FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

VERSLAG PROJECT MICROCONTROLLERS

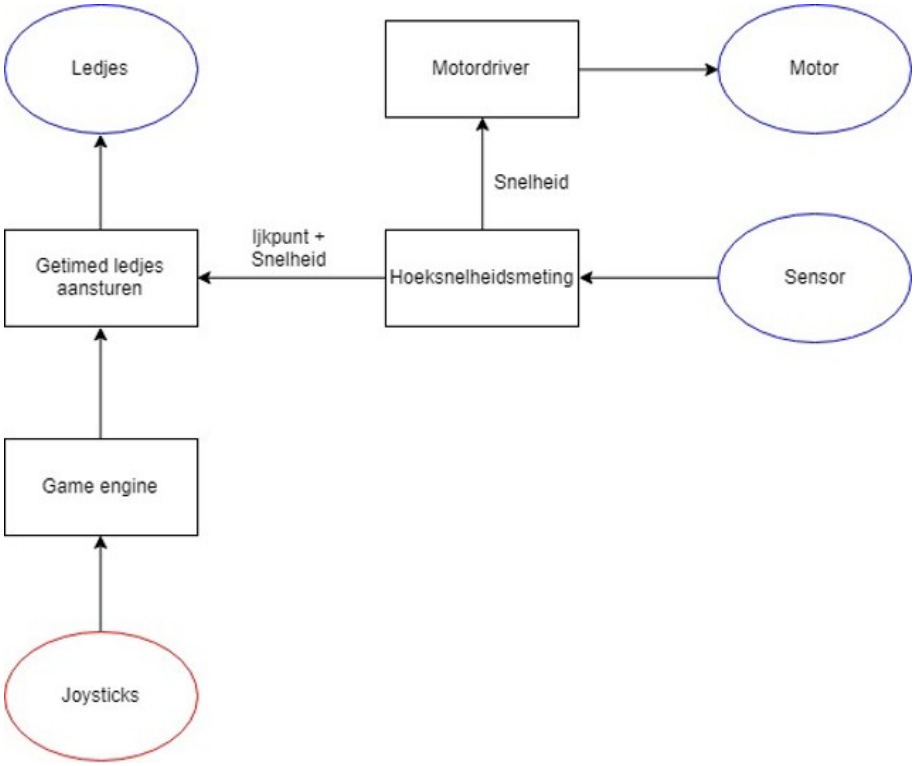
Marieke Louage, Stef Pletinck, Scoutt Verhelle

UNIVERSITEIT GENT
CAMPUS KORTRIJK

Inhoudsopgave

Lijst van figuren

Lijst van onafgewerkte taken



Figuur 1: Structuur van het microcontrollerprogramma

1 Doelstellingen

Doel van deze opgave was het maken van een spel op een microcontroller, specifiek een *Dwenguino*. Verdere doelen van het vak zijn het leren lezen en interpreteren van datasheets en werken met de taal *C*. Er werd besloten om een multiplayer spel te maken, voor meer interactiviteit.

Voor de weergave van het spel werden verschillende mogelijkheden overlopen. Een eerste mogelijkheid was het aansturen van een VGA-display, maar door de hoge klokfrequentie van VGA en de daaraan gelinkte problemen werd dit idee snel verworpen. Een tweede idee was het gebruik van een LED-matrix. De lage resolutie hiervan was echter een groot nadeel, dus werd uiteindelijk gekozen voor een scherm dat gebruik maakt van het principe van een hardeschijfklok. Een snel ronddraaiende ledstrip werd ook overwogen, maar dit is praktisch niet haalbaar.

Het spel zelf vindt plaats in een baan rond de aarde, waar twee ruimteschepen elkaar rond de aarde achtervolgen en proberen neer te schieten.

Als invoer van de spelers werd gekozen voor arcade-joysticks. Deze bestaan uit vier microswitches per joystick. Er waren verder nog vele ideeën om het spel verder uit te breiden.

2 Praktische aanpak

De nodige functionaliteit werd opgedeeld in logische blokken die met elkaar communiceren en op elkaar vertrouwen voor informatie. De bekomen structuur is zichtbaar in figuur 1.

2.1 Werking display

Een schijf met gaten draait snel rond over LED's, verspreid in sectoren, zie figuur 2 Door de LED's op de gepaste momenten in en uit te schakelen kan een zeer hoge resolutie bekomen worden rond het middelpunt. In de radiale richting is de resolutie beperkt door het aantal LED's, dit werd

Figuur 2: Schema van de draaischijf

Figuur 3: Schema van een led-pakket

gekozen op 16 om de aansturing werkbaar te houden.

Met deze aanpak is het mogelijk om willekeurig de resolutie te kiezen waarmee de hoek wordt beschreven. Hoe groter deze resolutie, hoe meer pixels. Deze waarde werd gekozen op 1024.

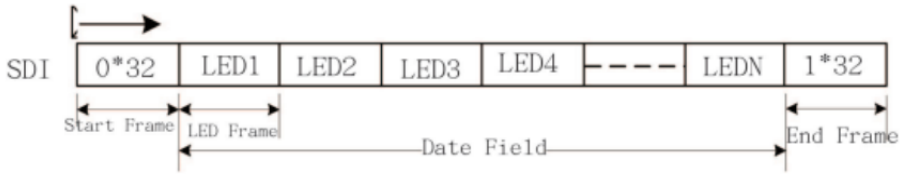
2.2 Aansturing LED's

De gebruikte LED's zijn ledstrips van het type `APA102`. Deze kunnen vrij eenvoudig via SPI data ontvangen. Elke LED krijgt een pakket bestaande uit 32 bits, zie figuur 3. Deze data moet byte per byte en bit per bit verzonden worden over één datalijn, met een kloksignaal.

De ledstrip luistert naar nieuwe commando's na een startpakket bestaande uit 32 0-bits. Vervolgens moet, zoals zichtbaar op figuur 4, een pakket per LED verstuurd worden gevolgd door nogmaals 32 bits om te zorgen dat alle data ver genoeg doorgeschoven is. De waarde van deze bits is onbelangrijk, maar 0 is handig om te voorkomen dat er een LED op volle helderheid wit licht geeft wanneer er niet naar elke LED gegevens worden gestuurd.

Er zijn twee mogelijke manieren om meerdere bytes te versturen over SPI, via een blokkerende **`while`**-lus en via een interrupt. Het is echter ook mogelijk om een interrupt te ontvangen wanneer een byte is verzonden, om vervolgens een volgende byte te verzenden. Op die manier kan CPU-tijd uitgespaard worden, maar dit blijkt zeer lastig foutloos te implementeren. Het spel gebruikt bijgevolg voorlopig nog de *blocking* aanpak.

Het is handig om te weten wat de maximale frequentie is waarmee alle LED's kunnen aangestuurd worden. Hiervoor is het belangrijk te weten dat de microcontroller een werkfrequentie heeft van



Figuur 4: Schema van het aansturen van een ledstrip

16 MHz en er maximaal 1 bit per 2 klokcycli kan verstuurd worden over SPI. Er zijn 16 LED's, dit geeft een maximale frequentie van 15 625 Hz.

2.3 Timing LED's

De juiste LED's moeten op correcte momenten aan en uit geschakeld worden om op de juiste plaats op de cirkel pixels op te laten lichten, en niet te lang zodat de pixels geen lijnen worden.

Daartoe wordt voor elk weer te geven object in een snelle lus gecontroleerd of het weer kan gegeven worden. Op de vergelijking van de hoek zit er een lichte marge, zodat elk element zeker wordt weergegeven ook al wordt het scherm niet snel genoeg opnieuw getekend.

Daartoe krijgt de functie zowel de tijd voor één volledige rotatie, als de tijd sinds de schijf het nulpunt is gepasseerd als argumenten. Daaruit kan namelijk de hoek waaronder de schijf staat bepaald worden.

2.4 Fysieke constructie

TODO De gehele constructie werd ontworpen in het CAD-pakket Siemens NX 11 en uitgesneden op een lasercutter, met uitzondering van enkele kleine stukken die ge-3D-print[1] zijn.

Fysieke constructie

De draaischijf bestaat uit dunne zwarte ABS, de behuizing uit MDF. De reflectieve achtergrond en tussenschotten achter en tussen de LED's zijn uit een vel wit plastic gesneden.

2.5 Aansturing motor

TODO De schijf draait rond met behulp van een brushless DC-motor uit een harde schijf. Om deze aan te sturen is een speciale ESC-module[2] nodig.

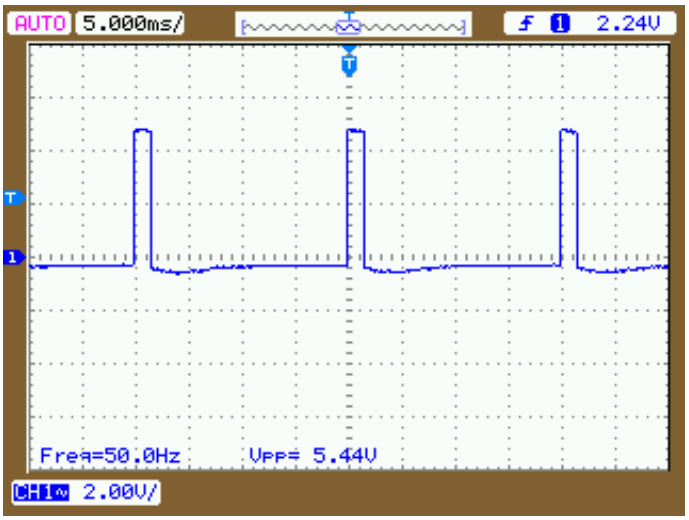
Keuze timers en verder aanvullen

Deze modules zijn ontworpen voor gebruik in quadcopters en verwachten bijgevolg een speciale aansturing. Het nodige signaal is een vorm van PWM, met een frequentie van 50 of 60 Hz en een pulsbreedte tussen 1 en 2 ms. De PWM-modules die ingebouwd zitten in de microcontroller kunnen niet opereren in deze frequenties en pulsbreedtelimieten, het protocol werd dus volledig in software geïmplementeerd. Zie figuur 5 voor een beeld het geproduceerde signaal.

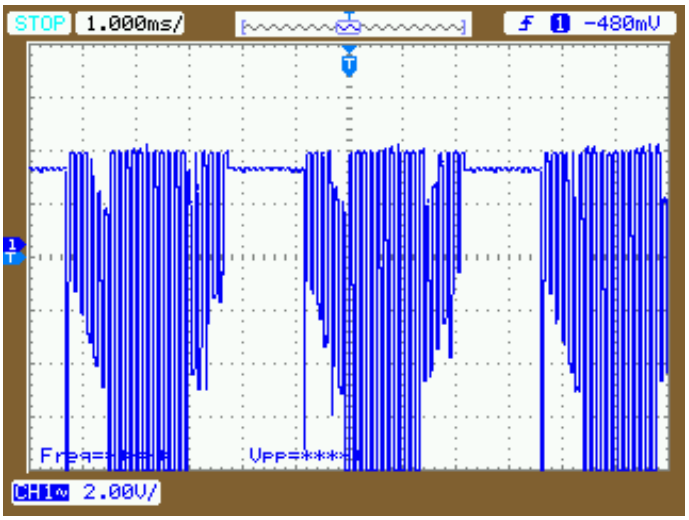
Een extra moeilijkheid is de opstartprocedure van de ESC. De microcontroller moet opgestart zijn en een signaal sturen dat overeen komt met 0 % motorvermogen wanneer de ESC stroom krijgt en wordt ingeschakeld. De ESC produceert dan een serie tonen, gevolgd door een langere toon wanneer het signaal herkend wordt. Daarna kan de motor aangestuurd worden en het vermogen verhoogd, liefst langzaam gezien het hoge gewicht van de draaischijf. Uiteindelijk zal de ESC een driefasig signaal naar de motor sturen, zoals zichtbaar in figuur 6.

Deze procedure maakt het moeilijk om microcontroller en ESC op één voeding te laten werken. Door de grote stroom is het ook lastig om bijvoorbeeld de ESC te schakelen met een transistor. Voorlopig moet de ESC manueel in de stekker gestoken worden. De driver wacht daartoe even met het opstarten van de motor, aangezien de stekker pas mag ingeplugd worden wanneer het stuursignaal reeds aanwezig is.

[1] Dit is de correcte spelling volgens het Groene Boekje.
[2] ESC: Electronic Speed Control



Figuur 5: Controlesignaal ESC



Figuur 6: Vermogenssignaal naar de motor

2.6 Toerenteller schijf

Op de schijf zit een optische sensor, die registreert wanneer een gaatje passeert. De microcontroller telt nauwkeurig op welk moment dit gebeurt, zodat afgeleid kan worden hoe snel de schijf draait.
TODO

Info toerenteller

2.7 Joysticks

De beide joysticks zijn relatief eenvoudig om uit te lezen. Ze bestaan telkens uit vier NO schakelaars, één per richting. Aangezien elke joystick op een aparte *port* van de microcontroller aangesloten is, kan met een simpele NOT operatie en enkele bitshifts een consistente weergave gegenereerd worden, met 1 bit per richting die hoog is wanneer de schakelaar actief is. Hiertoe moeten ook de pullups geactiveerd worden. De driver voor de joysticks controleert ook welke schakelaars sinds de laatste tick actief zijn geworden, en slaat deze apart op.

Enkele gemaksfuncties werden toegevoegd om uit deze bitweergave te bepalen of een gekozen richting actief is.

2.8 Game engine

De game engine werkt op een relatief eenvoudig principe. Op regelmatige basis wordt de functie `uint8_t tick(uint16_t time_since_zero)` opgeroepen, deze voert enkele stappen uit:

1. *Tick alle entities*[3], entities hebben een `tick()` functie,
   bijvoorbeeld `player_tick(Player *p)`, die ervoor zorgt dat de entity zich verplaatst volgens zijn snelheid en eventueel reageert op invoer.
2. *Controleer voor botsingen*, in deze stap wordt getest of er kogels een schip geraakt hebben, en eventueel levens van deze speler worden afgenomen.
3. *Check voor het einde van het spel* door te controleren of beide spelers nog in leven zijn. Dit bepaalt de returnwaarde van deze functie, `true` wanneer het spel moet verder gaan, anders `false`.

Deze functie wordt aan 30 Hz opgeroepen, met behulp van `Timer/Counter0` in *CTC* mode. De 8-bits resolutie is meer dan genoeg om een vrij groot bereik van mogelijke frequenties te bereiken met de prescaler op de maximale waarde van 1024. Apart daarvan wordt zo snel mogelijk de `render(uint8_t time_since_zero)` functie opgeroepen, die de LED's schakelt.

3 Problemen

3.1 ESC

Na een eerste poging bleek dat de ESC de motor niet kon laten draaien. De motor gaf wat schokjes en de ESC produceerde niet de gewenste opstarttonen. Na de massakabel van de motor los te maken bleek alles correct te functioneren. Het is onbekend waarom dit een probleem gaf.

3.2 Light bleed

Soms lichtten er pixels vaag op die niet moesten oplichten, dit bleek te zijn doordat er soms twee gaten over dezelfde sector hingen, wat mogelijk was omdat de scheiding tussen sectoren bestaat uit dun, reflectief plastic.

Er werd besloten om de verticale resolutie te verlagen, zodat de sectoren op de draaischijf groter worden, maar de sectoren met de LED's ongewijzigd te houden. Een andere optie zou zijn om de wanden tussen de sectoren dikker te maken.

[3] Entities zijn bijvoorbeeld kogels en spelers

4 Openstaande Problemen

Er is nog wat werk aan de fysieke constructie van de schijf en we twijfelen nog over het aantal sectoren. Een groot deel van de code, zoals de *game engine* en de driver voor LED-timing, is dus nog niet getest. Dit testen en debuggen zal waarschijnlijk nog vrij veel tijd innemen. Verder zijn er nog enkele verbeteringen te maken in de *game engine*, zo wordt er nog geen melding gegeven wanneer een speler sterft en kan het spel niet herstart worden zonder de microcontroller te resetten. Het spel begint ook van zodra de controller opstart, in plaats van wanneer de spelers klaar zijn. Dit is geen wenselijke werking.

5 Taakverdeling en Samenwerking

De motordriver werd geschreven door Marieke. Zij schreef ook de code die voor de juiste timing zorgt en de optische sensor uitleest.

De game engine en aansturing van LED's werd gedaan door Stef, net zoals uitlezen van Joysticks en debugfunctionaliteit.

Ontwerp en constructie van de behuizing is door Marieke.

Scoutt was verantwoordelijk voor een deel van het bestand `Pins`, dat bijhoudt welke pins op de microcontroller in gebruik zijn door welk systeem. Hij is ook verantwoordelijk voor commit `142748a:` *Nuttelose comment* (sic.).

Onze samenwerking gebeurt via Git en GitHub[4]. Taken worden verdeeld via *issues* op GitHub. Iedereen werkt aan een apart deel van de code of de hardware, die regelmatig samen worden gezet.

6 Conclusie en Toekomstig Werk

[4] `https://github.com/Epse/MCU_Project_PlanetFight`