**KSOM CAMPUS SINT-PAULUS**

**Kruisven 25 - 2400 Mol**

**Geïntegreerde proef**

Schooljaar 2023-2024

Studierichting: Industriële Wetenschappen

**Automatische speelkaartendeler**

Jochem Van Deuren, 6IW

Siebe Boënne, 6IW

**KSOM CAMPUS SINT-PAULUS**

**Kruisven 25 - 2400 Mol**

**Geïntegreerde proef**

Schooljaar 2023-2024

Studierichting: Industriële Wetenschappen

**Automatische speelkaartendeler**

Jochem Van Deuren, 6IW

Siebe Boënne, 6IW

**Woord vooraf**

Jochem

Tijdens onze GIP heb ik geleerd theorie in de praktijk uit te voeren en oplossend te denken. Door aan een project te werken dat mij interesseert, heb ik niet alleen mijn motivatie vergroot, maar ook mijn vaardigheden in het programmeren in C++, 3D-tekenen en 3D-printen verder ontwikkeld.

De mentoren de heren G. Daniëls en K. Van Genechten zou ik willen bedanken voor de hulp die ze geboden hebben tijdens het jaar. Ook wil ik mijn ouders bedanken. Ten slotte zou ik mijn GIP-partner Siebe ook willen bedanken voor de aangename tijd met hem en alle dingen die we hebben bijgeleerd.

Siebe

In september dacht ik dat een GIP veel werk zou, zijn maar ik had niet gedacht dat we onderweg ook nog problemen zouden tegenkomen.  
Sindsdien heb ik ondervonden dat een GIP voor 10% bestaat uit een

initiële visie, en 90% uit zowel verwachte als onverwachte problemen oplossen. Wat me hiervan is bijgebleven, is hoe ik aangeleerde kennis van de afgelopen jaren kan gebruiken om de meest passende oplossing te vinden voor elk probleem.

Bij deze wil ik dan ook de heren K. Van Genechten en G. Daniëls bedanken voor hun hulp tijdens het jaar. Ten slotte wil ik mijn GIP-partner Jochem bedanken voor een goede samenwerking en een fijn jaar.

Inhoud

[1 Inleiding 3](#_Toc166678895)

[2 Componenten 4](#_Toc166678896)

[2.1 Blokschema 4](#_Toc166678897)

[2.2 Uitleg blokschema 4](#_Toc166678898)

[2.3 Extra componenten 5](#_Toc166678899)

[3 Onderdelen 6](#_Toc166678900)

[3.1 Voeding 6](#_Toc166678901)

[3.1.1 Batterijen 6](#_Toc166678902)

[3.1.2 Battery management system 7](#_Toc166678903)

[7](#_Toc166678904)

[3.1.3 Buck converter 8](#_Toc166678905)

[3.2 FireBeetle 2 ESP32-E 10](#_Toc166678906)

[3.3 User interface 11](#_Toc166678907)

[11](#_Toc166678908)

[3.4 Stappenmotor 12](#_Toc166678909)

[3.5 DRV8825 aansturing stappenmotor 13](#_Toc166678910)

[3.6 DC-motor 16](#_Toc166678911)

[3.7 Hall-effectsensor 17](#_Toc166678912)

[3.8 DS3231 RTC 18](#_Toc166678913)

[3.9 Neopixel 19](#_Toc166678914)

[4 PCB 20](#_Toc166678915)

[5 Simulatie 21](#_Toc166678916)

[5.1 Wokwi 21](#_Toc166678917)

[5.2 Voorbeelden 22](#_Toc166678918)

[6 Code 24](#_Toc166678919)

[7 Grafische tekeningen 24](#_Toc166678920)

[8 Besluit 25](#_Toc166678921)

[9 Bibliografie 27](#_Toc166678922)

[10 Bijlage 29](#_Toc166678923)

[10.1 Code 29](#_Toc166678924)

# 1 Inleiding

We maken een machine die speelkaarten automatisch kan delen, zodat u dit niet handmatig hoeft te doen. U kunt via een interface een keuze maken welk spel u wil spelen en met hoeveel personen u speelt. De machine begint vervolgens te delen tot iedereen zijn hoeveelheid kaarten heeft gekregen.

Iedereen heeft waarschijnlijk al eens wiezen, zetten of pesten gespeeld. Deze populaire spellen worden gespeeld met twee of meer personen. Hierbij worden de kaarten handmatig verdeeld in een specifieke volgorde en met een vastgesteld aantal kaarten. Hoe gemakkelijk zou het zijn moest u dit niet meer zelf hoeven te doen?

Het doel van onze GIP was om het te presenteren tijdens de opendeurdag op 5 mei, en tegelijkertijd ervaring op te doen in het succesvol afronden van een project. We hebben problemen die we onderweg tegenkwamen opgelost door gebruik te maken van gekende theorie.

De noodzakelijke kennis om onze GIP te begrijpen omvat een basiskennis van elektromotoren, ook kennis van programmeren in C++. Ten slotte is een goede kennis van elektronica en elektriciteit ook essentieel.

Eerst bespreken we de componenten die we gebruikt hebben. Daarna wordt elk onderdeel volledig uitgelegd. Nadien hebben we het over de simulatie in wokwi.com en wordt heel het elektrisch gedeelte gepresenteerd met de nodige uitleg over de interface. In hoofdstuk 5 hebben we het over de werking van de code. Ten slotte gaan we de grafische tekeningen nog bespreken.

De problemen die we tegenkwamen omvatten een gebrek aan materiaal, waardoor ons oude ontwerp niet goed functioneerde. Daarnaast hadden we een te klein rubberen ringetje rond ons wieltje, daardoor was er onvoldoende wrijving tussen de kaart en het wieltje. Het gevolg hiervan was dat de kaarten niet afgeschoten konden worden.

# 2 Componenten

## 2.1 Blokschema

Afbeelding met elektronica, Elektronische engineering, machine, tekst

Automatisch gegenereerde beschrijving

1 Blokschema

## 2.2 Uitleg blokschema

We gebruiken een systeem met 3 lithium-ioncellen om onze componenten te voorzien van 10.6 - 12.6 volt. Dankzij de BMS kunnen we ervoor zorgen dat de batterij altijd optimaal wordt geladen en ontladen. Deze spanning wordt vervolgens verlaagd naar 5 volt, waarmee we zowel de FireBeetle als de DC-motor aansturen. De stappenmotor ontvangt echter rechtstreeks de volle 12.6 volt.

Met behulp van de FireBeetle lezen we de Hall-sensor uit, waarmee we de nulpositie kunnen bepalen. De stappenmotordriver stuurt vervolgens de stappenmotor aan om de machine te laten draaien. De DC-motor, die verantwoordelijk is voor het delen van kaarten, wordt ook aangestuurd door de FireBeetle.

Ten slotte worden de knoppen ingelezen door de FireBeetle en worden ze gebruikt om het OLED-scherm te bedienen, waardoor een volledig functioneel en geïntegreerd systeem ontstaat.

## 2.3 Extra componenten

**Afbeelding met tekst, elektronica, Elektronische engineering, scherm

Automatisch gegenereerde beschrijving**

2 Extra componenten

Dit systeem is een aan-uitmechanisme waarbij de FireBeetle kan worden in-uitgeschakeld met een eenvoudige knop. Wanneer je de knop indrukt, wordt er een spanning van 3 volt naar de FireBeetle gestuurd. Hierdoor start de FireBeetle op en voert hij een code uit die een relais activeert, dat fungeert als een overnamecontact. Vervolgens kun je de knop loslaten en blijft de FireBeetle ingeschakeld.

Wanneer je nogmaals op de knop drukt, zal de FireBeetle het relais uitschakelen. Hierdoor valt de spanning van de FireBeetle weg en stopt hij met werken. Dit biedt een eenvoudige en doeltreffende manier om de FireBeetle te bedienen en het systeem te activeren of te deactiveren.

We gebruiken een spanningsdeler om de spanning over een cel te verminderen voordat we deze met de FireBeetle meten. Hierdoor kunnen we de batterijspanning nauwkeurig aflezen en weergeven op het OLED-scherm. Als indicator voor het laadniveau maken we gebruik van een Neopixel. De tijd wordt gesynchroniseerd met een realtimeklok via I2C, waardoor we deze kunnen tonen op het OLED-scherm.

# 3 Onderdelen

## 3.1 Voeding

3.1.1 Batterijen

Afbeelding met plastic, smartphone, batterij

Automatisch gegenereerde beschrijving

Wij gebruiken 3 lithium-ionbatterijen 18650. Deze waarde staat voor de afmetingen van de batterij, namelijk een diameter van 18 mm en een lengte van 65 mm. Elke cel heeft een nominale voltage van 3,7 volt. Volledig ontladen is elke cel 3 volt en volledig opgeladen is deze 4,2 volt. Deze waarden kunnen afwijken afhankelijk van de ontladingssnelheid en temperatuur. Wij plaatsen deze cellen in serie en werken dus met een maximumvoltage van 12,6 volt en een minimum van 9.

3 Lithium-ioncel (18650)

De batterij bestaat uit een aantal essentiële onderdelen:

**Anode (negatieve elektrode):** De anode van een lithium-ionbatterij is meestal gemaakt van grafiet. Tijdens het ontladen van de batterij geeft de anode lithium-ionen en elektronen vrij.

**Kathode (positieve elektrode):** De kathode is vaak samengesteld uit een metaaloxide, zoals lithium-kobaltoxide. Bij ontlading van de batterij accepteert de kathode de lithium-ionen en elektronen.



4 Galvanische cel

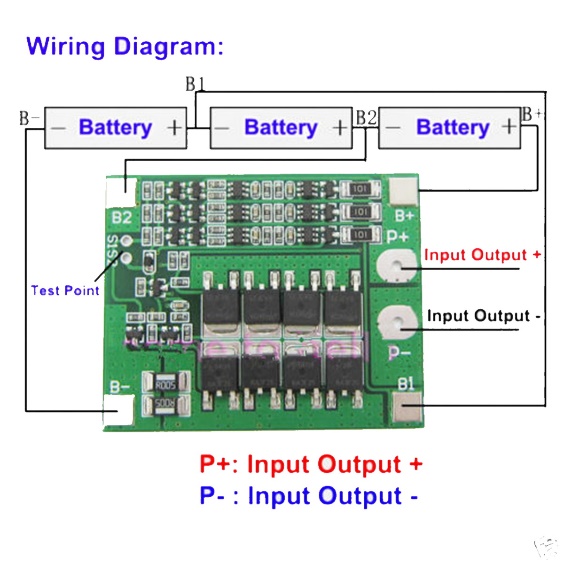
**Elektrolyt:** Een geleidende vloeistof, het elektrolyt, faciliteert het transport van lithium-ionen tussen de anode en kathode. Dit elektrolyt bevat zouten die dienen als geleiders voor de lithium-ionen.

**Separator:** Een poreus materiaal, de separator, scheidt de anode en kathode om kortsluiting te voorkomen. Desondanks laat de separator de doorgang van lithium-ionen toe.

**Elektrische stroom:** Tijdens het ontladen stromen de lithium-ionen van de anode naar de kathode door het elektrolyt, terwijl elektronen een externe stroomkring vormen om naar de kathode te bewegen. Dit genereert een elektrische stroom die apparaten van stroom voorziet.

**Opladen:** Bij het opladen wordt het proces omgekeerd. Een externe stroombron levert elektriciteit aan de batterij, waardoor lithium-ionen terugkeren naar de anode. Tegelijkertijd stromen elektronen via de externe stroomkring van de kathode naar de anode.

3.1.2 Battery management system



Een battery management system (BMS) wordt gebruikt om de lithium-ioncellen veilig te kunnen op- en ontladen. Hij beschermt de batterijen tegen overladen en te ver ontladen. Ook zorgt hij er voor dat de cellen gelijkmatig geladen en ontladen worden.

5 Battery Management System

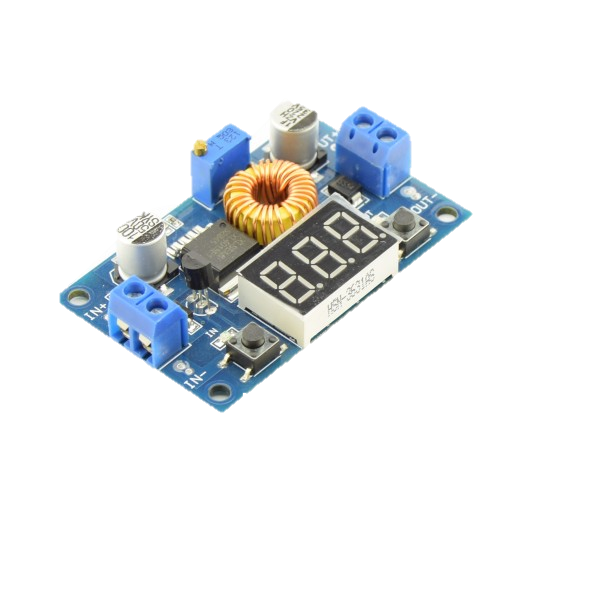
Afbeelding met lijn, schermopname, Rechthoek, Lettertype

Automatisch gegenereerde beschrijvingWe sluiten onze 3 cellen in serie aan met tussen 2 cellen een midden aftakking op de BMS.

We lezen ook de spanning over één cel analoog binnen via een spanningsdeler om het batterij percentage te kunnen weergeven op het scherm.

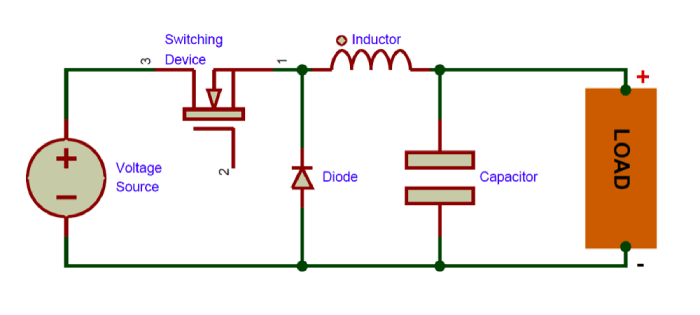
6 Gebalanceerde cellen

3.1.3 Buck converter

De buck converter of ook wel de step down converter genoemd is een elektronisch circuit dat wordt gebruikt om de spanning van een voedingsbron te verlagen.

7 Step down converter

Het principe achter een buck converter is gebaseerd op het schakelen van een spoel, een schakelaar (P-kanaal MOSFET), een diode en een condensator.



8 PWM-schakeling

Afbeelding met tekst, lijn, Perceel, diagram

Automatisch gegenereerde beschrijvingTijdens de inschakelfase wordt de schakelaar/ transistor kortstondig gesloten. Daardoor neemt de stroom door de spoel toe en slaat deze energie op. Vervolgens wordt de schakelaar geopend tijdens de uitschakelfase. Dan geeft de inductor zijn energie af aan de uitgangscondensator en de belasting via een schottkydiode.

9 Inductantie capacitantie grafiek

De schakelaar/ MOSFET zorgt dus eigenlijk voor een PWM-signaal.

Afbeelding met diagram, lijn, Lettertype, schermopname

Automatisch gegenereerde beschrijving

10 Duty cycle

Door het aanpassen van de schakeltijd/ duty cycle kan de buck converter de gemiddelde uitgangsspanning regelen. Om de rimpels in de uitgangsspanning te verminderen, wordt een condensator als filter toegevoegd. Wanneer de belasting verandert moet een regelkring worden gebruikt om de uitgangsspanning te bewaken en aanpassingen te maken aan de schakeltijd om de gewenste uitgangsspanning te kunnen blijven behouden.

Afbeelding met schermopname, groen, diagram, lijn

Automatisch gegenereerde beschrijving

11 Aansturing mosfet 1

Om de uitgangsspanning van de buck converter stabiel te houden, wordt een verschilversterker en een comparator (opamps) gebruikt. De uitgangsspanning van de buck converter wordt verlaagd door een spanningsdeler en toegevoerd aan de inverterende ingang van de eerste opamp. Deze eerste opamp fungeert als een verschilversterker, waarbij de fout wordt berekend tussen de werkelijke uitgangsspanning en een referentiespanning. Het doel is om deze fout te minimaliseren en zo de twee ingangsspanningen gelijk te houden. Wanneer de spanning aan de inverterende ingang afneemt, resulteert dit in een stijging van de uitgangsspanning van de eerste opamp.

Afbeelding met schermopname, plein, Kleurrijkheid, ontwerp

Automatisch gegenereerde beschrijvingDat signaal wordt vervolgens aangesloten op de comparator (vergelijkende opamp) samen met een zaagtand van 150KHz. De uitgangsspanning van de opamp is het PWM-signaal dat de MOSFET of schakelaar aanstuurt.

12 Verschilversterker

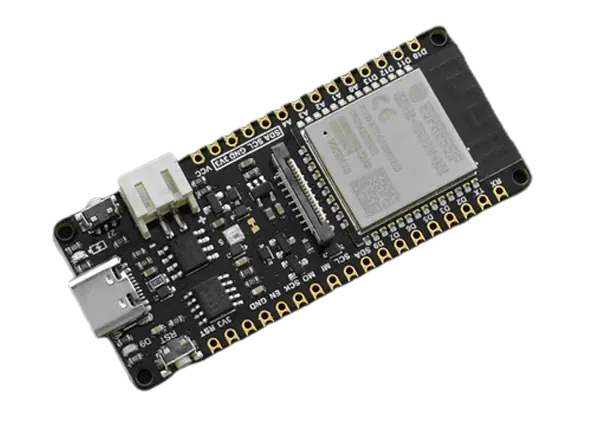
Afbeelding met schermopname, plein, Kleurrijkheid, Rechthoek

Automatisch gegenereerde beschrijvingWanneer de eerste opamp een andere uitgangsspanning geeft, zal de vergelijkende opamp een ander PWM-signaal uitsturen en verandert dus zo de uitgangsspanning van je buck converter.

13 Comparator

Dankzij dit schakelproces kan een buck converter de gewenste uitgangsspanning produceren met een hogere efficiëntie dan lineaire spanningsregelaars, vooral bij aanzienlijke spanningsverschillen tussen de invoer- en uitvoerspanningen.

## 3.2 FireBeetle 2 ESP32-E

De FireBeetle ESP32-E is een ontwikkelingsbord dat wordt aangedreven door een ESP32 dual-coreprocessor. Het bord ondersteunt zowel wifi als bluetooth (bluetooth 4.2) connectiviteit. Met 4 MB flashgeheugen en 520 KB SRAM biedt het voldoende ruimte voor het uitvoeren van applicaties en het opslaan van gegevens. Afhankelijk van het exacte model heeft de FireBeetle ESP32-E tussen de 12 digitale pinnen, evenals 5 analoge ingangen, waardoor het geschikt is voor het aansluiten van verschillende sensoren en apparaten.

14 FireBeetle 2 ESP32-E

Het bord ondersteunt programmering via de Arduino IDE en MicroPython, waardoor ontwikkelaars gemakkelijk kunnen beginnen met het bouwen van projecten. Met zijn compacte formaat en laag stroomverbruik is de FireBeetle ESP32-E ideaal voor batterijgevoede en draagbare toepassingen. De specificaties kunnen echter variëren afhankelijk van het exacte model en de versie van het bord.

Wij hebben gekozen voor een FireBeetle omdat deze wel 3 keer kleiner is dan een Arduino Mega, met toch 16 keer zoveel flashgeheugen. Deze FireBeetle voeden wij met 5V van de buck converter.

## 

## 3.3 User interface



Omhoog

Omlaag

Cancel

Enter



15 OLED-menu

## 

## 

16 UI met drukknoppen

Wij gebruiken een OLED-scherm van 2,42 inch met een resolutie van 128x64 pixels.

**I2C-interface**: Dit verwijst naar het communicatieprotocol dat wordt gebruikt om met het display te communiceren. I2C (Inter-Integrated Circuit) is een serieel communicatieprotocol dat vaak wordt gebruikt om langzame randapparatuur aan te sluiten op microcontrollers en microprocessoren. Het maakt het mogelijk om meerdere apparaten op dezelfde bus aan te sluiten met slechts twee draden (klok en data).

**Drukknoppen:**

17 Drukknop schakeling



GND

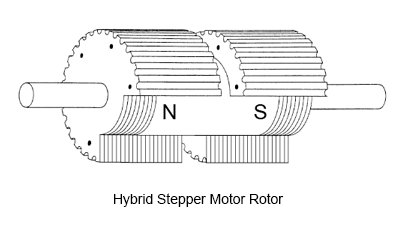
Digitale pin

Onze gebruikersinterface maakt gebruik van 4 drukknoppen (*selecteer*, *annuleren*, *omhoog* en *omlaag*) om de invoer van de gebruiker te lezen. De digitale ingangen van de FireBeetle gebruiken de pinmodus "INPUT\_PULLUP", wat betekent dat de digitale ingang standaard op 3.3V staat. Dit wordt bereikt door gebruik te maken van een interne pull-upweerstand in de FireBeetle. Wanneer de knop wordt ingedrukt, wordt de digitale ingang verbonden met 0V, wat gedetecteerd zal worden. Dit betekent dat in de code een ingedrukte knop als "LOW" wordt gelezen en een onbediende knop als "HIGH".

## Nema17 Pancake stepper 22mm - 3DINTHEBOX BV3.4 Stappenmotor

18 Nema-17 stappenmotor

Een bipolaire stappenmotor is een elektrische motor die wordt gebruikt in toepassingen waar nauwkeurige positionering en controle van bewegingen essentieel zijn. Deze worden dan ook gebruikt in CNC-machines, 3D-printers en robotica.

Het onderscheidende kenmerk van een bipolaire stappenmotor is het feit dat deze twee spoelen heeft en dus 4 aansluitingen, twee voor elke spoel. De wikkelingen worden in beide richtingen bekrachtigd en moeten daarom elk in een MOSFET H-brugschakeling geplaatst worden.

Afbeelding met tekst, schermopname, cirkel, logo

Automatisch gegenereerde beschrijvingDe kern van de stappenmotor is de magnetische rotor, die reageert op de magnetische velden die worden gegenereerd door de stroom die door de spoelen vloeit. Één helft van de rotor is noordgepolariseerd en de andere helft is zuidgepolariseerd.

19 As bipolaire stappenmotor

Afbeelding met tekst, schermopname, cirkel, Lettertype

Automatisch gegenereerde beschrijvingIn plaats van een continue rotatie beweegt de stappenmotor in stappen. Het aantal stappen per omwenteling hangt af van het ontwerp van de motor. Zowel de noord- als zuidgepolariseerde kant bevat 50 tanden die ten opzichte van elkaar geschrankt staan. De stator heeft tweemaal 48 tanden. Dit betekent dat niet alle tanden tegelijkertijd kunnen worden uitgelijnd.

20 Frame bipolaire stappenmotor

Door een stroom in een bepaalde richting door een spoel te sturen, wordt deze magnetisch, waardoor de rotor wordt aangetrokken. Wanneer we vervolgens de ene na de andere spoel bekrachtigen, zal de motor telkens een klein stukje draaien, ook wel een stap genoemd.

21 Spoelen bipolaire stappenmotor

Afbeelding met cirkel, schermopname, kunst, wiel

Automatisch gegenereerde beschrijvingDe grootte van een stap wordt berekend als

22 Magneetpolen bipolaire stappenmotor

(360° / 50 tanden) / 4 = 1,8°. De stator heeft bij de ene spoel twee mogelijke posities om uit te lijnen en bij de andere spoel ook twee, vandaar de "4" in de formule.

Bij de Nema-17 stappenmotor bedraagt het aantal stappen per omwenteling 200.

Om de bipolaire stappenmotor te laten draaien, wordt een speciale stappenmotorbestuurder gebruikt, zoals de DRV8825. Deze bestuurder regelt de stroomtoevoer naar de spoelen en bepaalt daarmee de bewegingsrichting en -snelheid van de motor.

## 3.5 DRV8825 aansturing stappenmotor



23 DRV8825 aansluiting

De DRV8825 is een stappenmotorbestuurder. Deze wordt gebruikt om bipolaire stappenmotoren aan te sturen. Stappenmotoren bewegen in stappen in plaats van een vloeiende beweging. Daardoor zijn ze ideaal voor toepassingen waar precisie en positionering cruciaal zijn.

Afbeelding met tekst, schermopname, Lettertype, nummer

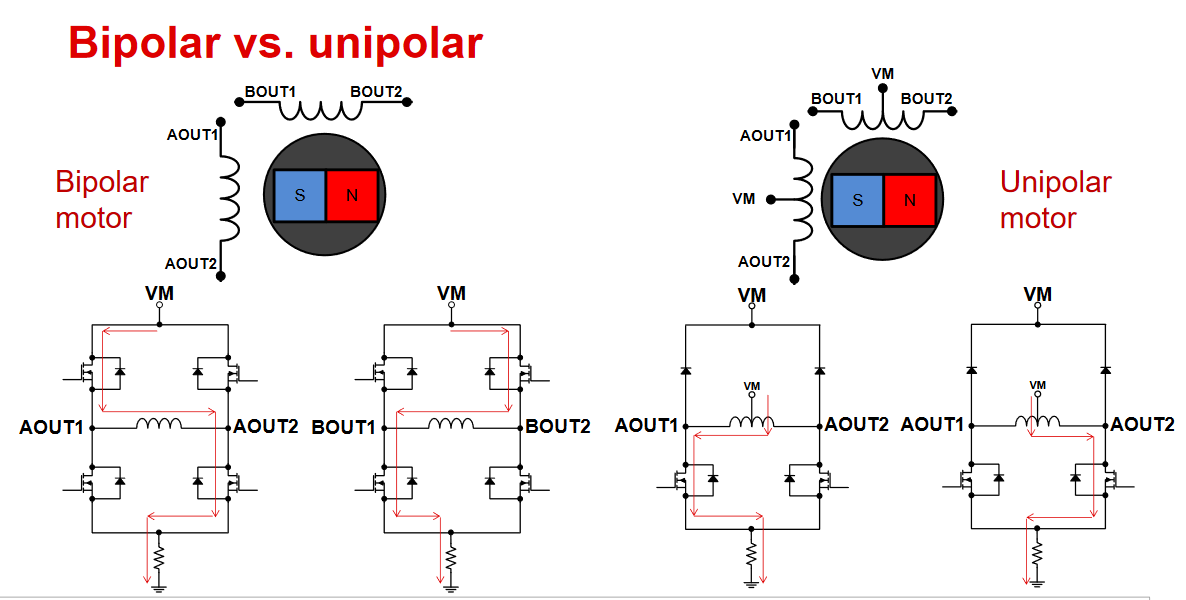
Automatisch gegenereerde beschrijving

Een opvallende functie van de DRV8825 is de mogelijkheid tot microstepping. Dat verdeelt elke stap van de stappenmotor verder in kleinere stappen, wat resulteert in soepelere bewegingen en een verbeterde nauwkeurigheid. De stroom die door de stappenmotor vloeit, wordt geregeld door een ingebouwde stroomregeling. Dit kan worden aangepast met een potentiometer op de module om oververhitting te voorkomen en de prestaties af te stemmen op de specificaties van de stappenmotor.

24 Microstepping op DRV8825

De DRV8825 wordt typisch aangestuurd met dir- (richting) en stepsignalen(stap). Door deze signalen te pulseren, kan de stappenmotor in de gewenste richting bewegen met de gewenste snelheid. Bovendien kan de snelheid van de stappenmotor worden geregeld met behulp van een PWM-signaal (Pulse Width Modulation).

Het gebruik van de DRV8825 vereist een externe stroombron, meestal geleverd door een aparte voeding. Het is essentieel om de voedingsspanning en stroomcorrecties correct in te stellen volgens de specificaties van de stappenmotor om optimale prestaties te bereiken.



25 Dubbele H-brug

In de DRV8825 worden twee H-bruggen toegepast. Een H-brug is een elektronische schakeling die de stroom in beide richtingen door een spoel kan leiden. Bij het toepassen van een stroom op een spoel zal deze zich polariseren in de noordelijke of zuidelijke richting.

Een H-brug bestaat uit vier schakelaars, meestal N-kanaal-MOSFET’s vanwege hun lagere weerstand tussen de drain en source in vergelijking met P-kanaal-MOSFET’s. Door de formule P = I² \* R begrijpen we dat N-kanaal-MOSFET’s minder warmteverlies veroorzaken. In een H-brug bevinden zich vier N-kanaal-MOSFET’s waarvan de twee bovenste high site worden geschakeld en dus voorzien zijn van bootstrapping.

Een bipolaire stappenmotor heeft twee spoelen die afwisselend worden geactiveerd om de motoras te laten roteren. Elke spoel vereist vier schakelaars om de stroomrichting te regelen, wat resulteert in de noodzaak van in totaal acht schakelaars voor 2 spoelen om een bipolaire stappenmotor aan te sturen.

De DRV8825 is voorzien van twee H-bruggen omdat het is ontworpen om twee spoelen van een bipolaire stappenmotor onafhankelijk aan te sturen. Deze configuratie maakt het mogelijk de stappenmotor nauwkeurig te positioneren door de stroom in elke spoel in de juiste volgorde en richting te sturen.

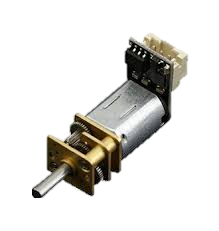
Afbeelding met diagram, tekst, lijn, Plan

Automatisch gegenereerde beschrijvingBootstrapping wordt gebruikt om de gates van de bovenste MOSFET’s in de H-brug aan te sturen met een spanning die hoger is dan de voedingsspanning van het stuurprogramma. Hierdoor kunnen de bovenste MOSFET’s volledig worden ingeschakeld, zelfs wanneer de voedingsspanning niet voldoende hoog is om de MOSFET’s volledig te openen.

26 Bootstrapping

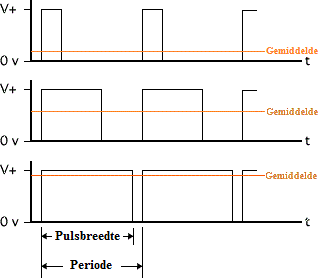
Dit wordt vaak bereikt door condensatoren te gebruiken om tijdelijk een hogere spanning op te slaan, en deze hogere spanning wordt dan toegepast op de gates van de bovenste MOSFET’s wanneer ze worden ingeschakeld. Dit verhoogt de efficiëntie van de H-brug doordat de bovenste MOSFET’s sneller en vollediger kunnen schakelen, wat resulteert in minder vermogensverlies.

## 3.6 DC-motor



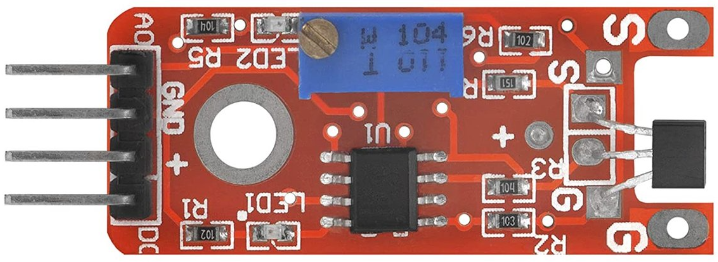
27 DC-motor met servocontroller

Dit is een standaard DC-motor met reductie 30:1 (Er zijn 30 omwentelingen op de motoras nodig om 1 omwenteling op de uitgaande as te bekomen). Een servodriver stuurt deze motor aan. Deze driver heeft 3 pinnen als aansluiting. Een GND-pin, een positieve pin met spanning van 3-8V en een pin waar je een PWM-signaal op aansluit.

Dit PWM-signaal wordt door de FireBeetle gegenereerd aan de hand van de servo.h library. Hoe breder de pulsen, hoe sneller de motor gaat draaien. Onbelast heeft deze een snelheid van 730 rpm (rotaties per minuut). Wij gebruiken deze motor omdat deze compact is en gemakkelijk aangestuurd kan worden in 2 richtingen.

28 Duty cycle voor snelheidscontrole

## 3.7 Hall-effectsensor

Een Hall-effectsensor (magneetsensor) maakt gebruik van het Hall-effect, een fenomeen dat optreedt wanneer een stroomvoerende geleider wordt blootgesteld aan een magnetisch veld. De sensor zelf bestaat meestal uit een dunne plaat, vaak gemaakt van halfgeleidermateriaal, waar een constante stroom doorheen wordt gestuurd. Plaats deze plaat in een magnetisch veld, en het Hall-effect komt in actie.

Wanneer de plaat in het magnetische veld is, ontstaat er een Hall-spanning loodrecht op zowel de stroomrichting als het magnetische veld. Deze Hall-spanning is evenredig met de sterkte en richting van het magnetische veld waaraan de sensor is blootgesteld. Het gegenereerde Hall-effectsignaal wordt vervolgens gemeten door de elektronica van de sensor. Deze meetwaarde wordt dan omgezet in digitale en analoge signalen.

29 Hall-effectsensor

Afbeelding met schermopname, tekst, ontwerp

Automatisch gegenereerde beschrijving

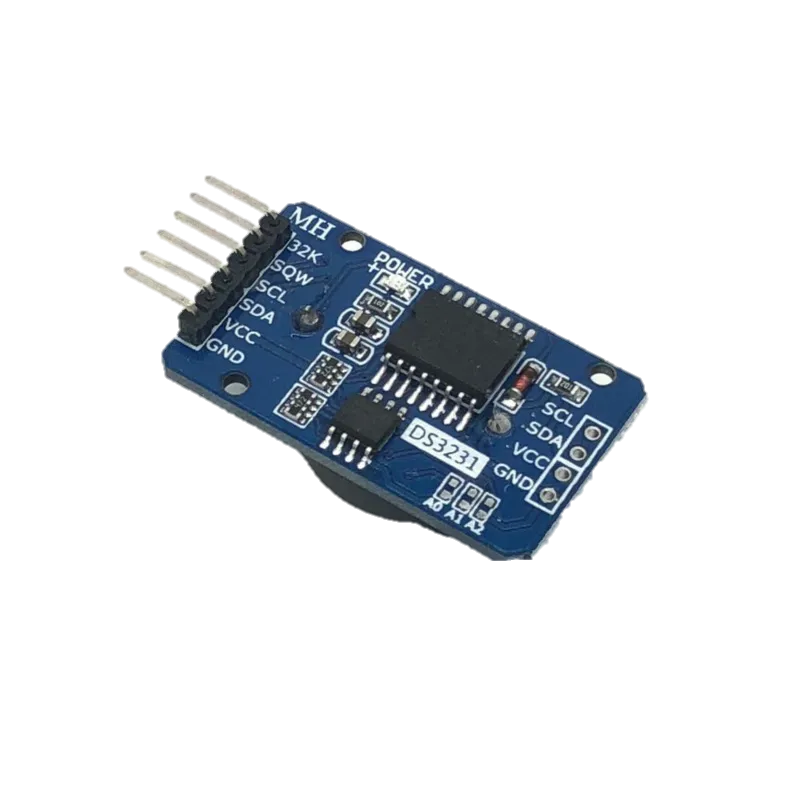
Wij gebruiken enkel het analoge signaal dat we met de FireBeetle binnen lezen.

30 Hall-effect 1

## 3.8 DS3231 RTC

Een RTC (real time clock) -module is een component die gebruikt wordt om de tijd en datum verder te tellen en op te slaan. Hij kan dit doet aan de hand van een siliciumdioxide (SiO2)-kristal, ook wel kwarts genoemd, dat trilt tegen een frequentie van 32kHz. Deze frequentie kan echter variëren naargelang de temperatuur. Hiervoor gebruikt de DS3231 een “temperature compensated crystal oscillator” (TCXO), wat betekent dat de temperatuur gemeten wordt en hij zo voor de frequentievariatie kan compenseren. Dit alles zorgt voor een precisie van ± 3ppm, wat overeen komt met 1min/jaar.

31 Real time clock



De DS3231 maakt gebruik van I2C-seriëlebuscommunicatie, wat de mogelijkheid biedt om andere componenten op dezelfde IC te plaatsen.

Zo heeft de DS3231 ook een klein opslaggeheugen (rode cirkel) van 56 bytes wat gebruikt kan worden door de gebruiker via dezelfde I2C-aansluiting. De temperatuur kan ook worden uitgelezen via I2C.  
Verder heeft de DS3231 ook nog een 32K-pin waarop je een zuivere frequentie van 32kHz kunt uitlezen, en een SQW-pin om alarmen in te stellen.

## 3.9 Neopixel

Afbeelding met tube

Automatisch gegenereerde beschrijving

32 Neopixel

Een Neopixel is een merknaam voor adresseerbare RGB LED's, geproduceerd door Adafruit. Deze LED's zijn populair vanwege hun compacte formaat, heldere kleuren en het vermogen om individueel aangestuurd te worden.

**Adressering**: Neopixels zijn adresseerbaar, wat betekent dat je elke LED afzonderlijk kunt aansturen zonder de anderen te beïnvloeden. Elke LED heeft een uniek digitaal adres, waardoor je commando's kunt sturen om de kleur en helderheid ervan te regelen.

**Data Protocol**: Neopixels communiceren via een digitaal seriëel protocol, meestal het WS2812- of SK6812-protocol. Dit betekent dat je een reeks gegevens verzendt die de kleuren van elke LED in de strip definiëren. De gegevens worden doorgegeven van de ene LED naar de andere, zodat je een lange keten van LED's kunt aansturen met slechts een paar draden.

**Programmeren**: Om Neopixels te gebruiken, moet je ze programmeren om de gewenste kleuren en effecten weer te geven. Dit kan worden gedaan met behulp van een verscheidenheid aan programmeertalen en ontwikkelomgevingen, afhankelijk van de microcontroller die je gebruikt.

Kortom, Neopixels bieden een flexibele en krachtige manier om kleurrijke verlichtingseffecten toe te voegen aan projecten, variërend van kunstinstallaties en kostuums tot domotica en hobbyprojecten.

De neopixel wordt gebruikt als een indicator voor de batterijspanning. Wanneer de batterijspanning boven de 20% ligt zal deze groen branden, daaronder zal hij rood knipperen. Tijdens het opladen knippert de neopixel groen totdat de batterij volledig is opgeladen, waarna hij continu groen brandt.

# 4 PCB

Afbeelding met stroomkring, Elektronische engineering, Elektronisch onderdeel, elektronica

Automatisch gegenereerde beschrijving

33 Printplaat

Een PCB (printed circuit board) is een dunne plaat gemaakt van een niet-geleidend materiaal, zoals glasvezel of epoxy, waarop elektronische componenten zijn gemonteerd en verbonden door geleidende sporen van koper die op het oppervlak van de plaat zijn gedrukt of ingebed.

Afbeelding met schermopname, stroomkring

Automatisch gegenereerde beschrijving

Wij hebben onze PCB getekend met het programma EasyEDA. Al onze componenten zijn hier op gesoldeerd en zijn inwendig verboden met banen (tracks). De rode lijnen stellen de ‘top layer’ voor en de blauwe lijnen de ‘bottom layer’. Deze connecties zijn van een bepaalde breedte voorzien om de juiste hoeveelheid stroom aan te kunnen.

34 Verbindingen printplaat

De printplaat beschikt ook over kleine opening (via’s) waardoor geleidende paden van de ene naar de andere laag kunnen gaan. Dit maakt het mogelijk om verbindingen tussen verschillende lagen van de printplaat tot stand te brengen.

De gele laag is de ‘top silk layer’ waarop je vormen en tekst kunt plaatsen om duidelijk te maken welke componenten op welke plaats gesoldeerd moeten worden

Het voordeel van een printplaat is dat het compact is. Ook is het betrouwbaar omdat de kans op losse bedrading en slechte verbindingen verminderd. Dit is een groot voordeel aangezien wij veel componenten en dus veel verbindingen hebben.

Toen het design af was bestelde we deze printplaat via JLCPCB. Hij werd gemaakt in China en werd dan naar ons opgestuurd.

# 5 Simulatie

## 5.1 Wokwi

Afbeelding met Lettertype, typografie, wit, kalligrafie

Automatisch gegenereerde beschrijving

Bij het gebruiken van hardware kan er vaak iets misgaan denk aan slechte verbindingen of kapotte componenten, maar hiervoor bestaat een oplossing. Tijdens het maken van onze GIP maakten we vaak gebruik van de online Wokwi

simulator. Hier hebben we geen nood aan hardware maar kunnen we wel nieuwe code testen, wat enorm helpt met de efficiëntie. Andere simulatieprogramma’s voor Arduino/ESP32 zijn Tinkercad en Xevro. We kozen echter voor Wokwi omdat zij enkele onderdelen hadden die andere simulators ons niet konden bieden, zoals werken met een ESP32 en het invoegen van een OLED-scherm.

35 Wokwi logo

Deze simulatie is zeer handig bij het schrijven van nieuwe code voor de user interface omdat je kleine blokjes kunt schrijven en dan snel testen.

In totaal hebben we 12 Wokwi simulaties die allerlei soorten code testen.

## 

Een simulatie moet natuurlijk zo realistisch mogelijk zijn. Dit is in de realiteit echter niet altijd het geval. Zo is er bij Wokwi bijvoorbeeld geen enkele soort spanningsbron aanwezig, wat het lastig maakt sommige delen code te testen. Toch beperkt de afwezigheid van voeding de simulatie niet te hard omdat je voor geen enkel van de componenten in de simulatie spanning moet voorzien. Het is dus niet op alle vlakken even realistisch maar wel geschikt om grote delen van onze code te testen.

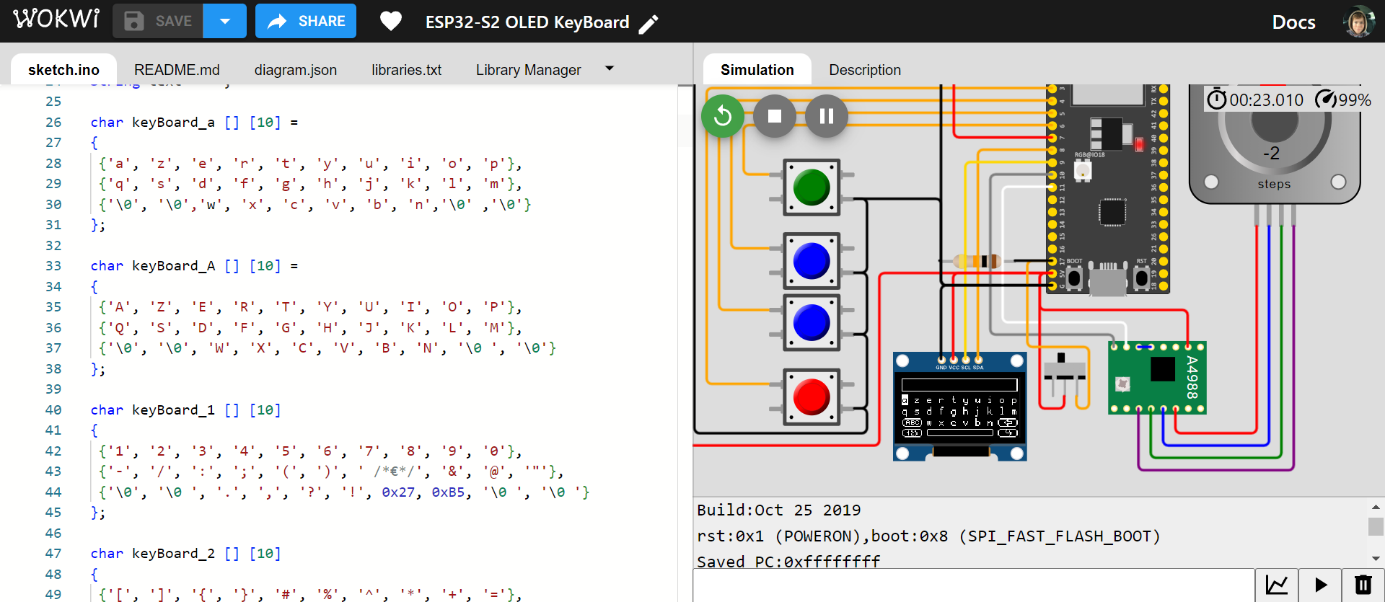
## Afbeelding met tekst, schermopname, software, Computerpictogram Automatisch gegenereerde beschrijving5.2 Voorbeelden

36 GIP simulatie

36 GIP simulatie

Dit is een voorbeeld van een Wokwi-simulatie. Rechts zien we het circuit waar alle componenten staan. Links zien we het venster waarin de code geschreven word.

Hier zien we een screenshot van de simulatie waarin de volledige GIP-simulatie staat. Rechts staan alle componenten behalve een Hall-sensor (vervangen door gele drukknop) en een DC-motor omdat deze niet beschikbaar waren in de simulatie. Links zien we het venster waarin de volledige GIP code staat. Enkele elementen van de volledige code worden echter omring door ‘/\*’ en ‘\*/’ om ze tijdens de simulatie even weg te laten. Dit moet omdat sommige libraries niet werken in de simulatie.



37 Toetsenbord simulatie



38 Toetsenbord a

In dit voorbeeld ziet u een simulatie waarin de code voor het toetsenbord gemaakt werd.  
Dit maakt het mogelijk om met de wifi te verbinden omdat we nu wachtwoorden kunnen invoeren.

Deze code werd in een aparte simulatie gemaakt   
omdat een toetsenbord coderen waarbij het mogelijk is om te wisselen tussen: hoofdletters, cijfers, en tekens tot ook mijn verbazing zeer complexe code is

Afbeelding met tekst, software, Computerpictogram, Webpagina

Automatisch gegenereerde beschrijving

39 UI simulatie

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

39 Hoofdmenu

Dit voorbeeld toont één van de eerste designs voor de user interface. De simulatie bestond uit niets meer dan 4 drukknoppen en een OLED-scherm. De code is nog niet compleet waardoor de simulatie na het selecteren van een item in één van de submenu’s stopt met werken.

# 6 Code

# 7 Grafische tekeningen

# 8 GitHub

# 9 Besluit

Jochem

We maakten een machine die automatisch speelkaarten kan delen, waardoor handmatig delen overbodig wordt. Het proces is eenvoudig: je plaatst je speelkaarten in de machine, kiest het gewenste spel en het aantal spelers. Vervolgens begint de machine met delen totdat elke speler de juiste hoeveelheid kaarten heeft ontvangen.

Ik probeerde zo goed mogelijk door te werken zodat we op het einde van het jaar niet in tijdsnood zouden komen. Ook heb ik geprobeerd altijd kritisch te kijken naar onze GIP om zo de goede en slechte punten te zien en mogelijke verbeteringen uit te voeren.

Ik vond het een heel leuk project en heb veel bijgeleerd. Ik heb mijn kennis die ik de voorbije jaren heb opgedaan kunnen toepassen om onze GIP tot een goed einde te brengen. Het leuke aan dit project was dat we zelf een GIP mochten kiezen en dus hebben we iets gekozen dat ons interesseerde. Hierdoor was ik gedreven om onze GIP tot een goed einde te brengen.

Af en toe moet je iets doen waar de leerkrachten geen ervaring mee hebben. In zulke situaties kunnen ze je niet altijd helpen als je vastloopt. Deze problemen hebben we dan door logisch na te denken en met opzoekwerk kunnen oplossen. Soms hadden we onderdelen nodig die niet direct beschikbaar waren op school. In deze gevallen moesten deze besteld worden. Dit is frustrerend omdat je tijdelijk niet verder kunt met datgene waarmee je mee bezig was.

Ik ben ontzettend blij met het eindresultaat. We hebben talloze uren aan werk besteed, en het is fantastisch om te zien dat dit zich heeft vertaald in een prachtig resultaat. Aan het begin van het schooljaar had ik nooit gedacht dat we zouden eindigen met zoiets moois. Het voelt geweldig om te zien hoe al onze inspanningen en toewijding hebben geleid tot iets wat echt de moeite waard is.

Siebe

Ons einddoel van deze GIP was om een tegen eind mei een automatische kaartendeler te maken voor diverse spellen. Deze worden geselecteerd aan de hand van een OLED-scherm en 4 drukknoppen.

Ik vind dat ik tijdens het schooljaar een zeer goede bijdrage heb geleverd aan deze GIP. Ik kwam tot dit besluit door het evalueren van de taken die door mij volbracht zijn. De eerste taak bestond vooral uit het schrijven van de software, deze omvat het menu op het OLED-scherm en het aansturen van verschillende componenten bijvoorbeeld het inlezen van sensoren en drukknoppen, bedienen van motoren en een relais. Vervolgens was er ook het ontwerpen van de elektronische schakelingen die vervolgens op een PCB werd geplaats. De communicatie tussen mij en Jochem was zeer goed, waardoor we ons er beide altijd van bewust waren wat er moest gebeuren, en hierdoor werden problemen snel en efficiënt aangepakt.

Werken aan een project waar je zelf het einddoel kiest is enorm aangenaam. Dit is natuurlijk omdat je grotendeels zelf kiest wat er op het einde van het jaar gemaakt moet zijn. Hierdoor zal je minder snel opgeven als bepaalde onderdelen van je opstelling niet werken. De gedachten om uiteindelijk het einddoel te bereiken zorgt bij mij voor enorm veel motivatie waardoor ik altijd zin had in de GIP-uren.

De GIP kwam voor mij met weinig frustraties of negatieve ervaringen vooral omdat ik gedreven was om dit project tot een goed einde te brengen, en dit leek dus ook, op een paar momenten na, wel te lukken. Eén van de ergerlijke dingen is dat

Het eindproduct overstijgt al mijn verwachting, in het begin van het jaar was er geen sprake van wifi of het gebruik van een PCB. De functies van het einddesign en de afwerking zijn allebei beter dan ik het me in september had voorgesteld. Ik ben dus ook enorm tevreden met het eindresultaat. Als ik terugkijk op dit project ben ik verbaasd dat we deze machine hebben gemaakt.

# 

# 

# 10 Bibliografie

Arduino (2019, 21 Feb). *WiFi – Arduino Reference.* Arduino.

(<https://www.arduino.cc/reference/en/libraries/wifi/>)

Blynk IoT. *IoT software that just works.* Blynk.

(<https://blynk.io/>)

DFRobot. DC Micro Metal Gear Motor w/Driver For Arduino - 30:1. (<https://www.dfrobot.com/product-1487.html>)

DFRobot WIKI. *FireBeetle\_Board\_ESP32\_E*. DFRobot.  
(<https://wiki.dfrobot.com/FireBeetle_Board_ESP32_E_SKU_DFR0654>)

EesyEda. *EasyEDA PCB design. EasyEDA.*

(<https://easyeda.com/nl>)

ESPRESSIF. *Preferences — Arduino-ESP32 2.0.14 documentation.* ESPRESSIF.

(<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/preferences.html>)

Mike McCauley (2022, 31 Okt). *AccelStepper Class Reference*. AirSpayce.

(<https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html>)

Olikraus (2023, 19 Dec,). *[u8g2reference · olikraus/u8g2 Wiki.](https://github.com/olikraus/u8g2/wiki/u8g2reference)* [GitHub](https://github.com/olikraus/u8g2/wiki/u8g2reference)

(<https://github.com/olikraus/u8g2/wiki/u8g2reference>)

Scott Minos (2023, 28 feb). How Lithium-ion Batteries Work. Energy Saver.

(<https://www.energy.gov/energysaver/articles/how-lithium-ion-batteries-work#:~:text=The%20anode%20and%20cathode%20store,at%20the%20positive%20current%20collector>.)

Stack Overflow. *stackoverflow* *– Where Developers Learn, Share & Build.* Stack Overflow

(<https://stackoverflow.com/>)

Youtube (2021, 20 jan). How does Buck Converter work? | DC-DC Converter – 1.

(<https://www.youtube.com/watch?v=W6NOV6b8kxs>)

Youtube (2016, 19 okt). How does a Stepper Motor work?.

(<https://www.youtube.com/watch?v=eyqwLiowZiU>)

Youtube (2021, 26 jun). How Stepper Motors Work - Electric motor.  
(<https://www.youtube.com/watch?v=09Mpkjcr0bo&t=720s>)

Youtube (2020, 20 jun). Buck Converter.

(<https://www.youtube.com/watch?v=rfChSvb8FX0>)

Youtube (2020, 9 jul). What is a Battery Management System?.

(<https://www.youtube.com/watch?v=GXYJ1xC10j4>)

Youtube (2020, 4 feb). Stepper motor driver basics.   
(<https://www.youtube.com/watch?v=15Ge4w_3Fyo>)

Youtube (2023, 15 dec). REAL-TIME-CLOCK MODULE (RTC) EXPLAINED.   
(<https://www.youtube.com/watch?v=dEWzm3AM_Tw>)

# 11 Bijlage

## 11.1 Code

/\*

--------Software - woord vooraf--------

Code

Gemaakt door: Siebe Boënne

Gemaakt voor: GIP 6IW, met Jochem Van Deuren

Code voor: Kaartendeler

huidige datum: 6/05/2024

Geschreven in: Arduino IDE 2.3.2, Wokwi online ESP32 simulatie

Taal: C++

Onderdelen

Microcontroller

FireBeetle 2 ESP32-E

UI

OLED SSD1309 128x64px

drukknop --> selecteren

drukknop --> annuleren

drukknop --> omhoog

drukknop --> omlaag

stappenmotor

library: AccelStepper.h --> object = stepper

DC-motor

library: ESP32\_Servo.h --> object = ESC

IoT

library: Blynk.h object = Blynk

Huidige mogenlijkheden

UI

Spelselectie

Wiezen

Zetten

Gelijk verdelen

Instellingen

Contrast

Tijd en datum

Reset

Geluid

Kalibreren

Wifi

Batterij

Batterijstaat

Informatie

''

Andere functies

!!!Deze code is gemaakt voor wanneer de HW allemaal op de PCB is aangebracht!!!

(Indien juist aangesloten kan de code gebruikt worden zonder PCB, maar dit word afgeraden door het grote aantal kabels)

De deler kan de laadspanning uitlezen,

batterijcapaciteit uitlezen,

een neopixel aansturen afhankelijk van batterijstaat bij standaard werking en bij opladen,

basis code voor Blynk Iot is aangebracht - moet nog uitgewerkt worden, in sub submenu 'Wifi' verbinden

met wifi netwerken en deze verbinding verbreken, codes invoeren om te verbinden met wifi

!!!Voor ontwikkeling!!!

Zoek de tag '//simultag' om plaatsen in de code te vinden waar aanpassingen moeten gebeuren

bij overschakelen van simulatie naar FireBeetle 2 ESP32-E

Zoek de tag '//devchange' om plaatsen in de code te vinden waar aanpassingen moeten gebeuren

bij overschakelen van ontwikkeling naar eindproduct

Wifi ERROR codes

WL\_NO\_SHIELD = 255

WL\_IDLE\_STATUS = 0

WL\_NO\_SSID\_AVAIL = 1

WL\_SCAN\_COMPLETED = 2

WL\_CONNECTED = 3

WL\_CONNECT\_FAILED = 4

WL\_CONNECTION\_LOST = 5

WL\_DISCONNECTED = 6

\*/

//libraries

//ESP32

#include <Insights.h>

//OLED display

#include <U8g2lib.h>

#include <U8x8lib.h>

//Servo controlled DC motor

#include <ESP32\_Servo.h>

//Stepper driver

#include <AccelStepper.h>

#include <MultiStepper.h>

//EEPROM

#include <Preferences.h>

//RTC

#include <uRTCLib.h>

//NeoPixel

#include <Adafruit\_NeoPixel.h>

//Blynk

#define BLYNK\_TEMPLATE\_ID "TMPL5zJOx3bI0"

#define BLYNK\_TEMPLATE\_NAME "Kaartendeler"

#define BLYNK\_AUTH\_TOKEN "iRhXnxVvD4joO0AFkEmNtXlxmaqelegZ"

#include <BlynkSimpleEsp32.h>

//WIFI

#include <WiFi.h>

#include <WiFiAP.h>

#include <WiFiClient.h>

#include <WiFiGeneric.h>

#include <WiFiMulti.h>

#include <WiFiSTA.h>

#include <WiFiScan.h>

#include <WiFiServer.h>

#include <WiFiType.h>

#include <WiFiUdp.h>

// bitmap arrays generated with image2cpp

// 'batterij', 64x64px

const unsigned char bitmap\_batterij [] =

{

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127,255,255,255,255,240,0,1,255,255,255,255,255,254,0,7,255,

255,255,255,255,255,0,7,0,0,0,0,0,7,128,14,0,0,0,0,0,3,128,14,33,8,66,0,0,1,128,14,115,156,231,24,0,1,

192,12,115,156,231,24,0,1,192,12,115,156,231,24,0,1,192,12,115,156,231,24,0,1,192,12,115,156,231,24,0,

1,192,12,115,156,231,24,0,1,248,12,115,156,231,24,0,1,252,12,115,156,231,24,0,0,28,12,115,156,231,24,0,

0,28,12,115,156,231,24,0,0,28,12,115,156,231,24,0,0,28,12,115,156,231,24,0,0,28,12,115,156,231,24,0,0,

28,12,115,156,231,24,0,0,28,12,115,156,231,24,0,0,28,12,115,156,231,24,0,1,252,12,115,156,231,24,0,1,

248,12,115,156,231,24,0,1,192,12,115,156,231,24,0,1,192,12,115,156,231,24,0,1,192,12,115,156,231,24,0,

1,192,12,115,156,231,24,0,1,192,14,33,8,66,0,0,1,192,14,0,0,0,0,0,3,128,7,0,0,0,0,0,7,128,7,255,255,255,

255,255,255,0,1,255,255,255,255,255,254,0,0,127,255,255,255,255,240,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

};

// 'wereldbol', 25x25px

const unsigned char bitmap\_wereldbol [] =

{

0x00, 0x7f, 0x00, 0x00, 0x01, 0x9c, 0xc0, 0x00, 0x06, 0x2a, 0x30, 0x00, 0x08, 0x2a, 0x08, 0x00,

0x10, 0x49, 0x04, 0x00, 0x2c, 0x88, 0x9a, 0x00, 0x23, 0x08, 0x62, 0x00, 0x43, 0x88, 0xe1, 0x00,

0x42, 0x7f, 0x21, 0x00, 0x84, 0x08, 0x10, 0x80, 0x84, 0x08, 0x10, 0x80, 0x84, 0x08, 0x10, 0x80,

0xff, 0xff, 0xff, 0x80, 0x84, 0x08, 0x10, 0x80, 0x84, 0x08, 0x10, 0x80, 0x84, 0x08, 0x10, 0x80,

0x42, 0x7f, 0x21, 0x00, 0x43, 0x88, 0xe1, 0x00, 0x23, 0x08, 0x62, 0x00, 0x2c, 0x88, 0x9a, 0x00,

0x10, 0x49, 0x04, 0x00, 0x08, 0x2a, 0x08, 0x00, 0x06, 0x2a, 0x30, 0x00, 0x01, 0x9c, 0xc0, 0x00,

0x00, 0x7f, 0x00, 0x00

};

// 'bliksem', 46x19px

const unsigned char bitmap\_bliksem [] =

{

0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xf0,

0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00,

0x7f, 0xff, 0xff, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xf0,

0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x0f, 0xff, 0xff, 0xe0,

0x00, 0x00, 0x00, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x07,

0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00, 0x00,

0x00, 0x00, 0x07, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x00, 0x00

};

// 'icon\_selectie', 16x16px

const unsigned char bitmap\_icon\_selectie[] =

{

0x00, 0x00, 0x60, 0x00, 0x30, 0x00, 0x19, 0xfc, 0x30, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00,

0x01, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xfc, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_informatie', 16x16px

const unsigned char bitmap\_icon\_informatie[] =

{

0x07, 0xe0, 0x08, 0x10, 0x30, 0x0c, 0x21, 0x84, 0x41, 0x82, 0x80, 0x02, 0x83, 0x81, 0x83, 0x81,

0x81, 0x81, 0x81, 0x81, 0x81, 0x82, 0x43, 0xc2, 0x23, 0xc4, 0x30, 0x0c, 0x0c, 0x30, 0x03, 0xc0

};

// 'icon\_battery', 16x16px

const unsigned char bitmap\_icon\_batterij[] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x40, 0x04, 0x5b, 0x66, 0x5b, 0x66,

0x5b, 0x66, 0x40, 0x04, 0x3f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_instellingen', 16x16px

const unsigned char bitmap\_icon\_instellingen[] =

{

0x00, 0x00, 0x00, 0x00, 0x20, 0x30, 0x30, 0x60, 0x18, 0x60, 0x0c, 0x7c, 0x06, 0xf8, 0x01, 0xc0,

0x03, 0x80, 0x07, 0x70, 0x0e, 0x78, 0x1c, 0x7c, 0x18, 0x3e, 0x00, 0x1e, 0x00, 0x0e, 0x00, 0x00

};

const unsigned char\* bitmap\_icons\_hoofdmenu [] =

{

bitmap\_icon\_selectie,

bitmap\_icon\_batterij,

bitmap\_icon\_informatie,

bitmap\_icon\_instellingen

};

//bitmaps: icons voor 1ste subMenu3, Instellingen

// 'icon\_geluid', 16x16px

const unsigned char bitmap\_icon\_geluid [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x04, 0x04, 0x0c, 0x14, 0x1c, 0x54, 0x3d, 0x54, 0x7d, 0x54,

0x7d, 0x54, 0x3d, 0x54, 0x1c, 0x54, 0x0c, 0x14, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_geluid\_uit', 16x16px

const unsigned char bitmap\_icon\_geluid\_uit [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x0c, 0x00, 0x1c, 0x82, 0x3c, 0x44, 0x7c, 0x28,

0x7c, 0x10, 0x3c, 0x28, 0x1c, 0x44, 0x0c, 0x82, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_contrast', 16x16px

const unsigned char bitmap\_icon\_contrast [] =

{

0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x0c, 0xf0, 0x10, 0xf8, 0x10, 0xf8, 0x20, 0xfc, 0x20, 0xfc,

0x20, 0xfc, 0x20, 0xfc, 0x10, 0xf8, 0x10, 0xf8, 0x0c, 0xf0, 0x03, 0xc0, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_reset', 16x16px

const unsigned char bitmap\_icon\_reset [] =

{

0x00, 0x00, 0x00, 0x00, 0x03, 0xc4, 0x0c, 0x34, 0x10, 0x0c, 0x10, 0x3c, 0x23, 0x80, 0x22, 0x40,

0x23, 0x84, 0x22, 0x84, 0x12, 0x48, 0x10, 0x08, 0x0c, 0x30, 0x03, 0xc0, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_datum', 16x16px

const unsigned char bitmap\_icon\_datum [] =

{

0x00, 0x00, 0x08, 0x88, 0x15, 0x54, 0x3f, 0xfc, 0x7f, 0xfe, 0x40, 0x02, 0x59, 0x9a, 0x59, 0x9a,

0x40, 0x02, 0x59, 0x9a, 0x59, 0x9a, 0x40, 0x02, 0x60, 0x06, 0x3f, 0xfc, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_tijd', 16x16px

const unsigned char bitmap\_icon\_tijd [] =

{

0x07, 0xc0, 0x18, 0x30, 0x21, 0x08, 0x51, 0x14, 0x41, 0x04, 0x81, 0x02, 0x81, 0x02, 0xc1, 0xe6,

0x80, 0x02, 0x80, 0x02, 0x40, 0x04, 0x50, 0x14, 0x20, 0x08, 0x19, 0x30, 0x07, 0xc0, 0x00, 0x00

};

// 'icon\_kalibreren', 16x16px

const unsigned char bitmap\_icon\_kalibreren [] =

{

0x00, 0x00, 0x00, 0x00, 0x03, 0xc4, 0x0c, 0x34, 0x10, 0x0c, 0x10, 0x3c, 0x22, 0x40, 0x22, 0x80,

0x23, 0x04, 0x22, 0x84, 0x12, 0x48, 0x10, 0x08, 0x0c, 0x30, 0x03, 0xc0, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_slot', 51x62px

const unsigned char bitmap\_slot [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00,

0x07, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x3f,

0xc0, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x0f, 0xfc, 0x00, 0x00, 0x03, 0xff, 0xe0, 0x01, 0xff, 0xf4,

0x00, 0x1f, 0xff, 0x00, 0x00, 0x3f, 0xfe, 0x00, 0x1f, 0xf0, 0x00, 0x00, 0x03, 0xfe, 0x00, 0x1e,

0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00,

0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00,

0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00,

0x1c, 0x00, 0x03, 0xf0, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x07, 0x38, 0x00, 0x0e, 0x00, 0x1c, 0x00,

0x0c, 0x0c, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x18, 0x06, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x18, 0x06,

0x00, 0x0e, 0x00, 0x1c, 0x00, 0x10, 0x02, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x10, 0x02, 0x00, 0x0e,

0x00, 0x1c, 0x00, 0x10, 0x02, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x10, 0x02, 0x00, 0x0e, 0x00, 0x1c,

0x00, 0x3f, 0xff, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x7f, 0xff, 0x80, 0x0e, 0x00, 0x1c, 0x00, 0x7f,

0xff, 0x80, 0x0e, 0x00, 0x1c, 0x00, 0x7f, 0xff, 0x80, 0x0e, 0x00, 0x1c, 0x00, 0x7f, 0xff, 0x80,

0x0e, 0x00, 0x1c, 0x00, 0x7f, 0xff, 0x80, 0x0e, 0x00, 0x1c, 0x00, 0x7f, 0xff, 0x80, 0x0e, 0x00,

0x0e, 0x00, 0x7f, 0xff, 0x80, 0x0e, 0x00, 0x0e, 0x00, 0x7f, 0xff, 0x80, 0x1c, 0x00, 0x0e, 0x00,

0x7f, 0xff, 0x80, 0x1c, 0x00, 0x0e, 0x00, 0x7f, 0xff, 0x80, 0x1c, 0x00, 0x07, 0x00, 0x7f, 0xff,

0x80, 0x38, 0x00, 0x07, 0x00, 0x7f, 0xff, 0x80, 0x38, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x78,

0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x70, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x01,

0xc0, 0x00, 0x00, 0x00, 0xe0, 0x00, 0x00, 0xe0, 0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0xf0, 0x00,

0x00, 0x03, 0xc0, 0x00, 0x00, 0x78, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x0f,

0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x3c, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x03, 0xe0, 0x01, 0xf0, 0x00, 0x00, 0x00, 0x00,

0xf8, 0x07, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x0f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x3f,

0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xf8, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00

};

// 'icon\_vinkje', 51x63px

const unsigned char bitmap\_vinkje [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x03, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00, 0x00, 0x00, 0x00,

0x00, 0x7f, 0xff, 0x80, 0x00, 0x00, 0x00, 0x03, 0xfe, 0x1f, 0xf0, 0x00, 0x00, 0x00, 0x7f, 0xf0,

0x03, 0xff, 0x80, 0x00, 0x1f, 0xff, 0x80, 0x00, 0x7f, 0xfe, 0x00, 0x1f, 0xfc, 0x00, 0x00, 0x0f,

0xfe, 0x00, 0x1f, 0x80, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00,

0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00,

0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00,

0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e,

0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c,

0x00, 0x00, 0x00, 0x40, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x01, 0xe0, 0x0e, 0x00, 0x1c, 0x00, 0x00,

0x01, 0xc0, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x03, 0xc0, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x07, 0x80,

0x0e, 0x00, 0x1c, 0x00, 0x00, 0x0f, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x1e, 0x00, 0x0e, 0x00,

0x1c, 0x00, 0x00, 0x3c, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x00, 0x78, 0x00, 0x0e, 0x00, 0x1c, 0x00,

0x70, 0xf0, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x79, 0xe0, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x3f, 0xc0,

0x00, 0x0e, 0x00, 0x0e, 0x00, 0x1f, 0x80, 0x00, 0x0e, 0x00, 0x0e, 0x00, 0x0f, 0x80, 0x00, 0x1c,

0x00, 0x0e, 0x00, 0x07, 0x00, 0x00, 0x1c, 0x00, 0x0e, 0x00, 0x06, 0x00, 0x00, 0x1c, 0x00, 0x07,

0x00, 0x00, 0x00, 0x00, 0x3c, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0x07, 0x80, 0x00,

0x00, 0x00, 0x38, 0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x70, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x00,

0xf0, 0x00, 0x01, 0xc0, 0x00, 0x00, 0x00, 0xe0, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x01, 0xe0, 0x00,

0x00, 0xf0, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x78, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x3c,

0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00,

0x3c, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x03, 0xe0, 0x01, 0xf0, 0x00,

0x00, 0x00, 0x00, 0xf8, 0x07, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x0f, 0x80, 0x00, 0x00, 0x00,

0x00, 0x3e, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07,

0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_scan', 51x51px

const unsigned char bitmap\_scan [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f,

0xff, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xf8,

0x00, 0x00, 0x00, 0x01, 0xfc, 0x01, 0xfc, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x00, 0x3e, 0x00, 0x00,

0x00, 0x07, 0xc0, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x1f, 0x87, 0x80, 0x00, 0x00, 0x0f,

0x00, 0x7f, 0x87, 0xc0, 0x00, 0x00, 0x1e, 0x01, 0xff, 0x83, 0xc0, 0x00, 0x00, 0x3e, 0x07, 0xfe,

0x01, 0xe0, 0x00, 0x00, 0x3c, 0x0f, 0xc0, 0x01, 0xe0, 0x00, 0x00, 0x3c, 0x1f, 0x1f, 0x80, 0xe0,

0x00, 0x00, 0x78, 0x3e, 0x7f, 0x80, 0xf0, 0x00, 0x00, 0x78, 0x3c, 0xff, 0x80, 0xf0, 0x00, 0x00,

0x78, 0x39, 0xf8, 0x00, 0x70, 0x00, 0x00, 0x78, 0x7b, 0xe3, 0x80, 0x70, 0x00, 0x00, 0x78, 0x73,

0xcf, 0x80, 0xf0, 0x00, 0x00, 0x78, 0xf7, 0x9f, 0x80, 0xf0, 0x00, 0x00, 0x78, 0xf7, 0x3f, 0x80,

0xf0, 0x00, 0x00, 0x78, 0xf7, 0x7c, 0x00, 0xf0, 0x00, 0x00, 0x78, 0xf7, 0x78, 0x00, 0xf0, 0x00,

0x00, 0x3c, 0xe7, 0x78, 0xe0, 0xe0, 0x00, 0x00, 0x3c, 0xe7, 0x79, 0xf1, 0xe0, 0x00, 0x00, 0x3e,

0x00, 0x01, 0xf1, 0xe0, 0x00, 0x00, 0x1e, 0x00, 0x01, 0xf3, 0xc0, 0x00, 0x00, 0x0f, 0x00, 0x00,

0xe7, 0xc0, 0x00, 0x00, 0x0f, 0x80, 0x00, 0x07, 0x80, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x1f, 0x00,

0x00, 0x00, 0x03, 0xe0, 0x00, 0x3f, 0x80, 0x00, 0x00, 0x01, 0xf8, 0x00, 0xff, 0xdc, 0x00, 0x00,

0x00, 0xff, 0x3f, 0xfb, 0xfe, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf1, 0xff, 0x00, 0x00, 0x00, 0x1f,

0xff, 0xc0, 0xff, 0x80, 0x00, 0x00, 0x03, 0xfe, 0x01, 0xf7, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x01,

0xe3, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf1, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0xf8,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x3e, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1f, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x8f, 0x80, 0x00, 0x00, 0x00,

0x00, 0x07, 0xc7, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x03, 0xe3, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x01,

0xf3, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0x80,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi\_groot', 42x31px

const unsigned char bitmap\_wifi\_groot [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x80,

0x00, 0x00, 0x00, 0x07, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x1f, 0xe1, 0xff, 0x00, 0x00, 0x00, 0x7c,

0x00, 0x0f, 0xc0, 0x00, 0x01, 0xe0, 0x00, 0x01, 0xe0, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x78, 0x00,

0x07, 0x00, 0x00, 0x00, 0x3c, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x00, 0x7f, 0x80,

0x06, 0x00, 0x00, 0x03, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0f, 0xc0, 0xfc, 0x00, 0x00, 0x00, 0x1e,

0x00, 0x1e, 0x00, 0x00, 0x00, 0x38, 0x00, 0x07, 0x80, 0x00, 0x00, 0x70, 0x00, 0x01, 0x80, 0x00,

0x00, 0x60, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00,

0x00, 0x00, 0x00, 0x00, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x01, 0xe0, 0xe0, 0x00, 0x00, 0x00, 0x01,

0x80, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x00,

0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi\_groot1', 42x31px

const unsigned char bitmap\_wifi\_groot1 [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x3f, 0x80,

0x00, 0x00, 0x1f, 0x07, 0xff, 0xf8, 0x00, 0x00, 0x07, 0x9f, 0xe1, 0xff, 0x00, 0x00, 0x03, 0xfc,

0x00, 0x0f, 0xc0, 0x00, 0x01, 0xf0, 0x00, 0x01, 0xe0, 0x00, 0x03, 0xf8, 0x00, 0x00, 0x78, 0x00,

0x07, 0x3c, 0x00, 0x00, 0x3c, 0x00, 0x0e, 0x1f, 0x00, 0x00, 0x0e, 0x00, 0x1c, 0x0f, 0xff, 0x80,

0x06, 0x00, 0x00, 0x07, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0f, 0xe0, 0xfc, 0x00, 0x00, 0x00, 0x1e,

0xf8, 0x1e, 0x00, 0x00, 0x00, 0x38, 0x7c, 0x07, 0x80, 0x00, 0x00, 0x70, 0x3e, 0x01, 0x80, 0x00,

0x00, 0x60, 0x1f, 0x00, 0x80, 0x00, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xe0,

0x00, 0x00, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x01, 0xe0, 0xf8, 0x00, 0x00, 0x00, 0x01,

0x80, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x80, 0x00,

0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x00, 0x00, 0x0c, 0x01, 0xe0, 0x00, 0x00, 0x00, 0x1e, 0x00,

0xf8, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x78, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x38, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi', 16x16px

const unsigned char bitmap\_icon\_wifi [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x10, 0x08, 0x20, 0x04, 0x47, 0xe2, 0x08, 0x10,

0x10, 0x08, 0x03, 0xc0, 0x04, 0x20, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi1', 16x16px

const unsigned char bitmap\_icon\_wifi1 [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xe0, 0x08, 0x10,

0x10, 0x08, 0x03, 0xc0, 0x04, 0x20, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi2', 16x16px

const unsigned char bitmap\_icon\_wifi2 [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x03, 0xc0, 0x04, 0x20, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi3', 16x16px

const unsigned char bitmap\_icon\_wifi3 [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'icon\_wifi4', 16x16px

const unsigned char bitmap\_icon\_wifi4 [] =

{

0x00, 0x00, 0x60, 0x00, 0x70, 0x00, 0x3f, 0xf0, 0x1c, 0x08, 0x2e, 0x04, 0x47, 0xe2, 0x0b, 0x90,

0x11, 0xc8, 0x03, 0xe0, 0x04, 0x70, 0x00, 0x38, 0x01, 0x9c, 0x00, 0x0e, 0x00, 0x06, 0x00, 0x00

};

const unsigned char\* bitmap\_icons\_subMenu3 [] =

{

bitmap\_icon\_geluid,

bitmap\_icon\_contrast,

bitmap\_icon\_datum,

bitmap\_icon\_reset,

bitmap\_icon\_kalibreren,

bitmap\_icon\_wifi

};

//items

const int maximumLengte = 16; // maximum characters for the item name

char menu\_items [] [maximumLengte] = //items in hoofdmenu

{

// array with item names

{ "Spel Selectie" }, //0

{ "Batterij" }, //1

{ "Informatie" }, //2

{ "Instellingen" } //3

};

char subMenu0\_items [] [maximumLengte] = // items in subMenu0, Spel Selectie

{

// array with item names

{ "Wiezen" }, //0

{ "Zetten" }, //1

{ "Black Jack" }, //2

{ "Gelijk Verdelen" } //3

};

char subMenu3\_items [] [maximumLengte] = // items in subMenu3, Instellingen

{

{ "Geluid" },

{ "Contrast" },

{ "Tijd en datum" },

{ "Reset" },

{ "Kalibreren" },

{ "Wifi" }

};

//algemeen

// 'scrollbar\_background', 8x64px

const unsigned char bitmap\_scrollbar\_background [] =

{

0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02,

0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02,

0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02,

0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x00

};

// 'item\_sel\_outline', 128x21px

const unsigned char bitmap\_item\_sel\_outline [] =

{

0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,

0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,

0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0,

0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0

};

// 'pop\_up\_outline', 128x22px

const unsigned char bitmap\_popUp\_outline [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7e, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'pop\_up\_vulling', 128x22px

const unsigned char bitmap\_popUp\_vulling [] =

{

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

};

// 'LOGO', 36x36px

const unsigned char bitmap\_LOGO [] =

{

0x00, 0x03, 0xfc, 0x00, 0x00, 0x00, 0x1f, 0xff, 0x80, 0x00, 0x00, 0x7f, 0xff, 0xe0, 0x00, 0x00,

0xfc, 0x03, 0xf8, 0x00, 0x03, 0xf0, 0x00, 0xfc, 0x00, 0x07, 0xc0, 0x1c, 0x3e, 0x00, 0x0f, 0x80,

0xfe, 0x1f, 0x00, 0x0e, 0x03, 0xf0, 0x07, 0x80, 0x1e, 0x07, 0x80, 0x07, 0x80, 0x3c, 0x0e, 0x00,

0x03, 0xc0, 0x38, 0x1c, 0x00, 0x01, 0xc0, 0x78, 0x3c, 0x00, 0x01, 0xe0, 0x70, 0x30, 0x00, 0x00,

0xe0, 0x70, 0x70, 0x00, 0x00, 0xe0, 0xe0, 0x60, 0x00, 0x00, 0x70, 0xe0, 0x60, 0x3f, 0xc0, 0x70,

0xe0, 0xe0, 0x7f, 0xe0, 0x70, 0xe0, 0xc0, 0x00, 0x60, 0x70, 0xe0, 0xc0, 0x00, 0x60, 0x70, 0xe0,

0xe0, 0x00, 0x60, 0x70, 0xe0, 0x60, 0x00, 0xe0, 0x70, 0xe0, 0x60, 0x01, 0xc0, 0x70, 0x70, 0x70,

0x03, 0x80, 0xe0, 0x70, 0x38, 0x0f, 0x00, 0xe0, 0x78, 0x1f, 0xff, 0x01, 0xe0, 0x38, 0x07, 0xf8,

0x01, 0xc0, 0x3c, 0x00, 0x00, 0x03, 0xc0, 0x1e, 0x00, 0x00, 0x03, 0x80, 0x1e, 0x00, 0x00, 0x07,

0x80, 0x0f, 0x80, 0x00, 0x0f, 0x00, 0x07, 0xc0, 0x00, 0x3e, 0x00, 0x03, 0xf0, 0x00, 0xfc, 0x00,

0x01, 0xfc, 0x03, 0xf8, 0x00, 0x00, 0x7f, 0xff, 0xe0, 0x00, 0x00, 0x1f, 0xff, 0x80, 0x00, 0x00,

0x03, 0xfc, 0x00, 0x00

};

// ------------------ end of bitmaps from image2cpp ---------------------------------

//define pins

//UI buttons

#define BUTTON\_SELECT\_PIN D1 // pin for SELECT button

#define BUTTON\_UP\_PIN D2 // pin for UP button

#define BUTTON\_DOWN\_PIN D3 // pin for DOWN button

#define BUTTON\_CANCEL\_PIN D5 // pin for CANCEL button

#define BUTTON\_POWER D13 //pin for POWER button

#define buzzer D6 //pin for buzzer

#define relaisPin D7 //pin voor overnamerelais

#define haltSensPin A0 //pin for haltsensor

#define servoControllerPin D0 //pin for DC motor

#define neoPixel D9 //pin voor neopixel

#define dirPin D10 //direction pin for stepper driver

#define stepPin D11 //step pin for stepper driver

#define sleepPin D12 //step enable pin

#define motorInterfaceType 1 //motor interface type for AccelStepper.h library, type 1 = driver

#define MOVING\_AVERAGE\_WINDOW\_SIZE 10

#define batterijSpanningPin A1 //pin voor batterijspanning

#define laadSpanningPin A2 //pin voor laadspanning

//Steppeer Accelstepper.h constructor

AccelStepper stepper = AccelStepper(motorInterfaceType, stepPin, dirPin);

//OLED U8g2lib.h constructor

U8G2\_SSD1309\_128X64\_NONAME0\_2\_HW\_I2C OLED(U8G2\_R0, U8X8\_PIN\_NONE /\*reset\*/); // Fast I2C

//Adafruit NeoPixel constructor

Adafruit\_NeoPixel pixel(1, neoPixel, NEO\_RGB);

//servo constructor

Servo ESC; //Electronic Speed Controller

//Preferences constructor

Preferences EEPROM;

//RTC constructor

uRTCLib rtc(0x68);

char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

//UI variabelen

uint8\_t items\_screen = 4; // number of items in the current screen, set to 4 because first screen is always menu

int8\_t item\_selected = 0; // geselecteerde item in huidige scherm

uint8\_t item\_selected1 = 0; // geselecteerde item in menu wanneer in submenu

uint8\_t item\_selected2 = 0; //geselecteerde item in submenu wanneer in subsubmenu

int8\_t item\_sel\_next = 1; // item onder het geselecteerde item, gebruikt in het menuscherm om het volgende item na het geselecteerde item te tekenen

int8\_t item\_sel\_previous = items\_screen - 1; // item boven het geselecteerde item, gebruikt in het menuscherm om het item voor het geselecteerde item te tekenen

uint8\_t current\_screen = 0; // 0 is hoofdmenu daarna current\_screen = current\_screen + 1 per selectie

//button bools

// zorgen ervoor dat als een knop bediend word een deel code maar 1 maal geïtereerd word

bool button\_up\_clicked = 0;

bool button\_down\_clicked = 0;

bool button\_select\_clicked = 0;

bool button\_cancel\_clicked = 0;

bool button\_power\_clicked = 0;

//buttonEnable

bool button\_up\_enable = 1; // kan de 'up' knop een andere code laten uitvoeren

bool button\_down\_enable = 1; // kan de 'down' knop een andere code laten uitvoeren

bool button\_select\_enable = 1; // kan de 'select' knop een andere code laten uitvoeren

bool button\_cancel\_enable = 1; // kan de 'cancel' knop een andere code laten uitvoeren

//algemene variabelen

bool i1 = 0; //in gebruik

int i2 = 0; //in gerbuik

bool i3 = 0; //in gebruik, neopixel

bool i4 = 0; //niet in gebruik

unsigned long i5 = 0; //word gebruikt om aantal seconden dat de deler aan één stuk aanstaat op te slaan

unsigned long mSec = 0; //word gebruikt om waarde van millis op te slaan

String str;

//calibration

int16\_t haltSensVal;

int16\_t lastHaltSensVal;

bool calibrated = 0;

bool automaticCalibration = 0;

int16\_t length = 0;

int16\_t stepsToZero;

//software versie

String softwareVersie = "v1";

//stepper

int16\_t currentPosition;

int16\_t stepsToRelativeZero;

//display String

String OLEDstring;

bool screen\_delay;

//instellingen

bool toggleOnOff;

bool geluid = 1;

uint8\_t settingValue;

uint8\_t contrast; //contrast in procent

String settingString;

//RTC

short seconden;

short minuten;

short uren;

short weekdag;

short dag;

short maand;

uint8\_t jaar;

//berekenen dag van de week

short jaarcode;

short maandcode;

short eeuwcode;

//schrikkeljaar

bool schrikkel\_jaar;

//hulpvariabelen

short seconden1;

short minuten1 ;

short uren1 ;

short dag1;

short maand1;

uint8\_t jaar1;

short Digit1;

short Digit2;

short Digit3;

short Digit4;

short Digit5;

short Digit6;

short Digit7;

short Digit8;

short Digit9;

short Digit10;

short Digit11;

short Digit12;

short h1; //word gebruikt in sub submenu 'Tijd en datum' en 'Wifi'

String sD1;

String sD2;

String sD3;

String sD4;

String sD5;

String sD6;

String sD7;

String sD8;

String sD9;

String sD10;

String sD11;

String sD12;

//spel selectie

//hulpvariabelen

short s1;

short s2;

//variabelen voor kaarten delen

uint8\_t nPersonen;

uint8\_t nKaarten;

int8\_t nDeelBegin;

bool wiezen;

bool zetten;

bool blackJack;

bool gelijkVerdelen;

//Wifi

//word gebruikt in sub submenu 'Wifi'

String ssid = "";

String pass = "";

String ssidArray[] = {};

String passArray[] = {};

int status;

short selected\_network;

int networks;

bool connect\_attempt = 0;

bool scan\_completed = 0;

//keyboard

int xKeyBoard = 0;

int yKeyBoard = 0;

int keyBoardMode = 0;

bool drawColorLetterShift;

bool drawColorSignShift;

bool errorMaxPasswordLength;

String text = "Geo-2050"; //testtag

char keyBoard\_a [] [10] =

{

{'a', 'z', 'e', 'r', 't', 'y', 'u', 'i', 'o', 'p'},

{'q', 's', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm'},

{'\0', '\0','w', 'x', 'c', 'v', 'b', 'n','\0' ,'\0'}

};

char keyBoard\_A [] [10] =

{

{'A', 'Z', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P'},

{'Q', 'S', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M'},

{'\0', '\0', 'W', 'X', 'C', 'V', 'B', 'N', '\0 ', '\0'}

};

char keyBoard\_1 [] [10]

{

{'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'},

{'-', '/', ':', ';', '(', ')', ' /\*€\*/', '&', '@', '"'},

{'\0', '\0 ', '.', ',', '?', '!', 0x27, 0xB5, '\0 ', '\0 '}

};

char keyBoard\_2 [] [10]

{

{'[', ']', '{', '}', '#', '%', '^', '\*', '+', '='},

{'\_', 0x5C, '|', 0x7E, '<', '>', '$', 0xA3, 0xA5, '§'},

{'\0', '\0 ', '.', ',', '?', '!', 0x27, 0xB5, '\0 ', '\0 '}

};

//power status

bool powerState = 0;

bool relaisState = 0;

//batterij

short int batterijPercentage;

short int batterijPercentageArray [MOVING\_AVERAGE\_WINDOW\_SIZE];

float batterijSpanning;

//opladen

bool charging;

short frame = 0;

bool lastCharging;

float laadSpanning;

//deler informatie

String serie\_nummer = ""; //serienummer deler

unsigned long kaarten\_gedeeld; //aantal kaarten gedeeld

unsigned long werk\_uren; //aantal werkuren van de deler

short int werk\_minuten; //telt tot 60 word dan omgezet naar 1 werkuur

short int werk\_seconden; //telt tot 60 word dan omgezet naar 1 werkminuut

short int last\_werk\_seconden;

//

//--------------------void setup--------------------

void setup()

{

delay(2000); //Geef de FireBeetle tijd om volledige werking te starten

//haal settings uit EEPROM geheugen

EEPROM.begin("instellingen", false);

geluid = EEPROM.getBool("geluid");

contrast = EEPROM.getInt("contrast");

EEPROM.end();

EEPROM.begin("wifi", false);

ssid = EEPROM.getString("ssid");

pass = EEPROM.getString("pass");

EEPROM.end();

EEPROM.begin("informatie", false);

serie\_nummer = EEPROM.getString("serieNummer");

kaarten\_gedeeld = EEPROM.getULong("kaartenGedeeld");

werk\_uren = EEPROM.getULong("werkUren");

werk\_minuten = EEPROM.getUShort("werkMinuten");

werk\_seconden = EEPROM.getUShort("werkSeconden");

EEPROM.end();

//Serial

Serial.begin(115200);

//Blynk

//Blynk.begin(BLYNK\_AUTH\_TOKEN, )

//servo.h

ESC.attach(servoControllerPin, 500, 2500); //Electronic Speed Controller

//RTC

URTCLIB\_WIRE.begin();

rtc.set\_model(URTCLIB\_MODEL\_DS3231);

rtc.set\_12hour\_mode(false);

//U8g2lib.h OLED

OLED.begin();

OLED.setBitmapMode(1);

//Neopixel

pixel.begin();

//Wifi

WiFi.mode(WIFI\_STA);

WiFi.begin(ssid, pass);

//pinMode

pinMode(BUTTON\_UP\_PIN, INPUT\_PULLUP); //up button

pinMode(BUTTON\_SELECT\_PIN, INPUT\_PULLUP); //select button

pinMode(BUTTON\_CANCEL\_PIN, INPUT\_PULLUP); //cancel button

pinMode(BUTTON\_DOWN\_PIN, INPUT\_PULLUP); //down button

pinMode(BUTTON\_POWER, INPUT); //power button

pinMode(batterijSpanningPin, INPUT);

pinMode(laadSpanningPin, INPUT);

pinMode(haltSensPin, INPUT); //haltsensor

pinMode(relaisPin, OUTPUT); //relais

pinMode(buzzer, OUTPUT); //buzzer

pinMode(sleepPin, OUTPUT); //DRV8825 sleep

pinMode(LED\_BUILTIN, OUTPUT);

//stepper

stepper.setCurrentPosition(0);

//check if power button is pressed

if(digitalRead(BUTTON\_POWER) == HIGH)

{

button\_power\_clicked = 1;

powerState = 1;

relaisState = 1;

digitalWrite(relaisPin, HIGH);

digitalWrite(sleepPin, HIGH); //haal de DRV8825 uit sleep mode

//call to void with start-up animation

opstartScherm();

}

else powerState = 0;

}

//--------------------void reset--------------------

//void to call, to reset arduino

void(\* resetFunc) (void) = 0;

//--------------------void loop--------------------

void loop()

{

//loops

//serial print, for development purposes only

//OLED

OLED.setContrast(map(contrast, 0, 100, 0, 255));

//time expirimenteel

while(1000000 <= (esp\_timer\_get\_time() - 1000000\*i5))

{

werk\_seconden++;

if(werk\_seconden == 60)

{

werk\_minuten++;

werk\_seconden = 0;

}

if(werk\_minuten == 60)

{

werk\_uren++;

werk\_minuten = 0;

}

i5++;

}

//bepaal batterijpercentage uit analogread

analogReadResolution(9);

batterijPercentage = constrain(map(analogReadMilliVolts(batterijSpanningPin), 2106, 3300, 0, 100), 0, 100);

for(int a = 0; a < MOVING\_AVERAGE\_WINDOW\_SIZE-1; a++)

{

batterijPercentageArray[1+a] = batterijPercentageArray [0+a];

batterijPercentageArray[0] = batterijPercentage;

}

batterijPercentage = 0;

for(int a = 0; a < MOVING\_AVERAGE\_WINDOW\_SIZE; a++) batterijPercentage += batterijPercentageArray[a];

batterijPercentage = batterijPercentage / MOVING\_AVERAGE\_WINDOW\_SIZE;

analogReadResolution(12);

//bepaal batterijspanning uit analogread

batterijSpanning = map(analogReadMilliVolts(batterijSpanningPin), 0, 3300, 0, 4386); //converteer 0-4095 naar batterijspanning

batterijSpanning /= (1000 / 3);

//batterijSpanning += 0.8; //kalibratiefactor

//bepaal laadspanning uit analogread

laadSpanning = map(analogReadMilliVolts(laadSpanningPin), 0, 3300, 0, 1577); //converteer 0-4095 naar laadspanning

laadSpanning /= 100;

//laadSpanning -= 2.8; //kalibratiefactor

//check of charging = TRUE

if(charging == 1) lastCharging = 1;

if(batterijSpanning < laadSpanning) charging = 1;

else

{

charging = 0;

lastCharging = 0;

}

if(powerState == 0) //als powerState 0 is moet de deler wel rechtsreeks spanning krijgen van een lader, charging zou dus ook TRUE moeten zijn

{

OLED.clear();

OLED.clearBuffer();

current\_screen = 0;

item\_selected = 0;

item\_sel\_previous = -1;

item\_sel\_next = 1;

items\_screen = 4;

if(charging == 1)

{

if(batterijPercentage < 97)

{

if(werk\_seconden != last\_werk\_seconden)

{

//laat neopixel groen pinken, indicatie voor opladen

if(i3 == 0)

{

pixel.clear();

pixel.show();

pixel.setPixelColor(0, 0, 100, 0); //Groen

pixel.show();

i3 = 1;

}

else

{

pixel.clear();

pixel.show();

i3 = 0;

}

}

}

else if(batterijPercentage >= 97)

{

//laat neopixel groen branden, indicatie voor volledig opgeladen

pixel.setPixelColor(0, 0, 100, 0); //Groen

pixel.show();

}

}

else if(charging == 0) //dit zou in normale omstandigheden nooit waar kunnen zijn, als dit toch het geval is dan zou wanneer de lader insteekt de neopixel rood zijn

{

//laat neopixel rood branden, indicatie dat de deler niet oplaad terwijl de lader inzit, dit kan als een ERROR melding beschouwt worden

pixel.setPixelColor(0, 100, 0, 0);

pixel.show();

}

digitalWrite(sleepPin, LOW); //zet de DRV8825 in sleep mode

if(relaisState == 1)

{

relaisState = 0;

digitalWrite(relaisPin, LOW);

}

if(digitalRead(BUTTON\_POWER) == HIGH && button\_power\_clicked == 0) //check for change of state

{

delay(2000); //consistancy

button\_power\_clicked = 1;

powerState = 1;

relaisState = 1;

digitalWrite(relaisPin, HIGH);

//call to void with start-up animation

opstartScherm();

}

if(digitalRead(BUTTON\_POWER) == LOW && button\_power\_clicked == 1) button\_power\_clicked = 0;

}

else if(powerState == 1)

{

//ON/OFF code

if(digitalRead(BUTTON\_POWER) == HIGH && button\_power\_clicked == 0)//check for change of state

{

button\_power\_clicked = 1;

powerState = 0;

//code vlak voor uitschakelen

EEPROM.begin("informatie", false);

EEPROM.putULong("werkUren", werk\_uren);

EEPROM.putUShort("werkMinuten", werk\_minuten);

EEPROM.putUShort("werkSeconden", werk\_seconden);

EEPROM.getString("serieNummer", serie\_nummer);

EEPROM.getULong("kaartenGedeeld", kaarten\_gedeeld);

EEPROM.end();

//buzzer

tone(buzzer, 1000); //tone 975, delay 225

delay(225);

tone(buzzer, 675); //tone 675, delay 250

delay(250);

tone(buzzer, 500); //tone 500, delay 500

delay(600);

noTone(buzzer);

delay(200);

//'Tot ziens' bericht op OLED

}

if(digitalRead(BUTTON\_POWER) == LOW && button\_power\_clicked == 1) button\_power\_clicked = 0;

if(charging == 0)

{

//loops if powerState = 1 and charging = FALSE / 0

//neopixel

if(batterijPercentage > 20)

{

pixel.setPixelColor(0, 0, 100, 0); //Groen

pixel.show();

}

else if(batterijPercentage <= 20)

{

pixel.setPixelColor(0, 100, 0, 0); //Rood

pixel.show();

}

//stepper

digitalWrite(sleepPin, HIGH); //haal de DRV8825 uit sleep mode

//Serial.print for developing only

//OLED

OLED.setContrast(map(contrast, 0, 100, 0, 255));

//code

if(calibrated == 0)

{

//OLED

OLED.firstPage();

do

{

stepper.run();

OLED.setColorIndex(1);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(0);

OLED.drawBitmap(12, 6, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(10 + 5 + 36, 20 + 8, "Industries");

OLED.setFont(u8g2\_font\_5x7\_tf);

OLED.drawStr(10 + 6 + 36, 20 + 16 + 5, "Card Dealer");

OLED.setFont(u8g2\_font\_5x7\_tf); //check

OLED.drawFrame(11, 48, 108, 10);

OLED.drawBox(12, 49, length, 8);

}while(OLED.nextPage());

if(automaticCalibration == 0) vAutomaticCalibration();

}

if (automaticCalibration == 1)

{

//looped als de deler aanstaat, niet oplaad en gekalibreerd is

//loops if powerState = 1 and charging = FALSE / 0 and automaticCalibration = TRUE / 1

//button up

if (digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0 && button\_up\_enable == 1)

{

button\_up\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

item\_selected--;

if (item\_selected < 0) item\_selected = items\_screen - 1;

}

//button down

if (digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0 && button\_down\_enable == 1)

{

button\_down\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

item\_selected++;

if (item\_selected >= items\_screen) item\_selected = 0;

}

//button select

if (digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0 && button\_select\_enable == 1)

{

button\_select\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 100

tone(buzzer, 1500); // 1500

delay(100); // 150

noTone(buzzer);

}

//code

current\_screen++;

if (current\_screen == 1) item\_selected1 = item\_selected;

else if (current\_screen == 2) item\_selected2 = item\_selected;

item\_selected = 0;

}

//button cancel

if (digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0 && button\_cancel\_enable == 1)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 100

tone(buzzer, 1000); // 1000

delay(100); // 150

noTone(buzzer);

}

//code

if (current\_screen > 0)

{

current\_screen--;

if (current\_screen == 0)

{

item\_selected = item\_selected1;

items\_screen = 4;

}

else if (current\_screen == 1) item\_selected = item\_selected2;

}

}

//reset button\_clicked... value, so button can be clicked again

if (digitalRead(BUTTON\_UP\_PIN) == HIGH && button\_up\_clicked == 1) button\_up\_clicked = 0;

if (digitalRead(BUTTON\_DOWN\_PIN) == HIGH && button\_down\_clicked == 1) button\_down\_clicked = 0;

if (digitalRead(BUTTON\_SELECT\_PIN) == HIGH && button\_select\_clicked == 1) button\_select\_clicked = 0;

if (digitalRead(BUTTON\_CANCEL\_PIN) == HIGH && button\_cancel\_clicked == 1) button\_cancel\_clicked = 0;

//set correct values for the previous and next items

item\_sel\_previous = item\_selected - 1;

if (item\_sel\_previous < 0) item\_sel\_previous = items\_screen - 1; // previous item would be below first --> make it the last

item\_sel\_next = item\_selected + 1;

if (item\_sel\_next >= items\_screen) item\_sel\_next = 0; // next item would be after last --> make it the first

//void screen

if (current\_screen == 0) menu();

else if (current\_screen == 1) screen1();

else if (current\_screen == 2) screen2();

}

}

else if(charging == 1)

{

//looped als de deler aanstaat en oplaad

//loops if powerState = 1 and charging = TRUE / 1

//neopixel

if(batterijPercentage < 97)

{

if(werk\_seconden != last\_werk\_seconden)

{

//laat neopixel groen pinken, indicatie voor opladen

if(i3 == 0)

{

digitalWrite(LED\_BUILTIN, HIGH);

pixel.clear();

pixel.show();

pixel.setPixelColor(0, 0, 150, 0);

pixel.show();

i3 = 1;

}

else

{

digitalWrite(LED\_BUILTIN, LOW);

pixel.clear();

pixel.show();

i3 = 0;

}

}

}

else if(batterijPercentage >= 97)

{

//laat neopixel groen branden, indicatie voor volledig opgeladen

pixel.setPixelColor(0, 0, 100, 0); //Groen

pixel.show();

}

//stepper

digitalWrite(sleepPin, LOW); //zet de DRV8825 in sleep mode

//code

//set correct values for the previous and next items

item\_sel\_previous = item\_selected - 1;

if (item\_sel\_previous < 0) item\_sel\_previous = items\_screen - 1; // previous item would be below first = make it the last

item\_sel\_next = item\_selected + 1;

if (item\_sel\_next >= items\_screen) item\_sel\_next = 0; // next item would be after last = make it the first

//void screen

if (current\_screen == 0) menu();

else if (current\_screen == 1) screen1();

else if (current\_screen == 2) screen2();

//button up

if (digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0 && button\_up\_enable == 1)

{

button\_up\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

item\_selected--;

if (item\_selected < 0) item\_selected = items\_screen - 1;

}

//button down

if (digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0 && button\_down\_enable == 1)

{

button\_down\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

item\_selected++;

if (item\_selected >= items\_screen) item\_selected = 0;

}

//button select

if (digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0 && button\_select\_enable == 1)

{

button\_select\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 100

tone(buzzer, 1500); // 1500

delay(100); // 150

noTone(buzzer);

}

//code

current\_screen++;

if (current\_screen == 1) item\_selected1 = item\_selected;

else if (current\_screen == 2) item\_selected2 = item\_selected;

item\_selected = 0;

}

//button cancel

if (digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0 && button\_cancel\_enable == 1)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 100

tone(buzzer, 1000); // 1000

delay(100); // 150

noTone(buzzer);

}

//code

if (current\_screen > 0)

{

current\_screen--;

if (current\_screen == 0)

{

item\_selected = item\_selected1;

items\_screen = 4;

}

else if (current\_screen == 1) item\_selected = item\_selected2;

}

}

//reset button\_clicked... value, so button can be clicked again

if (digitalRead(BUTTON\_UP\_PIN) == HIGH && button\_up\_clicked == 1) button\_up\_clicked = 0;

if (digitalRead(BUTTON\_DOWN\_PIN) == HIGH && button\_down\_clicked == 1) button\_down\_clicked = 0;

if (digitalRead(BUTTON\_SELECT\_PIN) == HIGH && button\_select\_clicked == 1) button\_select\_clicked = 0;

if (digitalRead(BUTTON\_CANCEL\_PIN) == HIGH && button\_cancel\_clicked == 1) button\_cancel\_clicked = 0;

//set correct values for the previous and next items

item\_sel\_previous = item\_selected - 1;

if (item\_sel\_previous < 0) item\_sel\_previous = items\_screen - 1; // previous item would be below first = make it the last

item\_sel\_next = item\_selected + 1;

if (item\_sel\_next >= items\_screen) item\_sel\_next = 0; // next item would be after last = make it the first

}

}

last\_werk\_seconden = werk\_seconden;

}

//--------------------void calibration--------------------

void vAutomaticCalibration() // kalibreer 0-punt stappenmotor

{

//code

haltSensVal = analogRead(haltSensPin);

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

stepper.moveTo(400);

stepper.run();

length = map(400 - stepper.distanceToGo(), 0, 400, 0, 107);

//calibration

if(haltSensVal > lastHaltSensVal)

{

stepsToZero = 400-stepper.distanceToGo();

lastHaltSensVal = haltSensVal;

}

if(stepper.distanceToGo() == 0)

{

delay(250);

stepper.setCurrentPosition(0);

if(stepsToZero<200) stepper.runToNewPosition(stepsToZero);

else if(stepsToZero>=200) stepper.runToNewPosition(stepsToZero-400);

stepper.setCurrentPosition(0);

calibrated = 1;

automaticCalibration = 1;

lastHaltSensVal = 0;

}

}

//--------------------void menu--------------------

void menu()

{

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//draw previous

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(25, 15, menu\_items[item\_sel\_previous]);

OLED.drawBitmap(4, 2, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_previous]);

//draw selected

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(25, 15 + 20 + 2, menu\_items[item\_selected]);

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_selected]);

//draw next

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(25, 15 + 20 + 20 + 2 + 2, menu\_items[item\_sel\_next]);

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

} while (OLED.nextPage());

}

//--------------------void screen1--------------------

void screen1()

{

if(item\_selected1 == 1)

{

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 1;

}

else

{

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

}

if (item\_selected1 == 0) //subMenu, Spel Sellectie

{

items\_screen = 4;

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//draw previous

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(10, 15, subMenu0\_items[item\_sel\_previous]);

//draw selected

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(10, 15 + 20 + 2, subMenu0\_items[item\_selected]);

//draw next

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(10, 15 + 20 + 20 + 2 + 2, subMenu0\_items[item\_sel\_next]);

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

} while (OLED.nextPage());

}

else if (item\_selected1 == 1) //subMenu, batterij

{

if(charging == 1)

{

delay(25);

frame++;

if(frame >= 4) frame = 0;

}

if(charging != lastCharging)

{

//start opladen animatie

for(int a = 0; a <= 25; a++)

{

OLED.firstPage();

do

{

OLED.setDrawColor(1);

OLED.drawBitmap(32, 0, 64/8, 64, bitmap\_batterij);

OLED.setDrawColor(0);

OLED.drawBox(40, 19, 30, 26);

OLED.setDrawColor(1);

OLED.drawBitmap(41, 23, 48 / 8, 19, bitmap\_bliksem);

}while(OLED.nextPage());

delay(20);

if(laadSpanning > batterijSpanning == 0)

{

charging = 0;

lastCharging = 0;

a = 25;

}

}

for(int a = 0; a <= 21; a++)

{

OLED.firstPage();

do

{

OLED.drawBitmap(32, 0-constrain(a, 0, 13), 64/8, 64, bitmap\_batterij);

OLED.setDrawColor(0);

OLED.drawBox(40, 19-constrain(a, 0, 13), 30, 26);

OLED.setDrawColor(1);

OLED.drawBitmap(41, 23+a, 48 / 8, 19, bitmap\_bliksem);

}while(OLED.nextPage());

delay(20);

if(laadSpanning > batterijSpanning == 0)

{

charging = 0;

lastCharging = 0;

a = 21;

}

}

lastCharging = 1;

}

OLED.firstPage();

do

{

if(charging == 0)

{

OLED.setDrawColor(1);

OLED.drawBitmap(32, 0, 64/8, 64, bitmap\_batterij);

OLED.setDrawColor(0);

OLED.drawBox(40, 19, 30, 26);

OLED.setDrawColor(1);

OLED.setFont(u8g\_font\_7x14);

OLEDstring = batterijPercentage;

OLED.drawStr(50, 36, OLEDstring.c\_str());

}

else if(charging == 1)

{

OLED.setDrawColor(1);

OLED.drawFrame(19, 23-5, 90, 36);

OLED.drawFrame(20, 24-5, 88, 34);

OLED.drawBitmap(41, 49-5, 48 / 8, 19, bitmap\_bliksem);

OLED.setDrawColor(0);

OLED.drawFrame(40, 47-5, 47, 22);

OLED.drawFrame(39, 47-5, 49, 22);

OLED.drawFrame(38, 47-5, 51, 22);

OLED.drawFrame(37, 47-5, 53, 22);

OLED.drawFrame(38, 23-5, 53, 2);

if(frame == 0)

{

//frame 1

//stroom, horizontaal links onder

OLED.drawFrame(33, 57-5, 2, 2);

OLED.drawFrame(29, 57-5,2, 2);

OLED.drawFrame(25, 57-5, 2, 2);

OLED.drawFrame(21, 57-5, 2, 2);

//stroom verticaal links

OLED.drawFrame(19, 55-5, 2, 2);

OLED.drawFrame(19, 51-5, 2, 2);

OLED.drawFrame(19, 47-5, 2, 2);

OLED.drawFrame(19, 43-5, 2, 2);

OLED.drawFrame(19, 39-5, 2, 2);

OLED.drawFrame(19, 35-5, 2, 2);

OLED.drawFrame(19, 31-5, 2, 2);

OLED.drawFrame(19, 27-5, 2, 2);

OLED.drawFrame(19, 23-5, 2, 2);

//stroom horizontaal links boven

OLED.drawFrame(19, 23-5, 2, 2);

OLED.drawFrame(23, 23-5, 2, 2);

OLED.drawFrame(27, 23-5, 2, 2);

OLED.drawFrame(31, 23-5, 2, 2);

OLED.drawFrame(35, 23-5, 2, 2);

//stroom horizontaal rechts boven

OLED.drawFrame(93, 23-5, 2, 2);

OLED.drawFrame(97, 23-5, 2, 2);

OLED.drawFrame(101, 23-5, 2, 2);

OLED.drawFrame(105, 23-5, 2, 2);

//stroom verticaal rechts

OLED.drawFrame(107, 25-5, 2, 2);

OLED.drawFrame(107, 29-5, 2, 2);

OLED.drawFrame(107, 33-5, 2, 2);

OLED.drawFrame(107, 37-5, 2, 2);

OLED.drawFrame(107, 41-5, 2, 2);

OLED.drawFrame(107, 45-5, 2, 2);

OLED.drawFrame(107, 49-5, 2, 2);

OLED.drawFrame(107, 53-5, 2, 2);

//stroom horizontaal rechts beneden

OLED.drawFrame(107, 57-5, 2, 2);

OLED.drawFrame(103, 57-5, 2, 2);

OLED.drawFrame(99, 57-5, 2, 2);

OLED.drawFrame(95, 57-5, 2, 2);

OLED.drawFrame(91, 57-5, 2, 2);

}

else if(frame == 1)

{

//frame 2

//stroom, horizontaal links onder

OLED.drawFrame(37-1, 57-5, 2, 2);

OLED.drawFrame(33-1, 57-5, 2, 2);

OLED.drawFrame(29-1, 57-5, 2, 2);

OLED.drawFrame(25-1, 57-5, 2, 2);

OLED.drawFrame(21-1, 57-5, 2, 2);

//stroom verticaal links

OLED.drawFrame(19, 55-1-5, 2, 2);

OLED.drawFrame(19, 51-1-5, 2, 2);

OLED.drawFrame(19, 47-1-5, 2, 2);

OLED.drawFrame(19, 43-1-5, 2, 2);

OLED.drawFrame(19, 39-1-5, 2, 2);

OLED.drawFrame(19, 35-1-5, 2, 2);

OLED.drawFrame(19, 31-1-5, 2, 2);

OLED.drawFrame(19, 27-1-5, 2, 2);

//stroom horizontaal links boven

OLED.drawFrame(19+1, 23-5, 2, 2);

OLED.drawFrame(23+1, 23-5, 2, 2);

OLED.drawFrame(27+1, 23-5, 2, 2);

OLED.drawFrame(31+1, 23-5, 2, 2);

OLED.drawFrame(35+1, 23-5, 2, 2);

//stroom horizontaal rechts boven

OLED.drawFrame(93+1, 23-5, 2, 2);

OLED.drawFrame(97+1, 23-5, 2, 2);

OLED.drawFrame(101+1, 23-5, 2, 2);

OLED.drawFrame(105+1, 23-5, 2, 2);

//stroom verticaal rechts

OLED.drawFrame(107, 25+1-5, 2, 2);

OLED.drawFrame(107, 29+1-5, 2, 2);

OLED.drawFrame(107, 33+1-5, 2, 2);

OLED.drawFrame(107, 37+1-5, 2, 2);

OLED.drawFrame(107, 41+1-5, 2, 2);

OLED.drawFrame(107, 45+1-5, 2, 2);

OLED.drawFrame(107, 49+1-5, 2, 2);

OLED.drawFrame(107, 53+1-5, 2, 2);

//stroom horizontaal rechts beneden

OLED.drawFrame(107-1, 57-5, 2, 2);

OLED.drawFrame(103-1, 57-5, 2, 2);

OLED.drawFrame(99-1, 57-5, 2, 2);

OLED.drawFrame(95-1, 57-5, 2, 2);

OLED.drawFrame(91-1, 57-5, 2, 2);

}

else if(frame == 2)

{

//stroom, horizontaal links onder

OLED.drawFrame(37-2, 57-5, 2, 2);

OLED.drawFrame(33-2, 57-5, 2, 2);

OLED.drawFrame(29-2, 57-5, 2, 2);

OLED.drawFrame(25-2, 57-5, 2, 2);

//stroom verticaal links

OLED.drawFrame(19, 59-2-5, 2, 2);

OLED.drawFrame(19, 55-2-5, 2, 2);

OLED.drawFrame(19, 51-2-5, 2, 2);

OLED.drawFrame(19, 47-2-5, 2, 2);

OLED.drawFrame(19, 43-2-5, 2, 2);

OLED.drawFrame(19, 39-2-5, 2, 2);

OLED.drawFrame(19, 35-2-5, 2, 2);

OLED.drawFrame(19, 31-2-5, 2, 2);

OLED.drawFrame(19, 27-2-5, 2, 2);

//stroom horizontaal links boven

OLED.drawFrame(19+2, 23-5, 2, 2);

OLED.drawFrame(23+2, 23-5, 2, 2);

OLED.drawFrame(27+2, 23-5, 2, 2);

OLED.drawFrame(31+2, 23-5, 2, 2);

OLED.drawFrame(35+2, 23-5, 2, 2);

//stroom horizontaal rechts boven

OLED.drawFrame(93+2, 23-5, 2, 2);

OLED.drawFrame(97+2, 23-5, 2, 2);

OLED.drawFrame(101+2, 23-5, 2, 2);

OLED.drawFrame(105+2, 23-5, 2, 2);

//stroom verticaal rechts

OLED.drawFrame(107, 25+2-5, 2, 2);

OLED.drawFrame(107, 29+2-5, 2, 2);

OLED.drawFrame(107, 33+2-5, 2, 2);

OLED.drawFrame(107, 37+2-5, 2, 2);

OLED.drawFrame(107, 41+2-5, 2, 2);

OLED.drawFrame(107, 45+2-5, 2, 2);

OLED.drawFrame(107, 49+2-5, 2, 2);

OLED.drawFrame(107, 53+2-5, 2, 2);

//stroom horizontaal rechts beneden

OLED.drawFrame(107-2, 57-5, 2, 2);

OLED.drawFrame(103-2, 57-5, 2, 2);

OLED.drawFrame(99-2, 57-5, 2, 2);

OLED.drawFrame(95-2, 57-5, 2, 2);

OLED.drawFrame(91-2, 57-5, 2, 2);

}

else if(frame == 3)

{

//stroom, horizontaal links onder

OLED.drawFrame(37-3, 57-5, 2, 2);

OLED.drawFrame(33-3, 57-5, 2, 2);

OLED.drawFrame(29-3, 57-5, 2, 2);

OLED.drawFrame(25-3, 57-5, 2, 2);

OLED.drawFrame(21-3, 57-5, 2, 2);

//stroom verticaal links

OLED.drawFrame(19, 59-3-5, 2, 2);

OLED.drawFrame(19, 55-3-5, 2, 2);

OLED.drawFrame(19, 51-3-5, 2, 2);

OLED.drawFrame(19, 47-3-5, 2, 2);

OLED.drawFrame(19, 43-3-5, 2, 2);

OLED.drawFrame(19, 39-3-5, 2, 2);

OLED.drawFrame(19, 35-3-5, 2, 2);

OLED.drawFrame(19, 31-3-5, 2, 2);

OLED.drawFrame(19, 27-3-5, 2, 2);

//stroom horizontaal links boven

OLED.drawFrame(19+3, 23-5, 2, 2);

OLED.drawFrame(23+3, 23-5, 2, 2);

OLED.drawFrame(27+3, 23-5, 2, 2);

OLED.drawFrame(31+3, 23-5, 2, 2);

OLED.drawFrame(35+3, 23-5, 2, 2);

//stroom horizontaal rechts boven

OLED.drawFrame(93+3, 23-5, 2, 2);

OLED.drawFrame(97+3, 23-5, 2, 2);

OLED.drawFrame(101+3, 23-5, 2, 2);

//stroom verticaal rechts

OLED.drawFrame(107, 21+3-5, 2, 2);

OLED.drawFrame(107, 25+3-5, 2, 2);

OLED.drawFrame(107, 29+3-5, 2, 2);

OLED.drawFrame(107, 33+3-5, 2, 2);

OLED.drawFrame(107, 37+3-5, 2, 2);

OLED.drawFrame(107, 41+3-5, 2, 2);

OLED.drawFrame(107, 45+3-5, 2, 2);

OLED.drawFrame(107, 49+3-5, 2, 2);

OLED.drawFrame(107, 53+3-5, 2, 2);

//stroom horizontaal rechts beneden

OLED.drawFrame(107-3, 57-5, 2, 2);

OLED.drawFrame(103-3, 57-5, 2, 2);

OLED.drawFrame(99-3, 57-5, 2, 2);

OLED.drawFrame(95-3, 57-5, 2, 2);

}

OLED.setDrawColor(1);

OLED.drawBitmap(32, -8-5, 64 / 8, 64, bitmap\_batterij);

OLED.setDrawColor(0);

OLED.drawBox(40, 11-5, 30, 26);

OLED.setDrawColor(1);

OLED.setFont(u8g\_font\_7x14);

OLEDstring = batterijPercentage;

OLED.drawStr(50, 23, OLEDstring.c\_str());

}

}while(OLED.nextPage());

}

else if (item\_selected1 == 2) //subMenu, Informatie

{

}

else if (item\_selected1 == 3) // subMenu, Instellingen

{

items\_screen = 6;

if(i1 == 0)

{

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//draw previous

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(22, 15, subMenu3\_items[item\_sel\_previous]);

OLED.drawBitmap(4, 2, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_sel\_previous]);

//draw selected

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(22, 15 + 20 + 2, subMenu3\_items[item\_selected]);

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_selected]);

//draw next

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(22, 15 + 20 + 20 + 2 + 2, subMenu3\_items[item\_sel\_next]);

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_sel\_next]);

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

}while(OLED.nextPage());

}

else if(i1 == 1)

{

OLED.firstPage();

do

{

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//draw previous

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(22, 15, subMenu3\_items[item\_sel\_previous]);

OLED.drawBitmap(4, 2, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_sel\_previous]);

//draw selected

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(22, 15 + 20 + 2, subMenu3\_items[item\_selected]);

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_selected]);

//draw next

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(22, 15 + 20 + 20 + 2 + 2, subMenu3\_items[item\_sel\_next]);

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_subMenu3[item\_sel\_next]);

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

//pop-up melding 'Opgeslagen

OLED.drawBitmap(0, 20-1, 128 / 8, 22, bitmap\_popUp\_outline);

OLED.setColorIndex(0);

OLED.drawBitmap(0, 20-1, 128 / 8, 22, bitmap\_popUp\_vulling);

OLED.setColorIndex(1);

OLED.drawStr(30, 34-1, "Opgeslagen");

}while(OLED.nextPage());

delay(1000);

i1 = 0;

}

}

}

//--------------------void screen2--------------------

void screen2()

{

if(item\_selected1 == 0) //--------------------submenu, 'Spel Selectie'--------------------

{

if(item\_selected2 == 0) //sub submenu, 'Wiezen'

{

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

if(i1 == 0)

{

nPersonen = 4;

nDeelBegin = 1;

item\_selected = 0;

items\_screen = 3; //items op huidige scherm

i1 = 1;

}

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

button\_up\_clicked = 1;

if(s1 == 0)

{

item\_selected--;

item\_selected = constrain(item\_selected, 0, 2);

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen++;

nPersonen = constrain(nPersonen, 3, 4);

}

else if(item\_selected == 1)

{

nDeelBegin++;

nDeelBegin = constrain(nDeelBegin, 1, 4);

}

}

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

button\_down\_clicked = 1;

if(s1 == 0)

{

item\_selected++;

item\_selected = constrain(item\_selected, 0, 2);

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen--;

nPersonen = constrain(nPersonen, 3, 4);

}

else if(item\_selected == 1)

{

nDeelBegin--;

nDeelBegin = constrain(nDeelBegin, 1, 4);

}

}

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//code

if(item\_selected == 0 || item\_selected == 1) s1 = 1;

else if(item\_selected == 2)

{

wiezen = 1;

zetten = 0;

blackJack = 0;

gelijkVerdelen = 0;

s1 = 1;

stepperCode();

}

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

}

//buton cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

if(s1 == 0)

{

//code

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

items\_screen = 4; //items op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

i1 = 0;

}

else if(s1 == 1) s1 = 0;

}

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 14, "Wiezen");

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

if(item\_selected == 0)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 2, "Personen:");

OLEDstring = nPersonen;

OLED.drawStr(75, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 20 + 2 + 2, OLEDstring.c\_str());

}

else if(item\_selected == 1)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 2, OLEDstring.c\_str());

//draw item selected outline

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

}

else if(item\_selected == 2)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

}

} while(OLED.nextPage());

}

else if(item\_selected2 == 1) // sub submenu, 'Zetten'

{

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

if(i1 == 0)

{

nPersonen = 4;

nDeelBegin = 1;

item\_selected = 0;

items\_screen = 3; //items op huidige scherm

i1 = 1;

}

items\_screen = 2; //items op huidige scherm

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

button\_up\_clicked = 1;

if(s1 == 0)

{

item\_selected--;

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen++;

nPersonen = constrain(nPersonen, 3, 4);

}

else if(item\_selected == 1)

{

nDeelBegin++;

nDeelBegin = constrain(nDeelBegin, 1, 4);

}

}

item\_selected = constrain(item\_selected, 0, 2);

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

button\_down\_clicked = 1;

if(s1 == 0)

{

item\_selected++;

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen--;

nPersonen = constrain(nPersonen, 3, 4);

}

else if(item\_selected == 1)

{

nDeelBegin--;

nDeelBegin = constrain(nDeelBegin, 1, 4);

}

}

item\_selected = constrain(item\_selected, 0, 2);

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//code

if(item\_selected == 0 || item\_selected == 1) s1 = 1;

else if(item\_selected == 2)

{

wiezen = 0;

zetten = 1;

blackJack = 0;

gelijkVerdelen = 0;

s1 = 1;

stepperCode();

}

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

}

//buton cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

if(s1 == 0)

{

//code

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

items\_screen = 4; //items op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

i1 = 0;

}

else if(s1 == 1) s1 = 0;

}

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 14, "Zetten");

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

if(item\_selected == 0)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 2, "Personen:");

OLEDstring = nPersonen;

OLED.drawStr(75, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 20 + 2 + 2, OLEDstring.c\_str());

}

else if(item\_selected == 1)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 2, OLEDstring.c\_str());

//draw item selected outline

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

}

else if(item\_selected == 2)

{

//draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "Begin Bij:");

OLEDstring = nDeelBegin;

OLED.drawStr(82, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

}

} while(OLED.nextPage());

}

else if(item\_selected2 == 2) // sub submenu, 'Black Jack'

{

}

else if(item\_selected2 == 3) //sub submenu, 'Gelijk Verdelen'

{

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

if(i1 == 0)

{

EEPROM.begin("instellingen", false);

nKaarten = EEPROM.getInt("nKaarten");

nPersonen = EEPROM.getInt("nPersonen");

EEPROM.end();

item\_selected = 0;

i1 = 1;

}

items\_screen = 3; //items op huidige scherm

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

if(s1 == 0)

{

item\_selected--;

button\_up\_clicked = 1;

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen++;

nPersonen = constrain(nPersonen, 1, 18);

if((nKaarten\*nPersonen) > 54) nPersonen--;

delay(150);

}

else if(item\_selected == 1)

{

nKaarten++;

nKaarten = constrain(nKaarten, 1, 54);

if((nKaarten\*nPersonen) > 54) nKaarten--;

delay(75);

}

}

item\_selected = constrain(item\_selected, 0, 2);

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

if(s1 == 0)

{

item\_selected++;

button\_down\_clicked = 1;

}

else if(s1 == 1)

{

if(item\_selected == 0)

{

nPersonen--;

nPersonen = constrain(nPersonen, 1, 18);

delay(150);

}

else if(item\_selected == 1)

{

nKaarten--;

nKaarten = constrain(nKaarten, 1, 54);

delay(75);

}

}

item\_selected = constrain(item\_selected, 0, 2);

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//code

if(item\_selected == 0 || item\_selected == 1) s1 = 1;

else if(item\_selected == 2)

{

wiezen = 0;

zetten = 0;

blackJack = 0;

gelijkVerdelen = 1;

s1 = 1;

EEPROM.begin("instellingen", false);

EEPROM.putInt("nKaarten", nKaarten);

EEPROM.putInt("nPersonen", nPersonen);

EEPROM.end();

stepperCode();

}

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

}

//buton cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

if(s1 == 0)

{

//code

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

items\_screen = 4; //items op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

i1 = 0;

}

else if(s1 == 1) s1 = 0;

}

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(12, 14, "Gelijk Verdelen");

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

if(item\_selected == 0)

{

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

OLEDstring = nPersonen;

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 2, "Personen:");

OLED.drawStr(75, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

OLEDstring = nKaarten;

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Kaarten:");

OLED.drawStr(68, 15 + 20 + 20 + 2 + 2, OLEDstring.c\_str());

}

else if(item\_selected == 1)

{

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLEDstring = nKaarten;

OLED.drawStr(4, 15 + 20 + 2, "Kaarten:");

OLED.drawStr(68, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

}

else if(item\_selected == 2)

{

if(s1 == 0)

{

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

OLEDstring = nKaarten;

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "Kaarten:");

OLED.drawStr(68, 15 + 20 + 2, OLEDstring.c\_str());

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

if(s1 == 0) OLED.setFont(u8g\_font\_7x14);

else if(s1 == 1) OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Start Delen");

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

}

}while(OLED.nextPage());

}

}

else if(item\_selected1 == 1) //--------------------sub menu, 'informatie'--------------------

{

//hier komt waarschijnlijk ook niets

}

else if(item\_selected1 == 2) //--------------------sub menu, 'Batterij'--------------------

{

//Hier staat niets

}

else if(item\_selected1 == 3) //--------------------submenu, 'Instellingen'--------------------

{

if(item\_selected2 == 0) //sub submenu -----'Geluid'-----

{

if(i1 == 0)

{

toggleOnOff = geluid;

i1 = 1;

}

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

items\_screen = 2; //word niet gebruikt

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

button\_up\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

toggleOnOff = 1;

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

button\_down\_clicked = 1;

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

toggleOnOff = 0;

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//code

geluid = toggleOnOff;

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

items\_screen = 5; //items op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

//sla verandering in waarde 'geluid' op in EEPROM

EEPROM.begin("instellingen", false);

EEPROM.putBool("geluid", geluid);

EEPROM.end();

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

}

//buton cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

//code

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

items\_screen = 5; //items op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

i1 = 0;

current\_screen--;

}

//AAN geselecteerd

if(toggleOnOff == 1)

{

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 15, "Geluid");

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 2, "aan");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_geluid);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "uit");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_geluid\_uit);

}while(OLED.nextPage());

}

//UIT geselecteerd

else if(toggleOnOff == 0)

{

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 15, "Geluid");

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 2, "aan");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_geluid);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "uit");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_geluid\_uit);

} while (OLED.nextPage());

}

}

else if (item\_selected2 == 1) //-----sub submenu 'Contrast'-----

{

if(i1 == 0)

{

settingValue = contrast;

i1 = 1;

}

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

items\_screen = 1; //word niet gebruikt

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

contrast = contrast + 5;

contrast = constrain(contrast, 0, 100);

delay(150); //I.P.V. button\_up\_clicked = 1, zodat de knop ingedrukt kan blijven om het percentage snel aan te passen

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

contrast = contrast - 5;

contrast = constrain(contrast, 0, 100);

delay(150); //I.P.V. button\_down\_clicked = 1, zodat de knop ingedrukt kan blijven om het percentage snel aan te passen

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

//code

items\_screen = 5; //items op het vorige frame

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

//sla verandering in waarde 'contrast' op in EEPROM

EEPROM.begin("instellingen", false);

EEPROM.putInt("contrast", contrast);

EEPROM.end();

}

//button cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

//code

contrast = settingValue;

i1 = 0;

items\_screen = 5; //items op het vorige frame

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

}

settingString = contrast;

settingString = settingString + "%";

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(35, 15, "Contrast");

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 2, settingString.c\_str());

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_contrast);

}while (OLED.nextPage());

}

else if(item\_selected2 == 2) //-----sub submenu 'Tijd en datum'-----

{

if(i1 == 0)

{

h1 = 0; //hulpvariabele voor UI in, sub submenu 'Tijd en datum' en 'Wifi'

item\_selected = 0;

items\_screen = 2;

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

i1 = 1;

}

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

if(h1 == 0) item\_selected = 0;

if(item\_selected == 0)

{

if(h1 == 1) Digit1++;

else if(h1 == 2) Digit2++;

else if(h1 == 3) Digit3++;

else if(h1 == 4) Digit4++;

else if(h1 == 5) Digit5++;

else if(h1 == 6) Digit6++;

}

else if(item\_selected == 1)

{

if(h1 == 1) Digit7++;

else if(h1 == 2) Digit8++;

else if(h1 == 3) Digit9++;

else if(h1 == 4) Digit10++;

else if(h1 == 5) Digit11++;

else if(h1 == 6) Digit12++;

}

//constrain tijd

Digit1 = constrain(Digit1, 0, 2);

if(Digit1 == 2) Digit2 = constrain(Digit2, 0, 3);

else constrain(Digit2, 0, 9);

Digit3 = constrain(Digit3, 0, 5);

Digit4 = constrain(Digit4, 0, 9);

Digit5 = constrain(Digit5, 0, 5);

Digit6 = constrain(Digit6, 0, 9);

//constrain datum

Digit9 = constrain(Digit9, 0, 1);

if(Digit9 == 1) Digit10 = constrain(Digit10, 1, 2);

else Digit10 = constrain(Digit10, 1, 9);

Digit11 = constrain(Digit11, 0, 9);

Digit12 = constrain(Digit12, 0, 9);

//!WAARSCHUWING! sD7 en sD8 worden hier gebruikt als hulpvariabelen en worden hierna onmiddelijk hersteld

sD9 = Digit9;

sD10 = Digit10;

sD9 = sD9 + sD10;

maand1 = sD9.toInt();

if(maand1 == 1 || maand1 == 3 || maand1 == 5 || maand1 == 7 || maand1 == 8 || maand1 == 10 || maand1 == 12) //maanden die 31 dagen hebben

{

Digit7 = constrain(Digit7, 0, 3);

if(Digit7 == 3) Digit8 = constrain(Digit8, 0, 1);

else if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else Digit8 = constrain(Digit8, 0, 9);

} else if(maand1 == 4 || maand1 == 6 || maand1 == 9 || maand1 == 11) //maanden die 30 dagen hebben

{

Digit7 = constrain(Digit7, 0, 3);

if(Digit7 == 3) Digit8 = 0;

else if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else Digit8 = constrain(Digit8, 0, 9);

} else if(maand1 == 2) //februari 🤬

{

//check voor schrikkeljaar 🤬

sD11 = Digit11;

sD12 = Digit12;

str = sD11 + sD12;

jaar1 = str.toInt();

//bepaal schrikkeljaar

if((2000 + jaar1) % 4 == 0)

{

schrikkel\_jaar = 1;

if((2000 + jaar1) % 100 == 0)

{

schrikkel\_jaar = 0;

if((2000 + jaar1) % 400 == 0)

{

schrikkel\_jaar = 1;

}

}

}

else schrikkel\_jaar = 0;

if(schrikkel\_jaar == 0)

{

Digit7 = constrain(Digit7, 0, 2);

if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else if(Digit7 == 1) Digit8 = constrain(Digit8, 0, 9);

else Digit8 = constrain(Digit8, 0, 8);

}

else if(schrikkel\_jaar == 1)

{

Digit7 = constrain(Digit7, 0, 2);

if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else if(Digit7 == 1) Digit8 = constrain(Digit8, 0, 9);

else Digit8 = constrain(Digit8, 0, 9);

}

}

if(h1 != 0) delay(300);

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

//code

if(h1 == 0) item\_selected = 1;

if(item\_selected == 0)

{

if(h1 == 1) Digit1--;

else if(h1 == 2) Digit2--;

else if(h1 == 3) Digit3--;

else if(h1 == 4) Digit4--;

else if(h1 == 5) Digit5--;

else if(h1 == 6) Digit6--;

}

if(item\_selected == 1)

{

if (h1 == 1) Digit7--;

else if(h1 == 2) Digit8--;

else if(h1 == 3) Digit9--;

else if(h1 == 4) Digit10--;

else if(h1 == 5) Digit11--;

else if(h1 == 6) Digit12--;

}

//constrain tijd

Digit1 = constrain(Digit1, 0, 2);

if(Digit1 == 2) Digit2 = constrain(Digit2, 0, 3);

else Digit2 = constrain(Digit2, 0, 9);

Digit3 = constrain(Digit3, 0, 5);

Digit4 = constrain(Digit4, 0, 9);

Digit5 = constrain(Digit5, 0, 5);

Digit6 = constrain(Digit6, 0, 9);

//constrain datum

Digit9 = constrain(Digit9, 0, 1);

if(Digit9 == 1) Digit10 = constrain(Digit10, 1, 2);

else Digit10 = constrain(Digit10, 1, 9);

Digit11 = constrain(Digit11, 0, 9);

Digit12 = constrain(Digit12, 0, 9);

//!WAARSCHUWING! sD7 en sD8 worden hier gebruikt als hulpvariabelen en worden hierna onmiddelijk hersteld

sD9 = Digit9;

sD10 = Digit10;

sD9 = sD9 + sD10;

maand1 = sD9.toInt();

if(maand1 == 1 || maand1 == 3 || maand1 == 5 || maand1 == 7 || maand1 == 8 || maand1 == 10 || maand1 == 12) //maanden die 31 dagen hebben

{

Digit7 = constrain(Digit7, 0, 3);

if(Digit7 == 3) Digit8 = constrain(Digit8, 0, 1);

else if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else Digit8 = constrain(Digit8, 0, 9);

}

else if(maand1 == 4 || maand1 == 6 || maand1 == 9 || maand1 == 11) //maanden die 30 dagen hebben

{

Digit7 = constrain(Digit7, 0, 3);

if(Digit7 == 3) Digit8 = 0;

else if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else Digit8 = constrain(Digit8, 0, 9);

}

else if(maand1 == 2) //februari 🤬

{

//check voor schrikkeljaar 🤬

sD11 = Digit11;

sD12 = Digit12;

str = sD11 + sD12;

jaar1 = str.toInt();

//bepaal schrikkeljaar

if((2000 + jaar1) % 4 == 0)

{

schrikkel\_jaar = 1;

if((2000 + jaar1) % 100 == 0)

{

schrikkel\_jaar = 0;

if((2000 + jaar1) % 400 == 0)

{

schrikkel\_jaar = 1;

}

}

}

else schrikkel\_jaar = 0;

if(schrikkel\_jaar == 0)

{

Digit7 = constrain(Digit7, 0, 2);

if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else if(Digit7 == 1) Digit8 = constrain(Digit8, 0, 9);

else Digit8 = constrain(Digit8, 0, 8);

}

else if(schrikkel\_jaar == 1)

{

Digit7 = constrain(Digit7, 0, 2);

if(Digit7 == 0) Digit8 = constrain(Digit8, 1, 9);

else if(Digit7 == 1) Digit8 = constrain(Digit8, 0, 9);

else Digit8 = constrain(Digit8, 0, 9);

}

}

if(h1 != 0) delay(300);

}

sD1 = Digit1;

sD2 = Digit2;

sD3 = Digit3;

sD4 = Digit4;

sD5 = Digit5;

sD6 = Digit6;

sD7 = Digit7;

sD8 = Digit8;

sD9 = Digit9;

sD10 = Digit10;

sD11 = Digit11;

sD12 = Digit12;

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

//code

h1++;

h1 = constrain(h1, 0, 7);

if(h1 == 1)

{

rtc.refresh();

seconden1 = rtc.second();

minuten1 = rtc.minute();

uren1 = rtc.hour();

dag1 = rtc.day();

maand1 = rtc.month();

jaar1 = rtc.year();

//tijd digits

Digit2 = uren1 % 10;

uren1 /= 10;

Digit1 = uren1 % 10;

Digit4 = minuten1 % 10;

minuten1 /= 10;

Digit3 = minuten1 % 10;

Digit6 = seconden1 % 10;

seconden1 /= 10;

Digit5 = seconden1 % 10;

//datum digits

Digit8 = dag1 % 10;

dag1 /= 10;

Digit7 = dag1 % 10;

Digit10 = maand1 % 10;

maand1 /= 10;

Digit9 = maand1 % 10;

Digit12 = jaar1 % 10;

jaar1 /= 10;

Digit11 = jaar1 % 10;

}

if(h1 == 7)

{

if(item\_selected == 0) //sla de verandering in tijd op in de RTC, DS3231

{

str = sD1 + sD2;

uren1 = str.toInt();

str = sD3 + sD4;

minuten1 = str.toInt();

str = sD5 + sD6;

seconden1 = str.toInt();

rtc.set(seconden1, minuten1, uren1, rtc.dayOfWeek(), rtc.day(), rtc.month(), rtc.year()); //rtc.set(sec, min, hour, day of week, day of month, month, year)

}

else if(item\_selected == 1) //sla de verandering in datum op in de RTC, DS3231

{

str = sD7 + sD8;

dag1 = str.toInt();

str = sD9 + sD10;

maand1 = str.toInt();

str = sD11 + sD12;

jaar1 = str.toInt();

//bepaal jaarcode

jaarcode = (jaar1 + (jaar1 / 4)) % 7;

//bepaal maandcode

if(maand1 == 1 || maand1 == 10) maandcode = 0;

else if(maand1 == 5) maandcode = 1;

else if(maand1 == 8) maandcode = 2;

else if(maand1 == 2 || maand1 == 3 || maand1 == 11) maandcode = 3;

else if(maand1 == 6) maandcode = 4;

else if(maand1 == 9 || maand1 == 12) maandcode = 5;

else if(maand1 == 4 || maand1 == 7) maandcode = 6;

//bepaal eeuwcode

/\*

eeuwcodes

1700s = 4

1800s = 2

1900s = 0

-> 2000s = 6 <- wij gebruiken momenteel standaard deze, deze gaat nog 76 jaar mee

2100s = 4

2200s = 2

2300s = 0

\*/

eeuwcode = 6;

//bepaal schrikkeljaar

if((2000 + jaar1) % 4 == 0)

{

schrikkel\_jaar = 1;

if((2000 + jaar1) % 100 == 0)

{

schrikkel\_jaar = 0;

if((2000 + jaar1) % 400 == 0)

{

schrikkel\_jaar = 1;

}

}

}

//bepaal weekdag

if(schrikkel\_jaar == 1)

{

if(maand1 == 1 || maand1 == 2)

{

weekdag = (jaarcode + maandcode + eeuwcode + dag1 - 1);

}

else

{

weekdag = (jaarcode + maandcode + eeuwcode + dag1);

}

}

else

{

weekdag = (jaarcode + maandcode + eeuwcode + dag1);

}

rtc.set(rtc.second(), rtc.minute(), rtc.hour(), weekdag, dag1, maand1, jaar1);

}

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(15, 15, "Tijd en datum");

if(item\_selected == 0)

{

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(30, 15 + 20 + 2, "Tijd");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_tijd);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "Datum");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_datum);

}

else if (item\_selected == 1)

{

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 2, "Tijd");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_tijd);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "Datum");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_datum);

}

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

//pop-up melding 'Opgeslagen

OLED.drawBitmap(0, 20 - 1, 128 / 8, 22, bitmap\_popUp\_outline);

OLED.setColorIndex(0);

OLED.drawBitmap(0, 20 - 1, 128 / 8, 22, bitmap\_popUp\_vulling);

OLED.setColorIndex(1);

OLED.drawStr(30, 34 - 1, "Opgeslagen");

}while (OLED.nextPage());

delay(1000);

h1 = 0;

}

}

//button cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

//code

h1--;

if(h1 < 0)

{

i1 = 0;

items\_screen = 5; //items op het vorige frame

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

}

}

//OLED

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

if(h1 == 0)

{

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(15, 15, "Tijd en datum");

if(item\_selected == 0)

{

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(30, 15 + 20 + 2, "Tijd");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_tijd);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "Datum");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_datum);

}

else if(item\_selected == 1)

{

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(30, 15 + 20 + 2, "Tijd");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_tijd);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(30, 15 + 20 + 20 + 2 + 2, "Datum");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_datum);

}

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

}

else

{

if(item\_selected == 0)

{

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(50, 15, "Tijd");

OLED.setColorIndex(1);

OLED.drawBitmap(12, 26, 16 / 8, 16, bitmap\_icon\_tijd);

OLEDstring = sD1 + sD2 + ":" + sD3 + sD4 + ":" + sD5 + sD6;

OLED.drawStr(36, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawHLine(32, 40, 64);

OLED.drawHLine(32, 41, 64);

}

else

{

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(46, 15, "Datum");

OLED.setColorIndex(1);

OLED.drawBitmap(7, 26, 16 / 8, 16, bitmap\_icon\_datum);

OLEDstring = sD7 + sD8 + "/" + sD9 + sD10 + "/" + "20" + sD11 + sD12;

OLED.drawStr(36 - 5, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawHLine(32 - 5, 40, 76);

OLED.drawHLine(32 - 5, 41, 76);

}

}

OLED.setColorIndex(0);

if(h1 > 4)

{

if(item\_selected == 0)

{

OLED.drawHLine(77 + (7 \* (h1 - 5)), 40, 8);

OLED.drawHLine(77 + (7 \* (h1 - 5)), 41, 8);

}

else if(item\_selected == 1)

{

OLED.drawHLine((91 - 5) + (7 \* (h1 - 5)), 40, 8);

OLED.drawHLine((91 - 5) + (7 \* (h1 - 5)), 41, 8);

}

}

else if(h1 > 2)

{

if(item\_selected == 0)

{

OLED.drawHLine(56 + (7 \* (h1 - 3)), 40, 8);

OLED.drawHLine(56 + (7 \* (h1 - 3)), 41, 8);

}

else if(item\_selected == 1)

{

OLED.drawHLine((56 - 5) + (7 \* (h1 - 3)), 40, 8);

OLED.drawHLine((56 - 5) + (7 \* (h1 - 3)), 41, 8);

}

}

else if(h1 > 0)

{

if(item\_selected == 0)

{

OLED.drawHLine(35 + (7 \* (h1 - 1)), 40, 8);

OLED.drawHLine(35 + (7 \* (h1 - 1)), 41, 8);

}

else if(item\_selected == 1)

{

OLED.drawHLine((35 - 5) + (7 \* (h1 - 1)), 40, 8);

OLED.drawHLine((35 - 5) + (7 \* (h1 - 1)), 41, 8);

}

}

}while (OLED.nextPage());

}

else if(item\_selected2 == 3) //-----sub submenu 'Reset'-----

{

digitalWrite(relaisPin, LOW); //checktag

resetFunc(); //call void resetFunc

}

else if(item\_selected2 == 4) //-----sub submenu 'Kalibreren'-----

{

//code

if(i1 == 0)

{

calibrated = 0;

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

i1 = 1;

}

haltSensVal = analogRead(haltSensPin);

stepper.setAcceleration(300);

stepper.setMaxSpeed(300);

stepper.move(400); //vorige: stepper.moveTo(400); , testen of dit beter werkt

stepper.run();

length = map(400 - stepper.distanceToGo() ,0 ,400 ,0 , 107);

//calibration

if(haltSensVal > lastHaltSensVal)

{

stepsToZero = 400-stepper.distanceToGo();

lastHaltSensVal = haltSensVal;

}

if(stepper.distanceToGo() == 0)

{

delay(250);

stepper.setAcceleration(150);

stepper.setMaxSpeed(300);

stepper.setCurrentPosition(0);

if(stepsToZero<200) stepper.runToNewPosition(stepsToZero);

else if(stepsToZero>=200) stepper.runToNewPosition(stepsToZero-400);

stepper.setCurrentPosition(0);

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

i1 = 0;

calibrated = 1;

lastHaltSensVal = 0;

item\_selected = item\_selected2;

current\_screen--;

}

}

else if(item\_selected2 == 5) ////-----sub submenu 'Wifi'-----

{

if(i1 == 0)

{

h1 = 0; //hulpvariabele voor UI in, sub submenu 'Tijd en datum' en 'Wifi'

item\_selected = 0;

selected\_network = 0;

items\_screen = 2;

screen\_delay = 1;

frame = 0;

WiFi.mode(WIFI\_STA);

WiFi.scanDelete();

button\_up\_enable = 0;

button\_down\_enable = 0;

button\_select\_enable = 0;

button\_cancel\_enable = 0;

i1 = 1;

}

//button up

if(digitalRead(BUTTON\_UP\_PIN) == LOW && button\_up\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

if(h1 == 0) item\_selected = 0;

else if(h1 == 1)

{

if(item\_selected == 1)

{

selected\_network--;

selected\_network = constrain(selected\_network, 0, networks);

button\_up\_clicked = 1;

}

}

else if(h1 == 2)

{

if(item\_selected == 1)

{

//toetsenbord functie

if(yKeyBoard == 2)

{

if(xKeyBoard == 0) xKeyBoard = 2;

else if(xKeyBoard == 8) xKeyBoard = 0;

else xKeyBoard++;

}

else if(yKeyBoard == 3)

{

if(xKeyBoard == 0) xKeyBoard = 2;

else if(1 < xKeyBoard && xKeyBoard < 7) xKeyBoard = 9;

else if(xKeyBoard == 8) xKeyBoard = 0;

else xKeyBoard++;

}

else xKeyBoard++;

if(9 < xKeyBoard) xKeyBoard = 0;

button\_up\_clicked = 1;

}

}

}

//button down

if(digitalRead(BUTTON\_DOWN\_PIN) == LOW && button\_down\_clicked == 0)

{

//buzzer

if(geluid == 1) tone(buzzer, 850, 50);

if(h1 == 0) item\_selected = 1;

else if(h1 == 1)

{

if(item\_selected == 1)

{

selected\_network++;

selected\_network = constrain(selected\_network, 0, networks);

button\_down\_clicked = 1;

}

}

else if(h1 == 2)

{

if(item\_selected == 1)

{

//toetsenbord functie

yKeyBoard++;

if(3 < yKeyBoard) yKeyBoard = 0;

button\_down\_clicked = 1;

}

}

}

//button select

if(digitalRead(BUTTON\_SELECT\_PIN) == LOW && button\_select\_clicked == 0)

{

button\_select\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1000); // 1000

delay(50); // 50

tone(buzzer, 1500); // 1500

delay(100); // 100

noTone(buzzer);

}

if(h1 == 0)

{

h1 = 1;

if(item\_selected == 1)

{

screen\_delay = 1;

scan\_completed = 0;

selected\_network = 0;

WiFi.scanDelete();

}

}

else if(h1 == 1)

{

button\_select\_clicked = 1;

if(item\_selected == 0) WiFi.disconnect();

else if(item\_selected == 1)

{

if(selected\_network == networks)

{

scan\_completed = 0;

selected\_network = 0;

WiFi.scanDelete();

}

else

{

ssid = WiFi.SSID(selected\_network);

h1 = 2;

}

}

}

else if(h1 == 2)

{

//toetsenbord functie

if((xKeyBoard == 0 || xKeyBoard == 1) && yKeyBoard == 2) //letter shift

{

if(keyBoardMode == 0) keyBoardMode = 1;

else if(keyBoardMode == 1) keyBoardMode = 0;

else if(keyBoardMode == 2) keyBoardMode = 0;

else if(keyBoardMode == 3) keyBoardMode = 0;

}

else if((xKeyBoard == 0 || xKeyBoard == 1) && yKeyBoard == 3) //teken shift

{

if(keyBoardMode == 0) keyBoardMode = 2;

else if(keyBoardMode == 1) keyBoardMode = 2;

else if(keyBoardMode == 2) keyBoardMode = 3;

else if(keyBoardMode == 3) keyBoardMode = 2;

}

else if((xKeyBoard == 8 || xKeyBoard == 9) && yKeyBoard == 2) //BACKSPACE

{

text.remove(text.length()-1);

delay(100);

}

else if((xKeyBoard == 8 || xKeyBoard == 9) && yKeyBoard == 3) //ENTER

{

//ENTER

pass = text;

h1 = 3;

button\_select\_clicked = 1;

WiFi.begin(ssid, pass);

connect\_attempt = 1;

}

else if(1 < xKeyBoard && xKeyBoard < 8 && yKeyBoard == 3) //spatiebalk

{

text = text + ' ';

delay(100);

}

else //print geselecteerde character in typ balk

{

if(text.length() < 63)

{

if(keyBoardMode == 0) text = text + keyBoard\_a[yKeyBoard][xKeyBoard];

else if(keyBoardMode == 1) text = text + keyBoard\_A[yKeyBoard][xKeyBoard];

else if(keyBoardMode == 2)

{

if(xKeyBoard == 6 && yKeyBoard == 1)

{

text = text + '€';

}

else text = text + keyBoard\_1[yKeyBoard][xKeyBoard];

}

else if(keyBoardMode == 3) text = text + keyBoard\_2[yKeyBoard][xKeyBoard];

delay(100);

}

else errorMaxPasswordLength = 1;

}

}

}

//button cancel

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

//buzzer

if(geluid == 1)

{

tone(buzzer, 1500); // 1500

delay(50); // 50

tone(buzzer, 1000); // 1000

delay(100); // 100

noTone(buzzer);

}

if(h1 == 0)

{

i1 = 0;

items\_screen = 5; //items op het vorige frame

item\_selected = item\_selected2; //geselecteerde item op het vorige frame

button\_up\_enable = 1;

button\_down\_enable = 1;

button\_select\_enable = 1;

button\_cancel\_enable = 1;

current\_screen--;

}

if(h1 == 1)

{

selected\_network = 0;

screen\_delay = 1;

scan\_completed = 0;

WiFi.scanDelete();

}

h1--;

}

if(digitalRead(BUTTON\_UP\_PIN) == HIGH && button\_up\_clicked == 1) button\_up\_clicked = 0;

if(digitalRead(BUTTON\_DOWN\_PIN) == HIGH && button\_down\_clicked == 1) button\_down\_clicked = 0;

if(digitalRead(BUTTON\_SELECT\_PIN) == HIGH && button\_select\_clicked == 1) button\_select\_clicked = 0;

if(digitalRead(BUTTON\_CANCEL\_PIN) == HIGH && button\_cancel\_clicked == 1) button\_cancel\_clicked = 0;

//OLED

OLED.firstPage();

do

{

if(h1 == 0)

{

OLED.setDrawColor(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(50, 15, "Wifi");

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

// draw scrollbar handle

OLED.drawBox(125, 64 / items\_screen \* item\_selected, 3, 64 / items\_screen);

if(item\_selected == 0)

{

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_profont12);

if(WiFi.status() == WL\_CONNECTED)

{

if(WiFi.SSID() == "")

{

OLED.drawStr(24, 15 + 20 + 2, "Niet verbonden");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi4);

}

else

{

OLEDstring = WiFi.SSID();

OLED.drawStr(24, 15 + 20 + 2, OLEDstring.c\_str());

if(WiFi.RSSI() < -90) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI() < -79) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI() < -50) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI() < 0) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi);

}

}

else

{

OLED.drawStr(24, 15 + 20 + 2, "Niet verbonden");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi4);

}

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, "Scan netwerken");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi);

}

else if(item\_selected == 1)

{

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_profont12);

if(WiFi.status() == WL\_CONNECTED)

{

OLEDstring = WiFi.SSID();

OLED.drawStr(24, 15 + 20 + 2, OLEDstring.c\_str());

if(WiFi.RSSI() < -90) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI() < -79) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI() < -50) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI() < 0) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi);

}

else

{

OLED.drawStr(24, 15 + 20 + 2, "Niet verbonden");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi4);

}

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, "Scan netwerken");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi);

}

}

else if(h1 == 1)

{

if(item\_selected == 0)

{

OLED.setDrawColor(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(50, 15, "Wifi");

OLED.setFont(u8g\_font\_7x14);

if(WiFi.status() == WL\_CONNECTED)

{

OLEDstring = WiFi.SSID();

OLED.drawStr(24, 15 + 20 + 2, OLEDstring.c\_str());

if(WiFi.RSSI() < -90) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI() < -79) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI() < -50) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI() < 0) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, "Verbreek");

OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi4);

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

else

{

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(24, 15 + 20 + 2, "Niet verbonden");

OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi4);

}

}

else if(item\_selected == 1)

{

if(scan\_completed == 0)

{

OLED.setDrawColor(1);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawBitmap(39, 2, 56 / 8, 51, bitmap\_scan);

OLED.drawStr(34, 62, "Scannen...");

}

else if(scan\_completed == 1)

{

if(screen\_delay == 1)

{

if(0 < networks)

{

OLED.drawBitmap(44, 17, 48 / 8, 31, bitmap\_wifi\_groot);

OLEDstring = networks;

OLEDstring += " netwerken gevonden";

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(2, 62, OLEDstring.c\_str());

}

else if(networks == 0)

{

OLED.drawBitmap(44, 17, 48 / 8, 31, bitmap\_wifi\_groot1);

OLEDstring = networks;

OLEDstring += " netwerken gevonden";

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(2, 62, OLEDstring.c\_str());

}

}

else if(screen\_delay == 0)

{

OLED.setDrawColor(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(32, 15, "Netwerken");

OLED.setFont(u8g\_font\_profont12);

OLEDstring = WiFi.SSID(selected\_network);

OLED.drawStr(24, 15 + 20 + 2, OLEDstring.c\_str());

if(WiFi.RSSI(selected\_network) < -90) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI(selected\_network) < -79) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI(selected\_network) < -50) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI(selected\_network) < 0) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = WiFi.SSID(selected\_network+1);

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, OLEDstring.c\_str());

if(WiFi.RSSI(selected\_network+1) < -90) OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI(selected\_network+1) < -79) OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI(selected\_network+1) < -50) OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI(selected\_network+1) < 0) OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icon\_wifi);

if(selected\_network == networks-1)

{

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, "Scan opnieuw");

OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

else if(selected\_network == networks)

{

OLED.drawStr(24, 15 + 20 + 20 + 2 + 2, "Scan opnieuw");

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = WiFi.SSID(selected\_network-1);

OLED.drawStr(24, 15 + 20 + 2, OLEDstring.c\_str());

if(WiFi.RSSI(selected\_network-1) < -90) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi3);

else if(WiFi.RSSI(selected\_network-1) < -79) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi2);

else if(WiFi.RSSI(selected\_network-1) < -50) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi1);

else if(WiFi.RSSI(selected\_network-1) < 0) OLED.drawBitmap(4, 24, 16 / 8, 16, bitmap\_icon\_wifi);

}

else OLED.drawBitmap(0, 22, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

}

}

}

else if(h1 == 2)

{

//toetsenbordcode

//spatiebalk

OLED.setDrawColor(1);

OLED.drawRFrame(29, 55, 71 , 8, 2);

//typ balk

OLED.setDrawColor(1);

OLED.drawFrame(2, 2, 124, 14);

//tekst

if(errorMaxPasswordLength == 0)

{

OLED.setFont(u8g\_font\_profont12);

if(OLED.getStrWidth(text.c\_str()) > 114)

{

OLED.drawStr(5-6-OLED.getStrWidth(text.c\_str())+120, 13, text.c\_str());

}

else OLED.drawStr(5, 13, text.c\_str());

}

//selectie box

OLED.setDrawColor(1);

drawColorLetterShift = 0;

drawColorSignShift = 0;

if((xKeyBoard == 0 || xKeyBoard == 1) && yKeyBoard == 2) //letter shift

{

drawColorLetterShift = 1;

}

else if((xKeyBoard == 0 || xKeyBoard == 1) && yKeyBoard == 3) //teken shift

{

drawColorSignShift = 1;

}

else if((xKeyBoard == 8 || xKeyBoard == 9) && yKeyBoard == 2) //BACKSPACE

{

OLED.drawRBox(105, 44, 21, 9, 3);

}

else if((xKeyBoard == 8 || xKeyBoard == 9) && yKeyBoard == 3) //ENTER

{

OLED.drawRBox(105, 55, 21, 9, 3);

}

else if(1 < xKeyBoard && xKeyBoard < 8 && yKeyBoard == 3) //spatiebalk

{

OLED.drawRBox(29, 55, 71 , 8, 2);

}

else OLED.drawBox(2+(13\*xKeyBoard), 19+(12\*yKeyBoard), 7, 11);

for(int rij = 0; rij < 3; rij++)

{

for(int kolom = 0; kolom < 10; kolom++)

{

if(keyBoardMode == 0)

{

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = keyBoard\_a[rij][kolom];

OLED.drawStr(3+(13\*kolom), 28+(12\*rij), OLEDstring.c\_str());

//shift letters key board

OLED.setDrawColor(drawColorLetterShift);

OLED.drawRBox(3, 44, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(7, 51, "ABC");

//shift tekens key board

OLED.setDrawColor(drawColorSignShift);

OLED.drawRBox(3, 55, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 62, "123");

}

else if(keyBoardMode == 1)

{

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = keyBoard\_A[rij][kolom];

OLED.drawStr(3+(13\*kolom), 28+(12\*rij), OLEDstring.c\_str());

//shift letters key board

OLED.setDrawColor(drawColorLetterShift);

OLED.drawRBox(3, 44, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 51, "abc");

//shift tekens key board

OLED.setDrawColor(drawColorSignShift);

OLED.drawRBox(3, 55, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 62, "123");

}

else if(keyBoardMode == 2)

{

if(rij == 1 && kolom == 6)

{

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g2\_font\_unifont\_t\_symbols);

OLED.drawGlyph(3+(13\*kolom)-2, 28+(12\*rij)+1, 0x20AC);

}

else

{

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = keyBoard\_1[rij][kolom];

OLED.drawStr(3+(13\*kolom), 28+(12\*rij), OLEDstring.c\_str());

}

//shift letters key board

OLED.setDrawColor(drawColorLetterShift);

OLED.drawRBox(3, 44, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 51, "abc");

//shift tekens key board

OLED.setDrawColor(drawColorSignShift);

OLED.drawRBox(3, 55, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(7, 62, "#+=");

}

else if(keyBoardMode == 3)

{

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g\_font\_profont12);

OLEDstring = keyBoard\_2[rij][kolom];

OLED.drawStr(3+(13\*kolom), 28+(12\*rij), OLEDstring.c\_str());

//shift letters key board

OLED.setDrawColor(drawColorLetterShift);

OLED.drawRBox(3, 44, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 51, "abc");

//shift tekens key board

OLED.setDrawColor(drawColorSignShift);

OLED.drawRBox(3, 55, 21, 9, 3);

OLED.setFontMode(1);

OLED.setDrawColor(2);

OLED.setFont(u8g2\_font\_tiny5\_tf);

OLED.drawStr(8, 62, "123");

}

}

}

//shift letters key board kader

OLED.setDrawColor(1);

OLED.drawRFrame(3, 44, 21, 9, 3);

//shift tekens key board kader

OLED.drawRFrame(3, 55, 21, 9, 3);

//BACKSPACE

OLED.setDrawColor(1);

OLED.drawRFrame(105, 44, 21, 9, 3);

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g2\_font\_unifont\_t\_symbols);

OLED.drawGlyph(112, 55, 0x21E6);

//ENTER

OLED.setDrawColor(1);

OLED.drawRFrame(105, 55, 21, 9, 3);

OLED.setDrawColor(2);

OLED.setFontMode(1);

OLED.setFont(u8g2\_font\_unifont\_t\_symbols);

OLED.drawGlyph(112, 66, 0x21AA);

//typ balk (zwart balkje links)

OLED.setDrawColor(0);

OLED.drawBox(0, 0, 2, 16);

if(errorMaxPasswordLength == 1)

{

OLED.setDrawColor(1);

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(5, 14, "Max. lengte bereikt!");

}

}

else if(h1 == 3)

{

if(connect\_attempt == 1) //IDLE

{

OLED.setDrawColor(1);

// 'icon\_slot', 51x62px

OLED.drawBitmap(1, 1, 56 / 8, 62, bitmap\_slot);

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(56, 28, "Wachtwoord");

OLED.drawStr(56, 44, "Verifieren");

}

else if(status == 3) //CONNECTED

{

OLED.setDrawColor(1);

// 'icon\_vinkje', 51x63px

OLED.drawBitmap(1, -5, 56 / 8, 63, bitmap\_vinkje);

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(56, 28, "Wachtwoord");

OLED.drawStr(56, 44, "Correct");

screen\_delay = 1;

}

else if(status == 6 || status == 1) //DISCONNECTED

{

OLED.setDrawColor(1);

// 'icon\_slot', 51x62px

OLED.drawBitmap(1, 1, 56 / 8, 62, bitmap\_slot);

OLED.setFont(u8g\_font\_profont12);

if(status == 1)

{

OLED.drawStr(56, 28, "Netwerk niet");

OLED.drawStr(56, 44, "beschikbaar");

}

else if(status == 6)

{

OLED.drawStr(56, 28, "Wachtwoord");

OLED.drawStr(56, 44, "foutief");

}

item\_selected = 1;

selected\_network = 0;

scan\_completed = 0;

WiFi.scanDelete();

screen\_delay = 1;

}

}

else if(h1 == 4)

{

OLED.setDrawColor(1);

// 'icon\_vinkje', 51x63px

OLED.drawBitmap(1, -5, 56 / 8, 63, bitmap\_vinkje);

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(56, 28, "Verbinden");

OLED.drawStr(56, 44, ssid.c\_str());

item\_selected = 0;

screen\_delay = 1;

}

}while(OLED.nextPage());

if(connect\_attempt == 1)

{

if(WiFi.status() == 3) status = 3;

else if(WiFi.status() == 1) status = 1;

else if(WiFi.status() == 6) status = 6;

delay(1000);

connect\_attempt = 0;

}

if(screen\_delay == 1 && item\_selected == 1 && h1 == 1 && scan\_completed == 1)

{

delay(1000);

screen\_delay = 0;

}

if((h1 == 3 || h1 == 4) && screen\_delay == 1)

{

if(h1 == 3)

{

if(status == 3) h1 = 4;

else if(status == 6)

{

h1 = 0;

WiFi.disconnect();

}

}

else h1 = 0;

delay(3000);

screen\_delay = 0;

}

if(scan\_completed == 0 && item\_selected == 1 && h1 == 1)

{

WiFi.scanDelete();

networks = WiFi.scanNetworks();

delay(1500);

scan\_completed = 1;

}

if(errorMaxPasswordLength == 1)

{

delay(1500);

errorMaxPasswordLength = 0;

}

}

}

}

//--------------------void stepper--------------------

void stepperCode()

{

//wiezen

if(wiezen == 1)

{

//OLED

OLED.firstPage();

do

{

if(s1 == 1)

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 14, "Wiezen");

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "Personen: ");

OLEDstring = nPersonen;

OLED.drawStr(75, 15 + 20 + 2, OLEDstring.c\_str());

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Aan Het Delen");

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

}while(OLED.nextPage());

//stepper

if(nPersonen == 4)

{

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

if(nDeelBegin <= 2) stepper.runToNewPosition(50 + ((nDeelBegin - 1) \* 100));

else if(nDeelBegin > 2) stepper.runToNewPosition(-50 + ((nDeelBegin - 4) \* 100));

for(int a = 0; a < 2; a++)

{

for(int b = 0; b < nPersonen; b++)

{

for(int c = 0; c < 4; c++)

{

ESC.write(0);

delay(150);

ESC.write(180);

delay(100);

ESC.write(90);

delay(500);

}

ESC.write(90);

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

stepper.move(100);

while(stepper.distanceToGo() != 0)

{

stepper.run();

currentPosition = stepper.currentPosition();

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

stepper.setAcceleration(100);

stepper.setMaxSpeed(200);

while(currentPosition >= 400)

{

currentPosition = currentPosition - 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

a = 2;

b = nPersonen;

}

}

}

}

if(button\_cancel\_clicked == 0)

{

for(int a = 0; a < 3; a++)

{

for(int b = 0; b < 5; b++)

{

ESC.write(0);

delay(150);

ESC.write(180);

delay(100);

ESC.write(90);

delay(500);

}

ESC.write(90);

stepper.move(100);

while(stepper.distanceToGo() != 0)

{

stepper.run();

currentPosition = stepper.currentPosition();

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

stepper.setAcceleration(100);

stepper.setMaxSpeed(200);

while(currentPosition >= 400)

{

currentPosition -= 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

}

}

}

for(int a = 0; a < 5; a++)

{

ESC.write(0);

delay(150);

ESC.write(180);

delay(100);

ESC.write(90);

delay(500);

}

ESC.write(90);

while(currentPosition >= 400)

{

currentPosition -= 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

}

}

else if(nPersonen == 3)

{

for(int a = 0; a < nPersonen; a++)

{

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

if(nDeelBegin <= 2) stepper.runToNewPosition(50 + ((nDeelBegin - 1) \* 100));

else if(nDeelBegin > 2) stepper.runToNewPosition(-50 + ((nDeelBegin - 4) \* 100));

for(int a = 0; a < 2; a++)

{

}//tag

//for(int a = 0; a)

}

}

stepper.setCurrentPosition(0);

nDeelBegin++;

s1 = 0;

}

//zetten

else if(zetten == 1)

{

//OLED

OLED.firstPage();

do

{

if(s1 == 1)

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(43, 14, "Zetten");

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "Personen: ");

OLEDstring = nPersonen;

OLED.drawStr(75, 15 + 20 + 2, OLEDstring.c\_str());

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Aan Het Delen");

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

}while(OLED.nextPage());

if(nPersonen == 4)

{

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

if(nDeelBegin <= 2) stepper.runToNewPosition(50 + ((nDeelBegin - 1) \* 100));

else if(nDeelBegin > 2) stepper.runToNewPosition(-50 + ((nDeelBegin - 4) \* 100));

for(int a = 0; a < 7; a++)

{

for(int b = 0; b < 4; b++)

{

ESC.write(0);

delay(100);

ESC.write(180);

delay(100);

ESC.write(90);

delay(500);

}

ESC.write(90);

stepper.move(100);

while(stepper.distanceToGo() != 0)

{

stepper.run();

currentPosition = stepper.currentPosition();

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

button\_cancel\_clicked = 1;

stepper.setAcceleration(100);

stepper.setMaxSpeed(200);

while(currentPosition >= 400)

{

currentPosition = currentPosition - 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

a = 7;

}

}

}

for(int a = 0; a < 4; a++)

{

ESC.write(0);

delay(100);

ESC.write(180);

delay(100);

ESC.write(90);

delay(500);

}

ESC.write(90);

}

while(currentPosition >= 400)

{

currentPosition -= 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

nDeelBegin++;

s1 = 0;

}

//blackjack

else if(blackJack == 1)

{

}

//gelijk verdelen

else if(gelijkVerdelen == 1)

{

//OLED

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawHLine(10, 17, 108);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(12, 14, "Gelijk verdelen");

// draw scrollbar background

OLED.drawBitmap(128 - 8, 0, 8 / 8, 64, bitmap\_scrollbar\_background);

if(s1 == 1)

{

OLED.setFont(u8g\_font\_7x14);

OLED.drawStr(4, 15 + 20 + 2, "P: ");

OLEDstring = nPersonen;

OLED.drawStr(28, 15 + 20 + 2, OLEDstring.c\_str());

OLED.drawStr(64, 15 + 20 + 2, "K: ");

OLEDstring = nKaarten;

OLED.drawStr(88, 15 + 20 + 2, OLEDstring.c\_str());

//OLED.drawBitmap(4, 46, 16 / 8, 16, bitmap\_icons\_hoofdmenu[item\_sel\_next]);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(4, 15 + 20 + 20 + 2 + 2, "Aan Het Delen");

OLED.drawBitmap(0, 44, 128 / 8, 21, bitmap\_item\_sel\_outline);

}

}while(OLED.nextPage());

//stepper

for(int a = 0; a<nPersonen; a++)

{

stepper.setAcceleration(250);

stepper.setMaxSpeed(500);

stepper.move(400/nPersonen);

while(stepper.distanceToGo() != 0)

{

stepper.run();

currentPosition = stepper.currentPosition();

if(digitalRead(BUTTON\_CANCEL\_PIN) == LOW && button\_cancel\_clicked == 0)

{

stepper.setAcceleration(100);

stepper.setMaxSpeed(200);

while(currentPosition >= 400)

{

currentPosition = currentPosition - 400;

}

if(currentPosition < 200) stepper.move(-currentPosition);

else if(currentPosition >= 200) stepper.move(400-currentPosition);

while(stepper.distanceToGo() != 0) stepper.run();

stepper.setCurrentPosition(0);

a = nPersonen;

}

}

for(int b = 0; b < nKaarten; b++)

{

ESC.write(0);

delay(100);

ESC.write(180);

delay(200);

ESC.write(90);

delay(500);

}

ESC.write(90);

}

stepper.setCurrentPosition(0);

s1 = 0;

}

}

//--------------------void opstartScherm--------------------

void opstartScherm()

{

//LOGO

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(0);

OLED.drawBitmap(12, 13, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr( 10 + 5 + 36, 20 + 15, "Industries");

}while(OLED.nextPage());

//buzzer

tone(buzzer, 500); //tone 500, delay 500

delay(225);

tone(buzzer, 675); //tone 675, delay 250

delay(250);

tone(buzzer, 1000); //tone 975, delay 225

delay(600);

noTone(buzzer);

delay(200);

OLED.firstPage();

do

{

OLED.setColorIndex(0);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(1);

OLED.drawBitmap(12, 13, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr( 10 + 5 + 36, 20 + 15, "Industries");

}while(OLED.nextPage());

delay(300);

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(0);

OLED.drawBitmap(12, 13, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr( 10 + 5 + 36, 20 + 15, "Industries");

}while(OLED.nextPage());

delay(1250);

//card dealer

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(0);

OLED.drawBitmap(12, 13, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(10 + 5 + 36, 20 + 15, "Industries");

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(10 + 6 + 36, 20 + 16 + 12, "Card Dealer");

}while(OLED.nextPage());

delay(1250);

//software versie

OLED.firstPage();

do

{

OLED.setColorIndex(1);

OLED.drawBox(0, 0, 128, 64);

OLED.setColorIndex(0);

OLED.drawBitmap(12, 13, 40 / 8, 36, bitmap\_LOGO);

OLED.setFont(u8g\_font\_7x14B);

OLED.drawStr(10 + 5 + 36, 20 + 15, "Industries");

OLED.setFont(u8g\_font\_profont12);

OLED.drawStr(10 + 6 + 36, 20 + 16 + 12, "Card Dealer");

OLED.setFont(u8g2\_font\_5x7\_tf);

OLED.drawStr(53 ,46 + 10 ,softwareVersie.c\_str());

}while(OLED.nextPage());

delay(3000);

}

**Bijlage 2: Een titel**

**Bijlage 3: Een titel**