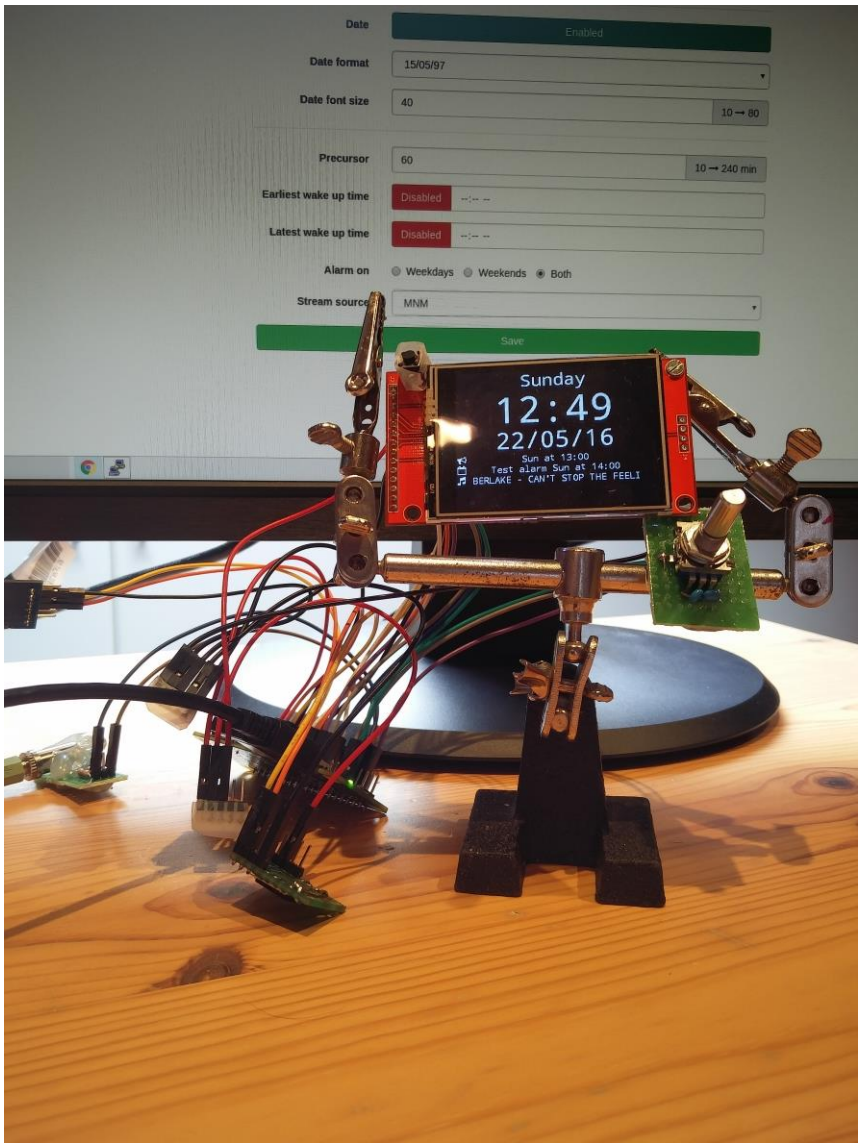


SmartAlarmClock

PROJECT II – ICT-ELEKTRONICA

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

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, michiel.bellekens@student.thomasmore.be.

Inhoudsopgave

1 Hasrdware

1.1 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.

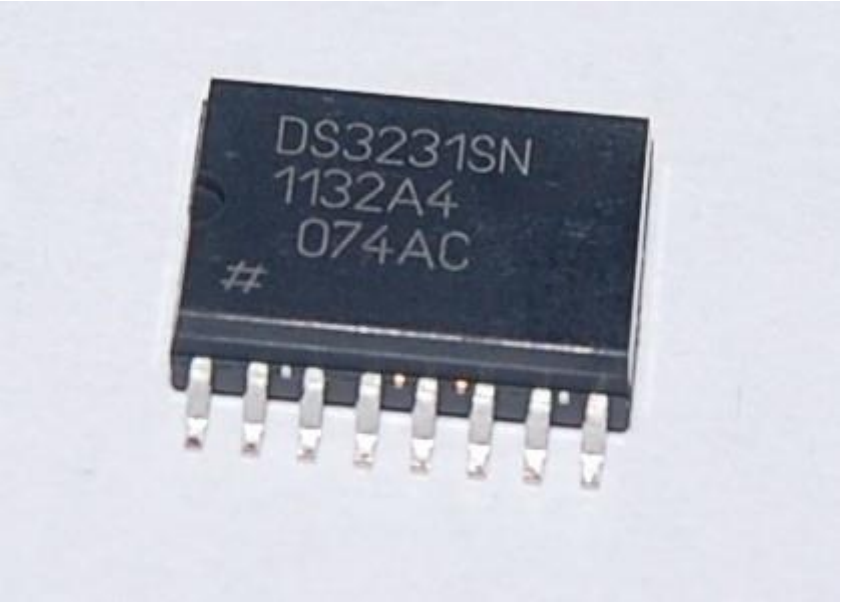
1.2 Rotary encoder

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

1.3 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

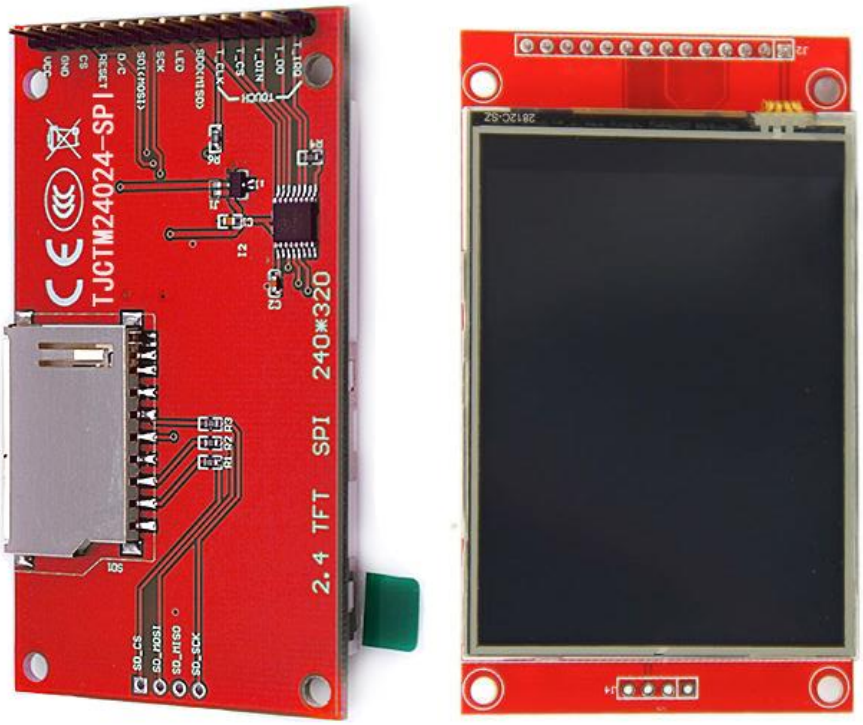
1.4 Spanningsregelaar (MCP1703)

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.

1.5 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.

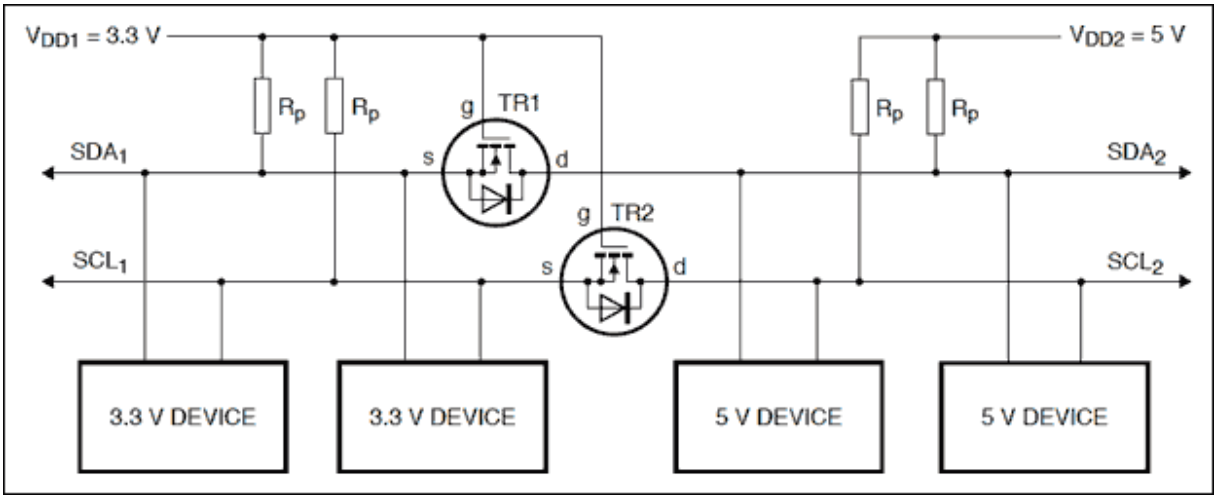
1.6 Buzzer

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

1.7 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.



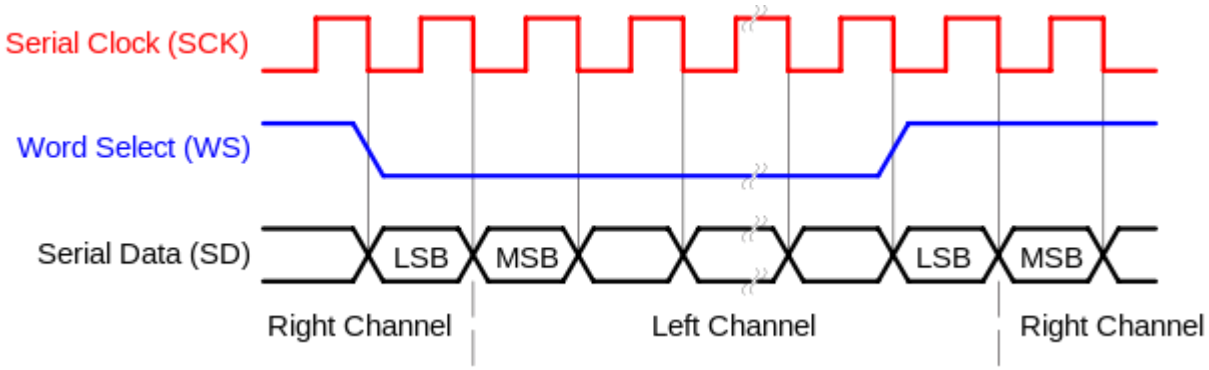
Figuur 1-6: Shiften tussen 3.3 V en 5 V op een I2C bus.

1.8 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.

1.8.1 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.

1.9 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.

1.10 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.

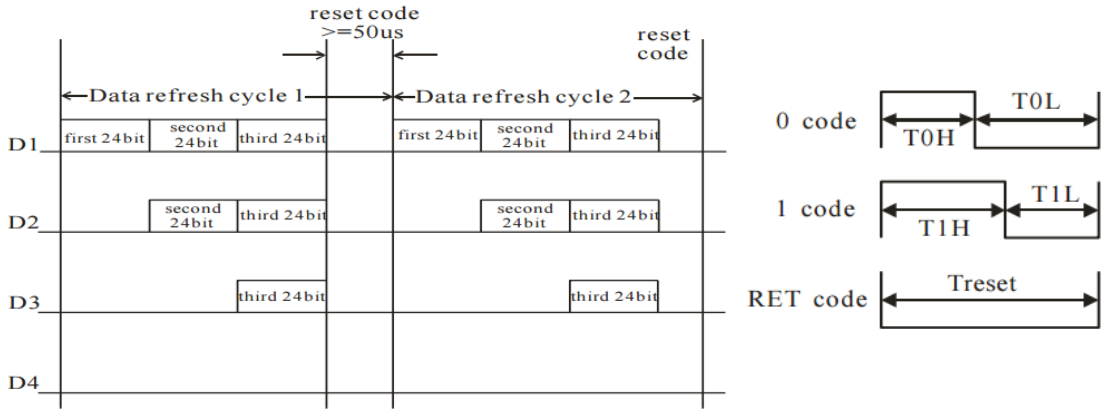
1.11 Supercap

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

1.12 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

1.13 Het schema & PCB

Zie bijlage 6.1.

2 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.

2.1 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.

2.1.1 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.

2.1.2 “.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.

2.1.3 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.

2.2 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.

2.2.1 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.

2.2.2 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]]

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

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

DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

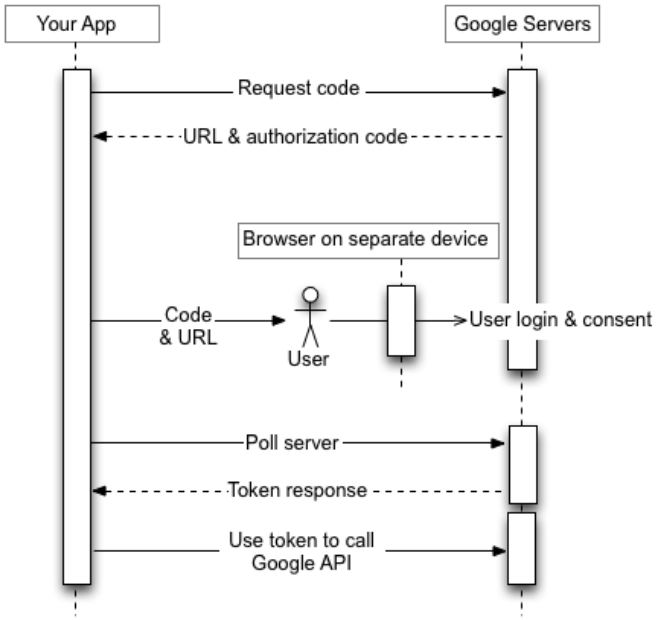
Figuur 2-2:Deze waardes bepalen bij welke overeenkomst de interrupt pin moet worden geactiveerd.

2.2.3 Google Calendar

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

2.2.3.1 *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.

2.2.3.2 *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.

2.2.3.3 *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’]”.

2.2.3.4 *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

2.2.4 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.

2.2.5 Netwerk en acces point

De eerste stap in verband met het netwerk is het controleren van het bestaan van de “wlano” adapter. Dit kan door te controleren of het pad “/sys/class/net/wlano” 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 “wlano” 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.

2.2.6 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.

2.2.7 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”.

2.2.8 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.

2.2.8.1 *Status tab*

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

2.2.8.2 *Wifi settings tab*

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

2.2.8.3 *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.

2.2.8.4 *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 .

3 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**

4 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.

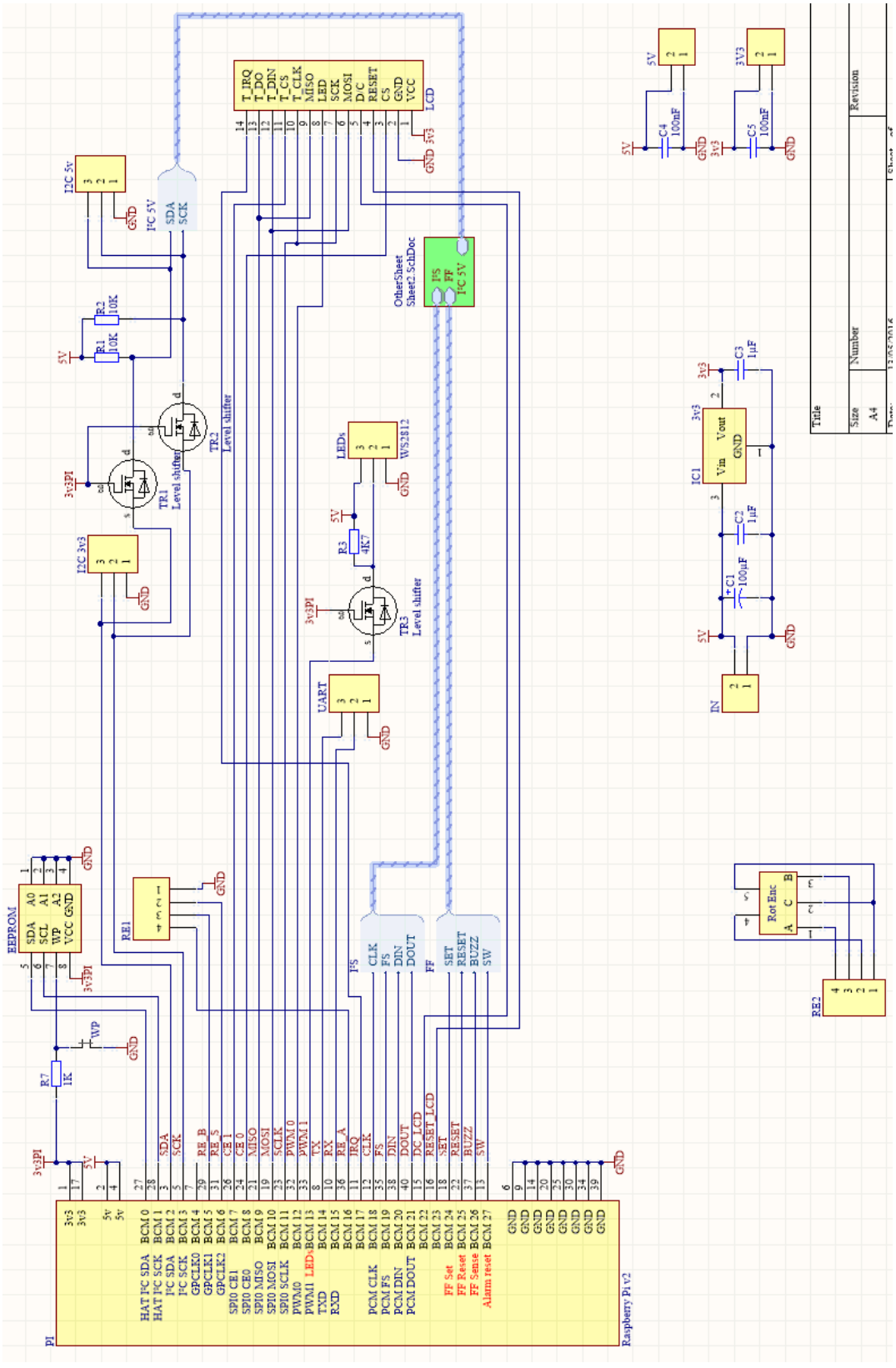
Voor de muziek werd initieel gedacht om de I²S interface te gebruiken. Aangezien de printen niet tijdig werden geleverd ging dit echter niet meer. Om rond dit probleem te werken hebben we dan enkel zaken aangepast. Voor het geluid besloten we om via PWM te werken waardoor de ws2812 leds moesten worden 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.

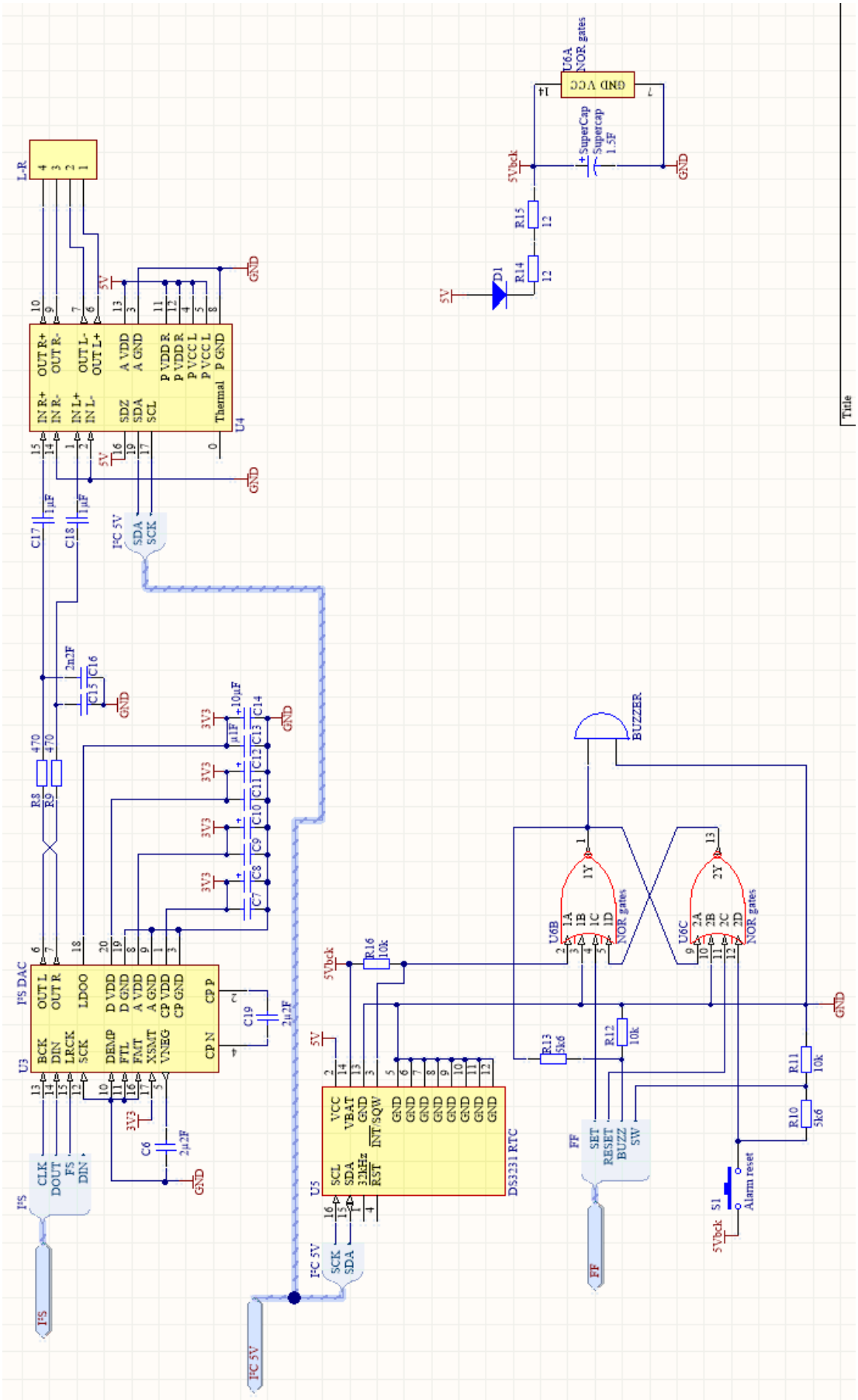
5 Bijlagen

5.1 Schema's & Lay-out printplaat

5.1.1 De schema's

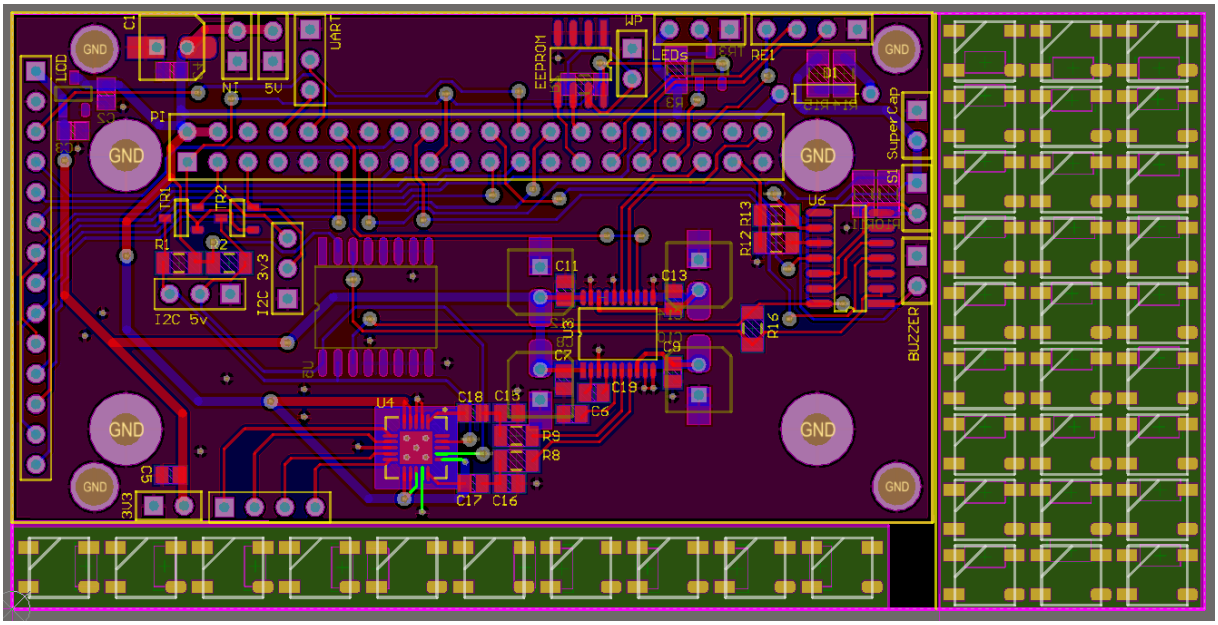


Figuur 5-1: Het schema, sheet 1.



Figuur 5-2: Het schema, sheet 2.

5.1.2 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.

5.2 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))
```

```
77      subprocess.call(['amixer', 'sset', 'PCM,0', '%.0f%%' % vol, '-M'])
78
79    # Save settings
80    def save():
81      json.dump(settings, open(SETTINGS_FILE, 'w'), indent=2, cls=EnumEncoder)
82
83    # Clamp value between 0 and 100 by default
84    def clamp(n, minn=0, maxn=100):
85      return max(min(maxn, n), minn)
86
87    # Go back from "<Class>.<Name>" to actual enum instance
88    def as_enum(full):
89      if full is None:
90        return None
91      name, member = full.split(".")
92      return getattr(globals()[name], member)
93
94    def stream_start():
95      global music_process
96      if music_process is not None:
97        return
98      stream = streams[settings['alarm']['stream']] if settings['alarm'][
99                                                          'stream'] in streams else \
100     settings['alarm']['stream']
101     status['streaming'] = True
102     music_process = subprocess.Popen(['mpg123', '-T', stream], universal_newlines=True,
103                                      bufsize=1, stderr=subprocess.PIPE)
104
105     def stream_parser():
106       re_title = re.compile(r"ICY-META: StreamTitle='(.*)';")
107       while music_process is not None and music_process.poll() is None:
108         line = music_process.stderr.readline()
109         matcher = re_title.search(line)
110         if matcher:
111           status['draw']['title'] = matcher.group(1)
112       if 'title' in status['draw']:
113         del status['draw']['title']
114
115     threading.Thread(target=stream_parser, name='StreamParser', daemon=True).start()
116
117   def stream_stop():
118     global music_process
119     if music_process is None:
120       return
121     if music_process.poll() is None:
122       music_process.terminate()
123       music_process.wait()
124     status['streaming'] = False
125     music_process = None
126
127   # ################################ Sequential code
128   # set pwm pin
129   # subprocess.call(['gpio', '-g', 'mode', '12', 'pwm'])
130   # set pwm pin sound
131   subprocess.call(['gpio_alt', '-p', '13', '-f', '0'])
132   # Kill any existing ap
133   subprocess.call(['create_ap', '--stop', 'wlan0'])
134   # Sometimes the above isn't enough
135   subprocess.call(['killall', 'hostapd'])
136
137   if os.getenv('SDL_FBDEV') is None:
138     print('SDL_FBDEV not set.')
139     sys.exit(1)
140
141   # Write to SDL_FBDEV (normally /dev/fb1)
142   with open(os.getenv('SDL_FBDEV'), 'wb') as outfile:
143     # Raw image data of 'Booting..' centered on the lcd. Made by a little custom C program
144     frame = bytearray([0x00] * 44446 + [0xff] * 6 + [0x00] * ..........................)
145     outfile.write(frame)
146     outfile.close()
147
148   # ######################################## BOOT SEQ, PART 2 - GPIO
149   print('Boot sequence part 2 - GPIO (%s)' % datetime.datetime.now())
150   # ################################ Imports
151   # noinspection PyUnresolvedReferences
152   import RPi.GPIO as GPIO
153
```

```
154   # ################################ Definitions
155   RE_A = 16
156   RE_B = 5
157   RE_S = 6
158   ALARM_S = 27
159   PWM_BG = 12
160
161   # Set PWM of LCD background LED via gpio program because the GPIO python module
162   def set_brightness(percent=100):
163     p.ChangeDutyCycle(percent)
164     # subprocess.call(['gpio', '-g', 'pwm', '12', '%.0f' % (clamp(percent) * 10.23)])
165
166   # While booting, ramp up the LCD backlight from 0 to 100% over 2 seconds
167   def pre_boot_pwm():
168     i = 0
169     while status['booting'] and i < 100:
170       set_brightness(i)
171       time.sleep(0.2)
172       i += 1
173     set_brightness(100)
174
175   # 'Interrupt' handler of alarm button
176   def int_btn_alarm(chan):
177     print('BTN: ALARM')
178     if 'alarm' in status and not status['skipped']:
179       ts_alarm = status['alarm'].replace(second=0, microsecond=0).timestamp()
180       ts_now = datetime.datetime.now().replace(second=0, microsecond=0).timestamp()
181       if int(ts_alarm - ts_now) < 600:
182         status['skipped'] = True
183         if 'alarm' in status['draw']:
184           del status['draw']['alarm']
185         if status['streaming']:
186           stream_stop()
187           del status['alarm']
188           next_event()
189         return
190     if status['streaming']:
191       stream_stop()
192     else:
193       stream_start()
194
195   # 'Interrupt' handler button of rot encoder
196   def int_btn_ok(chan):
197     print('BTN: OK')
198     if not status['draw']['clock']:
199       return
200     if status['menu']:
201       status['draw']['option'] = None if status['menu'] == Menu.Exit else status['menu']
202       status['menu'] = None
203     elif status['draw']['option']:
204       status['draw']['option'] = None
205       save()
206     else:
207       status['menu'] = Menu.Show_IP
208
209   # 'Interrupt' handler for A of rotary encoder
210   def int_rot(chan):
211     a = GPIO.input(RE_A)
212     # Since A triggers on falling edge, it should be 0, if not, debounce
213     if a:
214       return
215     # To get the direction, pull B
216     b = GPIO.input(RE_B)
217     print('Rotate: %s' % ('Right,+' if b else 'Left,-'))
218     # If we are in menu, go left or right
219     if status['menu']:
220       items = list(Menu)
221       i = items.index(status['menu'])
222       status['menu'] = items[(i + (1 if b else -1)) % len(items)]
223     # If we are doing a setting
224     elif status['draw']['option']:
225       if status['draw']['option'].value['setting']:
226         setting = status['draw']['option'].value['setting']
227         current = settings
228         # Go down the the sencond to last object and property name, so it can be set
229         for key in setting[:-1]:
230           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:
```

```
308         settings.update(json.load(open(SETTINGS_FILE)))
309         settings['alarm']['days'] = as_enum(settings['alarm']['days'])
310         set_volume()
311       except json.decoder.JSONDecodeError:
312         print('Config file unreadable, lets just throw it away and start fresh')
313     else:
314       save()
315
316     # ######################################## BOOT SEQ, PART 4 - Scheduler
317     print('Boot sequence part 4 - Scheduler (%s)' % datetime.datetime.now())
318     # ############################### Definitions
319
320     # Draw text on screen, below other height, in font & color, centered by default
321     def draw_text(message, font=FONT_S, color=WHITE, height=0, center=True):
322       # if height is 0, assume the screen needs clearing
323       if height == 0:
324         SCREEN.fill(BLACK)
325       # Render the text on a new surface
326       text = font.render(message, False, color)
327       x = 0
328       # Center the text
329       if center:
330         x = (SCREEN.get_width() - text.get_width()) / 2
331       # Draw the text surface on the screen
332       SCREEN.blit(text, (x, height))
333       # 'Commit' the changes
334       pygame.display.update()
335       # Return the new height
336       return height + text.get_height()
337
338     # Special message display, used for 'fatal crashes'
339     def error(message):
340       SCREEN.fill(BLACK)
341       height = draw_text('SmartAlarmClock (%s)' % VERSION, FONT_M)
342       height = draw_text('Fatal Error', color=RED, height=height)
343       draw_text(message, height=height)
344       # Goodbye cruel world. We can't exit because it would clear the screen.
345       while True:
346         time.sleep(1)
347
348     # Task to periodically update the IP to be displayed
349     def task_update_ip():
350       threadLocal.ip = socket.gethostbyname(socket.gethostname())
351       CLOCK.enter(10, 10, task_update_ip)
352
353     # To make the font objects updatable, but not waste resources
354     def task_update_font():
355       threadLocal.font_day = pygame.font.SysFont('notomono', settings['day']['size'])
356       threadLocal.font_clock = pygame.font.SysFont('notomono', settings['clock']['size'])
357       threadLocal.date_clock = pygame.font.SysFont('notomono', settings['date']['size'])
358
359     # Task to do the background brightness, handles pulsing effect if required
360     def task_update_pwm():
361       if status['streaming']:
362         set_brightness(100)
363       else:
364         bgt = settings['brightness']
365         bgt['target'] = clamp(bgt['target'], bgt['min'], bgt['max'])
366         # If we are drawing the brightness slider, give instant feedback
367         if status['draw']['option'] == Menu.Set_Brightness:
368           set_brightness(bgt['preference'])
369         # If not drawing the brightness slider
370         else:
371           # If we are not at target level brightness
372           if bgt['target'] != bgt['now']:
373             # Deviate predetermined step size from the current brightness towards the
374
375             if bgt['target'] < bgt['now']:
376               bgt['now'] = max(bgt['target'], bgt['now'] - bgt['step'], bgt['min'], 0)
377             else:
378               bgt['now'] = min(bgt['target'], bgt['now'] + bgt['step'], bgt['max'], 100)
379             set_brightness(bgt['now'])
380           # if we are at target brightness
381           else:
382             # if pulsing
383             if status['pulsing']:
384               if bgt['target'] <= bgt['min']:
```

```
385               bgt['target'] = bgt['max']
386             else:
387               bgt['target'] = bgt['min']
388           # if not pulsing
389           else:
390             bgt['target'] = bgt['preference']
391     CLOCK.enter(0.1, 2, task_update_pwm)
392
393   def truncate_scroll_text(string, length=30):
394     if len(string) <= length:
395       return string
396     td = int(datetime.datetime.now().timestamp() * 2) % (len(string) - length + 1)
397     return string[td:td + length]
398
399   # Task that draws to the LCD
400   def task_draw_clock():
401     # To make adding code easy, leftover/unnecessary height assignments are left!
402     height = 0
403     # draw the main clock
404     if status['draw']['clock']:
405       if settings['day']['enabled']:
406         height = draw_text(datetime.datetime.now().strftime('%A'), height=height,
407                            font=threadLocal.font_day)
408         height -= (threadLocal.font_day.get_height() * 0.1)
409       height = draw_text(datetime.datetime.now().strftime(settings['clock']['format']),
410                          height=height, font=threadLocal.font_clock)
411       height -= (threadLocal.font_clock.get_height() * 0.1)
412       if settings['date']['enabled']:
413         height = draw_text(datetime.datetime.now().strftime(settings['date']['format']),
414                            height=height, font=threadLocal.date_clock)
415     # if we need to register the devie with gcal
416     if 'user_code' in status['gcal']:
417       height = draw_text(status['gcal']['verification_url'], height=height, font=FONT_S)
418       height = draw_text('Code: ' + status['gcal']['user_code'], height=height, font=FONT_S)
419     # If the menu needs drawing
420     if status['menu']:
421       height = draw_text('Menu', height=height, font=FONT_S)
422       height = draw_text(status['menu'].value['name'], height=height, font=FONT_S)
423     # if we have a menu option selected
424     elif status['draw']['option']:
425       if status['draw']['option'] == Menu.Show_IP:
426         height = draw_text(threadLocal.ip, height=height, font=FONT_S)
427       # If the option is tied to a setting
428       elif status['draw']['option'].value['setting']:
429         setting = status['draw']['option'].value['setting']
430         current = settings
431         # Move down the list of keys: ('foo', 'bar') => settings['foo']['bar]
432         for key in setting:
433           current = current[key]
434         height = draw_text(status['draw']['option'].value['name'] + ': %d%%' % current,
435                            height=height, font=FONT_S)
436         # Draw the rectangle below the text and % value
437         pygame.draw.rect(SCREEN, WHITE, (0, height, SIZE[0] * (current / 100), 5))
438         # Need to move 5 px down manually
439         height += 5
440     else:
441       if 'alarm' in status['draw']:
442         draw_text('\uf0a1', height=height, font=FONT_ICO, center=False)
443         height = draw_text(status['draw']['alarm'], height=height, font=FONT_S)
444       if 'next' in status['draw']:
445         draw_text('\uf133', height=height, font=FONT_ICO, center=False)
446         height = draw_text(truncate_scroll_text(status['draw']['next']), height=height,
447                            font=FONT_S)
448       if 'title' in status['draw']:
449         draw_text('\uf001', height=height, font=FONT_ICO, center=False)
450         height = draw_text(truncate_scroll_text(status['draw']['title']), height=height,
451                            font=FONT_S)
452     # Actually commit the LCD
453     pygame.display.update()
454     CLOCK.enter(0.5, 1, task_draw_clock)
455
456   def task_check_gcal():
457     print('Checking gcal (%s)' % datetime.datetime.now())
458     gcal_get_events()
459     CLOCK.enter(60 * 30, 2, task_check_gcal)
460
461   # Helper method, to run the task once manually before passing it off to the scheduler
```

```
462   def run_clock_thread():
463     # We are now out of booting
464     status['booting'] = False
465     task_update_font()
466     task_update_ip()
467     task_update_pwm()
468     task_draw_clock()
469     task_check_gcal()
470     task_alarm_check()
471     CLOCK.run()
472
473   def task_alarm_check():
474     if 'alarm' in status:
475       print('Alarm poll')
476       ts_alarm = status['alarm'].replace(second=0, microsecond=0).timestamp()
477       ts_now = datetime.datetime.now().replace(second=0, microsecond=0).timestamp()
478       if int(ts_alarm - ts_now) == 600:
479         print('10 minute mark')
480         status['pulsing'] = True
481       if int(ts_alarm - ts_now) == 0:
482         print('alarm time')
483         if status['skipped']:
484           status['skipped'] = False
485           del status['alarm']
486           next_event()
487         else:
488           stream_start()
489         if 'alarm' in status['draw']:
490           del status['draw']['alarm']
491     CLOCK.enter(30, 1, task_alarm_check)
492
493   # Try to connect to the wifi network set via settings['wifiProfile']
494   def attempt_connect(height=0):
495     # Just to be sure
496
497     subprocess.call(['killall', 'hostapd'])
498     subprocess.call(['create_ap', '--stop', 'wlan0'])
499     height = draw_text('Connecting to %s' % settings['wifiProfile'], height=height)
500     # Use switch to make sure no other network is using wlan0
501     if subprocess.call(['netctl', 'switch-to', settings['wifiProfile']]) == 0:
502       # Congratulations, we have liftoff
503       status['network'] = True
504       # Draw the IP in screen to make accessing the web interface easy
505       ip = socket.gethostbyname(socket.gethostname())
506       height = draw_text(ip, height=height, font=FONT_M)
507       height = draw_text('Syncing time & date', height=height)
508       # Use NTP to get internet time
509       subprocess.call(['systemctl', 'restart', 'ntpd'])
510       # Make sure we actually have NTP time
511       if subprocess.call(['ntp-wait', '-n', '5']) != 0:
512         error('NTP sync failed.')
513       else:
514         # Write to RTC
515         subprocess.call(['hwclock', '-w'])
516         # Now we can start rendering the clock
517         status['draw']['clock'] = True
518     # Connecting to wifi failed
519     else:
520       height = draw_text('Failed...', height=height)
521       status['network'] = False
522     return height
523
524   # Refresh our token
525   def gcal_refresh():
526     if not status['network'] or 'refresh_token' not in settings['gcal']:
527       return
528     out = json.loads(requests.post('https://www.googleapis.com/oauth2/v4/token',
529                                    data={'refresh_token': settings['gcal']['refresh_token'],
530                                      'client_id': gcal['client_id'],
531                                      'client_secret': gcal['client_secret'],
532                                      'grant_type': 'refresh_token'}).text)
533     status['gcal'] = out
534     if 'expires_in' in out:
535       status['gcal']['expires'] = datetime.datetime.now() + \
536                                   datetime.timedelta(seconds=out['expires_in'] - 10)
537
538   # Poll with device token to see if user has granted us permission yet
```

```
539   def gcal_poll():
540     out = json.loads(requests.post('https://www.googleapis.com/oauth2/v4/token',
541                                    data={'client_id': gcal['client_id'],
542                                      'client_secret': gcal['client_secret'],
543                                      'code': status['gcal']['device_code'],
544                                      'grant_type':
545                                           'http://oauth.net/grant_type/device/1.0'}).text)
546     # Some information, 'authorization_pending' is normal
547     if "error" in out:
548       print(out['error'])
549     # No error = good
550     else:
551       status['gcal'] = out
552       # 10 sec for safety
553       status['gcal']['expires'] = datetime.datetime.now() + \
554                                   datetime.timedelta(seconds=out['expires_in'] - 10)
555       # Store the token, and make the primary calendar default
556       settings['gcal'] = {'refresh_token': out['refresh_token'], 'calendar_id': 'primary'}
557       # Save the new settings
558       save()
559       # Fetch the upcoming events
560       gcal_get_events()
561       # to prevent re-registering
562       return
563
564     # if the token request is expired, drop it
565     if datetime.datetime.now() > status['gcal']['expires']:
566       print('Token request expired. (%s)' % datetime.datetime.now())
567       status['gcal'] = {}
568     # Poll at the rate requested by Google
569     else:
570       CLOCK.enter(status['gcal']['interval'], 3, gcal_poll)
571
572   # Do a new token request, overrides all old gcal data, except for appointments
573   def gcal_request_token():
574     out = json.loads(requests.post('https://accounts.google.com/o/oauth2/device/code',
575                                    data={'client_id': gcal['client_id'],
576                                      'scope': gcal['scope']}).text)
577     status['gcal'] = {}
578     status['gcal']['device_code'] = out['device_code']
579     status['gcal']['interval'] = out['interval']
580     status['gcal']['user_code'] = out['user_code']
581     status['gcal']['verification_url'] = out['verification_url']
582     # 10 sec for safety
583     status['gcal']['expires'] = datetime.datetime.now() + \
584                                 datetime.timedelta(seconds=out['expires_in'] - 10)
585     if 'gcal' in settings:
586       del settings['gcal']
587       save()
588     CLOCK.enter(out['interval'], 3, gcal_poll)
589
590   # Pull in new events, if token is expired / missing it will request a new one.
591   def gcal_get_events():
592     if not status['network'] or 'gcal' not in settings:
593       return
594     if 'calendar_id' not in status['gcal'] or \
595             datetime.datetime.now() > status['gcal']['expires']:
596       gcal_refresh()
597     if 'access_token' not in status['gcal']:
598       # Something exploded.
599       return
600     out = json.loads(requests.get(
601       'https://www.googleapis.com/calendar/v3/calendars/%s/events' % settings['gcal'][
602         'calendar_id'], headers={
603         'Authorization': status['gcal']['token_type'] + ' ' + status['gcal'][
604           'access_token']}, params={
605         'timeMin': datetime.datetime.now(datetime.timezone.utc).astimezone().isoformat('T'),
606         'timeMax': (datetime.datetime.now(datetime.timezone.utc) + datetime.timedelta(
607           days=7)).astimezone().isoformat('T'), 'singleEvents': True,
608         'orderBy': 'startTime'}).text)
609     if 'items' not in out:
610       print('No items in response gcal_get_events')
611       print(out)
612       status['items'] = None
613       return False
614     status['items'] = out['items']
615     next_event()
```

```
616
617   def next_event():
618     for item in status['items']:
619       alarm = dateutil.parser.parse(item['start']['dateTime'])
620       status['draw']['next'] = item['summary'] + ' ' + \
621                     dateutil.parser.parse(item['start']['dateTime']).strftime('%a at %H:%M')
622       if alarm.isoweekday() not in settings['alarm']['days'].value:
623         print('Event not in target days, skipping. %s' % item['summary'])
624         continue
625       alarm -= datetime.timedelta(minutes=settings['alarm']['offset'])
626       if settings['alarm']['min'] != -1 and \
627                   (alarm.hour * 60) + alarm.minute < settings['alarm']['min']:
628         alarm = alarm.replace(hour=settings['alarm']['min'] // 60,
629                               minute=settings['alarm']['min'] % 60)
630       if settings['alarm']['max'] != -1 and \
631                   (alarm.hour * 60) + alarm.minute > settings['alarm']['max']:
632         alarm = alarm.replace(hour=settings['alarm']['max'] // 60,
633                               minute=settings['alarm']['max'] % 60)
634       if alarm.timestamp() < datetime.datetime.now().timestamp():
635         print('Alarm time passed, skipping. %s' % item['summary'])
636         continue
637       status['alarm'] = alarm
638       status['draw']['alarm'] = alarm.strftime('%a at %H:%M')
639       return
640       # Only get here if no (correct) items
641
642     if 'alarm' in status:
643       del status['alarm']
644     if 'alarm' in status['draw']:
645       del status['draw']['alarm']
646
647   # ################################ Sequential code
648   # Network adapter not plugged in => panic
649   if not os.path.exists('/sys/class/net/wlan0'):
650     error('No wifi interface')
651
652   # For the IP & fonts
653   threadLocal = threading.local()
654   # Make the scheguler
655   CLOCK = sched.scheduler(time.time, time.sleep)
656   # Start the 'main' thread
657   threading.Thread(target=run_clock_thread, name='ClockThread', daemon=True).start()
658   # Draw name & version
659   h = draw_text('SmartClock (%s)' % VERSION, font=FONT_M)
660
661   # If we have a wifi network saved, try to connect
662   if 'wifiProfile' in settings and settings['wifiProfile'] != '':
663     h = attempt_connect(h)
664
665   # if we have connected
666   if status['network']:
667     if 'gcal' in settings:
668       # if we have some gcal stuff saved, update the event list
669       gcal_get_events()
670     else:
671       # Start the token procedure
672       gcal_request_token()
673   # If we don't have network
674   else:
675     # Pull in the clock/date from the RTC, if it got reset, it'll be at 1 jan 2000
676     subprocess.call(['hwclock', '-s'])
677     h = draw_text('Starting AP', height=h)
678     subprocess.call(['create_ap', '-n', '--daemon', '--redirect-to-localhost', 'wlan0',
679                      'SmartAlarmClock'])
680     # Give the adapter some time
681     time.sleep(5)
682     h = draw_text('Connect to wifi network for setup:', height=h)
683     h = draw_text('SmartAlarmClock', height=h, font=FONT_M)
684     h = draw_text('And browse to:', height=h)
685     # Display IP
686     h = draw_text(socket.gethostbyname(socket.gethostname()), height=h, font=FONT_M)
687
688   # ######################################## BOOT SEQ, PART 5 - Flask
689   print('Boot sequence part 5 - Flask (%s)' % datetime.datetime.now())
690   # ################################ Imports
691   from flask import Flask
692   from flask import json
```

```
693   from flask import Response
694   from flask import request
695   from flask import url_for
696   # ############################## Definitions
697   app = Flask(__name__)
698
699   # List of possible access points
700   @app.route('/')
701   def api():
702     output = []
703     for rule in app.url_map.iter_rules():
704       options = {}
705       for arg in rule.arguments:
706         options[arg] = '[%s]' % arg
707       output.append({'name': rule.endpoint, 'methods': ','.join(rule.methods),
708                      'url': '/api' + urllib.parse.unquote(
709                        url_for(rule.endpoint, **options))})
710     return Response(json.dumps(output, cls=EnumEncoder), mimetype='text/javascript')
711
712   # Get the list of wifi networks OR set the wifi profile settings (ssid & pass) (via POST)
713   @app.route('/wifi', methods=['GET', 'POST'])
714   def api_wifi():
715     # Set wifi settings by making a profile file for it, so we can (ab)use netctl
716     if request.method == 'POST':
717       # stop drawing the clock for a second
718       status['draw']['clock'] = False
719       text = "Description='Automatically generated profile by python'\n"
720       text += 'Interface=wlan0\n'
721       text += 'Connection=wireless\n'
722       text += 'IP=dhcp\n'
723       text += "ESSID='%s'\n" % request.form['ssid']
724       if 'pass' in request.form and request.form['pass'] != '':
725         text += "Security=wpa\nKey='%s'" % request.form['pass']
726       else:
727         text += 'Security=none'
728       settings['wifiProfile'] = 'GEN-wlan0-%s' % request.form['ssid']
729       # Write the file
730       f = open('/etc/netctl/%s' % settings['wifiProfile'], 'w')
731       f.write(text)
732       f.close()
733       save()
734       attempt_connect()
735       if 'gcal' in settings:
736         # if we have some gcal stuff saved, update the event list
737         gcal_get_events()
738       else:
739         # Start the token procedure
740         gcal_request_token()
741       # Return a readable value
742       if status['network']:
743         return 'OK'
744       return 'ERROR'
745     # GET
746     else:
747       # Regex yey
748       re_cell = re.compile(r'Cell \d+')
749       re_mac = re.compile(r'Address: (?P<Address>.*)')
750       re_ssid = re.compile(r'ESSID:"(?P<SSID>.*)"')
751       re_quality = re.compile(r'Quality=(?P<Quality>\d+)/100')
752       re_signal = re.compile(r'Signal level=(?P<SignalLevel>\d+)/100')
753       re_encrypted = re.compile(r'Encryption key:(?P<Protected>on|off)')
754       re_encryption = re.compile(r'IE: (?P<Protection>.*)')
755       re_authentication = re.compile(
756         r'Authentication Suites \(1\) : (?P<Authentication>.*)')
757       proc = subprocess.Popen(['iwlist', 'wlan0', 'scan'], stdout=subprocess.PIPE,
758                               universal_newlines=True)
759       out, err = proc.communicate()
760       # data array
761       data = []
762       # Used to remeber the last object we are working on
763       cell = None
764       for line in out.split('\n'):
765         # strip whitespace
766         line = line.strip()
767         matcher = re_cell.search(line)
768         # if this is the start of a new network
769         if matcher:
```

```
770           # if cell wasn't None, add it to the data list
771           if cell:
772             data.append(cell)
773           cell = {}
774         for regex in [re_mac, re_ssid, re_quality, re_signal, re_encrypted, re_encryption,
775                       re_authentication]:
776           matcher = regex.search(line)
777           # if this regex matched
778           if matcher:
779             # add the match to the list of properties in the dict
780             cell.update(matcher.groupdict())
781       return Response(json.dumps(data, cls=EnumEncoder), mimetype='text/javascript')
782 
783   # GET: Dump the settings POST: Set settings
784   @app.route('/settings', methods=['GET', 'POST'])
785   def api_settings():
786     if request.method == 'GET':
787       return Response(json.dumps(settings, cls=EnumEncoder), mimetype='text/javascript')
788     else:
789       settings.update(request.get_json())
790       settings['alarm']['days'] = as_enum(settings['alarm']['days'])
791       CLOCK.enter(0, 1, task_update_font)
792       set_volume()
793       next_event()
794       save()
795       return 'OK'
796 
797   # Reset (and restart) the gcal linking process
798   @app.route('/pollgcal', methods=['POST'])
799   def api_pollgcal():
800     gcal_get_events()
801     return 'OK'
802 
803   # Reset (and restart) the gcal linking process
804   @app.route('/resetgcal', methods=['POST'])
805   def api_resetgcal():
806     gcal_request_token()
807     return 'OK'
808 
809   # GET: Dump the status POST: Set status
810   @app.route('/status', methods=['GET', 'POST'])
811   def api_status():
812     if request.method == 'GET':
813       return Response(json.dumps(status, cls=EnumEncoder), mimetype='text/javascript')
814     else:
815       status.update(request.get_json())
816       status['draw']['option'] = as_enum(status['draw']['option'])
817       status['menu'] = as_enum(status['menu'])
818       save()
819       return 'OK'
820 
821   # ################################ Sequential code
822   print('Starting webserver... (%s)' % datetime.datetime.now())
823   # Blocking call!
824   app.run(host='127.0.0.1', port=5000, use_reloader=False)
825 
826   print('EXIT: Flask died (%s)' % datetime.datetime.now())
827   pygame.quit()
828   GPIO.cleanup()
```

5.3 Web interface

5.3.1 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 <i class="fa fa-long-arrow-right"
aria-hidden="true"></i> 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 <i class="fa fa-long-arrow-right"
aria-hidden="true"></i> 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 <i class="fa fa-long-arrow-right"
aria-hidden="true"></i> 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 <i class="fa fa-long-arrow-right"
aria-hidden="true"></i> 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">WWWWWWWWWWWWWW</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>
```

5.3.2 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);
```

```
77        }
78        else if (data['gcal']['access_token']) {
79          GCAL_LIST.show();
80          GCAL_LINK.hide();
81          GCAL_RESET.show();
82        }
83        else if (data['gcal']) {
84          GCAL_RESET.show();
85        }
86
87        if (data['items']) {
88          GCAL_LIST_ITEMS.empty();
89          data['items'].forEach(function (item) {
90            GCAL_LIST_ITEMS.append('<li><b>' + item['summary'] + '</b> at ' +
91    item['start']['dateTime'] + '</li>')
92          })
93        }
94
95      }).error(function () {
96        setTimeout(loadStatus, 500);
97      });
98    }
99
100   var showTimeout = setTimeout(function () {
101     showModal('The program is still booting...', 'Please wait');
102   }, 250);
103   loadStatus();
104
105   // WiFi tab
106   const WIFI_DROPDOWN = $('#wifi-dropdown');
107   const WIFI_FORM = $('#wifi-form');
108   const WIFI_RELOAD = $('#wifi-reload');
109   const WIFI_SSID = $('#wifi-ssid');
110   const WIFI_PASS = $('#wifi-pass');
111
112   function onChangeSSID() {
113     WIFI_SSID.parents('.form-group').removeClass('has-error');
114     WIFI_DROPDOWN.parents('.form-group').removeClass('has-error');
115     // index of selected option, or -1
116     var i = WIFI_DROPDOWN.val();
117     if (i != -1) // One of the items on the list
118     {
119       WIFI_PASS.val('').attr('readonly', wifi_data[i].Protected == 'off');
120       WIFI_SSID.val(wifi_data[i].SSID).attr('readonly', true);
121     }
122     else // Manual
123     {
124       WIFI_SSID.attr('readonly', false).val('');
125       WIFI_PASS.attr('readonly', false).val('');
126     }
127   }
128
129   function loadWifi() {
130     WIFI_RELOAD.prop('disabled', true);
131     disableTabs();
132     WIFI_DROPDOWN.html('<option value="-1" disabled>Scanning...</option>');
133     $.getJSON('/api/wifi', function (data) {
134       wifi_data = data;
135       var html = "<option value='-1' disabled selected>Pick an option...</option>";
136       for (var i = 0; i < data.length; i++) {
137         if (data[i].SSID === "") continue;
138         html += "<option value='" + i + "'" + (data[i].Authentication === '802.1x' ? '
139   disabled ' : '') + ">" + data[i].SSID + "</option>";
140       }
141       html += "<option value='-1'>Other...</option>";
142       WIFI_DROPDOWN.html(html);
143       WIFI_RELOAD.prop('disabled', false);
144       enableTabs();
145     });
146   }
147   WIFI_RELOAD.click(function () {
148     loadWifi();
149   });
150   WIFI_DROPDOWN.change(onChangeSSID);
151   WIFI_SSID.focusin(function () {
152     WIFI_SSID.parents('.form-group').removeClass('has-error');
153   });
```

```
154   WIFI_FORM.submit(function () {
155     if (WIFI_SSID.val() == '') {
156       if (WIFI_DROPDOWN.val() == -1) WIFI_SSID.parents('.form-group').addClass('has-error');
157       else WIFI_DROPDOWN.parents('.form-group').addClass('has-error');
158       return false;
159     }
160
161     disableTabs();
162     showModal('Connecting to the wifi network...<br>Make sure you connect to the same wifi as
163   the SmartAlarmClock.', 'Connecting to wifi');
164     $.post('/api/wifi', {ssid: WIFI_SSID.val(), pass: WIFI_PASS.val()}, function () {
165       MODAL.modal('hide');
166       enableTabs();
167       loadStatus();
168     });
169
170     return false;
171   });
172
173
174   // Clock tab
175   function setBnt(jq, b) {
176     if (b) jq.removeClass('btn-danger').addClass('btn-success').text('Enabled');
177     else jq.removeClass('btn-success').addClass('btn-danger').text('Disabled');
178   }
179
180   function toggleBnt() {
181     var us = $(this);
182     setBnt(us, !us.hasClass('btn-success'));
183   }
184
185   const CLOCK_FORM = $("#clock-form");
186   const CLOCK_WEEKDAY_ENABLE = $("#clock-weekday-enable").click(toggleBnt);
187   const CLOCK_WEEKDAY_SIZE = $("#clock-weekday-size");
188   const CLOCK_FORMAT1 = $("#clock-format1");
189   const CLOCK_SIZE1 = $("#clock-size1");
190   const CLOCK_FORMAT2_ENABLE = $("#clock-format2-enable").click(toggleBnt);
191   const CLOCK_FORMAT2 = $("#clock-format2");
192   const CLOCK_SIZE2 = $("#clock-size2");
193   const CLOCK_OFFSET = $("#clock-offset");
194   const CLOCK_MIN = $("#clock-min");
195   const CLOCK_MIN_ENABLE = $("#clock-min-enable").click(toggleBnt);
196   const CLOCK_MAX = $("#clock-max");
197   const CLOCK_MAX_ENABLE = $("#clock-max-enable").click(toggleBnt);
198   const CLOCK_DAYS_1 = $("#clock-days-1");
199   const CLOCK_DAYS_2 = $("#clock-days-2");
200   const CLOCK_DAYS_3 = $("#clock-days-3");
201   const CLOCK_STREAM = $("#clock-stream");
202
203   CLOCK_FORM.submit(function () {
204     $.ajax({
205       type: 'POST',
206       url: '/api/settings',
207       data: JSON.stringify(
208           {
209             'day': {
210               'enabled': CLOCK_WEEKDAY_ENABLE.hasClass('btn-success'),
211               'size': parseInt(CLOCK_WEEKDAY_SIZE.val())
212             },
213             'clock': {
214               'format': CLOCK_FORMAT1.val(),
215               'size': parseInt(CLOCK_SIZE1.val())
216             },
217             'date': {
218               'enabled': CLOCK_FORMAT2_ENABLE.hasClass('btn-success'),
219               'format': CLOCK_FORMAT2.val(),
220               'size': parseInt(CLOCK_SIZE2.val())
221             },
222             'alarm': {
223               'offset': parseInt(CLOCK_OFFSET.val()),
224               'min': CLOCK_MIN_ENABLE.hasClass('btn-success') ? timeToMin(CLOCK_MIN.val()) : -
225   1,
226               'max': CLOCK_MAX_ENABLE.hasClass('btn-success') ? timeToMin(CLOCK_MAX.val()) : -
227   1,
228               'days': $('input:radio[name="clock-days"]:checked').val(),
229               'stream': CLOCK_STREAM.val()
230             }
```

```
231           }),
232         contentType: 'application/json',
233         dataType: 'json',
234         success: function () {
235           loadStatus();
236           loadSettings();
237         }
238       });
239       return false;
240     });
241
242     //Settings tabs
243     function loadSettings() {
244       $.getJSON('/api/settings', function (data) {
245         console.log('Update settings data.');
246         settings_data = data;
247         setBnt(CLOCK_WEEKDAY_ENABLE, data['day']['enabled']);
248         CLOCK_WEEKDAY_SIZE.val(data['day']['size']);
249         CLOCK_FORMAT1.val(data['clock']['format']);
250         CLOCK_SIZE1.val(data['clock']['size']);
251         setBnt(CLOCK_FORMAT2_ENABLE, data['date']['enabled']);
252         CLOCK_FORMAT2.val(data['date']['format']);
253         CLOCK_SIZE2.val(data['date']['size']);
254         CLOCK_OFFSET.val(data['alarm']['offset']);
255         CLOCK_MIN.val(minToTime(data['alarm']['min']));
256         setBnt(CLOCK_MIN_ENABLE, data['alarm']['min'] != -1);
257         CLOCK_MAX.val(minToTime(data['alarm']['max']));
258         setBnt(CLOCK_MAX_ENABLE, data['alarm']['max'] != -1);
259         if (data['alarm']['days'] == 'Days.Weekdays') CLOCK_DAYS_1.prop('checked', true);
260         else if (data['alarm']['days'] == 'Days.Weekends') CLOCK_DAYS_2.prop('checked', true);
261         else if (data['alarm']['days'] == 'Days.Both') CLOCK_DAYS_3.prop('checked', true);
262         CLOCK_STREAM.val(data['alarm']['stream']);
263       });
264     }
265
266     GCAL_POLL_FORM.submit(function () {
267       $.post('/api/pollgcal', function (data) {
268         loadStatus();
269         loadSettings();
270       });
271       return false;
272     });
273
274     GCAL_RESET_FORM.submit(function () {
275       $.post('/api/resetgcal', function (data) {
276         loadStatus();
277         loadSettings();
278       });
279       return false;
280     });
281
282
283     //Global things
284     TAB_STATUS.on('show.bs.tab', function () {
285       console.log('Tab status');
286     });
287     TAB_WIFI.on('show.bs.tab', function () {
288       console.log('Tab wifi');
289       loadWifi();
290     });
291     TAB_CLOCK.on('show.bs.tab', function () {
292       console.log('Tab clock');
293       loadSettings();
294     });
295     TAB_GCAL.on('show.bs.tab', function () {
296       console.log('Tab gcal');
297       loadSettings();
298     });
299
300     if (document.location.toString().match('#')) {
301       $('.nav-tabs a[href="#' + document.location.toString().split('#')[1] + '"]').tab('show');
302     }
303     $('.nav-tabs a').on('shown.bs.tab', function (e) {
304       loadStatus();
305       window.location.hash = e.target.hash;
306     });
```

5.4 Overige programma's & configuratie bestanden

5.4.1 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;
    }
}
```

5.4.2 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
```

5.4.3 Kernelmodules (/etc/modules-load.d/raspberrypy.conf)

```
snd-bcm2835
i2c-dev
i2c-bcm2835
spi-bcm2835
rtc-ds1307
fbtft_device
```