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

SmartAlarmClock

Project II – ICT-Elektronica



# Voorwoord

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

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

De gebruiker kan via een web interface de klok configureren naar zijn/haar eigen wensen. Ten eerste kan de gebruiker instellen hoe lang voor de eerste afspraak de wekker moet afgaan. Ten tweede kan men kiezen of men gewekt wenst te worden via een muziekfile, muziekstream of via voorgeprogrammeerde geluiden. Ten derde kan de gebruiker ook een minimum en maximum wek tijd instellen. Dit betekent dat onafhankelijk van de afspraken de wekker nooit vroeger dan het minimum en later dan het maximum mag afgaan. Tot slot zijn er nog enkele instellingen voor de lay-out op het scherm zoals het formaat en de grootte van het lettertype van de tijd en de datum.

Ons contacteren, kan via email, [dries.kennes@student.thomasmore.be](mailto:dries.kennes@student.thomasmore.be) , michiel.bellekens@student.thomasmore.be, of via de website, [dries007.net](http://dries007.net).

Inhoudsopgave

[1 Voorwoord 2](#_Toc451524630)

[2 Hardware 4](#_Toc451524631)

[2.1 Raspberry Pi Zero Essentials kit 4](#_Toc451524632)

[2.2 Rotary encoder 4](#_Toc451524633)

[2.3 I2C RTC (3.3 V DS3231) 5](#_Toc451524634)

[2.4 Spanningsregelaar (MCP1703) 5](#_Toc451524635)

[2.5 2.4” 240x320 spi tft LCD (3.3 V) 6](#_Toc451524636)

[2.6 Buzzer 6](#_Toc451524637)

[2.7 Levelshifter (AN10441) 7](#_Toc451524638)

[2.8 Audio stereo DAC (PCM 5102A 3.3 V) 7](#_Toc451524639)

[2.8.1 Intermezzo I2S 8](#_Toc451524640)

[2.9 Audio amplifier (TPA2016D2 5V) 8](#_Toc451524641)

[2.10 NOR gate flipflop 9](#_Toc451524642)

[2.11 Supercap 9](#_Toc451524643)

[2.12 WS2812 led’s 10](#_Toc451524644)

[2.13 Het schema & PCB 10](#_Toc451524645)

[3 Software 11](#_Toc451524646)

[3.1 Het hoofdprogramma (app.py) 11](#_Toc451524647)

[3.1.1 Google Calendar authenticatie 11](#_Toc451524648)

[3.1.1.1 Registreren van SmartAlarmClock 11](#_Toc451524649)

[3.1.1.2 Device specifieke codes verkrijgen 12](#_Toc451524650)

[3.1.1.3 Calendar informatie opvragen 13](#_Toc451524651)

[3.1.1.4 Refresh token gebruiken 13](#_Toc451524652)

[3.1.2 Netwerk en acces point 13](#_Toc451524653)

[3.1.3 LCD aansturen 14](#_Toc451524654)

[3.1.4 Web interface verwerking/Flask 14](#_Toc451524655)

[3.1.5 RTC aansturen 15](#_Toc451524656)

[3.1.6 Rotary encoder 15](#_Toc451524657)

[3.1.7 WS2812 led’s aansturen 15](#_Toc451524658)

[3.1.8 Web interface 16](#_Toc451524659)

[3.1.8.1 Status tab 16](#_Toc451524660)

[3.1.8.2 Wifi settings tab 16](#_Toc451524661)

[3.1.8.3 Clock settings tab 16](#_Toc451524662)

[3.1.8.4 Google Calendar tab 16](#_Toc451524663)

[4 Budget/kostenraming 17](#_Toc451524664)

[5 Besluit 18](#_Toc451524665)

[6 Bijlagen 19](#_Toc451524666)

[6.1 PCB Schema’s & Lay-out 20](#_Toc451524667)

[6.1.1 De schema’s 20](#_Toc451524668)

[6.1.2 De PCB 21](#_Toc451524669)

[6.2 Scripts 22](#_Toc451524670)

[6.3 Broncode (app.py) 23](#_Toc451524671)

[6.4 Foto’s 23](#_Toc451524672)

# Hardware

## 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 2‑1: De inhoud van de Raspberry Pi Zero Essentials kit.

## Rotary encoder

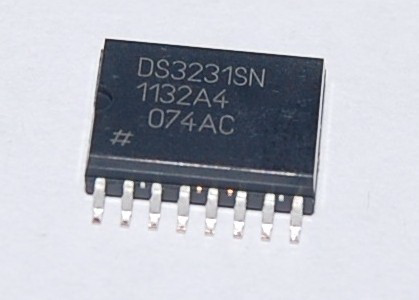
Als input voor bv. het volume van het geluid of de helderheid van het scherm, wordt gebruik gemaakt van een rotary encoder. De gebruikte rotary encoder heeft naast een rotatie functie ook een drukknop functie.



Figuur 2‑2: Een rotary encoder

## I2C RTC (3.3 V DS3231)

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



Figuur 2‑3: de DS3231 chip

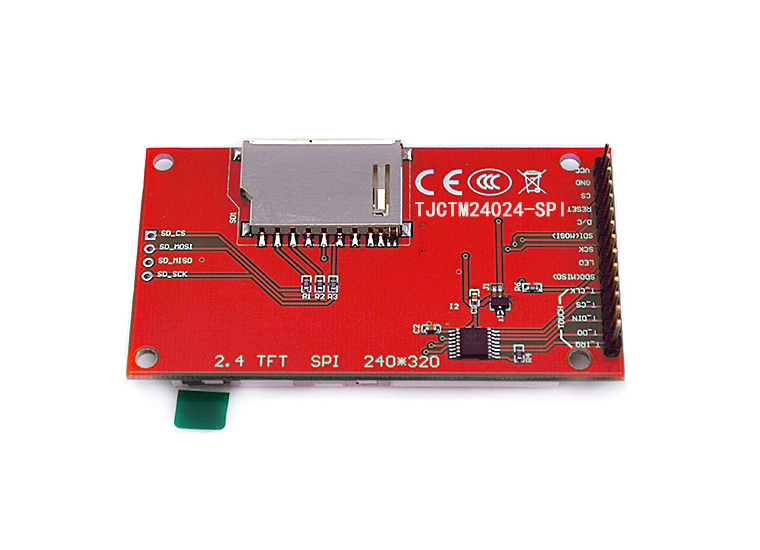
## Spanningsregelaar (MCP1703)

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

## 2.4” 240x320 spi tft LCD (3.3 V)

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

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



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

## Buzzer

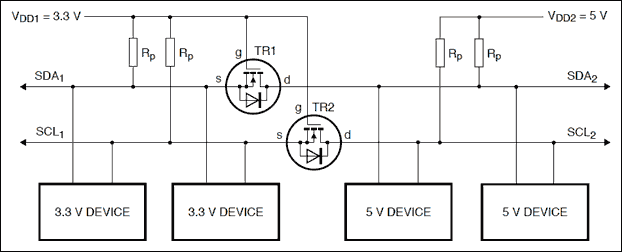
De buzzer dient hoofdzakelijk als back-up alarm. Indien de voedingspanning wegvalt, zal door de back-up voeding van de RTC deze toch nog blijven werken. Tijdens normale werking wordt het alarm in de RTC afgezet vooraleer het zijn interrupt kan geven en zullen de speakers voor het alarm zorgen. Indien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



Figuur 2‑5: De buzzer

## Levelshifter (AN10441)

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



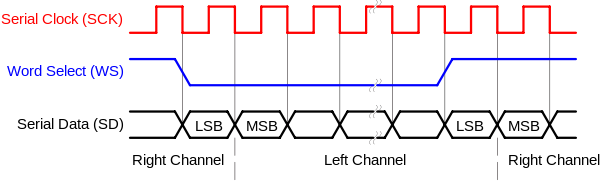
Figuur 2‑6: Shiften tussen 3.3 V en 5 V op een I2C bus.

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

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

### Intermezzo I2S

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



Figuur 2‑7: De timing van een I2S bus interface.

## Audio amplifier (TPA2016D2 5V)

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

## NOR gate flipflop

De flipflop wordt gebruikt om te kunnen bepalen hoelang de buzzer moet aflopen. Wanneer de flipflop wordt geset

!! TODO !!

## Supercap

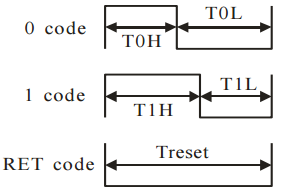
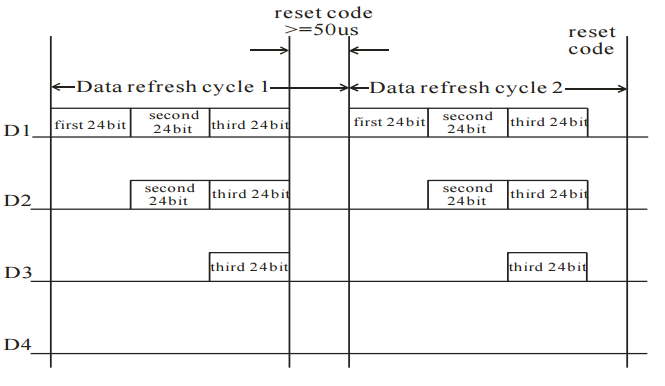
Als back-up voeding voor de basisfunctionaliteit van de klok wordt een condensator van 1.5 farad (“supercap”) gebruikt. De RTC schakelt over naar de voeding van de condensator indien de voedingsspanning wegvalt. Aangezien de RTC en flipflop slechts enkele µA gebruiken zou de backup functionaliteit zeer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



Figuur 2‑8: Een 'supercap' condensator

## WS2812 led’s

De WS2812 led’s 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.



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

## Het schema & PCB

Zie bijlage 6.1.

# Software

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

## De nodige drivers en initialisatiecode

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

### Kernel-parameters en –modules

De Linux kernel heeft de mogelijkheid om via “serial console” te worden bestuurd, hiervoor moeten in “/boot/cmdline.txt” het loglevel naar 5 worden veranderd. Om dan ook een login console te krijgen en niet enkel kernel debug informatie word 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 nodige 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 fps=23” laat de driver weten waar de relevante pinnen zijn aangesloten. De SPI pinnen moeten niet worden beschreven.
* “fps=23” bepaald de maximale vernieuwingsfrequentie.
* “speed=42000000” legt de snelheid van de SPI bus vast op 42MHz.  
  (Experimenteel bepaalde maximale frequentie voor foutloze communicatie)
* “rotate=90” draait de framebuffer 90° zodat de LCD in landschap modus kan worden gebruikt zonder extra werk in de applicatiecode.

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

### “.bash\_profile”

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

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

### Nginx

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

Deze manier van werken heeft nog als voordeel dat Nginx veel sneller start dan Flask, en dus is de web-interface altijd klaar voor dat de gebruiker naar het IP adres surft. Als dit voorkomt falen natuurlijk wel 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 na korte tijd opnieuw word geprobeerd.

## Het hoofdprogramma (app.py)

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

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

De code word hier gegroepeerd per functionaliteit.

### LCD aansturen

Om tekst op het scherm te krijgen wordt gebruik gemaakt van de Python module pygame. Deze module is bedoelt 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 in aan te sturen omdat dit minder CPU gebruikt dan de Python GPIO module.

### RTC aansturen

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

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

Om de alarmfuncties van de RTC te gebruiken zijn 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]]

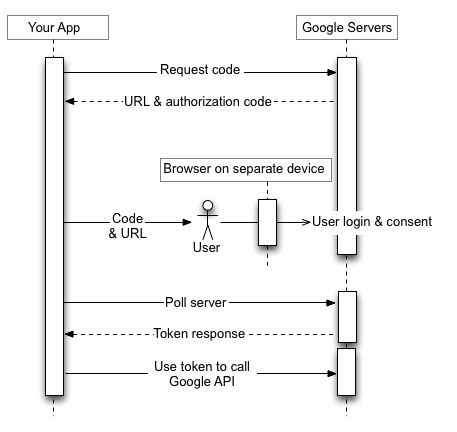
Ook deze commando’s worden gebruikt via “subprocess.call”.

### Google Calendar

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

#### Registreren van SmartAlarmClock

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



Figuur 3‑1: Schematische voorstelling van het stappenplan om toegang te krijgen tot de Google Calendar van de gebruiker. Dit is oAuth 2.0 voor embedded devices.

#### Toestemming krijgen van de gebruiken

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

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

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

#### Kalender afspraken opvragen

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

#### Een vervallen “Access token” vervangen

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

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

### Flask (Web interface API)

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

### Netwerk en acces point

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

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

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

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

Via het extern programma “iwlist” wordt de scan naar wifi netwerken uitgevoerd. De output van dit programma is echter niet optimaal (het bevat veel overbodige info en onbruikbare netwerken) dus wordt het eerst met RegEx omgezet naar een JSON vriendelijk formaat.

### Rotary encoder

Om de rotary encoder aan te sturen wordt gebruikt 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 vermijd domme typfoutjes en lang debug werk.

### WS2812 led’s aansturen

### Web interface

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

#### Status tab

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

#### Wifi settings tab

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

#### Clock settings tab

De settings tab is de belangrijkste tab na het in gebruik nemen van het toestel. Hier kan de gebruiker de SmartAlarmClock instellen naar zijn eigen wensen. Ten eerste kan het format en de size 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 size dit moet. De laatste instelling bepaalt welk type van alarm er moet worden gebruikt tijdens het wekken. De gebruiker heeft hierbij de keuze uit een muziekfile, muziekstream of 1 van de ingebouwde geluiden.

#### Google Calendar tab

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

# Budget/kostenraming

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

|  |  |
| --- | --- |
| Naam | Kostprijs |
| Wifi dongle | € 2 |
| Buzzer | € 3 |
| USB adapter | € 1 |
| RTC | € 0.50 |
| Supercap | € 2 |
| Levelshifters | € 0.20 |
| 2 x speaker | € 2.50 |
| 3.3V spanningsregulator | € 0.50 |
| Micro SD | € 8 |
| Led’s | € 0.50 |
| Power adapter | € 2 |
| Amplifier | € 2.50 |
| Rotary encoders | € 0.50 |
| PCB | € 3.50 |
| Raspberry Pi Zero essentials kit | € 8 |
| LCD scherm | € 7 |
| DAC | € 2.50 |
| 8 x WS2812 led’s | € 0.80 |
| **Totaal** | **€ 47** |

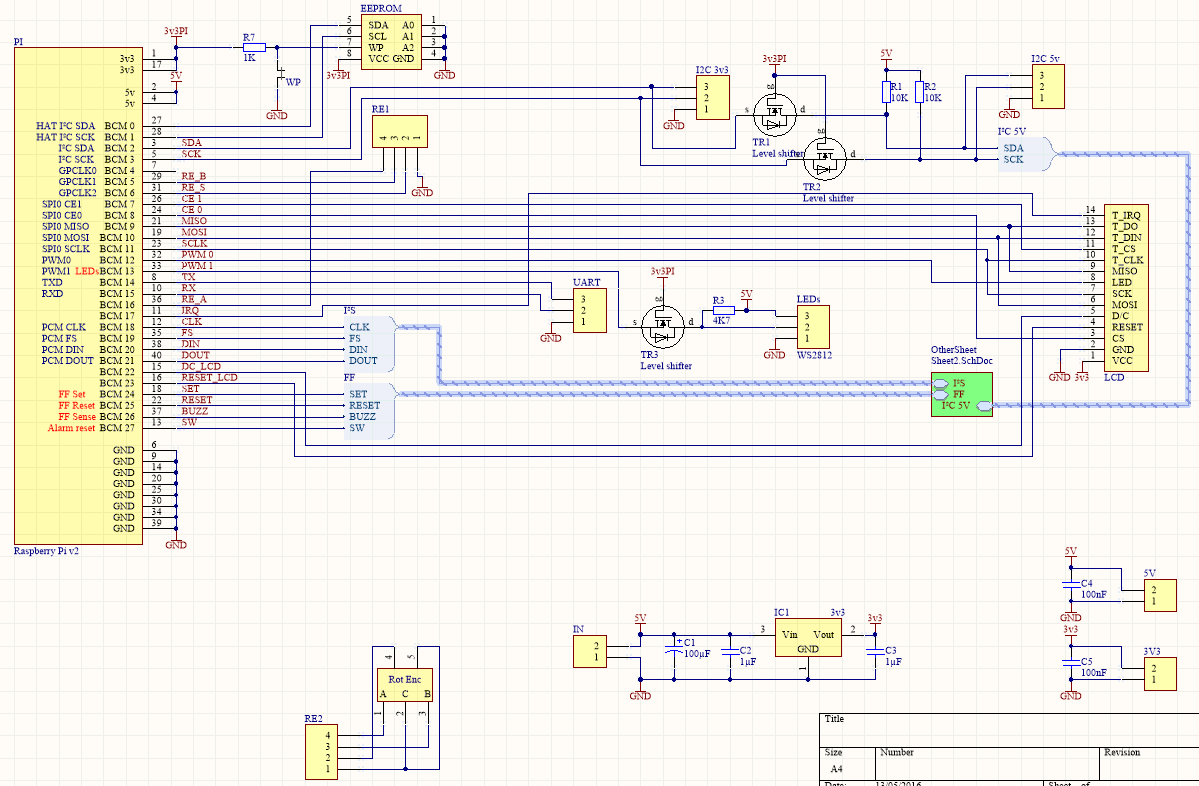
# Besluit

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 display. Aangezien er voor de bestelde display niet echt eenduidige datasheets (pin lay-out) 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 toch voor een extra tijdsdruk.

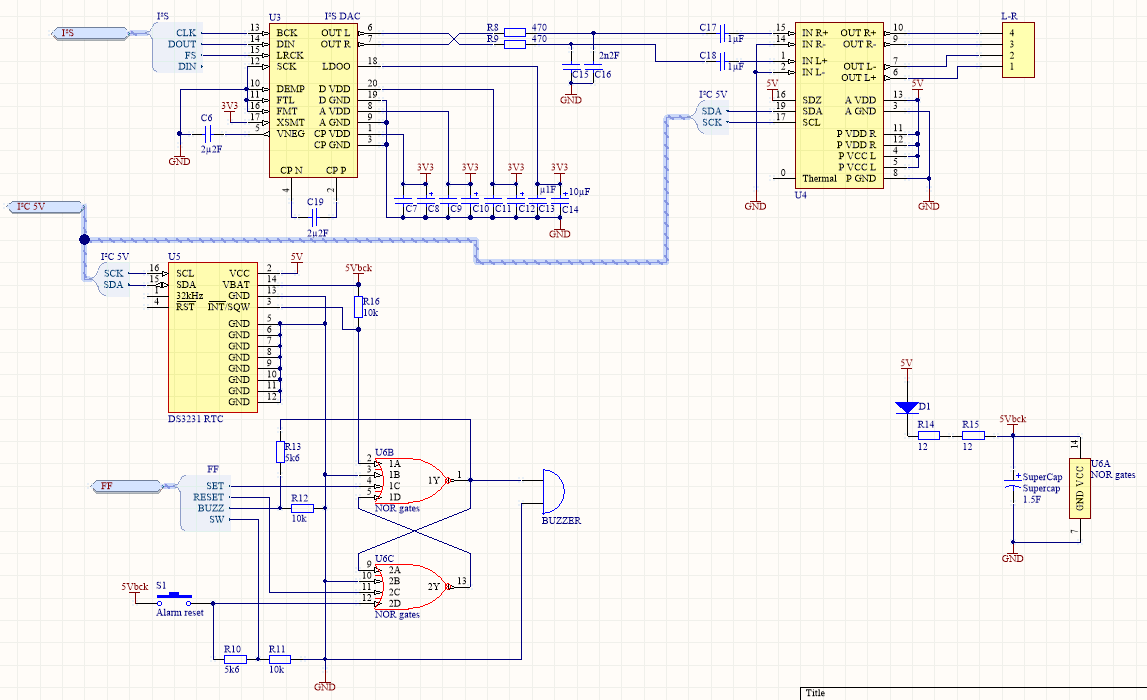
# Bijlagen

## PCB Schema’s & Lay-out

### De schema’s

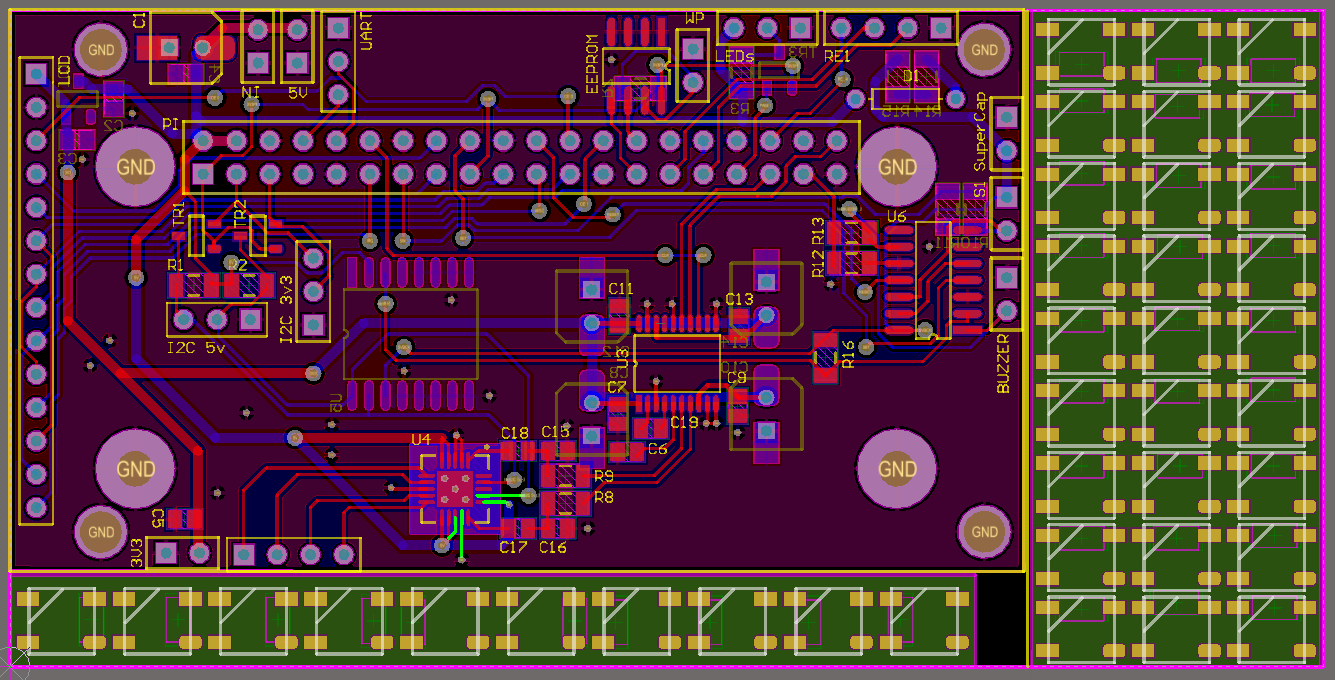


Figuur 6: PCB schema sheet 1.



Figuur 7: PCB schema sheet 2.

### De PCB



Figuur 8: 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 led’s voorzien.

## Scripts

## Broncode (app.py)

De volledige broncode is ook beschikbaar op [github.com/\*](https://github.com/dries007/ModernMasterMind).

## Foto’s