%I:%M"**>**05:20 (12h clock)**</option>

<option value=**"%I:%M:%S %p"**>**05:20:30 PM (12h clock)**</option>

<option value=**"%I:%M %p"**>**05:20 PM (12h clock)**</option>

<option value=**"%H:%M:%S"**>**17:20:30 (24h clock)**</option>

<option value=**"%H:%M"**>**17:20 (24h clock)**</option>

</select>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-size1"** class=**"col-sm-3 control-label"**>**Clock font size**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<input id=**"clock-size1"** class=**"form-control"** type=**"number"** name=**"clock-size1"** min=**"10"** max=**"80"** value=**"-1"**>

<span class=**"input-group-addon"**>**10**&nbsp;<i class=**"fa fa-long-arrow-right"** aria-hidden=**"true"**></i>&nbsp;**80**</span>

</div>

</div>

</div>

<hr/><!-- ----------------------------------------------------- -->

<div class=**"form-group"**>

<label for=**"clock-format2-enable"** class=**"col-sm-3 control-label"**>**Date**</label>

<div class=**"col-sm-9"**>

<button id=**"clock-format2-enable"** type=**"button"** class=**"btn btn-block btn-success"**>**Enabled**</button>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-format2"** class=**"col-sm-3 control-label"**>**Date format**</label>

<div class=**"col-sm-9"**>

<select class=**"form-control"** id=**"clock-format2"**>

<option value=**"%d/%m/%y"**>**15/05/97**</option>

<option value=**"%m/%d/%y"**>**05/15/97**</option>

<option value=**"%d-%m-%y"**>**15-05-97**</option>

<option value=**"%m-%d-%y"**>**05-15-97**</option>

<option value=**"%d/%m/%Y"**>**15/05/1997**</option>

<option value=**"%m/%d/%Y"**>**05/15/1997**</option>

<option value=**"%d-%m-%Y"**>**15-05-1997**</option>

<option value=**"%m-%d-%Y"**>**05-15-1997**</option>

<option value=**"%d %b %Y"**>**15 may 1997**</option>

<option value=**"%b %d %Y"**>**may 15 1997**</option>

</select>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-size2"** class=**"col-sm-3 control-label"**>**Date font size**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<input id=**"clock-size2"** class=**"form-control"** type=**"number"** name=**"clock-size2"** min=**"10"** max=**"80"** value=**"-1"**>

<span class=**"input-group-addon"**>**10**&nbsp;<i class=**"fa fa-long-arrow-right"** aria-hidden=**"true"**></i>&nbsp;**80**</span>

</div>

</div>

</div>

<hr/><!-- ----------------------------------------------------- -->

<div class=**"form-group"**>

<label for=**"clock-offset"** class=**"col-sm-3 control-label"** title=**"How many minutes before an appointment the alarm should go off."**>**Precursor**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<input id=**"clock-offset"** class=**"form-control"** type=**"number"** name=**"clock-offset"** min=**"0"** max=**"240"** value=**"-1"**>

<span class=**"input-group-addon"**>**10**&nbsp;<i class=**"fa fa-long-arrow-right"** aria-hidden=**"true"**></i>&nbsp;**240 min**</span>

</div>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-min"** class=**"col-sm-3 control-label"**>**Earliest wake up time**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<div class=**"input-group-btn"**>

<button id=**"clock-min-enable"** type=**"button"** class=**"btn btn-block btn-success"**>**Enabled**</button>

</div>

<input id=**"clock-min"** class=**"form-control"** name=**"clock-min"** type=**"time"** value=**"-1"**>

</div>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-max"** class=**"col-sm-3 control-label"**>**Latest wake up time**</label>

<div class=**"col-sm-9"**>

<div class=**"input-group"**>

<div class=**"input-group-btn"**>

<button id=**"clock-max-enable"** type=**"button"** class=**"btn btn-block btn-success"**>**Enabled**</button>

</div>

<input id=**"clock-max"** class=**"form-control"** name=**"clock-max"** type=**"time"** value=**"-1"**>

</div>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-days"** class=**"col-sm-3 control-label"**>**Alarm on**</label>

<div class=**"col-sm-9"**>

<label class=**"radio-inline"**>

<input id=**"clock-days-1"** name=**"clock-days"** type=**"radio"** value=**"Days.Weekdays"**> **Weekdays**

</label>

<label class=**"radio-inline"**>

<input id=**"clock-days-2"** name=**"clock-days"** type=**"radio"** value=**"Days.Weekends"**> **Weekends**

</label>

<label class=**"radio-inline"**>

<input id=**"clock-days-3"** name=**"clock-days"** type=**"radio"** value=**"Days.Both"**> **Both**

</label>

</div>

</div>

<div class=**"form-group"**>

<label for=**"clock-stream"** class=**"col-sm-3 control-label"**>**Stream source**</label>

<div class=**"col-sm-9"**>

<select class=**"form-control"** id=**"clock-stream"**>

<option value=**"MNM"**>**MNM**</option>

<option value=**"MNM Hits"**>**MNM Hits**</option>

<option value=**"Studio Brussel"**>**Studio Brussel**</option>

<option value=**"Klara"**>**Klara**</option>

<option value=**"Radio 1"**>**Radio 1**</option>

<option value=**"Radio 2 Antwerpen"**>**Radio 2 Antwerpen**</option>

</select>

</div>

</div>

<button type=**"submit"** class=**"btn btn-block btn-success"**>**Save**</button>

</form>

</div>

<!-- Google Calender -->

<div role=**"tabpanel"** class=**"tab-pane fade"** id=**"gcal"**>

<div id=**"gcal-link"** class=**"pad-bottom"**>

<p>**Please link your account by going to this url and entering the device code.**</p>

<a id=**"gcal-link-url"** target=**"\_blank"** href=**"https://www.google.com/device"**>**https://www.google.com/device**</a>

<p>**Device code:** <code id=**"gcal-link-code"**>**WWWWWWWWWWWWWWW**</code></p>

</div>

<div id=**"gcal-list"** class=**"pad-bottom"**>

<form id=**"gcal-list-form"** class=**"form-horizontal pad-bottom"**>

<div class=**"form-group"**>

<label for=**"gcal-list-select"** class=**"col-sm-3 control-label"**>**Select calendar**</label>

<div class=**"col-sm-9"**>

<select class=**"form-control"** id=**"gcal-list-select"**>

<option value=**"primary"**>**Primary Calender**</option>

</select>

</div>

</div>

<button type=**"submit"** class=**"btn btn-block btn-success"**>**Save**</button>

</form>

<p>**Next appointments:**</p>

<ul id=**"gcal-list-items"**>

</ul>

<form id=**"gcal-poll-form"** class=**"form-horizontal"**>

<button class=**"btn btn-block btn-warning"**>**Force event synchronisation**</button>

</form>

</div>

<div id=**"gcal-reset"** class=**"pad-bottom"**>

<p>**You can reset your link with google calendar, or re-initialize the linking procedure if it expired the first time.**</p>

<form id=**"gcal-reset-form"** class=**"form-horizontal"**>

<button class=**"btn btn-block btn-danger"**>**Reset link**</button>

</form>

</div>

</div>

</div>

</div>

</div>

<div class=**"modal fade"** tabindex=**"-1"** role=**"dialog"** id=**"modal"**>

<div class=**"modal-dialog modal-sm"**>

<div class=**"modal-content"**>

<div class=**"modal-header"**><h4 class=**"modal-title"**>**Placeholder**</h4></div>

<div class=**"modal-body"**>**Placeholder**</div>

</div>

</div>

</div>

<script src=**"js/jquery.js"** type=**"application/javascript"**></script>

<script src=**"js/bootstrap.js"** type=**"application/javascript"**></script>

<script src=**"js/custom.js"** type=**"application/javascript"**></script>

</body>

</html>

### custom.js

"use strict"**;**

**const** TAB\_STATUS **=** $**(**'a[data-toggle="tab"][href="#status"]'**);**

**const** TAB\_WIFI **=** $**(**'a[data-toggle="tab"][href="#wifi"]'**);**

**const** TAB\_CLOCK **=** $**(**'a[data-toggle="tab"][href="#clock"]'**);**

**const** TAB\_GCAL **=** $**(**'a[data-toggle="tab"][href="#gcal"]'**);**

**const** MODAL **=** $**(**'#modal'**);**

**const** GCAL\_RESET **=** $**(**'#gcal-reset'**);**

**const** GCAL\_LINK **=** $**(**'#gcal-link'**);**

**const** GCAL\_LIST **=** $**(**'#gcal-list'**);**

**const** GCAL\_LIST\_FORM **=** $**(**'#gcal-list-form'**);**

**const** GCAL\_POLL\_FORM **=** $**(**'#gcal-poll-form'**);**

**const** GCAL\_RESET\_FORM **=** $**(**'#gcal-reset-form'**);**

**const** GCAL\_LIST\_ITEMS **=** $**(**'#gcal-list-items'**);**

**var** settings\_timeout **=** null**;**

**var** status\_data **=** null**;**

**var** settings\_data **=** null**;**

**var** wifi\_data **=** null**;**

**function** pad**(**n**,** width**,** z**)** **{**

z **=** z **||** '0'**;**

n **=** n **+** ''**;**

**return** n**.**length **>=** width **?** n **:** **new** Array**(**width **-** n**.**length **+** 1**).**join**(**z**)** **+** n**;**

**}**

**function** minToTime**(**min**)** **{**

**return** pad**(**min **/** 60**,** 2**)** **+** ':' **+** pad**(**min **%** 60**,** 2**);**

**}**

**function** timeToMin**(**time**)** **{**

**var** split **=** time**.**split**(**':'**);**

**return** **(parseInt(**split**[**0**])** **\*** 60**)** **+** **parseInt(**split**[**1**]);**

**}**

**function** disableTabs**()** **{**

**[**TAB\_STATUS**,** TAB\_WIFI**,** TAB\_CLOCK**,** TAB\_GCAL**].**forEach**(function** **(**e**)** **{**

e**.**prop**(**'disabled'**,** **true).**addClass**(**'disabled'**)**

**});**

**}**

**function** enableTabs**()** **{**

**[**TAB\_STATUS**,** TAB\_WIFI**,** TAB\_CLOCK**,** TAB\_GCAL**].**forEach**(function** **(**e**)** **{**

e**.**prop**(**'disabled'**,** **false).**removeClass**(**'disabled'**)**

**});**

**}**

**function** showModal**(**message**,** title**)** **{**

disableTabs**();**

MODAL**.**find**(**'.modal-body'**).**html**(**message**);**

MODAL**.**find**(**'.modal-title'**).**html**(**title**);**

MODAL**.**modal**({**backdrop**:** 'static'**,** keyboard**:** **false,** show**:** **true});**

**}**

**function** loadStatus**()** **{**

disableTabs**();**

$**.**getJSON**(**'/api/status'**,** **function** **(**data**)** **{**

console**.**log**(**'Update status data.'**);**

status\_data **=** data**;**

**if** **(**data**[**'booting'**])** **return;**

MODAL**.**modal**(**'hide'**);**

enableTabs**();**

loadSettings**();**

**clearTimeout(**showTimeout**);**

**if** **(!**data**[**'network'**])** TAB\_WIFI**.**tab**(**'show'**);**

**else** **if** **(**data**[**'gcal'**][**'user\_code'**])** **{**

GCAL\_LINK**.**show**();**

GCAL\_LIST**.**hide**();**

GCAL\_RESET**.**hide**();**

$**(**'#gcal-link-url'**).text(**data**[**'gcal'**][**'verification\_url'**]).**attr**(**'href'**,** data**[**'gcal'**][**'verification\_url'**]);**

$**(**'#gcal-link-code'**).text(**data**[**'gcal'**][**'user\_code'**]);**

TAB\_GCAL**.**tab**(**'show'**);**

**setTimeout(**loadStatus**,** 2000**);**

**}**

**else** **if** **(**data**[**'gcal'**][**'access\_token'**])** **{**

GCAL\_LIST**.**show**();**

GCAL\_LINK**.**hide**();**

GCAL\_RESET**.**show**();**

**}**

**else** **if** **(**data**[**'gcal'**])** **{**

GCAL\_RESET**.**show**();**

**}**

**if** **(**data**[**'items'**])** **{**

GCAL\_LIST\_ITEMS**.**empty**();**

data**[**'items'**].**forEach**(function** **(**item**)** **{**

GCAL\_LIST\_ITEMS**.**append**(**'<li><b>' **+** item**[**'summary'**]** **+** '</b> at ' **+** item**[**'start'**][**'dateTime'**]** **+** '</li>'**)**

**})**

**}**

**}).**error**(function** **()** **{**

**setTimeout(**loadStatus**,** 500**);**

**});**

**}**

**var** showTimeout **=** **setTimeout(function** **()** **{**

showModal**(**'The program is still booting...'**,** 'Please wait'**);**

**},** 250**);**

loadStatus**();**

// WiFi tab

**const** WIFI\_DROPDOWN **=** $**(**'#wifi-dropdown'**);**

**const** WIFI\_FORM **=** $**(**'#wifi-form'**);**

**const** WIFI\_RELOAD **=** $**(**'#wifi-reload'**);**

**const** WIFI\_SSID **=** $**(**'#wifi-ssid'**);**

**const** WIFI\_PASS **=** $**(**'#wifi-pass'**);**

**function** onChangeSSID**()** **{**

WIFI\_SSID**.**parents**(**'.form-group'**).**removeClass**(**'has-error'**);**

WIFI\_DROPDOWN**.**parents**(**'.form-group'**).**removeClass**(**'has-error'**);**

// index of selected option, or -1

**var** i **=** WIFI\_DROPDOWN**.**val**();**

**if** **(**i **!=** **-**1**)** // One of the items on the list

**{**

WIFI\_PASS**.**val**(**''**).**attr**(**'readonly'**,** wifi\_data**[**i**].**Protected **==** 'off'**);**

WIFI\_SSID**.**val**(**wifi\_data**[**i**].**SSID**).**attr**(**'readonly'**,** **true);**

**}**

**else** // Manual

**{**

WIFI\_SSID**.**attr**(**'readonly'**,** **false).**val**(**''**);**

WIFI\_PASS**.**attr**(**'readonly'**,** **false).**val**(**''**);**

**}**

**}**

**function** loadWifi**()** **{**

WIFI\_RELOAD**.**prop**(**'disabled'**,** **true);**

disableTabs**();**

WIFI\_DROPDOWN**.**html**(**'<option value="-1" disabled>Scanning...</option>'**);**

$**.**getJSON**(**'/api/wifi'**,** **function** **(**data**)** **{**

wifi\_data **=** data**;**

**var** html **=** "<option value='-1' disabled selected>Pick an option...</option>"**;**

**for** **(var** i **=** 0**;** i **<** data**.**length**;** i**++)** **{**

**if** **(**data**[**i**].**SSID **===** ""**)** **continue;**

html **+=** "<option value='" **+** i **+** "'" **+** **(**data**[**i**].**Authentication **===** '802.1x' **?** ' disabled ' **:** ''**)** **+** ">" **+** data**[**i**].**SSID **+** "</option>"**;**

**}**

html **+=** "<option value='-1'>Other...</option>"**;**

WIFI\_DROPDOWN**.**html**(**html**);**

WIFI\_RELOAD**.**prop**(**'disabled'**,** **false);**

enableTabs**();**

**});**

**}**

WIFI\_RELOAD**.**click**(function** **()** **{**

loadWifi**();**

**});**

WIFI\_DROPDOWN**.**change**(**onChangeSSID**);**

WIFI\_SSID**.**focusin**(function** **()** **{**

WIFI\_SSID**.**parents**(**'.form-group'**).**removeClass**(**'has-error'**);**

**});**

WIFI\_FORM**.submit(function** **()** **{**

**if** **(**WIFI\_SSID**.**val**()** **==** ''**)** **{**

**if** **(**WIFI\_DROPDOWN**.**val**()** **==** **-**1**)** WIFI\_SSID**.**parents**(**'.form-group'**).**addClass**(**'has-error'**);**

**else** WIFI\_DROPDOWN**.**parents**(**'.form-group'**).**addClass**(**'has-error'**);**

**return** **false;**

**}**

disableTabs**();**

showModal**(**'Connecting to the wifi network...<br>Make sure you connect to the same wifi as the SmartAlarmClock.'**,** 'Connecting to wifi'**);**

$**.**post**(**'/api/wifi'**,** **{**ssid**:** WIFI\_SSID**.**val**(),** pass**:** WIFI\_PASS**.**val**()},** **function** **()** **{**

MODAL**.**modal**(**'hide'**);**

enableTabs**();**

loadStatus**();**

**});**

**return** **false;**

**});**

// Clock tab

**function** setBnt**(**jq**,** b**)** **{**

**if** **(**b**)** jq**.**removeClass**(**'btn-danger'**).**addClass**(**'btn-success'**).text(**'Enabled'**);**

**else** jq**.**removeClass**(**'btn-success'**).**addClass**(**'btn-danger'**).text(**'Disabled'**);**

**}**

**function** toggleBnt**()** **{**

**var** us **=** $**(this);**

setBnt**(**us**,** **!**us**.**hasClass**(**'btn-success'**));**

**}**

**const** CLOCK\_FORM **=** $**(**"#clock-form"**);**

**const** CLOCK\_WEEKDAY\_ENABLE **=** $**(**"#clock-weekday-enable"**).**click**(**toggleBnt**);**

**const** CLOCK\_WEEKDAY\_SIZE **=** $**(**"#clock-weekday-size"**);**

**const** CLOCK\_FORMAT1 **=** $**(**"#clock-format1"**);**

**const** CLOCK\_SIZE1 **=** $**(**"#clock-size1"**);**

**const** CLOCK\_FORMAT2\_ENABLE **=** $**(**"#clock-format2-enable"**).**click**(**toggleBnt**);**

**const** CLOCK\_FORMAT2 **=** $**(**"#clock-format2"**);**

**const** CLOCK\_SIZE2 **=** $**(**"#clock-size2"**);**

**const** CLOCK\_OFFSET **=** $**(**"#clock-offset"**);**

**const** CLOCK\_MIN **=** $**(**"#clock-min"**);**

**const** CLOCK\_MIN\_ENABLE **=** $**(**"#clock-min-enable"**).**click**(**toggleBnt**);**

**const** CLOCK\_MAX **=** $**(**"#clock-max"**);**

**const** CLOCK\_MAX\_ENABLE **=** $**(**"#clock-max-enable"**).**click**(**toggleBnt**);**

**const** CLOCK\_DAYS\_1 **=** $**(**"#clock-days-1"**);**

**const** CLOCK\_DAYS\_2 **=** $**(**"#clock-days-2"**);**

**const** CLOCK\_DAYS\_3 **=** $**(**"#clock-days-3"**);**

**const** CLOCK\_STREAM **=** $**(**"#clock-stream"**);**

CLOCK\_FORM**.submit(function** **()** **{**

$**.**ajax**({**

type**:** 'POST'**,**

url**:** '/api/settings'**,**

data**:** JSON**.**stringify**(**

**{**

'day'**:** **{**

'enabled'**:** CLOCK\_WEEKDAY\_ENABLE**.**hasClass**(**'btn-success'**),**

'size'**:** **parseInt(**CLOCK\_WEEKDAY\_SIZE**.**val**())**

**},**

'clock'**:** **{**

'format'**:** CLOCK\_FORMAT1**.**val**(),**

'size'**:** **parseInt(**CLOCK\_SIZE1**.**val**())**

**},**

'date'**:** **{**

'enabled'**:** CLOCK\_FORMAT2\_ENABLE**.**hasClass**(**'btn-success'**),**

'format'**:** CLOCK\_FORMAT2**.**val**(),**

'size'**:** **parseInt(**CLOCK\_SIZE2**.**val**())**

**},**

'alarm'**:** **{**

'offset'**:** **parseInt(**CLOCK\_OFFSET**.**val**()),**

'min'**:** CLOCK\_MIN\_ENABLE**.**hasClass**(**'btn-success'**)** **?** timeToMin**(**CLOCK\_MIN**.**val**())** **:** **-**1**,**

'max'**:** CLOCK\_MAX\_ENABLE**.**hasClass**(**'btn-success'**)** **?** timeToMin**(**CLOCK\_MAX**.**val**())** **:** **-**1**,**

'days'**:** $**(**'input:radio[name="clock-days"]:checked'**).**val**(),**

'stream'**:** CLOCK\_STREAM**.**val**()**

**}**

**}),**

contentType**:** 'application/json'**,**

dataType**:** 'json'**,**

success**:** **function** **()** **{**

loadStatus**();**

loadSettings**();**

**}**

**});**

**return** **false;**

**});**

//Settings tabs

**function** loadSettings**()** **{**

$**.**getJSON**(**'/api/settings'**,** **function** **(**data**)** **{**

console**.**log**(**'Update settings data.'**);**

settings\_data **=** data**;**

setBnt**(**CLOCK\_WEEKDAY\_ENABLE**,** data**[**'day'**][**'enabled'**]);**

CLOCK\_WEEKDAY\_SIZE**.**val**(**data**[**'day'**][**'size'**]);**

CLOCK\_FORMAT1**.**val**(**data**[**'clock'**][**'format'**]);**

CLOCK\_SIZE1**.**val**(**data**[**'clock'**][**'size'**]);**

setBnt**(**CLOCK\_FORMAT2\_ENABLE**,** data**[**'date'**][**'enabled'**]);**

CLOCK\_FORMAT2**.**val**(**data**[**'date'**][**'format'**]);**

CLOCK\_SIZE2**.**val**(**data**[**'date'**][**'size'**]);**

CLOCK\_OFFSET**.**val**(**data**[**'alarm'**][**'offset'**]);**

CLOCK\_MIN**.**val**(**minToTime**(**data**[**'alarm'**][**'min'**]));**

setBnt**(**CLOCK\_MIN\_ENABLE**,** data**[**'alarm'**][**'min'**]** **!=** **-**1**);**

CLOCK\_MAX**.**val**(**minToTime**(**data**[**'alarm'**][**'max'**]));**

setBnt**(**CLOCK\_MAX\_ENABLE**,** data**[**'alarm'**][**'max'**]** **!=** **-**1**);**

**if** **(**data**[**'alarm'**][**'days'**]** **==** 'Days.Weekdays'**)** CLOCK\_DAYS\_1**.**prop**(**'checked'**,** **true);**

**else** **if** **(**data**[**'alarm'**][**'days'**]** **==** 'Days.Weekends'**)** CLOCK\_DAYS\_2**.**prop**(**'checked'**,** **true);**

**else** **if** **(**data**[**'alarm'**][**'days'**]** **==** 'Days.Both'**)** CLOCK\_DAYS\_3**.**prop**(**'checked'**,** **true);**

CLOCK\_STREAM**.**val**(**data**[**'alarm'**][**'stream'**]);**

**});**

**}**

GCAL\_POLL\_FORM**.submit(function** **()** **{**

$**.**post**(**'/api/pollgcal'**,** **function** **(**data**)** **{**

loadStatus**();**

loadSettings**();**

**});**

**return** **false;**

**});**

GCAL\_RESET\_FORM**.submit(function** **()** **{**

$**.**post**(**'/api/resetgcal'**,** **function** **(**data**)** **{**

loadStatus**();**

loadSettings**();**

**});**

**return** **false;**

**});**

//Global things

TAB\_STATUS**.**on**(**'show.bs.tab'**,** **function** **()** **{**

console**.**log**(**'Tab status'**);**

**});**

TAB\_WIFI**.**on**(**'show.bs.tab'**,** **function** **()** **{**

console**.**log**(**'Tab wifi'**);**

loadWifi**();**

**});**

TAB\_CLOCK**.**on**(**'show.bs.tab'**,** **function** **()** **{**

console**.**log**(**'Tab clock'**);**

loadSettings**();**

**});**

TAB\_GCAL**.**on**(**'show.bs.tab'**,** **function** **()** **{**

console**.**log**(**'Tab gcal'**);**

loadSettings**();**

**});**

**if** **(document.location.**toString**().**match**(**'#'**))** **{**

$**(**'.nav-tabs a[href="#' **+** **document.location.**toString**().**split**(**'#'**)[**1**]** **+** '"]'**).**tab**(**'show'**);**

**}**

$**(**'.nav-tabs a'**).**on**(**'shown.bs.tab'**,** **function** **(**e**)** **{**

loadStatus**();**

**window.location.**hash **=** e**.**target**.**hash**;**

**});**

## Overige programma’s & configuratie bestanden

### Nginx configuratie (nginx.conf)

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

root /root/www/static/;

index index.html index.htm;

location /nginx\_status {

stub\_status on;

access\_log off;

}

# Dynamic python

location /api/ {

proxy\_pass http://127.0.0.1:5000/;

proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_read\_timeout 600s;

}

}

### Startup script (.bash\_profile)

[[ -f ~/.bashrc ]] && . ~/.bashrc

# Terminal resize

[[ $TERM != "screen" ]] && resize 45 180

getLocalIP() {

local \_ip \_myip \_line \_nl=$'\n'

while IFS=$': \t' read -a \_line ;do

[ -z "${\_line%inet}" ] &&

\_ip=${\_line[${#\_line[1]}>4?1:2]} &&

[ "${\_ip#127.0.0.1}" ] && \_myip=$\_ip

done< <(LANG=C /sbin/ifconfig)

printf ${1+-v} $1 "%s${\_nl:0:$[${#1}>0?0:1]}" $\_myip

}

echo "Hostname: `hostname`"

echo "Local IP: `getLocalIP`"

echo "Date: `date`"

echo "HwClock: `hwclock`"

runApp() {

echo ds3231 0x68 > /sys/class/i2c-adapter/i2c-1/new\_device

www/app.py

}

export APP\_GCAL\_ID="806788990556-i96frm71oem88mn63qvsudbmvhrusesf.apps.googleusercontent.com"

export APP\_GCAL\_SECRET="MFDHJWCHUl\_CHAwokNvPXyR4"

export SDL\_FBDEV=/dev/fb1

[ "$TERM" != "screen" ] && [ "$SSH\_TTY" == "" ] && runApp

### Kernelmodules (/etc/modules-load.d/raspberrypy.conf)

snd-bcm2835

i2c-dev

i2c-bcm2835

spi-bcm2835

rtc-ds1307

fbtft\_device