VRIJE
UNIVERSITEIT
BRUSSEL

Bachelorproef
Bachelor of Science in de Industriële Wetenschappen:
Elektromechanica

AIRHOCKEY ROBOT

Niels Vannerom & Amirgan Bouakhounov

2021-2022

promotoren: Jonas Verbeke & Lieven Standaert
begeleiders: Jef Verbaenen, Jeroen Schelkens, Jonathan Sterckx & Michiel Vrolix

INGENIEURSWETENSCHAPPEN

woord vooraf

Wij zijn Amirgan Bouakhounov en Niels Vannerom, studenten uit derde bachelor Industriële Wetenschappen aan de Vrije Universiteit Brussel. Voor het behalen van ons bachelordiploma dienen wij een bachelorproef te maken. Dit is een eindwerk waarin we al onze kennis en vaardigheden die we de voorbije jaren geleerd hebben in één geheel gaan verwerken.

Dit deden we aan de hand van een project onze voorkeur te geven aan het begin van het academiejaar. Hierbij hebben wij de voorkeur gegeven aan het project: Air hockey robot. Dit houdt in dat we een robot moesten ontwerpen en bouwen dat in staat is om zelfstandig een Air hockey wedstrijd spelende te houden.

Graag willen we onze begeleiders J. Verbaenen, J. Schelkens, J. Sterckx en M. Vrolix bedanken voor al hun hulp en ondersteuning gedurende het gehele project. Ook onze promotoren J. Verbeke en L. Standaert willen we bedanken voor hun mening wanneer we voor een probleem stonden.

Inhoudsopgave

Lijst van figuren

Acroniemen

fps	frames per seconde
rpm	rotaties per minuut

Griekse symbolen

π	pi	$3,14$
ρ	massadichtheid	$\mathrm{kg/m^3}$

Romeinse symbolen

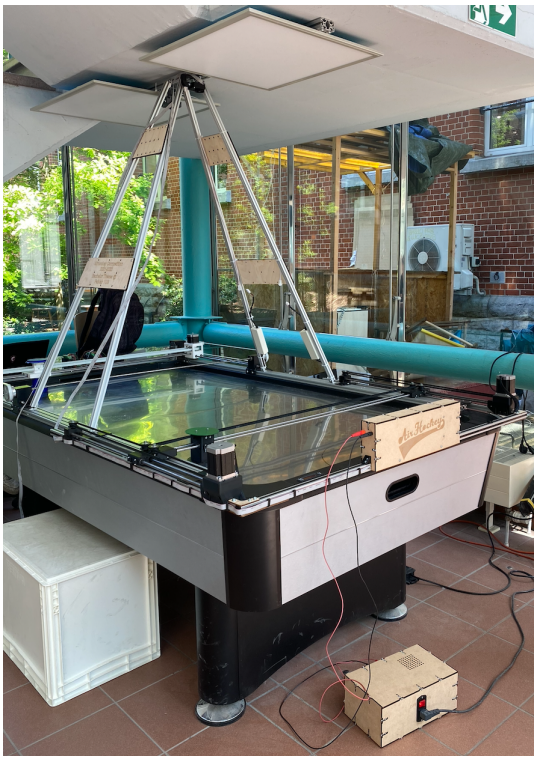
a	versnelling	$\mathrm{m/s^2}$
F	kracht	N
L	lengte	m
l	lengte	m
m	massa	kg
r	straal	m
s	afstand	m
T	koppel	Nm
t	tijd	s
V	volume	$\mathrm{m^3}$
v	snelheid	m/s

Subscripts

0	begin positie
acc	versnellen
decc	vertragen
defstr	defensieve striker
max	maximaal
p	puck
play	speelbaar
rest	resterend
tot	totaal

In dit verslag van onze bachelorproef zullen we het hebben over onze eigen versie van een autonome airhockey robot dat we volledig zelf ontworpen en geprogrammeerd hebben. De striker is het onderdeel van deze machine die we zo nauwkeurig mogelijk zullen positioneren om de puck af te blokken. Deze striker wordt volledig bestuurd door een x-y mechanisme, aangedreven door stappenmotoren. Deze striker kan zich dan volledig autonoom positioneren in functie van de puck.

Dit verslag gaan we beginnen met kort de probleemstelling te schetsen, wat moet onze robot juist kunnen doen? Hierna komen de twee grootste hoofdstukken, deze gaan zowel over de elektronica als de mechanica die we gebruikt hebben in dit project. Hier zullen we de grootste beslissingen toelichten, waarom we deze gekozen hebben en eventueel staven we deze beslissing met een berekening. Uiteraard is dit project geen realiteit geworden door enkel theoretisch na te denken, de praktische realisatie komt hierna aan bod. Hier zullen we onze evolutie doorheen het jaar kort toelichten. Tot slot resten er zich nog twee hoofdstukken over een reflectie voor ons zelf. Wat hebben we geleerd uit dit project en stel dat we dit project opnieuw zouden mogen doen, wat zouden we dan anders aanpakken?



Figuur 1: Air hockey robot groep 1

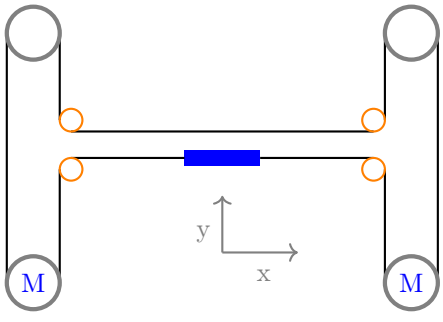
probleemstelling

Het doel van deze bachelorproef is om een robot te bouwen die in staat is om zelfstandig air-hockey te spelen. Deze robot moet bevestigd kunnen worden op een commerciële air-hockey tafel. Volledig commercieel is deze air-hockey tafel niet meer, aangezien er aluminium extrusie profielen op de bovenkant van de tafel gemonteerd staan. Met enkel deze extrusie profielen moesten we voldoende hebben om onze robot te kunnen bevestigen.

Deze robot is in staat om volledig zelfstandig een air-hockey wedstrijd te kunnen spelen. Ofwel tegen een andere air-hockey robot ofwel tegen een mens. Eén van de vereisten hiervoor is dat onze robot in staat is om zijn eigen doel te kunnen verdedigen om te voorkomen dat de puck in zijn eigen doel terecht komt. Verder moet de puck met voldoende kracht terug naar de overkant gespeeld kunnen worden.

De vereisten die werden meegegeven is dat we in staat zijn om de puck te kunnen tracken, de positie en richting van de puck afleiden en tot slot moeten we het pad van de puck kunnen voorspellen. Om dit waar te kunnen maken, maken we gebruik van een micro controller van het type ESP-32 en een camera van het type Pixy2.1. De communicatie tussen deze micro controller en camera zal over SPI verlopen.

Een volgende vereiste is dat we in staat moeten zijn om onze striker met voldoende nauwkeurigheid te kunnen positioneren, tot slot moeten we de striker zowel voorwaarts als achterwaarts kunnen bewegen om voldoende kracht op de puck uit te kunnen oefenen. Om dit mogelijk te kunnen maken zullen we een x- en y-systeem bouwen op het principe in fig. 1.1. We maken gebruik van twee statische motoren, in ons geval NEMA 23 stappen motoren. Door deze in dezelfde richting te laten draaien zal de striker zijwaarts kunnen bewegen. Door de twee motoren in tegengestelde richting te draaien zal de striker voorwaarts of achterwaarts bewegen. Door met deze verhoudingen te spelen kunnen we ook diagonaal bewegen door slechts twee motoren te gebruiken. Uiteraard worden deze twee motoren en striker verbonden met een open riem.



Figuur 1.1: h-bot mechanisme

Wat het frame zelf betreft zullen we gebruik maken van 20 X 20 aluminium extrusie profielen. Dit omdat we zo een eenvoudige en stevige constructie kunnen bouwen.

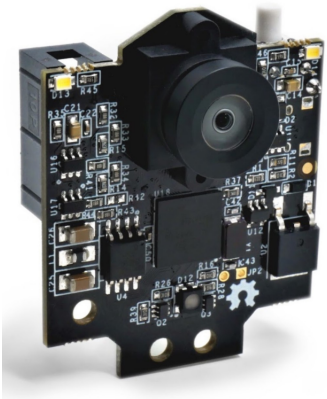
In dit hoofdstuk zullen we bespreken welke elektronica we allemaal gebruikt hebben voor dit project. Op vlak van elektronica kunnen we toch wel stellen dat de camera niet kan ontbreken. Dit komt vooral omdat we onze puck willen kunnen detecteren. Wij kiezen dus voor een camera om dit te doen. Vervolgens moeten we de gegevens van deze camera kunnen verwerken. Om dit waar te maken , gebruiken we een microcontroller. Verder in dit hoofdstuk zullen we bespreken welke microcontroller we juist gebruiken.

Deze verwerkte data moeten we natuurlijk kunnen gebruiken om motoren aan te sturen, door het feit dat we kiezen voor stappenmotoren hebben we dus natuurlijk ook stepper drivers nodig. Al deze componenten zullen we dan samenbrengen op een printed circuit boad.

2.1 Camera

De camera die we gebruiken voor het detecteren van zowel de striker als puck is een Pixy 2.1 camera. Dit is een camera die gemaakt is om kleuren te kunnen herkennen en onderscheiden van andere kleuren. Dit wil zeggen dat de camera zelf het grootste deel van de beeldverwerking voor zich neemt en enkel de positiedata doorstuurt naar een microcontroller. Deze microcontroller is dan weer voldoende krachtig om met deze data verder te kunnen rekenen.

De positiedata die de microcontroller ontvangt van de camera is helemaal niet zo zwaar meer om mee te werken. Dit komt door het feit dat de Pixy 2.1 camera een eigen microprocessor aan boord heeft. Deze microcontroller verwerkt al de data die binnenkomt waarbij de camera de coördinaten, uitgedrukt in een aantal pixels, al heeft omgevormd. Deze coördinaten worden dan ontvangen door de uiteindelijke microcontroller.



Figuur 2.1: Pixy 2.1

2.1.1 Frames per second

Uit de specificaties van deze camera blijkt dat deze 60 frames per seconde kan waarnemen.

Op basis van een ruwe schatting veronderstellen we dat de maximale snelheid dat onze puck behaalt 70 $\frac{km}{h}$ of 19,44 $\frac{m}{s}$is. Wat de lengte van de tafel betreft hebben we een lengte van 203,3 cm.

Met deze gegevens kunnen we berekenen hoe lang het duurt voor dat de puck de overkant bereikt heeft :

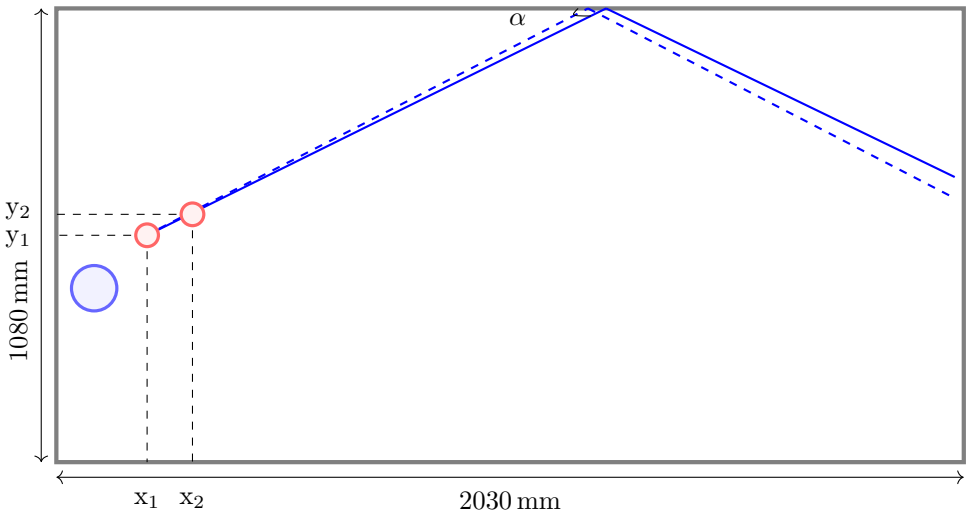
$t = \frac{l}{v_{\max}} = \frac{2{,}033\,\mathrm{m}}{19{,}44\,\frac{\mathrm{m}}{\mathrm{s}}} = 0,105\,\mathrm{s}$

Nu rest ons enkel nog te berekenen hoeveel frames we in dit geval zouden binnen krijgen :
$\#\ \text{frames} = 0,105\,\mathrm{s} \cdot 60\,\mathrm{fps} = 6,27\ \text{frames} \approx 6\ \text{frames}.$

Deze hoeveelheid aan frames is niet ideaal maar wel haalbaar, rekening houdend dat men meestal meer frames zal hebben aangezien we hier met de maximale snelheid van 70 $\frac{\mathrm{km}}{\mathrm{h}}$ gerekend hebben.

2.1.2 Resolutie

Een volgende beperking van de camera waar we rekening mee moeten houden, Is de resolutie van onze camera. Om een idee te krijgen van welke 'fouten' deze ons mee gaat geven wanneer we het traject willen voorspellen, het kan door de resolutie van onze camera heel wat centimeters verschillen tussen de werkelijke locatie waar de puck terecht zal komen en de voorspelde locatie. Om dit te staven hebben we volgende berekening gemaakt:



Figuur 2.2: schema air-hockey tafel

2.1.2.1 Ideaal

Indien de camera een voldoende hoge resolutie heeft zodat da fouten zo klein zijn dat ze geen impact hebben veronderstellen we de volgende gegevens, deze zijn willekeurig gekozen:
De coördinaten van punt één : (200; 540)
De coördinaten van punt twee : (300; 590)

Met deze gegevens kunnen we eenvoudig de vergelijking opstellen van de rechte door deze twee punten:
$\implies \mathrm{y} - \mathrm{y}_1 = \frac{y_2 - y_1}{x_2 - x_1} \cdot (\mathrm{x} - \mathrm{x}_1) \iff \mathrm{y} - 540 = \frac{590 - 540}{300 - 200} \cdot (\mathrm{x} - 200)$
$\iff \mathrm{y} = 0,5\mathrm{x} + 440$

Indien we een puck hebben met een diameter van 5 cm, raakt de puck de bovenkant op y = 1055. Dit komt overeen met een x-waarde van 1230 mm.

Met deze gegevens kunnen we de hoek α berekenen :
$\alpha = \arctan(\frac{1055-440}{1230}) = 26,57°$

De richtingscoëfficiënt is 0,5; dus na de weerkaatsing bekomen we een richtingscoëfficiënt van -0,5.

De vergelijking van deze rechte is:
$y - 1055 = -0,5x - 1230$
$y = -0,5x + 1670$

Deze rechte raakt de onderkant wanneer y = 25 mm. Dit is op x = 3290 mm, wat al voorbij onze striker ligt. Onze striker ligt op x = 2010mm, wat betekent dat we met onze striker op een hoogte y = 680 mm gaan moeten staan om de puck tegen te houden. Het traject van dit geval kun je in fig. 2.2 zien in de volle blauwe lijn.

2.1.2.2 Realiteit

Omdat we werken met een Pixy2.1 zijn we beperkt met een resolutie van 1295 X 976. Indien we de camera perfect kunnen positioneren komt dit overeen met 1295 pixels voor de lengte (2030 mm). Wat er op neer komt dat we met 1 pixel een aftsand van 1,57 mm overbruggen, wat dus betekend dat we een fout van 1,57 mm kunnen hebben.

Vervolgens veronderstellen we de volgende uiterste gegevens:

De coördinaten van punt één : (198,43; 538,43)
De coördinaten van punt twee : (301,57; 591;57)

We krijgen volgende vergelijking:
$\implies y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1)$
$\iff y - 538,43 = \frac{591,57 - 538,43}{301,57 - 198,43} \cdot (x - 198,43)$
$\iff y = 0,52x + 436,19$

In dit geval zal de puck de bovenkant raken bij een x-waarde van 1190,02 mm.

Met deze gegevens kunnen we de hoek α berekenen:
$\alpha = \arctan(\frac{1055-436,19}{1190,02}) = 27,47°$

We krijgen een rico van -0,52, wat volgende vergelijking geeft:
$y - 1055 = -0,52 \cdot (x - 1190,02)$
$y = -0,52x + 1673,81$

Deze rechte raakt de onderkant wanneer y = 25 mm, dit is op x = 3170,79 mm, wat al voorbij onze striker ligt. De striker ligt op x = 2010 mm, wat betekent dat we onze striker op een hoogte van y = 628,61 mm moeten positioneren. Het traject van dit geval kun je in fig. 2.2 zien in de blauwe stippellijn.

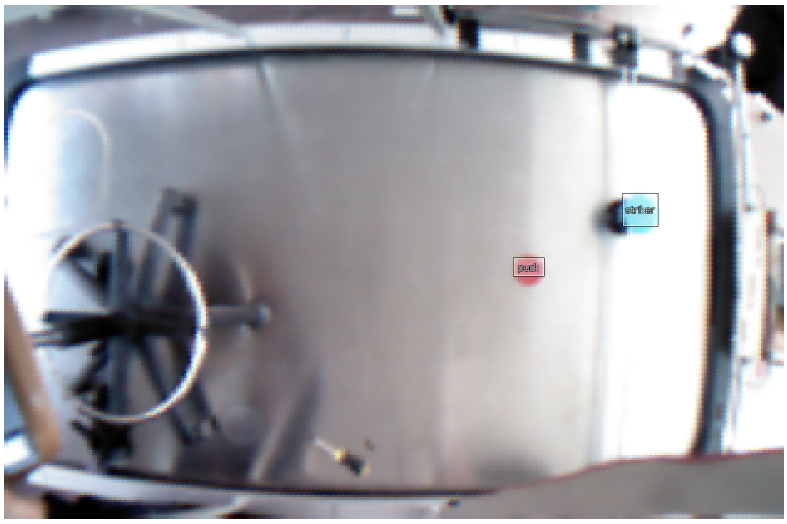
Het verschil tussen de ideale situatie en de situatie gerekend met de resolutie van de camera verschilt toch 5,1 cm in de y-richting van de striker. Dit is dus een zeer belangrijk aspect om rekening mee te houden tijdens de voorspelling van het traject. In dit specifieke geval hebben we het dus berekend indien de puck niet zo snel beweegt, hier hebben we dus meer tijd om gedurende het traject te corrigeren. Indien de puck wel snel zou bewegen zouden onze twee punten verder uit elkaar liggen en hebben we een kleinere fout op de hoek maar veel minder tijd om nog te kunnen corrigeren.

2.1.3 TRACKING

Voor het kunnen tracken van de objecten moeten we vooraf via een meegeleverd programma genaamd Pixymon de te tracken kleuren instellen. Hier stelt men simpelweg de RGB kleuren in, dus met andere woorden welke exacte kleur dat het te tracken object heeft. Maar het algoritme gaat veel verder dan enkel de kleur volgen.

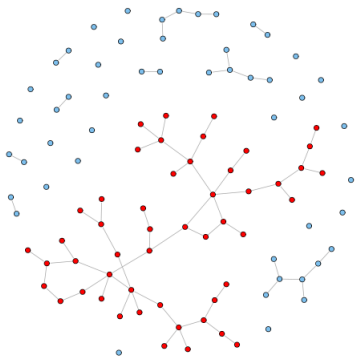
De Pixy 2.1 maakt gebruik van het zogenaamde Color Connected Components algoritme. Dit algoritme houdt rekening met de verzadiging van iedere pixel wat maakt dat een verschil in belichting relatief weinig verschil maakt voor de tracking van een object. In fig. 2.3 kan men zien dat de belichting links en rechts niet gelijk zijn aan elkaar, toch kan de pixy 2.1 de objecten hier over de hele tafel tracken.

Men mag echter niet te optimistisch zijn voor dit algoritme. Wanneer de belichting te extreem wordt, vervagen de kleuren te hard en verloopt de tracking ook niet meer optimaal.



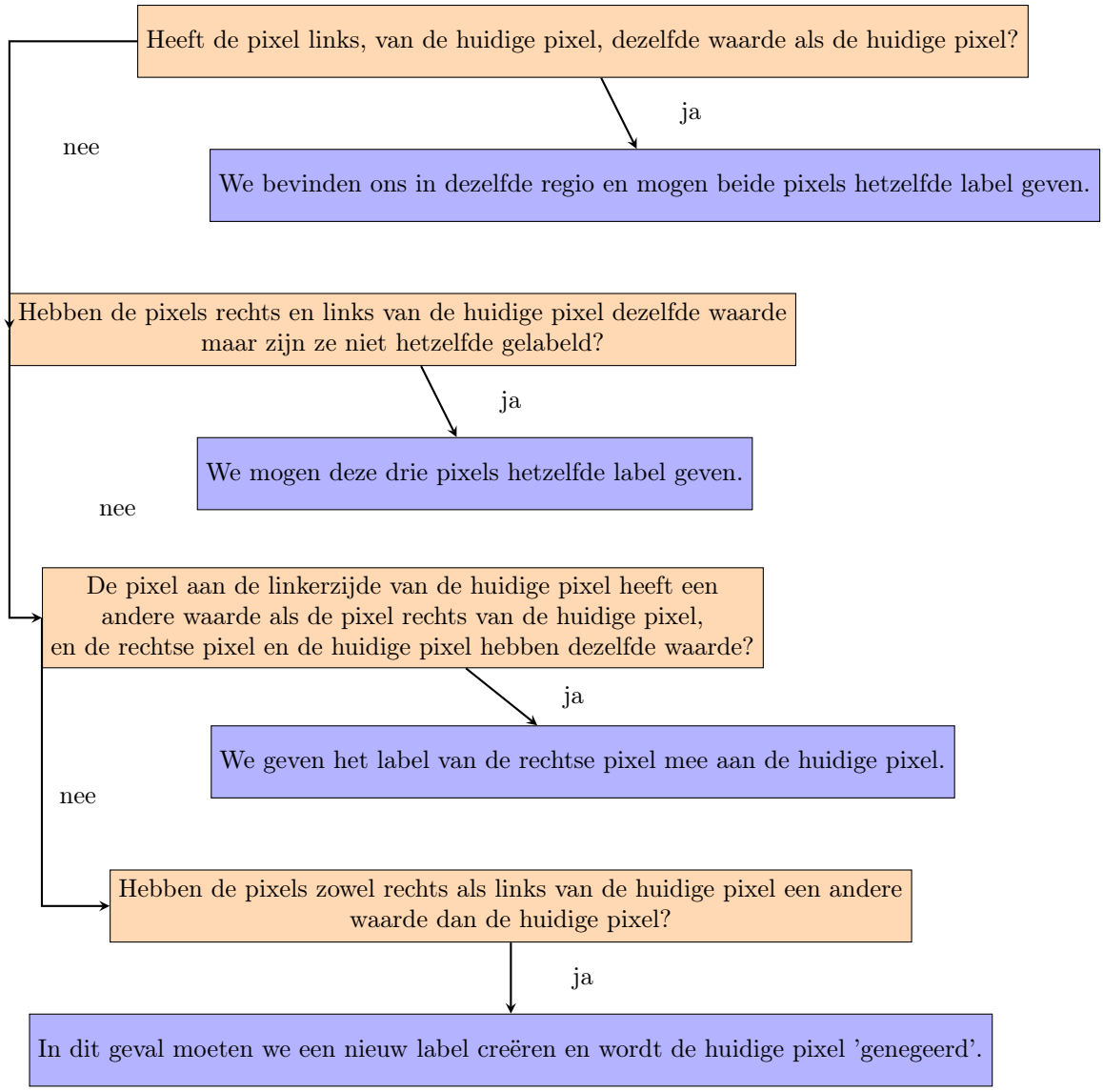
Figuur 2.3: Tracking striker en puck

Het algoritme rust op het principe waar men bij iedere pixel een label gaat meeleveren. Hoe dan alles juist gelabeld is, ziet men in fig. 2.4. Zo is het rode netwerk hier bijvoorbeeld de puck, hoe dit juist bepaald wordt, kan men zien in fig. 2.5.



Figuur 2.4: gelabeld netwerk

Het Color Connected Components (CCC) algoritme werkt als volgt:

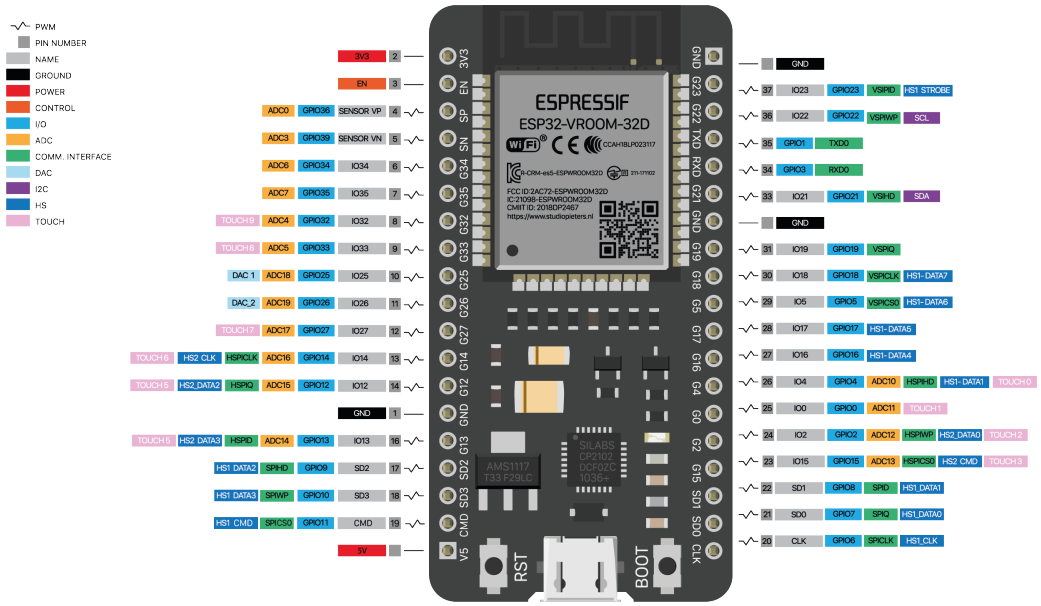


Figuur 2.5: flow diagram Color Connected Components algoritme

Op deze manier probeert de pixy 2.1 camera dus zowel onder als overbelichting te onderdrukken zodat de tracking hier zo min mogelijk invloed van ondervindt.

2.2 Micro controller

Als micro controller kiezen we voor een ESP32-WROOM-32D, de opvolger van de ESP8266. De hoofdreden achter deze keuze ligt hem in de snelheid van deze controller, deze ESP 32 is ongeveer 100 maal sneller dan de klassieke Arduino UNO. Het feit dat we twijfelden tussen een Arduino UNO en een ESP 32 en dat deze ESP zoveel sneller is heeft de doorslag gegeven. Doordat deze zoveel sneller is ligt voor onze toepassingen enkel maar in ons voordeel. Een bijkomende parameter om voor deze micro controller te kiezen, is de relatief goedkope kostprijs. Het enige nadeel waar we rekening mee moeten houden, is dat de pinnen enkel overweg kunnen met een spanning van 3,3V.



Figuur 2.6: PinOut ESP32 StudioPieters, 2022

zoals we verwachten heeft niet iedere pin op deze microcontroller dezelfde functionaliteiten heeft. Daarom is het belangrijk om een overzichtelijk beeld te krijgen van welke pin men waarvoor kan gebruiken. Deze functionaliteiten staan opgelijst in tabel 2.1. Indien een pin in het groen weergegeven wordt, kan men deze zonder problemen gebruiken. Een pin die in het geel aangeduid staat, kan men gebruiken maar tijdens het opstarten kunnen deze pinnen een vreemd gedrag vertonen. Een pin dat in het rood aangeduid staat, kan men niet gebruiken.

Tabel 2.1: table: input en output pinnen StudioPieters, 2022

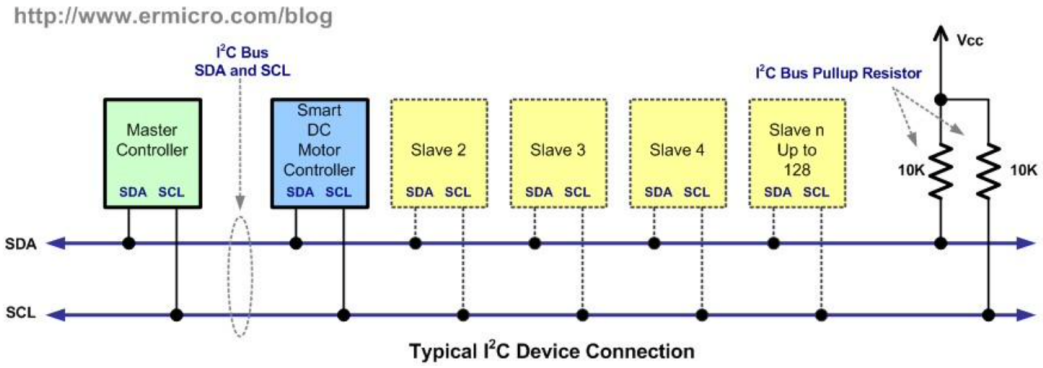
GPIO	Input	Output	Notities
0	pulled up	OK	uitgangen PWM signalen bij opstarten
1	TX Pin	OK	debug uitgang bij opstarten
2	OK	OK	aangesloten op on-board-LED
3	OK	RX PIN	HIGH bij opstarten
4	OK	OK	
5	OK	OK	uitgangen PWM signalen bij opstarten
6	X	X	aangesloten op geïntegreerde SPI flash
7	X	X	aangesloten op geïntegreerde SPI flash
8	X	X	aangesloten op geïntegreerde SPI flash
9	X	X	aangesloten op geïntegreerde SPI flash
10	X	X	aangesloten op geïntegreerde SPI flash
11	X	X	aangesloten op geïntegreerde SPI flash
12	OK	OK	
13	OK	OK	uitgagen PWM signalen bij opstarten
14	OK	OK	uitgangen PWM signalen bij opstarten
15	OK	OK	
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
20	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
24	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
28	OK	OK	
29	OK	OK	
30	OK	OK	
31	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		
35	OK		
36	OK		enkel input
37	OK		enkel input
38	OK		enkel input
39	OK		enkel input

2.2.1 Interface

De ESP32 kan gebruik maken van verschillende interfaces zoals SPI en I^{2}C,... Enkele worden hieronder verder besproken.

2.2.1.1 I2C

I^{2}C is een seriële synchrone datacommunicatie tussen twee apparaten of met andere woorden tussen de master en de slave.

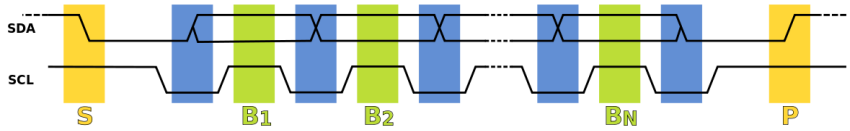


Figuur 2.7: I^{2}C schema Van Ham, 2021

In fig. 2.7 kun je het aansluitschema zien van een I^{2}C bus. Men heeft hiervoor een SDA lijn (Serial Data Line) en een SCL lijn (Serial Clock Line) nodig. Op onze ESP32 komt dit overeen met volgende pinnen:

- SDA = GPIO 21
- SCL = GPIO 22

werking: De dataoverdracht gebeurt over de serial data line. Het kloksignaal, wanneer wat gebeurt, gaat over de SCL lijn.



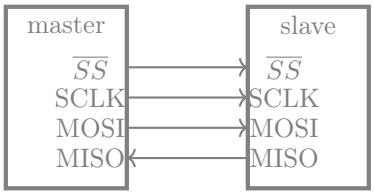
Figuur 2.8: I^{2}C tijdsdiagram

1. We starten met een STARTbit, dit gebeurt wanneer men de SDA laag houdt, ondertussen blijft de SCL pin hoog.
2. Het veranderen van SDA van laag naar hoog of omgekeerd gebeurt wanneer men de SCL lijn laag houdt. Wanneer men de SCL lijn hoog trekt, gaat de data, dat op dat moment op de SDA lijn staat, doorgegeven worden.
3. Wanneer men de hele reeks van data doorgestuurd heeft, plaatst men een STOPbit. Dit doet men door zowel de SDA als SCL lijn hoog te houden.

Doordat deze communicatie dus serieel en over eenzelfde lijn gebeurd, is deze communicatie dus ook vrij traag. Hoe meer data er wordt doorgestuurd, hoe meer vertraging men krijgt. Wat voor onze toepassing maakt dat we een redelijke vertraging hebben op de binnenkomende data van de camera. Daarom is I^{2}C voor onze toepassing onbruikbaar.

2.2.1.2 SPI

Serial Peripheral Interface (SPI) is net zoals I^2C een seriële synchrone datacommunicatie tussen ten minste twee apparaten.



Figuur 2.9: SPI schema

- $\overline{SS}$: slave select: deze lijn wordt laag aangestuurd wanneer men de geselecteerde slave wil aansturen.
- SCLK: seriele klok: het kloksignaal dat wordt meegeleverd door de master.
- MOSI: master output slave input: via deze lijn wordt data van de master naar de slave gestuurd.
- MISO: master input slave output: via deze lijn wordt data van de slave naar de master gestuurd.

In fig. 2.9 kan men het aansluitschema zien voor een SPI interface. Men ziet dat men voor iedere slave een aparte slave select verbinding nodig heeft. Dit heeft zowel voor- als nadelen. De communicatie verloopt redelijk snel aangezien er geen adres moet gelezen worden en er meer data op eenzelfde moment doorgestuurd kan worden. Langs de andere kant kunnen we snel heel veel draden hebben als we meerdere slaves willen gebruiken.

De communicatie gebeurt volgens volgend principe:
Eerst wordt er door de master een geschikte klok frequentie meegegeven die zowel voor de slave als de master past. Om vervolgens te kunnen selecteren van welke slave we data willen ontvangen, sturen we een logische 0 voor zijn slave select verbinding. Vervolgens wordt er over MOSI een boodschap gestuurd naar de slave, deze slave antwoordt dan over MISO op deze boodschap.

Voordelen:

- snel
- "maar" één extra pin per toegevoegde slave
- geen adressering nodig
- kan op hoge kloksnelheden werken

Nadelen:

- meer draden nodig dan bij I^2C
- gevoeliger voor ruis door bit verschuiving ten gevolge van een vervormd kloksignaal
- maximaal 1 master

Als we de voor- en de nadelen afwegen van SPI en I^2C, kiezen we SPI als interface voor de communicatie tussen de camera en de ESP32. Deze werkt sneller maar is gevoeliger voor ruis. Aangezien onze kabel niet heel erg lang is, leggen we zijn gevoeligheid voor ruis wat aan de kant.

2.2.2 Gebruiken van dual-core ESP32

De ESP32 kan zowel in een single-core als een dual-core uitvoering gebruikt worden. Wij maken voor onze toepassing gebruik van de dual-core uitvoering.

De voornaamste reden voor de keuze voor de dual-core uitvoering ligt hem in de programmatie van de aansturing van de stappen motoren en het uitlezen van de Pixy2.1 camera.

Voor het aansturen van de stappenmotoren wordt gebruik gemaakt van een delay. Dit zorgt ervoor dat het programma gedurende de periode, vermeld door deze delay, "stilgelegd"wordt. Dit zorgt er voor dat we tijdens het aansturen van de motoren geen data van onze camera meer kunnen uitlezen.

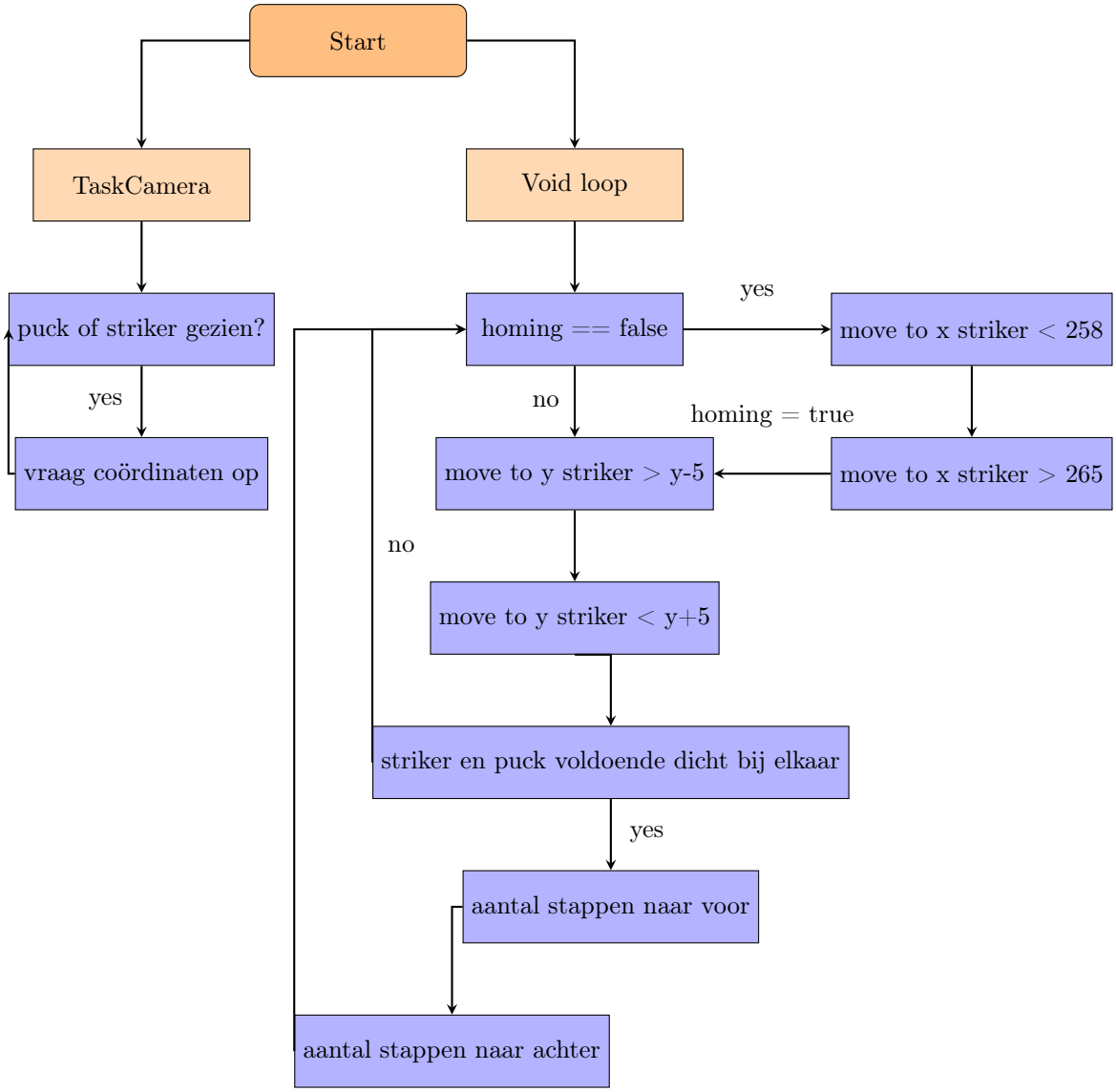
Om dit probleem op te lossen, creëren we twee oneindige loops in onze code die onafhankelijk van elkaar kunnen werken. De code om de data uit te lezen van onze camera runnen we bijvoorbeeld op core 0. De code om de stappenmotoren aan te sturen, runnen we op core 1. Hierdoor hinderen beide handelingen elkaar niet. In listing 2.1 kun je zien dat de code binnen Task1code op core 0 loopt en de code binnen de void loop runt bij Arduino IDE standaard op core 1, twee gescheiden cores dus.

Listing 2.1: Dual core voorbeeld Santos, 2018

```
TaskHandle_t Task1;

void setup() {
  Serial.begin(115200);

  xTaskCreatePinnedToCore(
                    Task1code,   /* Task function. */
                    "Task1",     /* naam van de taak. */
                    10000,       /* Stack size van de taak */
                    NULL,        /* parameter van de taak */
                    1,           /* prioriteit van de taak */
                    &Task1,      /*Task handle om de taak bij te houden*/
                    0);          /* voer taak uit op core 0 */
  delay(500);
}

//Task1code: code dat moet runnen op core 0
void Task1code( void * pvParameters ){
  for(;;){
    /* oneindige loop */
    vTaskDelay(10);
  }
}
//void loop(): code dat moet runnen op core 1
void loop() {

}
```

2.3 Programma van de air hockey robot

Om heel deze robot te laten werken moest er natuurlijk ook geprogrammeerd worden. Om dit alles een beetje duidelijker te maken hebben we in fig. 2.10 een flow diagram gemaakt. Hierin kan men eenvoudig de opbouw van het programma volgen zonder diepe kennis van het programma te hebben.



Figuur 2.10: flow diagram

Zoals in vorige subsectie vermeld, maken we gebruik van twee core's op de ESP32. Dit kan men ook eenvoudig zien in bovenstaand flow diagram. Zo runt TaskCamera in een oneindige loop op core 0 en runt alle code die te maken heeft met de stappenmotoren op core 1. Zo hinderen beide loops elkaar niet.

De Void loop staat dus in voor de stappenmotoren. We starten met een homing, eens deze voltooid is, gaat de striker op de correcte positie positioneren. Wanneer de puck dan weer voldoende dicht is, valt de striker ook aan zodat de puck terug aan de overkant geraakt.

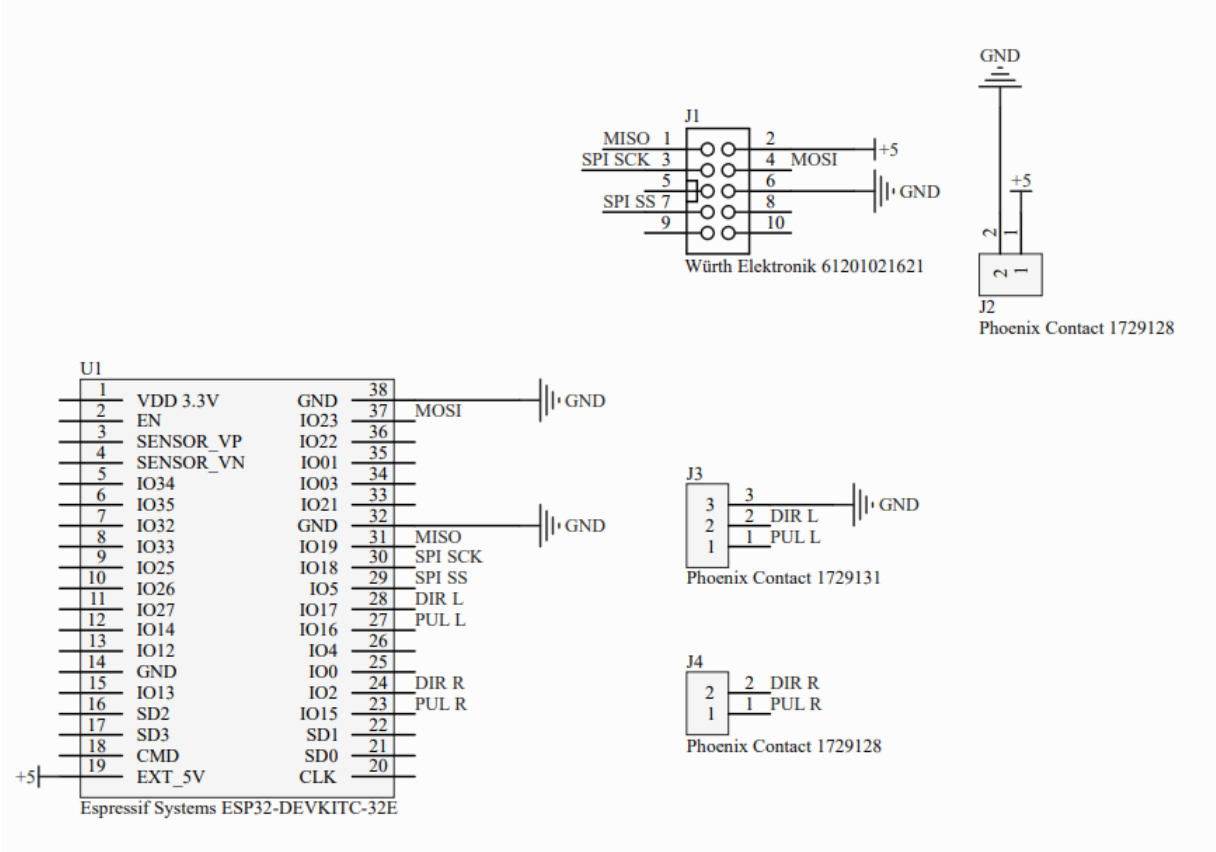
2.4 Elektrisch schema

Bij dit project hoort natuurlijk ook een elektrisch schema. Deze hebben we getekend via Circuitmaker en hiervan hebbben we eerst een prototype op een protoboard gemaakt. Dit is een bord waarop je losse componenten op kan solderen voor een properdere versie met betere contacten te hebben dan bv. een schakeling op een breadbord. Vervolgens hebben we natuurlijk een finale versie op een printed circuit board (PCB) te maken.

2.4.1 Elektrisch schema met microcontroller

Het elektrisch schema fig. 2.11 bevat de volgende componenten:

- ESP32 microcontroller voor het programmeren en besturen van onze Airhockey-robot.
- Box Header male 10pin connector voor onze Pixy2 op aan te sluiten.
- 2 screwblocks die de ESP32 met de Step en Direction pinnen van de DM542T stepper motor driver verbindt en een andere screwblock waarbij de GND verbonden is met de GND van de Mosfets (zie deel hieronder) en de 5V verbonden is met een ventilator die zorgt voor de koeling van het elektronisch gedeelte en de DM542T stepper motor drivers.

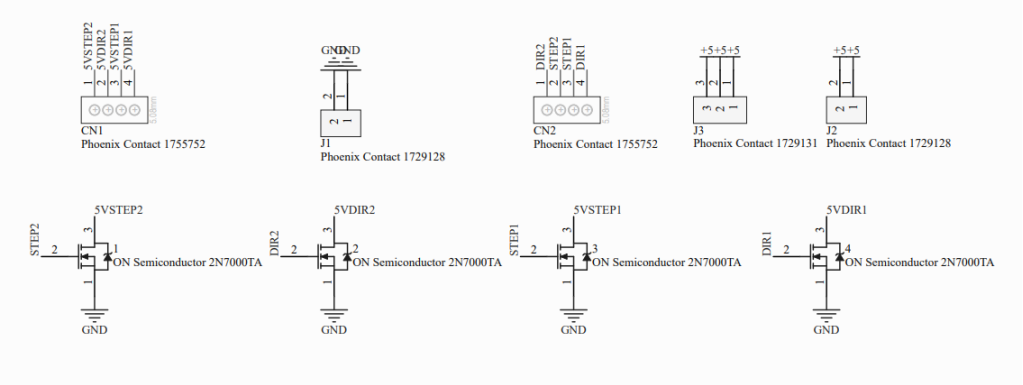


Figuur 2.11: Elektrisch schema van de Airhockey-tafel zonder mosfets

2.4.2 Elektronisch schema met mosfets

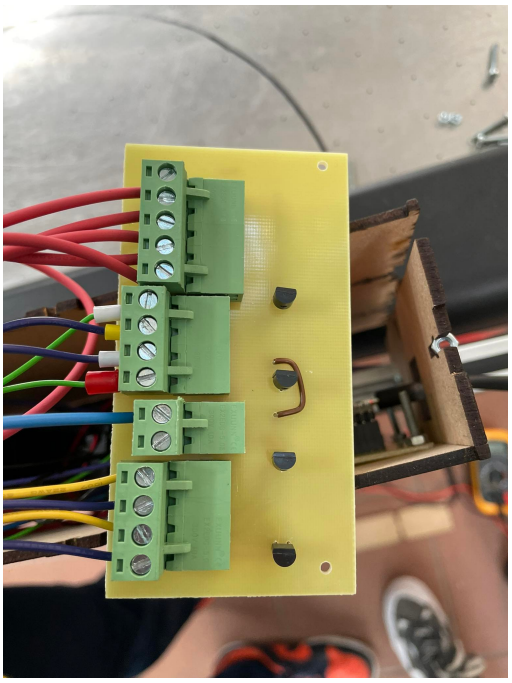
Door het continu testen van de opstelling hebben we ondervonden dat we niet genoeg kracht konden leveren om onze striker snel genoeg te laten bewegen. We hebben dit kunnen oplossen door het implementeren van een 48V voeding en sterkere DM542T stepper drivers met een range van 1-4.2A en 20-50VDC. Hiervoor gebruikten we de TB6600 stepper drivers.

Bij deze sterkere drivers moeten de signalen voor de besturing 4 à 5V kunnen leveren i.p.v. de standaard 3,3V van de ESP32. Dit hebben we dan kunnen oplossen door het omzetten van 3,3V signalen naar 5V signalen d.m.v. een mosfet schakeling die zichtbaar is op fig. 2.12. We konden echter niet zomaar eender welke mosfet gebruiken. Aangezien de ESP32 maar 3,3V kan leveren, willen we een mosfet gebruiken die men met deze spanning al in geleiding krijgt. We gaan dus op zoek naar een mosfet met een $V_{\mathrm{GS,on}}$ van 3,0 à 3,3V, hier kwamen we uit bij een 2N7000 mosfet.

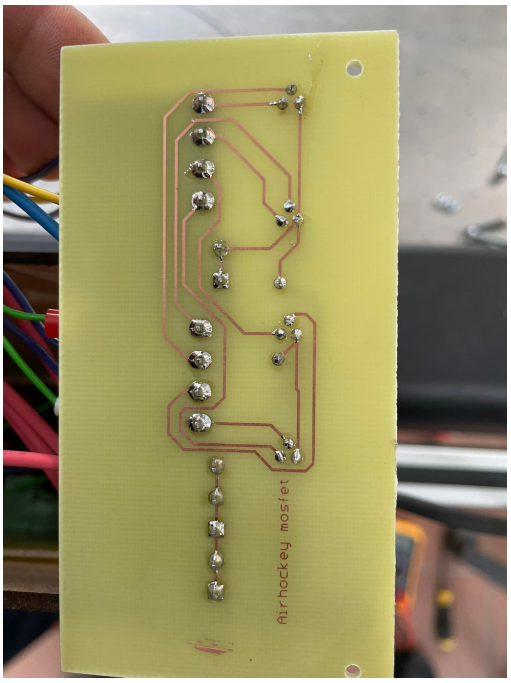


Figuur 2.12: Elektrisch schema van de 2N7000 mosfets

Een voorbeeld van deze gerealiseerde PCB met zijn componenten fig. 2.13 en het soldeerwerk fig. 2.14 ziet u op de volgende afbeeldingen:



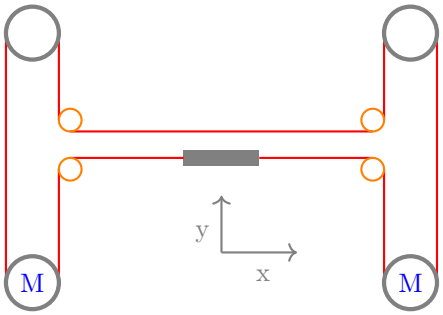
Figuur 2.13: PCB met 2N7000 mosfets



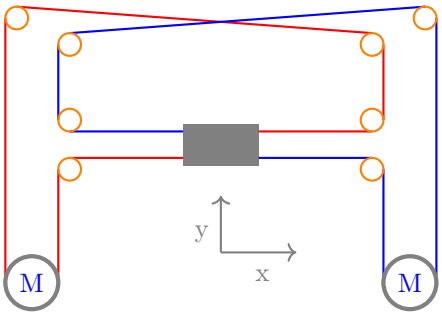
Figuur 2.14: Onderzijde PCB met mosfets

3.1 X, Y-systeem

Er bestaan twee "grote"x, y-mechanismen. Het ene is H-bot (fig. 3.1) en het andere heet coreXY fig. 3.2. Beide mechanismen hebben hun voor- en hun nadelen.



Figuur 3.1: H-bot systeem



Figuur 3.2: core XY systeem

3.1.1 CoreXY

CoreXY verwijst naar het bewegingsmechanisme van een machine volgens de 2 assen X en Y. Dit bewegingsmechanisme heeft 2 stappenmotoren nodig om zijn werk te doen en deze motoren kunnen gelijktijdig en onafhankelijk van elkaar bewegen.

De CoreXY is verbeterd ten opzichte van het cartesiaanse systeem. Het heeft verschillende vlakken zodat de riemen vrij kunnen bewegen en er minder torsie optreedt wanneer de motoren dezelfde kant op gaan.

Bovendien worden de motoren voor zowel de X- als de Y-as stationair gehouden. Dat heeft tot gevolg dat men zich minder zorgen hoeft te maken over bewegende delen. Doordat deze motoren niet op hun plaats bewegen, is er minder traagheid, wat een snellere acceleratie mogelijk maakt.

CoreXY is dus samengevat een eenvoudig concept waarbij men verschillende componenten uit verschillende materialen kan implementeren.

De nadelen zijn wel dat men meer tijd zou moeten besteden aan het onderhouden van een CoreXY-mechanisme t.o.v. H-bot vanwege de verscheidenheid aan vlakken en elke riem zou correct moeten worden aangespannen. Het systeem in fig. 3.2 is dus hevig afhankelijk van zijn riemen.

3.1.2 H-bot

Het H-Bot ontwerp dankt zijn naam aan het feit dat de opstelling eruitziet als de letter H zoals men in figuur fig. 3.1 kan zien. Hier maakt men gebruik van twee motoren, loodrecht op elkaar gemonteerde rails en een distributieriem. Het H-Bot ontwerp is dus veel eenvoudiger dan de CoreXY, mede doordat er minder componenten nodig zijn en het werkingsmechanisme simpeler is.

Kort samengevat de voor- en nadelen:
Voordelen:

- Eenvoudig, met slechts één riem
- Het kan een zeer goede werking hebben als het goed is gebouwd.

Nadelen:

- Alle componenten moet perfect op elkaar zijn afgestemd, dus de afmetingen van corresponderende parts moeten overeenkomen, riemen moeten goed aangespannen worden etc.
- Minimale koppels geleverd door motoren kunnen resulteren in een slechtere kwaliteit van werking.

In het algemeen zijn de belangrijkste verschillen tussen het CoreXY en H-bot mechanisme dat:

- De CoreXY gebruikt doorgaans langere of meer riemen, terwijl het H-Bot mechanisme veel eenvoudiger is.
- De CoreXY is veel stabieler en de motoren zijn stationair, terwijl het H-Bot mechanisme erg onstabiel is doordat het hele systeem een beetje buigt als er beweging op de assen is en hierdoor minder nauwkeurig kan zijn.
- CoreXY-mechanismen zijn compact, nauwkeurig, lineair en herhaalbaar, terwijl de H-Bot nauwe toleranties moet hebben om hem stabieler te maken.

In onze omstandigheden van een hockey-tafel is het het makkelijkst om het H-bot mechanisme te gebruiken omdat dit ontwerp veel eenvoudiger op een frame te monteren is en het dus samengevat veel praktischer is voor onze toepassing.

3.2 Motoren

Om een idee te krijgen van welke motoren we zouden kunnen gebruiken moeten we enkele berekeningen uitvoeren. Om een beter beeld te krijgen, proberen we eerst een aantal inschattingen te maken.

Het gewicht van een aantal componenten:

- 3D-print striker: $\pm$100g
- 4 glijlagers in deze 3D-print: $4 \cdot 15\text{g} = 60\text{g}$
- 2x sliders 3D-print zijkanten: $2 \cdot 50\text{g} = 100\text{g}$
- 2 gladde assen D 8mm, L 120cm $\implies C45 - staal : \rho = 7,85\,\text{g/cm}^3 \implies m = \rho \cdot V = 7,85\text{g/cm}^3 \cdot \pi \cdot 0,4\,\text{cm}^2 \cdot 120\text{cm} = 473,50\text{g} \implies m_{\text{tot}} = 2 \cdot m = 947,0\text{g}$

  $\implies m_{\text{totale constructie}} = 100\,\text{g} + 60\,\text{g} + 100\,\text{g} + 60\,\text{g} + 947\,\text{g} = 1267\,\text{g} = 1,267\,\text{kg}$

Vervolgens onderstellen we de volgende gegevens:

De puck heeft een maximum snelheid van 70 km/h.

We zouden idealiter een snelheid/koppel-grafiek willen die zoveel en zo traag mogelijk lineair daalt.

Het veronderstelde gewicht van de puck is 18g.

Het basismodel is hier de 8 foot airhockey tafel (96"L x 50"W x 32"H inch => 243,84L x 127W x 81,28H cm).

Vervolgens onderstellen we dat de tegenstander van de helft van zijn speelhelft de puck schiet met zijn striker.

Volgens de regels wordt er door de striker op 20cm van zijn doel verdedigd.

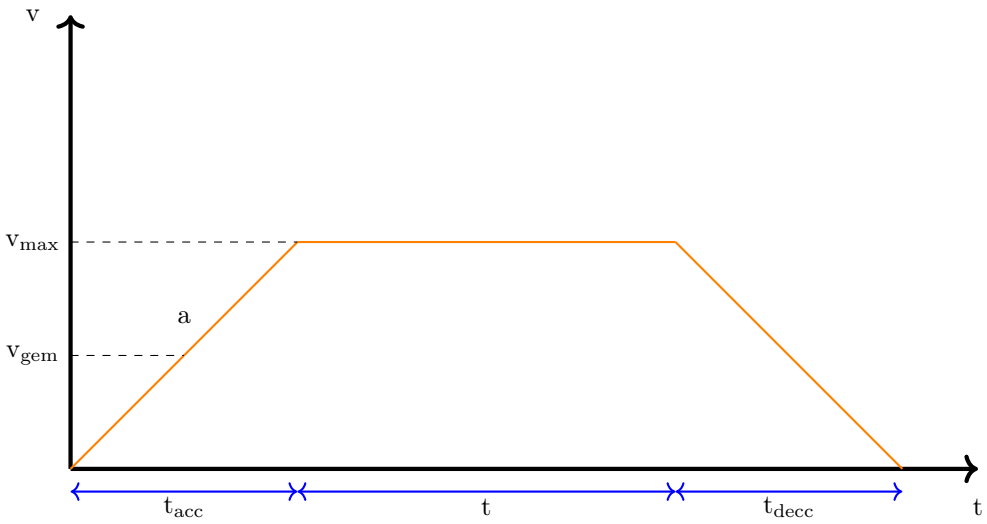
Stel: de vijandige striker schiet de puck en de verdedigende striker maakt een voorwaartse beweging naar de puck toe. Dit zou gebeuren van pos1 = 20cm naar pos2 = 40cm over een tijd t = ?

We weten nu dus de snelheid van de puck, dus kunnen we de afgelegde afstand (van de aanvallende tot verdedigende striker) eenvoudig berekenen met gegevens waaruit de tijd die de puck nodig heeft om dit traject af te leggen, volgt:

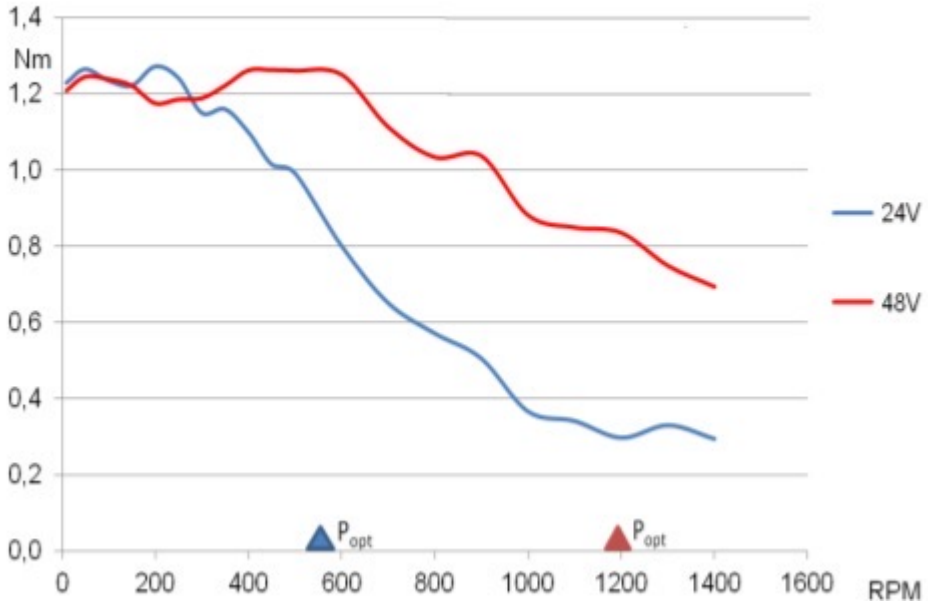
$s_{\text{p}} = L_{\text{play}} - L_{\text{defstr}} - L_{\text{play}/4} = 243,84 - 40 - 243,84/4 = 142,88\text{cm}$

$t = \frac{s}{v} = \frac{1,4288\,\text{m}}{19,44\,\text{m/s}} = 0,07348\,\text{s}$

Voor de berekeningen betreffende de tijd dat de motoren er over doen om een bepaalde afstand af te leggen hanteren we volgend snelheidsprofiel zoals afgebeeld in fig. 3.3.



Figuur 3.3: snelheidsprofiel



Figuur 3.4: Rpm t.o.v. koppel uit de datasheet van NEMA23 - 57BYGH450E-05

We kiezen in fig. 3.4 voor het toerental van 1200 toeren per minuut. Als we voor dit toerental het koppel aflezen, indien we een driver van 48 V nemen, komen we op een koppel van ongeveer 0,85 Nm.

3.2.1 BEWEGING IN X-RICHTING

Indien we de beweging in de x-richting bestuderen hebben we enkel de striker die beweegt. Deze heeft een gewicht van 0,160 kg en verplaatst zich maximaal over een afstand van 60 cm. Tot slot kiezen en we hier voor een pulley met een diameter van 3 cm.

Eerst berekenen we de maximale versnelling die de motor kan leveren met dit gewicht:
$\Longrightarrow \mathrm{F} = \frac{T}{r} = \frac{0{,}85}{0{,}015} = 56{,}67\,\mathrm{N}$
$\Longrightarrow \mathrm{a} = \frac{F}{m} = \frac{56{,}67}{0{,}160} = 354{,}17\,\frac{\mathrm{m}}{\mathrm{s}^2}$

We vormen de snelheid uit de datasheet om:
$\Longrightarrow \mathrm{v_{max}} = \frac{rpm \cdot 2 \cdot \pi \cdot r}{60} = \frac{1200 \cdot 2 \cdot \pi \cdot 0{,}015}{60} = 1{,}885\,\frac{\mathrm{m}}{\mathrm{s}}$

De tijd dat we er over doen om tot deze snelheid te komen is:
$\Longrightarrow \mathrm{t_{acc}} = \frac{v_{\mathrm{max}} - v_0}{a} = \frac{1{,}885}{354{,}17} = 0{,}0053222\,\mathrm{s}$
De tijd voor het accelereren en deccelereren is gelijk aan elkaar, maar wat is de afstand dat we overbruggen tijdens het accelereren?
$\Longrightarrow \mathrm{s_{acc}} = (\mathrm{v_0} \cdot \mathrm{t_{acc}}) + (\frac{1}{2} \cdot \mathrm{a} \cdot \mathrm{t_{acc}^2}) = (0 \cdot 0{,}0053222) + (\frac{1}{2} \cdot 354{,}17 \cdot 0{,}005322^2) = 0{,}5016\,\mathrm{cm}$

De resterende afstand dat we dus met een constante snelheid gaan afleggen, is:
$\Longrightarrow \mathrm{s_{rest}} = 60\,\mathrm{cm} - 2 \cdot \mathrm{s_{acc}} = 60\,\mathrm{cm} - 2 \cdot 0{,}5016\,\mathrm{cm} = 58{,}89968\,\mathrm{cm}$

Deze afstand overbruggen we in volgende tijd:
$\mathrm{t} = \frac{s_{\mathrm{rest}}}{v_{\mathrm{max}}} = \frac{58{,}89968\,\mathrm{cm}}{1{,}886\,\frac{\mathrm{m}}{\mathrm{s}}} = 0{,}312988\,\mathrm{s}$

De totale tijd om deze afstand te overbruggen bekomen we dus door alle deeltijden op te tellen:
$\mathrm{t_{tot}} = \mathrm{t_{acc}} + \mathrm{t_{decc}} + \mathrm{t} = 0{,}32363\,\mathrm{s}$

3.2.2 BEWEGING IN DE Y-RICHTING

Indien we de beweging in de y-richting bestuderen hebben we de striker met de assen en de sliders die bewegen. Deze hebben een gewicht van 1,267 kg en verplaatsen zich maximaal over een afstand van 20 cm. Ook hier hebben we een pulley met een diameter van 3 cm.

Eerst berekenen we de maximale versnelling die de motor kan leveren met dit gewicht:
$\Longrightarrow \mathrm{F} = \frac{T}{r} = \frac{0{,}85}{0{,}015} = 56{,}67\,\mathrm{N}$
$\Longrightarrow \mathrm{a} = \frac{F}{m} = \frac{56{,}67}{1{,}267} = 44{,}73\,\frac{\mathrm{m}}{\mathrm{s}^2}$

We vormen de snelheid uit de datasheet om:
$\Longrightarrow \mathrm{v_{max}} = \frac{rpm \cdot 2 \cdot \pi \cdot r}{60} = \frac{1200 \cdot 2 \cdot \pi \cdot 0{,}015}{60} = 1{,}885\,\frac{\mathrm{m}}{\mathrm{s}}$

De tijd dat we er over doen om tot deze snelheid te komen is:
$\Longrightarrow \mathrm{t_{acc}} = \frac{v_{\mathrm{max}} - v_0}{a} = \frac{1{,}885}{44{,}73} = 0{,}04215\,\mathrm{s}$
De tijd voor het accelereren en deccelereren is gelijk aan elkaar, maar wat is de afstand dat we overbruggen tijdens het accelereren?
$\Longrightarrow \mathrm{s_{acc}} = (\mathrm{v_0} \cdot \mathrm{t_{acc}}) + (\frac{1}{2} \cdot \mathrm{a} \cdot \mathrm{t_{acc}^2}) = (0 \cdot 0{,}04215) + (\frac{1}{2} \cdot 44{,}73 \cdot 0{,}04215^2) = 3{,}972\,\mathrm{cm}$

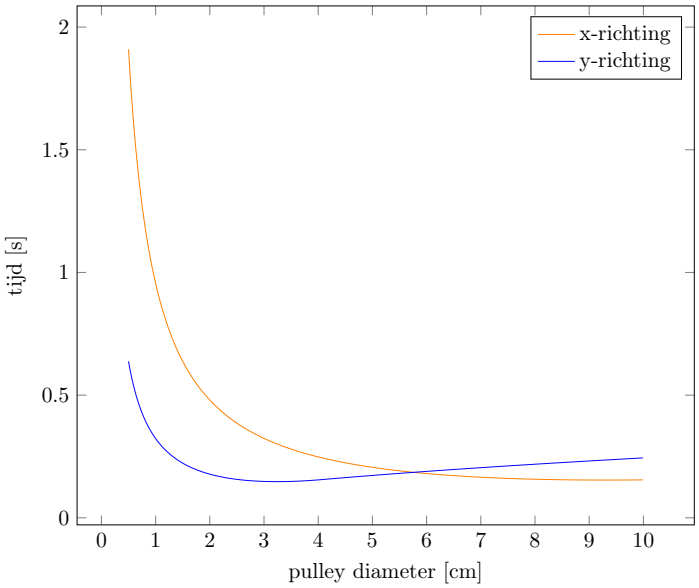
De resterende afstand dat we dus met een constante snelheid gaan afleggen, is:
$\Longrightarrow \mathrm{s_{rest}} = 20\,\mathrm{cm} - 2 \cdot \mathrm{s_{acc}} = 20\,\mathrm{cm} - 2 \cdot 3{,}972\,\mathrm{cm} = 12{,}056\,\mathrm{cm}$

Deze afstand overbruggen we in volgende tijd:
$\mathrm{t} = \frac{s_{\mathrm{rest}}}{v_{\mathrm{max}}} = \frac{12{,}056\,\mathrm{cm}}{1{,}886\,\frac{\mathrm{m}}{\mathrm{s}}} = 0{,}06396\,\mathrm{s}$

De totale tijd om deze afstand te overbruggen bekomen we dus door alle deeltijden op te tellen:
$\mathrm{t_{tot}} = \mathrm{t_{acc}} + \mathrm{t_{decc}} + \mathrm{t} = 0{,}14825\,\mathrm{s}$

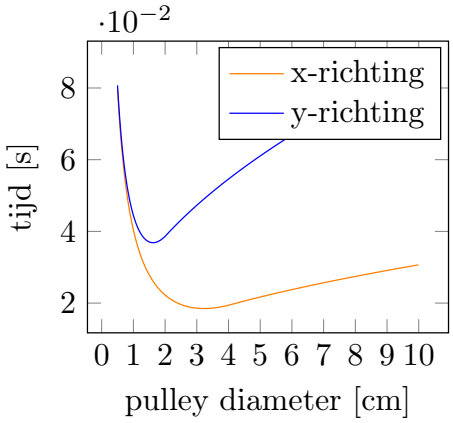
3.3 Bepalen diameter pulley

Indien we voorgaande berekeningen telkens opnieuw maken voor een andere diameter van pulley kunnen we eenvoudig een beeld krijgen van met welke pulley we de afstand het snelst kunnen afleggen. We verwachten dat we met een kleine pulley een relatief hoog koppel kunnen leveren, dit zorgt ervoor dat we een lagere maximale snelheid. Wanneer we voor een grote pulley kiezen kunnen we slechts een beperkt koppel leveren, het voordeel hier is dat we een hogere maximale snelheid kunnen halen. Hieruit kunnen we concluderen dat een kleine pulley geschikt is voor korte afstanden en een grote pulley eerder geschikt is voor langere afstanden. Om een middenweg tussen deze pulleys te bepalen hanteren we volgende berekeningen:

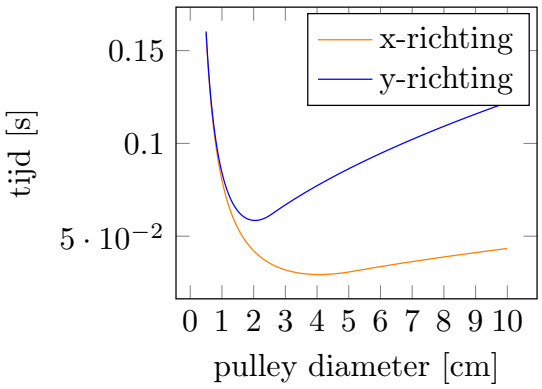


Figuur 3.5: tijd in functie van de pulley diameter

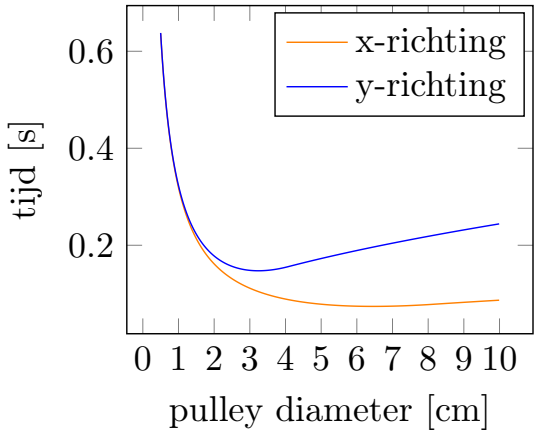
Idealiter willen we een pulley waarbij beide tijden zo goed als 0 s zijn, dit is niet mogelijk dus zoeken we een compromis. Uit fig. 3.5 kunnen we afleiden dat een pulley met een diameter van ongeveer 4,5 cm het beste zou zijn.



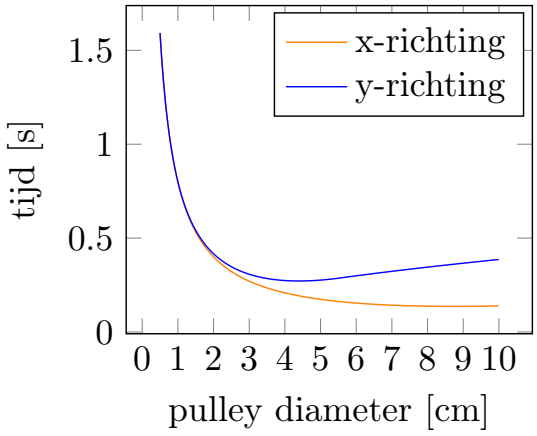
Figuur 3.6: tijd in functie van de pulley diameter voor een verplaatsing van 2,5 cm



Figuur 3.7: tijd in functie van de pulley diameter voor een verplaatsing van 5 cm



Figuur 3.8: tijd in functie van de pulley diameter voor een verplaatsing van 20 cm



Figuur 3.9: tijd in functie van de pulley diameter voor een verplaatsing van 50 cm

In fig. 3.6 tot en met fig. 3.9 kun je het verloop zien van de tijd in functie van de pulley diameter afhankelijk van de te overbruggen afstand. We zien dat het dal het meest links gelegen is indien we de kortste afstand overbruggen. Wanneer we een langere afstand willen overbruggen zien we dat het dal steeds meer naar rechts opschuift.

Dit maakt het natuurlijk wat moeilijker om een geschikte pulley te gaan kiezen. We zullen goed moeten overwegen wat de voor- en de nadelen zijn.

Uit de data die we hierboven hebben, kunnen we best voor een pulley kiezen met een diameter van ongeveer 3,5 cm of 4 cm. Als we dan door de beschikbare pulley's gaan zoeken voor een GT2 type riem, komen we uit op een pulley met 40 tanden.

Figuur 3.10: pulley 40 tanden GT2 riem

3.4 Constructie

Ons CAD model toont eenvoudig al onze ontwerp keuzes. Zo hebben we gekozen voor een U-vormig frame gemaakt uit aluminium extrusie profielen I-type 5 zoals men kan zien in fig. 3.11. Dit was echter de enige manier om snel en eenvoudig een stevige constructie te bouwen.

Figuur 3.11: Samenstelling Air Hockey robot

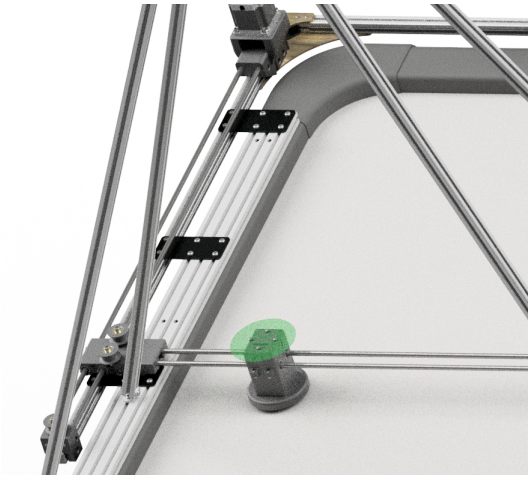
De belangrijkste reden die we gekozen hebben voor een stevig frame, is om zo de montage en demontage tijd van heel onze robot sterk te kunnen reduceren. Door simpelweg een constructie te bouwen die we met een paar moeren en bouten volledig kunnen vastschroeven besparen we ons heel wat tijd.

In fig. 3.12 en fig. 3.13 kunnen we zien hoe het frame precies in elkaar zit. We hebben gekozen om de hoeken extra te verstevigen omdat tijdens het transport dit toch wel het grootste pijnpunt zal zijn. Hier komen namelijk de grootste krachten op.

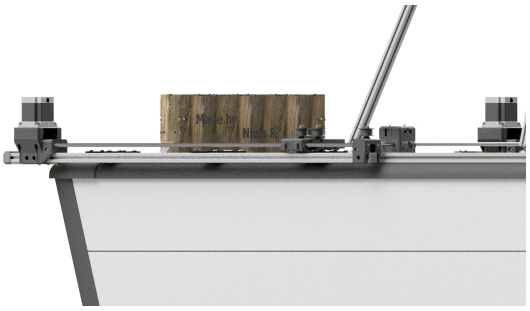
Figuur 3.12: samenstelling frame

Figuur 3.13: montage frame

Dit frame moet natuurlijk ook bevestigd kunnen worden op de Air hockey tafel. Om dit mogelijk te kunnen maken hebben we door het relatief hoge gewicht van de volledige constructie gekozen voor metalen plaatjes zoals men op fig. 3.14 kan zien.



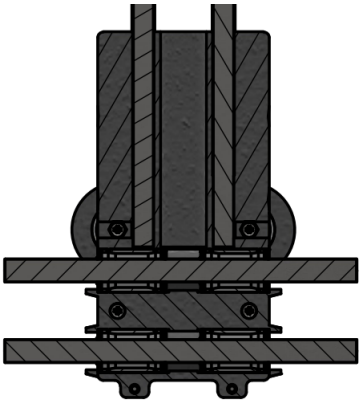
Figuur 3.14: bevestiging plaatje



Figuur 3.15: mounts

Uiteindelijk moet alles ook op dit frame bevestigd kunnen worden. Hiervoor hebben we ons design zo eenvoudig mogelijk gemaakt zodat we deze onderdelen simpelweg over het extrusie profiel kunnen schuiven. Verder heeft dit nog als voordeel dat we de mounts (fig. 3.15) kunnen verschuiven, dit maakt dat we eenvoudig de riem op correcte spanning kunnen brengen.

Om de geleiding vlot te laten verlopen, zowel voorwaarts als achterwaarts, maken we gebruik van lineaire glijlagers. Hoe deze juist gepositioneerd zijn, kunnen we zien in fig. 3.16. Er zitten 2 glijlagers per as om zoveel mogelijk moment op te kunnen vangen. Aangezien we gebruik maken van 2 assen per part, een part bevat dus 4 glijlagers.



Figuur 3.16: sliders doorsnede

In dit hoofdstuk zullen we wat meer spreken over de realisatie van ons project. Het ontwerpen van de Airhockey robot was uiteraard een uitgebreid proces dat natuurlijk gepaard ging met veel trial en error, zoals u al misschien in een eerder deel van dit verslag hebt kunnen lezen en dat men bijgehouden heeft in een logfile zoals men kan zien in appendix A.

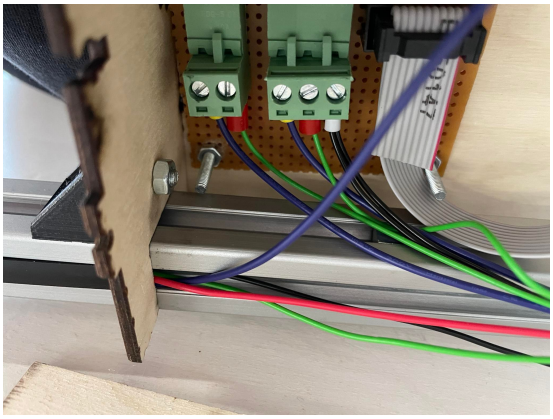
We zijn het project dus begonnen met te brainstormen over hoe we onze praktische realisatie van de Airhockey robot gingen ontwerpen, rekening houdend met onder andere de opstelling, het materiaal, de afmetingen enz. van het door de promotoren gekozen model van de Airhockey tafel. Dit moesten we dan volgens een bepaald mechanisme kunnen laten werken. Deze ideeën hebben we dan als gevolg vertaald in een CAD-model, zichtbaar in fig. 3.11. Hier hebben wegens zijn eenvoudig design voor het H-bot mechanisme gekozen die reeds uitgebreid werd besproken in dit deel fig. 1.1 van het hoofdstuk Mechanica.

In de praktische realisatie hebben we dan o.a. gebruik gemaakt van aluminium extrusieprofielen, riemen, assen, lagers, pulleys en 3D-prints. Natuurlijk moesten we de praktische realisatie van het project afstemmen op de elektronische parts zoals o.a. de stappenmotoren met zijn bijhorende drivers, een ventilator, de pixy2.1 camera en natuurlijk ook de ESP 32 microcontroller. Deze hebben we dan vervolgens in een behuizing van berk geïntegreerd die op zijn beurt netjes op het frame gemonteerd is.

Onze eerste versie was dan ook een frame met 3D-printed parts en stappenmotoren op. Dit frame hebben we dan met betrekking van stalen plaatjes vastgemaakt aan de aluminium extrusieprofielen.

Figuur 4.1: Eerste versie frame

Voor onze elektronica hebben we dan een houten behuizing gemaakt waarin we alle elektronische componenten mooi konden monteren. Uiteraard is er ook aan een gepaste afwerking / cable-management van de constructie gedacht zoals u op de onderstaande afbeelding fig. 4.2 ziet.



Figuur 4.2: Montage elektronische behuizing en afwerking elektronica

Op vlak van de werking van het project hebben we ook een aantal dingen kunnen realiseren. We zijn dit project gestart met eerst op een simpele manier een stappenmotor met een driver aan te kunnen sturen.

Vervolgens hebben we een tweede stappenmotor hieraan kunnen toevoegen met zijn respectievelijke driver.

Hierna hebben we dan de pixy2.1 camera leren kennen en zijn we hier begonnen van het tracken van een object en zijn coördinaten te kunnen uitlezen. Hierop hebben we dan vervolgens de baan berekend die het object tussen 2 tijdstippen aflegt en hierbij horend ook het toekomstig traject voorspeld a.d.h.v. de richtingscoëfficiënt en eventuele weerkaatsing waarbij de richtingscoëfficiënt dan simpelweg van teken verwisselt. Tijdens de praktische uitvoering van het te voorspellen traject kregen we te maken met de vervorming van de lens wat maakte dat het voorspelde traject niet klopte en dus meer kwaad dan goed deed. Hierdoor hebben we beslist om de trajectvoorspelling links te laten liggen.

Figuur 4.3: Finale praktische realisatie air hockey robot

Ten laatste moesten we dus alle deelprogramma's samenvoegen in 1 grote werkende code, wat eigenlijk één van de grootste moeilijkheidsgraden van dit project was door de complexiteit om alle componenten synchroon met elkaar te kunnen laten werken. Na veel trial & error zijn we er uiteindelijk in geslaagd om de door stappenmotoren aangestuurde striker de puck te doen kunnen laten volgen en deze zelf met een impuls terug te kunnen laten spelen.

Het enige probleem dat we dan nog tegen kwamen was dat men niet voldoende snel kon bewegen. Om dit op te lossen kozen we voor een sterkere driver voor de stappenmotoren. Deze verwacht een signaal tussen 4V en 5V maar onze microcontroller kon maar een signaal van 3,3V leveren. Hierdoor moest het signaal afkomstig van onze ESP 32 naar een mosfet gestuurd worden. Deze mosfet (2N7000) werd dan in geleiding gebracht zodat hij 5V naar de driver kon sturen. Door al deze aanpassingen kan de striker een veel hogere snelheid halen.

Lessons learned

Nu de praktische realisatie van onze Bachelorproef volledig afgerond is, kunnen we nagaan wat we nu eigenlijk allemaal geleerd hebben gedurende dit traject.

Aangezien dit ons eerste project was waar we stappenmotoren aan een hoge snelheid moesten laten draaien, hebben we natuurlijk redelijk veel bijgeleerd over stappenmotoren. Wat we vooral gemerkt hebben, is dat de snelheid waarmee we kunnen bewegen, en dus ook het koppel geleverd door de motoren, aanzienlijk stijgt met een stijgende spanning te gebruiken op de drivers. Natuurlijk hadden de drivers op zich hier ook wel een impact in. Iets krachtigere stepper drivers die een iets hogere spanning aan kunnen, hadden voor ons een immens verschil.

Het switchen van stepper drivers was niet van zelf sprekend, zo zijn we te weten gekomen dat niet iedere driver kan aangestuurd worden door een ESP32. Het feit dat deze pinnen werken op 3,3V en de meeste drivers een ingangssignaal willen ontvangen tussen 4V en 5V heeft ons aan het denken gezet. Zo hebben we weer probleemoplossend proberen denken en zo een mosfet schakeling ontwerpen.

Ook was het voor ons beiden de eerste keer dat we gebruik maakten van een camera. Aangezien we dit nog niet eerder gedaan hadden, hadden we er ook een beetje schrik voor. Om deze reden kozen we dan ook voor een keuze dat op het eerste zicht het eenvoudigst leek, de Pixy2.1 camera. Achteraf gezien bleek dit niet meer de eenvoudigste keuze door het niet zo goed overweg kunnen met overbelichting van deze camera. Hierdoor moesten we aan de slag met polarisatiefilters, wat voor ons compleet nieuw was maar nu weten we ook hoe we deze kunnen gebruiken en wat zowel de voor- als nadelen hiervan zijn.

Op vlak van databussen is onze kennis uiteraard ook niet constant gebleven bij de start van het project kenden we nauwelijks het verschil tussen SPI en I^2C terwijl we door trail and error de verschillen wel hebben ondervonden.

Wat als we het opnieuw zouden doen?

Wat zouden we anders doen moesten we het project nu opnieuw moeten doen? Als eerste zouden we opteren voor een ander X, Y-systeem dan H-bot. Op zich is er niets mis met het H-bot systeem maar we hebben de breedte van de tafel wat onderschat. Dit mechanisme trekt sowieso al wat scheef maar door deze lengte trekt hij wel heel erg scheef. Door dit scheef trekken zet het mechanisme zichzelf soms vast waardoor we de striker niet meer kunnen bewegen.

Een volgend punt dat we zouden aanpassen, is de camera keuze. We dachten een slimme keuze gemaakt te hebben door voor de Pixy2.1 te kiezen. Achteraf gezien zouden we eerder voor een klassieke webcam met wide angle kiezen. Deze zouden we dan aan de hand van een Raspberry pi de object detectie laten doen. Door hiervoor te kiezen hebben we veel meer vrijheid in het detecteren van een object.

Dit brengt ons bij het volgende punt. In plaats van het object te tracken aan de hand van kleuren zouden we er voor durven kiezen om een patroon op zowel de puck als de striker aan te brengen indien dit toegelaten zou zijn. De reden voor deze keuze is dat men een patroon veel moeilijker verstoord krijgt door de belichting dan dat men een kleur verstoord krijgt. Op deze manier zouden we dan beter met verschillende soorten belichtingen overweg moeten kunnen wat toch wel het grootste pijnpunt van dit project was.

Bijlage

In fig. A.1 kunnen we een deel zien van het logboek dat we doorheen het academiejaar bijgehouden hebben. Indien men wenst om het volledige logboek te bekijken is er de volgende link naar onze logboek.

Logfile				
Naam:	**Datum:**	**Duur:**	**Wat heb je gedaan?**	**Waar zijn de documenten te vinden?**
Niels Vanneromn	9/10/2021	02:00:00	Stepper mounts getekend in inventor.	Op de pc van Niels
Niels Vanneromn	10/10/2021	03:00:00	Eerste samenstelling van het mechanisme getekend in inventor.	Op de pc van Niels
Amirgan Bouakhounov	11/10/2021	01:00:00	Research + bespreking shopping list + opmaak 'drive' & logboek + INDIWORKS bespreking bachelorproef : how?	Op het tablab pagina's van de groep op Canvas
Niels Vanneromn	11/10/2021	01:00:00	Research + bespreking shopping list + opmaak 'drive' & logboek + INDIWORKS bespreking bachelorproef : how?	Op het tablad pagina's van de groep op Canvas
Niels Vanneromn	13/10/2021	01:00:00	Opzoeken welke camera, stappenmotoren en drivers voor stappenmotoren we het best zullen gebruiken	Op het tablad pagina's van de groep op Canvas
Niels Vanneromn	15/10/2021	01:00:00	Eerste versie van de Bill Of Material gemaakt.	Op het tablad pagina's van de groep op Canvas
Niels Vanneromn	18/10/2021	01:00:00	INDIWORKS bespreking bachelorproef: voorstelling concept. + berekening voor het aantal fps van de camera	bij bestanden in de Teams groep
Amirgan Bouakhounov	18/10/2021	00:30:00	INDIWORKS bespreking bachelorproef: voorstelling concept.	
Amirgan Bouakhounov	21/10/2021	04:00:00	Samenstellen BOM + berekeningen holding torque stappenmotoren	Bij bestanden in de Teams groep
Niels Vanneromn	21/10/2021	05:00:00	Verfijnen BOM + creëeren Gantt chart	Bij bestanden in de Teams groep
Amirgan Bouakhounov	22/10/2021	03:00:00	Vervolledigen BOM	Bij bestanden in de Teams groep
Niels Vanneromn	22/10/2021	04:00:00	Aanpassingen maken aan de inventor tekening + alternatief zoeken voor de pixy 2 camera → pixy 2.1 heeft een wijdere lens.	Op de pc van Niels
Niels Vanneromn	23/10/2021	04:00:00	Aanpassingen maken aan de inventor tekening .	Op de pc van Niels
Niels Vanneromn	24/10/2021	05:00:00	Verfijnen Inventor tekening.	Op de pc van Niels
Niels Vanneromn	29/10/2021	04:00:00	Tekenen van de striker en drive pulley, poging gedaan om een syncronous belt toe te voegen.	Op de pc van Niels
Amirgan Bouakhounov	29/10/2021	04:00:00	Parts uit Fablab verzamelen + uittesten stepper motor + driver + esp32	Op de pc van Amirgan
Niels Vanneromn	9/11/2021	00:45:00	INDIWORKS bespreking bachelorproef: bespreking planning, concept en elektronica	Verslag is te vinden bij bestanden in de Teams groep
Amirgan Bouakhounov	9/11/2021	00:45:00	INDIWORKS bespreking bachelorproef: bespreking planning, concept en elektronica	Verslag is te vinden bij bestanden in de Teams groep
Niels Vanneromn	11/11/2021	00:45:00	Berekening voorspelling voor het traject van de puck	Bij bestanden in de Teams groep
Amirgan Bouakhounov	12/11/2021	04:00:00	Testen van de steppers.	Op de pc van Amirgan
Niels Vanneromn	13/11/2021	01:30:00	Berekenen van de fout van het voorspelde traject + opstellen eenvoudig flow diagram	Bij bestanden in de Teams groep
Niels Vanneromn	14/11/2021	00:30:00	Opzoekingswerk voor een geschikte riem met volgwiel en eventueel geschikt tandwiel	In de Bill of Material (BOM) in deze excel sheet
Amirgan Bouakhounov	18/11/2021	03:30:00	Stappenmotoren testen op Arduino UNO (esp geeft problemen bij uploaden - write timeout)	Bij bestanden in de Teams groep
Niels Vanneromn	24/11/2021	01:30:00	Testen van de camera en tracken van een object in combinatie met een Arduino UNO	Op de pc van Niels
Amirgan Bouakhounov	25/11/2021	06:30:00	Stappenmotoren met millis laten draaien + esp testen en errors oplossen	Bij bestanden in de Teams groep
Niels Vanneromn	25/11/2021	08:00:00	Bepalen van de richting, snelheid & positie van het te tracken object + stepper laten draaien a.d.h.v. micros()	Op de pc van Niels
Niels Vanneromn	26/11/2021	01:00:00	Beslissen welke stappenmotor we gaan gebruiken adhv Amirgan zijn berekeningen van het nodige koppel + proberen uitzoeken waarom de pixy lib	In de Bill of Material (BOM) in deze excel sheet
Niels Vanneromn	27/11/2021	01:00:00	Error opgelost voor het kunnen uploaden van een code met pixy2 library op een ESP32 en pixy2SPI_SS library aangepast zodat deze kan communice	Op de pc van Niels
Niels Vanneromn	28/11/2021	02:00:00	Case gemaakt voor de pixy2. 1 camera en nagedacht over hoe we het traject van de puck kunnen voorspellen.	Op de pc van Niels
Amirgan Bouakhounov	1/12/2021	07:15:00	Stappenmotoren aansturen met behulp van een library en aantal testen doen van het koppel bij hoge snelheid + nadenken over het frame + INDIW(	Op de pc van Niels
Niels Vanneromn	1/12/2021	08:45:00	Stappenmotoren aansturen met behulp van een libary en aantal testen doen van het koppel bij hoge snelheid + nadenken over het frame + INDIWC	Op de pc van Niels
Amirgan Bouakhounov	2/12/2021	01:30:00	Elektrisch schema van het project opstellen + PCB tekenen in Circuitmaker	Op de pc van Amirgan
Niels Vanneromn	2/12/2021	01:30:00	CAD tekening aanpassen → tekenen van het frame	Op de pc van Niels
Niels Vanneromn	14/12/2021	01:00:00	Opstellen van het verslag: tussentijdse evaluatie einde eerste semester	In het Overleaf document
Amirgan Bouakhounov	15/12/2021	05:30:00	Berekenen van het nodige koppel van de steppers via Python en datasheets, kijken welkmateriaal er van Motedis aanwezig is	Op de pc van Amirgan
Niels Vanneromn	15/12/2021	06:30:00	Berekenen van het nodige koppel van de steppers, kijken welkmateriaal er van Motedis aanwezig is en aanpassingen doen aan het CAD model	Op de pc van Amirgan en Niels
Amirgan Bouakhounov	16/12/2021	02:30:00	Opstellen van het verslag: tussentijdse evaluatie einde eerste semester	In het Overleaf document
Niels Vanneromn	17/12/2021	02:00:00	Aanpassingen doen aan het CAD model	Op de pc van Niels

Figuur A.1: logboek Air hockey robot groep 1

Figuur A.2: demo video air hockey robot groep 1.

Bibliografie

Dy, A. (2021). *CoreXY vs HBot 3D Printers: Which is Right For You?* URL: https://total3dprinting.org/corexy-vs-hbot/ (bezocht op 04-03-2022).

Mecheltron (2021). *Hybrid Stepper Motor 57BYGH450E-05*. URL: https://www.rocketronics.de/download/datasheet/stepper/57BYGH450E-05.pdf (bezocht op 19-12-2021).

Pixy (2021). *Introducing Pixy 2*. URL: https://pixycam.com (bezocht op 19-12-2021).

Santos, S. (2018). *How to use ESP32 dual core with Arduino IDE*. URL: https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/ (bezocht op 16-02-2022).

StudioPieters (2022). *ESP32 - PinOut*. URL: https://www.studiopieters.nl/esp32-pinout/ (bezocht op 17-02-2022).

Van Ham, R. (2021). *Mechatronica 6 microcontrollers*. Vrije Universiteit Brussel.

Wikipedia (2021). *Connected-component labeling*. URL: https://en.wikipedia.org/wiki/Connected-component_labeling (bezocht op 30-04-2022).

— (2022a). *ESP32*. URL: https://nl.wikipedia.org/wiki/ESP32 (bezocht op 28-02-2022).

— (2022b). *Serial Peripheral Interface*. URL: https://nl.wikipedia.org/wiki/Serial_Peripheral_Interface (bezocht op 28-02-2022).