



# Flipperkast met betaalmodule

6<sup>e</sup> jaar Elektriciteit-elektronica

Schooljaar 2020-2021

Robin Monseré

Begeleiders: Dhr. K. Hertens



# Voorwoord

Voor onze geïntegreerde proef kregen we de opdracht om zelf een project te kiezen en dit te realiseren. Ik was dan ook blij dat we zelf die keuze mochten maken. Ik ben voor een flipperkast gegaan, en heb er mijn eigen twist aan te gegeven. Mijn focus ligt niet enkel op de flipperkast zelf maar ook op de betaalmodule die ik er heb bijgemaakt. Deze werkt met RFID-tags. Wil je de flipperkast gebruiken? Zet wat Credits op je RFID-tag en je kan aan de slag! De gedachte hierachter is dat andere “arcade” games dan ook gespeeld worden met diezelfde badge. Om alles te realiseren maakte ik gebruik van meerdere Arduinos, solenoïdes, stuurschakelingen en andere elektronica zoals een I2C LCD, RFID-reader, etc.

Graag zou ik enkele mensen bedanken die me dit jaar geholpen hebben met mijn GIP, zonder hun hulp zou deze GIP niet tot stand zijn gekomen. Eerst en vooral wil ik meneer Hertens bedanken, die me doorheen het jaar tips en uitleg gaf. Daarnaast wil ik Gauthier Vanhove bedanken die met zijn 3D printer stukjes heeft gemaakt, Yoni Beheydt die me geholpen heeft met het maken van het eerste prototype, Johan Vanthournout die geholpen heeft met de constructie van de flipperkast, Lukas Oliver die me Paint.net heeft leren kennen en Bas Vercruyse en Lien Vanthournout die deze portfolio gelezen hebben en verbeterd hebben. Tot slot wil ik ook mijn klasgenoten die de flipperkast getest en hun oprochte mening gaven bedanken.

# Abstractum

Deze GIP was geen simpele opdracht en ik heb er zeker veel van geleerd. Niet enkel over de elektronica en de programmatie maar ook over planning, testen, 3D printen, prototyping, je werk bijhouden en documenteren. Voor mij is de opdracht zeker geslaagd als ik kijk naar de hoeveelheid kennis die ik hier uit gehaald heb.

# Inhoudsopgave

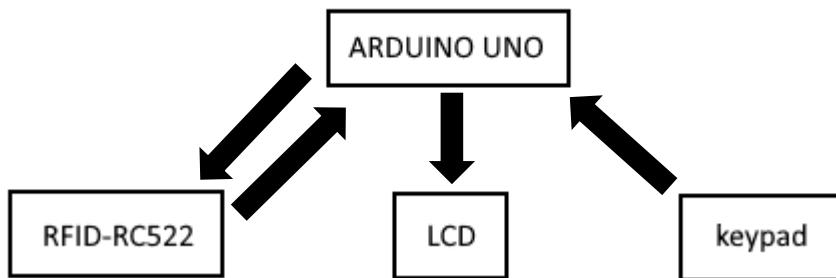
1	Inleiding	1
2	Betaalmodule	2
2.1	Werking	3
2.2	Toetsenbord	4
2.2.1	Werking	4
2.2.2	Aansluitingen	6
2.2.3	Software	6
2.2.3.1	Bibliotheek	6
2.2.3.2	Declaratie	6
2.2.3.3	keypad.getKey()	7
2.2.3.4	keypad.waitForKey()	8
2.3	RFID module	9
2.3.1	Werking	9
2.3.2	Aansluitingen	10
2.3.3	SPI bus	11
2.3.4	RFID badge	11
2.3.4.1	Geheugen	12
2.3.5	Software	13
2.4	I2C-LCD	14
2.4.1	Werking	14
2.4.2	Aansluiting	14
2.5	Software	15
2.5.1	Diagram	15
2.5.2	Uitleg	16
2.6	Omhulsel	22
2.7	Tussenbesluit	24
3	Flipperkast	25
3.1	Constructie	26
3.2	Solenoïdes	27
3.2.1	Werking	27
3.2.2	Aansturing	27
3.2.2.1	MOSFET	28
3.2.2.2	Blusdiode	28

3.2.2.3	Schema	28
3.3	Flippers	30
3.4	Drukknoppen	31
3.4.1	Software	31
3.4.1.1	Schema	32
3.5	Joystick en LCD	33
3.5.1	Werking	33
3.5.2	Aansluitingen	34
3.5.3	Software	34
3.6	Micro SD module	35
3.6.1	Aansluitingen	35
3.6.2	Software	35
3.7	RFID Module	36
3.8	Stappenmotor	37
3.8.1	Aansluitingen	37
3.8.2	Software	37
3.9	Software	38
3.10	Bron	39
4	Besluit	39
4.1	Fouten	39
4.2	To do	40
5	Componentenlijst	41
6	Bronnenlijst	42

# 1 Inleiding

Mijn GIP valt op te delen in 2 stukken, de flipperkast en de betaalmodule. Ik begin bij de betaalmodule, deze is volledig ge-3D-print. Hier wordt een Arduino Uno gebruikt, een LCD, keypad en een RFID-RC522.

Schematische voorstelling:



Figuur 1. Schematische voorstelling betaalmodule.

De flipperkast bestaat uit veel meer onderdelen dan de betaalmodule. Als de speler de knoppen indrukt aan de zijkant, zullen de flippers bewegen met behulp van 2 solenoïdes die met MOSFETS via een Arduino Mega worden aangestuurd. Aan de voorkant is er een LCD en joystick aanwezig. Op de LCD kan je navigeren naar verschillende menu's met behulp van de joystick. Je kan het klassement bekijken die op een SD kaard wordt opgeslagen. Of het aantal Credits bekijken die nog op de RFID-badge staat.



Figuur 2. De flipperkast.



Figuur 3. De betaalmodule.

## 2 Betaalmodule

De betaalmodule wordt gebruikt om “Credits” op een RFID-badge te zetten, dit kan je vergelijken met een bankkaart of de Prizma kaart waarmee je een maaltijd betaalt.

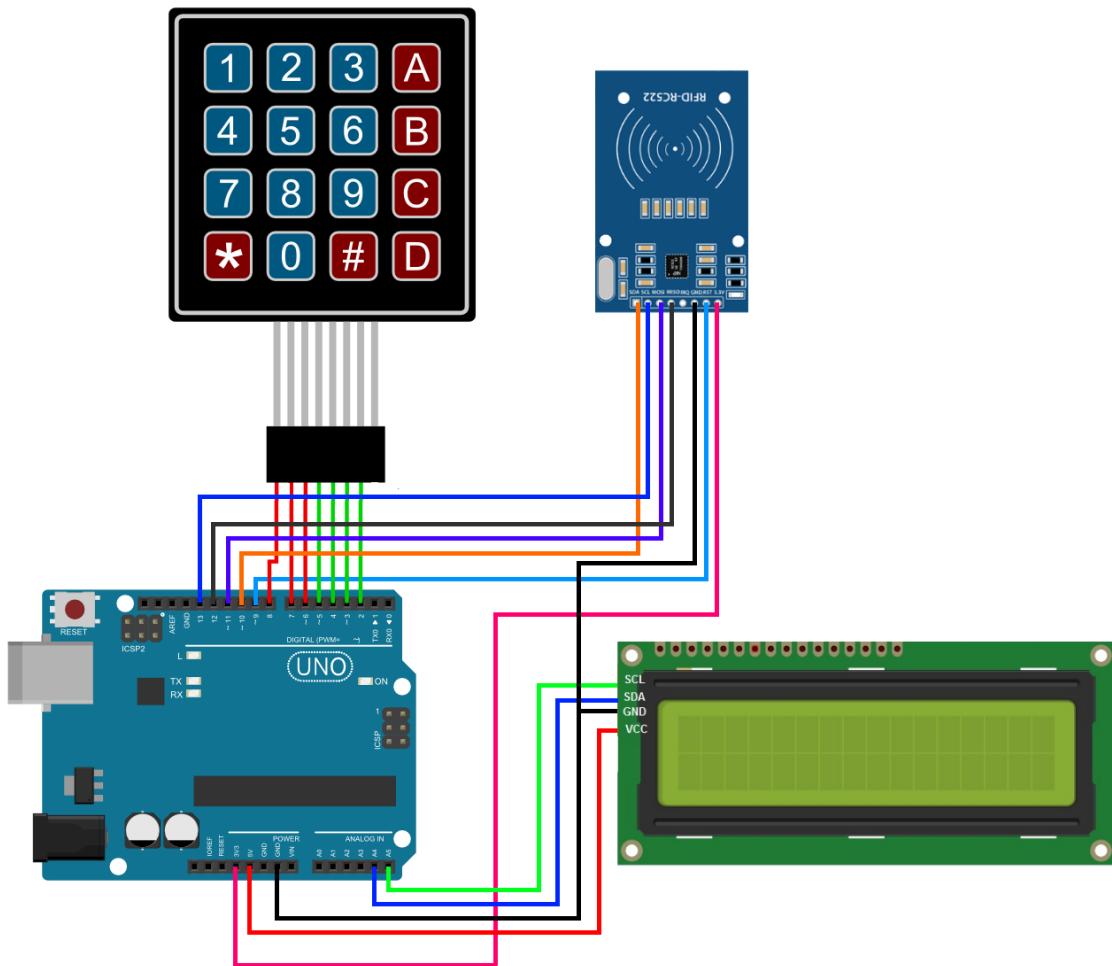
Deze Credits heb je nodig om de flipperkast te kunnen gebruiken.

De badge wordt ingelezen via de RFID-RC552, en via het toetsenbord kan je navigeren op de LCD en ingeven hoeveel credits je op je badge wilt zetten.

De betaalmodule bestaat uit een Arduino Uno, een LCD, een membraan toetsenbord en RFID-lezer. De Arduino Uno had net genoeg digitale ingangen om alle componenten aan te sluiten.



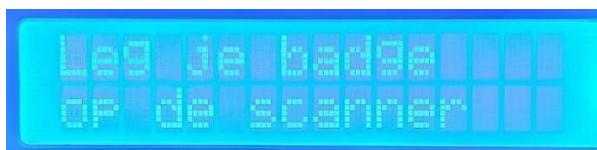
Figuur 4. RFID-badge.



Figuur 5. Verbindingen voor de betaalmodule.

## 2.1 Werking

De betaalmodule wacht tot er een RFID-badge gedetecteerd wordt, op de LCD staat er “leg je badge op de scanner”, zoals in figuur 7. Als die een badge detecteert, zal er “welkom [naam badge]” op de LCD komen, dit komt omdat elke badge een specifieke UID (Unique identifier) heeft (vb: 5D 68 BD 02). Dit is een nummer dat gegarandeerd uniek is voor deze badge. Deze UID wordt ingelezen en vergeleken met UID’s die in het programma zitten. Zo weet het programma welke badge er op de scanner ligt. Het is de bedoeling dat de gebruiker de badge laat liggen tot alles klaar is. Als dit echter niet gebeurt, en de gebruiker neemt de badge van de scanner, komt er een error op het scherm. Daarna kan de gebruiker kiezen uit 2 menu’s, Credits storten of kijken wat het saldo is van de badge (zie figuur 6). Door op \* of # te drukken op het toetsenbord, kom je in het corresponderend menu.



Figuur 7. LCD, passieve toestand.



Figuur 6. LCD, keuzemenu.

Druk je op \*, dan krijg je te zien hoeveel credits op de badge staan, indien je dan weer op # duwt, kom je terug in het keuzemenu. Als je kiest om credits te storten word je gevraagd hoeveel er moeten bijkomen. Door in te geven op het toetsenbord, met een maximum van 999, komt de waarde op het scherm. Je wordt gevraagd om te bevestigen door op \* te duwen. Als het programma de credits succesvol heeft gestort, verschijnt ‘Storting geslaagd’ op het scherm. (zie figuur 9).



Figuur 8. LCD, storting.



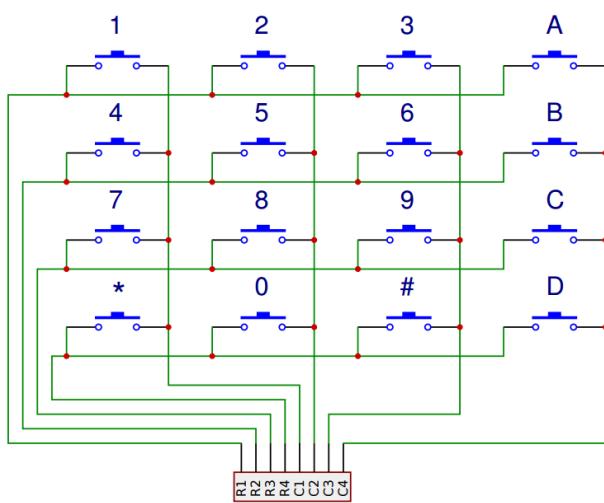
Figuur 9. LCD, storting geslaagd.

Als dit gedaan is, keert het programma terug naar het keuzemenu. Wanneer er plots een badge van de scanner wordt gehaald, dan zal de LCD terug naar het eerste scherm springen. Dan kan een andere persoon zijn badge op de scanner leggen.

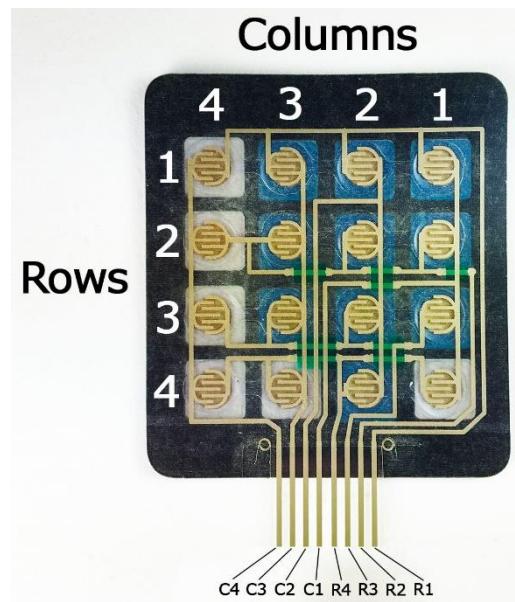
## 2.2 Toetsenbord

### 2.2.1 Werking

De knoppen op het toetsenbord zijn gerangschikt in rijen en kolommen. Onder elke knop zit een membraanschakelaar. Elke schakelaar in een rij is verbonden met de andere schakelaars in die rij, en elke schakelaar in een kolom is ook verbonden met elkaar (zie figuur 10 en figuur 11).

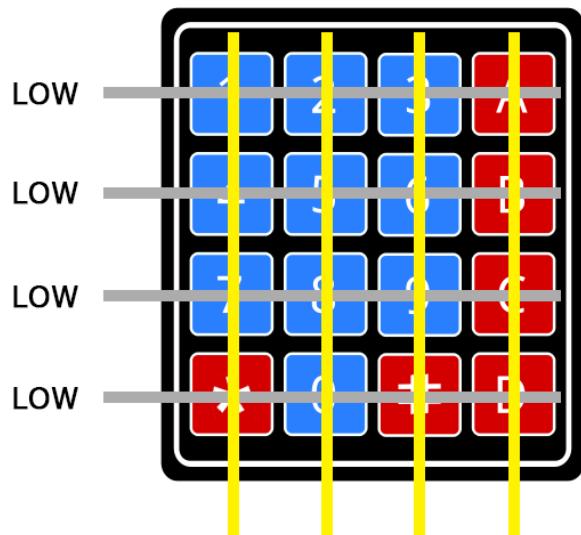


Figuur 10. Schema toetsenbord.



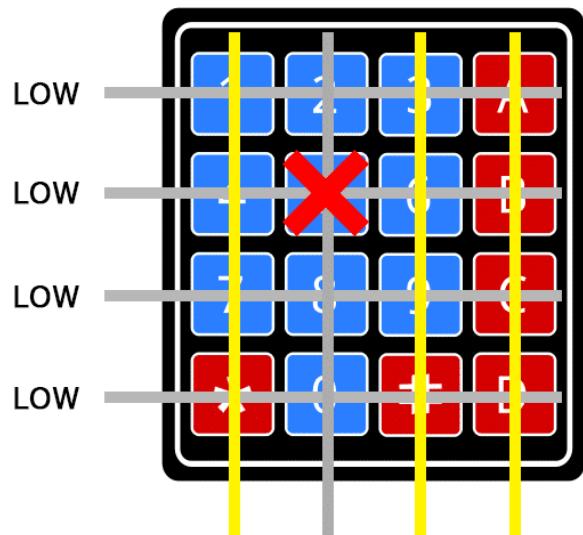
Figuur 11. Binnenkant toetsenbord.

Dit toetsenbord heeft 16 knoppen en toch maar 8 aansluitingen. Het bepalen van welke knop er ingeduwd wordt, gebeurt in het programma. Dit gebeurt in 4 stappen. Eerst is elke kolom pin hoog, en elke rij pin laag. (Zie figuur 12). Wanneer er een knop wordt ingedrukt, wordt de kolom pin daarbij laag getrokken (zie figuur 13). De kolom van de ingedrukte knop is nu bepaald, de Arduino weet nu wel nog niet welke knop het precies is, het moet nu enkel de rij nog bepalen. Dit gebeurt door elke kolom pin nu laag te trekken en 1 voor 1 de rij pinnen kort hoog te maken. (Zie figuur 14) De Arduino leest de kolom pinnen in en wanneer die hoog komt (zie figuur 15) weet die nu ook in welke kolom de knop zit. Met die kennis beslist de Arduino welke knop je net ingedrukt hebt. Welk cijfer of letter waar staat op het toetsenbord, moet je in het programma declareren.



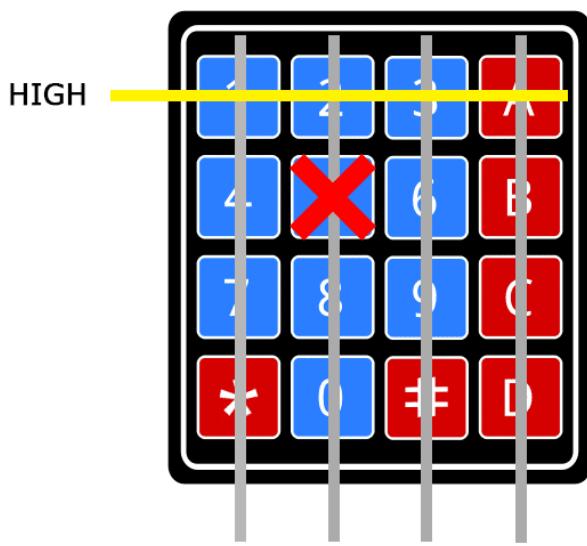
**HIGH HIGH HIGH HIGH**

Figuur 13. Geen knop ingedrukt.



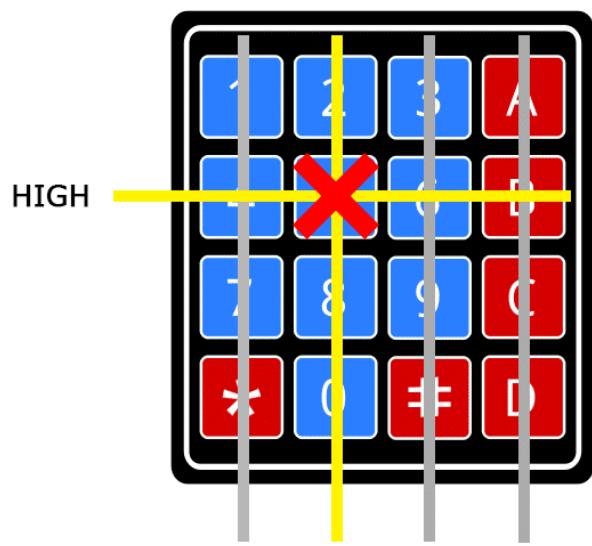
**HIGH LOW HIGH HIGH**

Figuur 12. "5" ingedrukt en kolom pin laag.



**LOW LOW LOW LOW**

Figuur 15. . Kolom pinnen laag, rij pinnen 1 voor 1 hoog.



**LOW HIGH LOW LOW**

Figuur 14. Rij pin trekt kolom pin hoog.

## 2.2.2 Aansluitingen

Het aansluiten van dit membraantoetsenbord aan een Arduino is simpel, er zijn 8 pinnen die moeten aangesloten zijn. Deze kunnen rechtsreeks aan de digitale ingangen van de Arduino verbonden worden. Op het schema (figuur 5) kan je zien dat er echter maar 7 verbindingen worden gemaakt. Dit komt omdat de kolom met de knoppen A, B, C en D niet gebruikt worden. En die hoef ik dan ook niet aan te sluiten.

## 2.2.3 Software

### 2.2.3.1 Bibliotheek

Het programmeren van een Arduino toetsenbord is zeer gemakkelijk. Door gebruik te maken van een bibliotheek hoef je zelf niet veel te programmeren maar kan je een functie oproepen die standaard in de bibliotheek zit. Voor vele sensoren en actuatoren kan je een bibliotheek vinden en gebruiken om het programma makkelijker te maken. Alle informatie over de bibliotheek kan je hier vinden: <https://playground.arduino.cc/Code/Keypad/>

### 2.2.3.2 Declaratie

Je begint in het programma met de nodige variabelen te declareren. Ook wordt er een matrix aangemaakt met de karakters van het toetsenbord. Hier worden de karakters A, B, C en D ook niet gedeclareerd, ik heb die toch niet nodig.

```
const int ROW_NUM = 4;
const int COLUMN_NUM = 3;
char keys[ROW_NUM][COLUMN_NUM] =
{
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};

byte pin_rows[ROW_NUM] = {8, 7, 6, 5}; // de rijen van het keyboard
byte pin_column[COLUMN_NUM] = {4, 3, 2}; // de kolommen van het keyboard
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
```

Daarna maak je een “keypad” object aan, dit gebeurt hierboven. Als je meerdere toetsenborden gebruikt voor je project dan maak je meerdere objecten aan en wijs je de correcte pinnen van de Arduino toe aan het juiste object. Voor het oproepen van een functie die in de bibliotheek zit, plaats je altijd het object ervoor. Zo weet de Arduino over welk toetsenbord je het hebt. Bij mijn GIP kan er geen verwarring zijn want ik gebruik maar 1.

### 2.2.3.3 keypad.getKey()

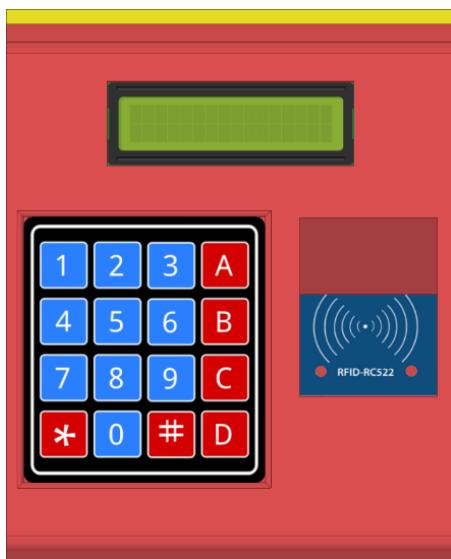
Om te weten welke toets er op een bepaald moment is ingedrukt, roep je de functie “keypad.getKey()” uit de bibliotheek op, deze geeft een karakter van de matrix terug. Wanneer er geen knop wordt ingedrukt op het moment dat je de functie oproept, dan keert de “getKey” functie ‘NO\_KEY’ terug. Dit karakter steek ik in een variabele die ik “keypressed” noem. Nu gaat het programma door 3 ‘if’ en ‘else if’ statements. Deze controleren of de variabele “keypressed” een ‘#’ of een ‘\*’ is, dan gaat het programma verder in de correcte functies. Als er echter op een andere knop gedrukt is, bv: 1 of 5, dan zal het programma in de laatste if else functie komen. Er komt een tekst op het scherm die de gebruiker laat weten dat je op ‘#’ of ‘\*’ moet drukken. En dan gaat het programma verder.

```
char keypressed = keypad.getKey();
if (keypressed == '#')
{
    KeuzeMenu = false;
    STORT();
}
else if (keypressed == '*') // * voor naar balans menu te gaan.
{
    KeuzeMenu = false;
    BALANS();
}
else if (keypressed != NO_KEY)
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Druk op # of *"));
    delay(1000);
    lcd.clear();
    lcd.print(F("#: Stort Credits"));
    lcd.setCursor(0, 1);
    lcd.print(F("*: Bekijk balans"));
}
```

#### 2.2.3.4 keypad.waitForKey()

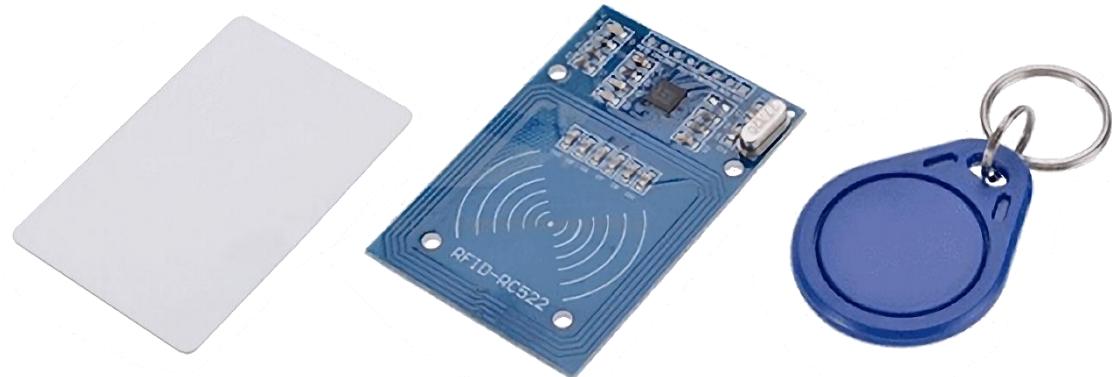
Soms is het nodig dat het programma moet wachten op een toets die ingedrukt wordt, dit kan gemakkelijk gedaan worden met de ‘waitForKey’ functie. Dit stopt het volledige programma en wacht tot er een toets wordt ingedrukt. Als je ook wilt weten welke toets dit is dan ken je die toe aan een variabele. In het voorbeeld zit het programma in een while lus en telkens wanneer er op een toets wordt gedrukt, dan wordt deze gecontroleerd, als het een ‘#’ is dan gaat het programma uit de while lus en verder door het programma.

```
char KeyPressed = ' ';
while (KeyPressed != '#')
{
    KeyPressed = keypad.waitForKey();
}
KeuzeMenu = true;
KEUZEMENU();
```



## 2.3 RFID module

Gebruik maken van een RFID kit is essentieel om te weten wie speelt op de flipperkast. Zo kan er een verdienmodel aan gekoppeld worden. Ik heb voor de RC522 gekozen omdat deze zeer bekend is. Het is goedkoop, vaak minder dan 5 euro, het is betrouwbaar en heeft een laag verbruik. Er zijn 2 belangrijke onderdelen, namelijk de RC522: een sensor die een RFID chip kan uitlezen en naar schrijven en de RFID chip, die zit meestal in een kaart of badge. (Zie figuur 16)

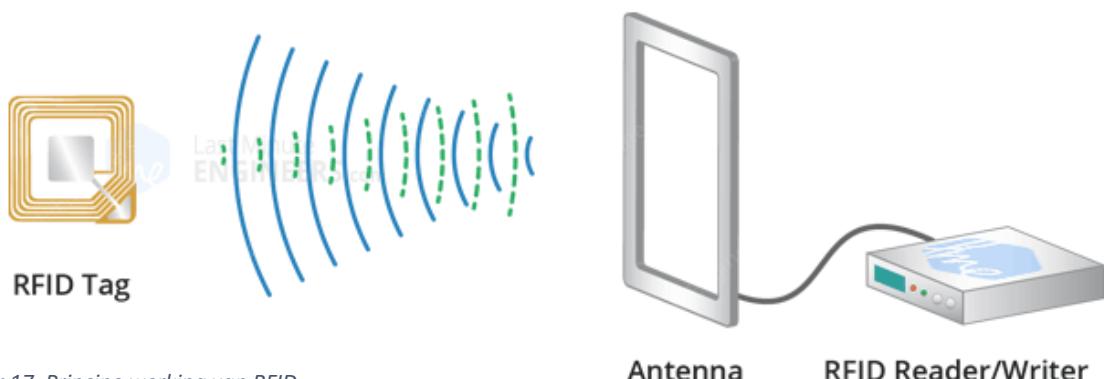


Figuur 16. Een volledige RFID-RC522 kit met kaart en badge.

RFID staat voor: Radio Frequency Identification. Deze technologie wordt vaak toegepast in de praktijk: in grote bedrijven wordt het gebruikt om te weten wanneer arbeiders beginnen aan hun shift of wanneer ze naar huis gaan. Ook de Prizma eetkaart werkt met dit systeem. Dit kan ook gebruikt worden om in een winkel automatisch af te rekenen.

### 2.3.1 Werking

De RC522 module bestaat uit een radiofrequentiemodule en een antenne die een hoogfrequent, elektromagnetisch veld genereert. De RFID chip is meestal een passief apparaat, wat betekent dat er geen batterij in zit. In plaats daarvan bevat het een microchip die informatie opslaat en verwerkt, en een antenne om het signaal te ontvangen en te verzenden.

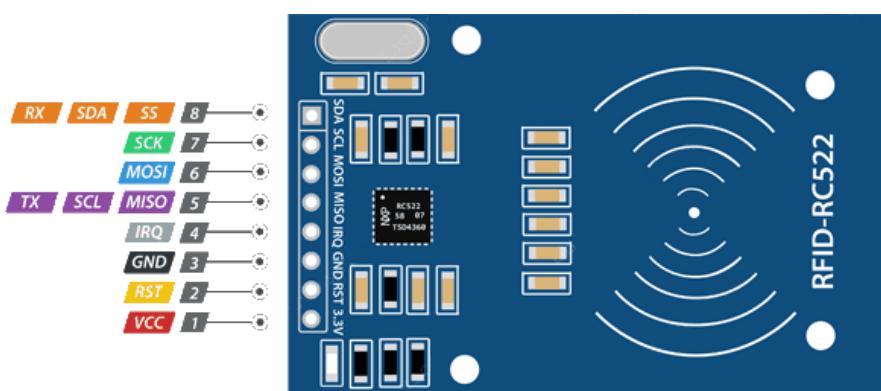


Figuur 17. Principe werking van RFID.

Om de informatie die op een badge opgeslagen is te lezen, wordt deze in de buurt van de RC522 geplaatst (hoeft geen direct contact te hebben). De RC522 genereert een elektromagnetisch veld waardoor elektronen door de antenne van de RFID chip bewegen en vervolgens de chip van stroom voorzien. De aangedreven RFID chip in de tag reageert vervolgens door de opgeslagen informatie terug te sturen naar de lezer in de vorm van een ander radiosignaal. Dit wordt ‘backscatter’ genoemd. De verandering in het elektromagnetisch veld wordt gedetecteerd en opgevangen door de RC522, die de gegevens vervolgens naar een computer of microcontroller stuurt, in dit geval een Arduino Uno. De communicatie tussen Arduino en de RC522 gebeurt via de SPI bus. Door opnieuw gebruik te maken van een bibliotheek kan dit vlot en simpel verlopen.

## 2.3.2 Aansluitingen

De RC522 werkt op een spanning van 3.3V dus we beginnen met de VCC pin aan de 3.3V te verbinden en de GND pin aan de GND van de Arduino. De RST pin kan aangesloten worden op elke digitale ingang van de Arduino, hier is die op poort 9 aangesloten. De RST pin is de reset pin en wordt door de bibliotheek gebruikt, zelf hoef je niets te doen. De rest van de pinnen zijn voor de SPI communicatie, deze worden best verbonden met hardware SPI pinnen van de microcontroller. Deze zijn het snelst. Elk type Arduino bord heeft andere pinnummers daarvoor. De Arduino Uno gebruikt pin 11 voor de MOSI (Master Out, Slave In), pin 12 voor MISO (Master In, Slave Out), pin 13 voor SCK (Serial Clock) en pin 10 voor de CS (Chip Select). Hoe deze werken wordt later uitgelegd.

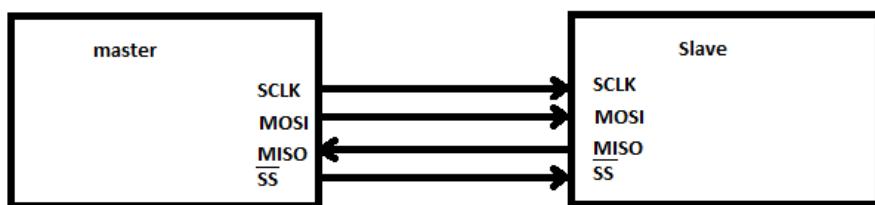


Figuur 18. Aansluitingen RC522

### 2.3.3 SPI bus

De RC522 maakt gebruik van de SPI bus, dit is een welbekend en wereldwijd gebruikt communicatie protocol. De start van de communicatie gebeurt door de masterchip. Het is mogelijk om meerdere slaves te hebben, maar ze moeten dan ieder een aparte chip select hebben. De master is meestal een microcontroller, en de slaves zullen sensoren of actuatoren zoals een LCD zijn. Er zijn altijd vier verbindingen voor communicatie nodig, namelijk:

- MISO: Master In Slave Out, de data van de slave wordt hierover naar de master gestuurd.
- MOSI: Master Out Slave In, op deze lijn wordt de data verzonden van master naar slave.
- SCLK: Seriële Clock, wordt geleverd door de master.
- SS: Slave Select, Deze lijn wordt actief laag aangestuurd. De lijn voor de geselecteerde slave zal laag zijn. Wanneer de communicatie met de slave gedaan is, zal de master de lijn hoog maken.



Figuur 19. Schematische voorstelling SPI bus.

### 2.3.4 RFID badge

Ik heb beslist om een badge te gebruiken en geen kaart, de badge is wat kleiner en past op de RC522, de kaart heeft echter meer ruimte nodig. Deze zou niet op de betaalmodule passen. In de bibliotheek die ik gebruik voor de RC522 zit een voorbeeld-programma genaamd “Dumpinfo”. Dit programma is zeer handig, het schrijft niets van informatie naar de badge maar het toont alle informatie die in de badge opgeslagen is op de pc waar de Arduino is op aangesloten. Dit is ook het merk, type, UID en hoeveel opslag er is. De badges die ik gebruik, hebben 1 kilobyte aan data en de UID is “20 C3 93 5E” (zie figuur 20).

```
Firmware Version: 0x92 = v2.0
Scan PICC to see UID, SAK, type, and data blocks...
Card UID: 20 C3 93 5E
Card SAK: 08
PICC type: MIFARE 1KB
```

Figuur 20. Gegevens RFID-badge.

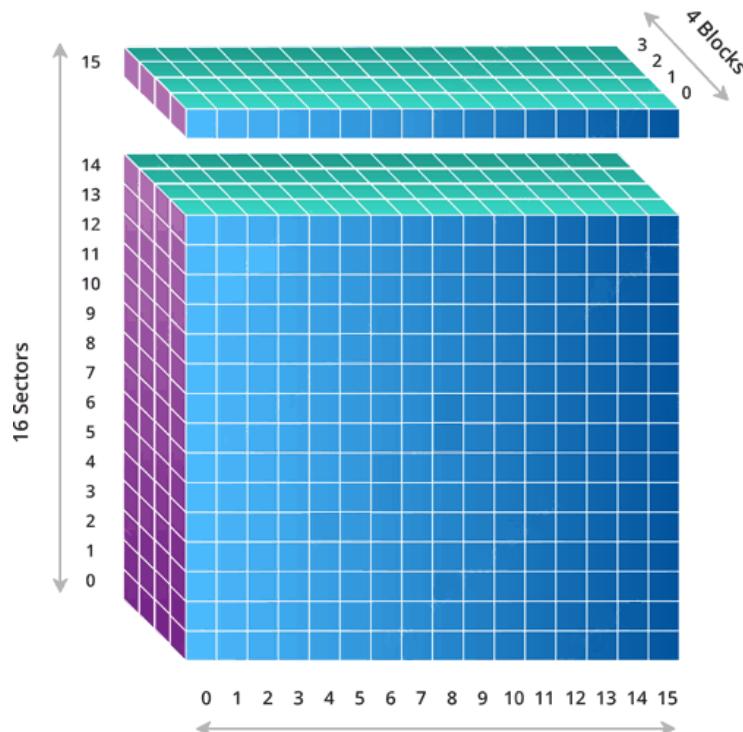
### 2.3.4.1 Geheugen

De RFID badge heeft een geheugen van 1kB, ook dit kunnen we perfect zien via het voorbeeld programma. Dit geheugen is georganiseerd in 16 **sectoren** (van 0 tot 15). Elke sector is verder verdeeld in 4 **blokken**, elke blok kan 16 bytes aan **data** opslaan.

Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AccessBits
15	63	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[ 0 0 1 ]	
	62	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
14	59	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]	
	58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	57	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	56	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
13	55	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]	
	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
	52	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[ 0 0 0 ]	
12	51	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	[ 0 0 1 ]	

Figuur 21. Geheugen RFID-badge.

Dus als we het even uitrekenen, 16 bytes aan data per blok, en 4 blokken per sector, voor 16 sectoren:  $16 \times 4 \times 16 = 1024$  bytes = 1kB (zie figuur 22). Dit geheugen kunnen we gebruiken om naar te schrijven en van te lezen, hier zetten we de hoeveelheid Credits op. Er moet enkel afgesproken worden op welke plaats we die data plaatsen. Je kan kiezen waar je dit doet, ik koos voor block 60.



Figuur 22. Schematische voorstelling 1kB.

## 2.3.5 Software

Om de RC522 te programmeren gebruik ik de MFRC522 bibliotheek, gemaakt door ‘miguelbalboa’ op GitHub. Ook de SPI bus bibliotheek van Arduino wordt gebruikt. Om te starten maak je een object aan die je later terug kan oproepen. Dit is mfrc522 in het programma. Ook worden de SDA (pin 10) en RST (pin 9) pin gedeclareerd.

```
MFRC522 mfrc522(10, 9);
```

In de setup wordt dit object geïnitialiseerd. En dan kan het gebruikt worden. In de MFRC522 bibliotheek zitten verschillende voorbeeldprogramma’s die ik gebruikt heb en waarvan ik onderdelen heb gekopieerd. Zoals dit stuk:

```
while (KeuzeMenu == true)
{
    MFRC522::StatusCode status;
    MFRC522::MIFARE_Key key;
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
    Block, &key, &(mfrc522.uid));

    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("Authentication failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        CorrectRFID = false;
        RFIDSCAN();
    }
}
```

Dit stuk wordt gebruikt terwijl de badge op de scanner ligt en wacht op de gebruiker die kiest of er Credits moeten bijkomen of het saldo wil bekijken. Het programma zit in een while lus en blijft dit doorlopen. De RC522 stuurt een signaal met de key (deze is voor elke badge gelijk) naar de badge en kijkt of die een reactie terug krijgt. Zo niet, dan zal de status niet gelijk zijn aan ‘STATUS\_OK’ en gaat het programma terug naar de RFIDSCAN functie. De RFIDSCAN functie wacht op een nieuwe badge, leest deze in en bepaalt welke badge het is aan de hand van de UID. Alle UID’s die in het systeem zijn gedefinieerd, worden in het begin van het programma in een array gestoken, en ook alle namen van de badges.

```
String AllPlayers[13] = {"Admin" , "Blauw" , ... , "Wit"};
String UIDtags[13] = {" 23 AA E9 1B" , " 5D 68 BD 02",... , " D7 4B 21 03"};
```

Er wordt gekeken op welke plaats de UID staat in de rij. De naam van de badge staat op dezelfde plaats in de andere array. Als de UID niet in het systeem zit, dan komt er een error op de LCD, en het programma herhaalt zich.

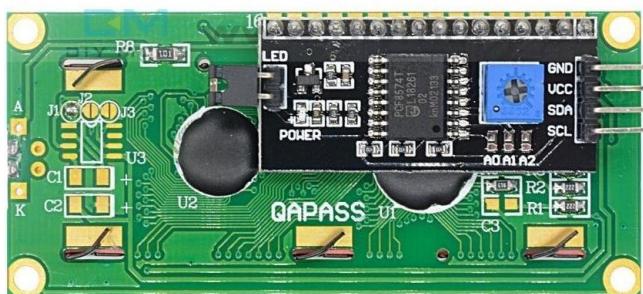
De volledige RFIDSCAN functie staat op pagina 16 en 17.

## 2.4 I2C-LCD

De LCD is cruciaal, hier komt alle nodige informatie op. De LCD is relatief gemakkelijk om aan te sturen omdat we dit al vroeg geleerd hadden op school. Ook hier gebruik ik weer een bibliotheek.

### 2.4.1 Werking

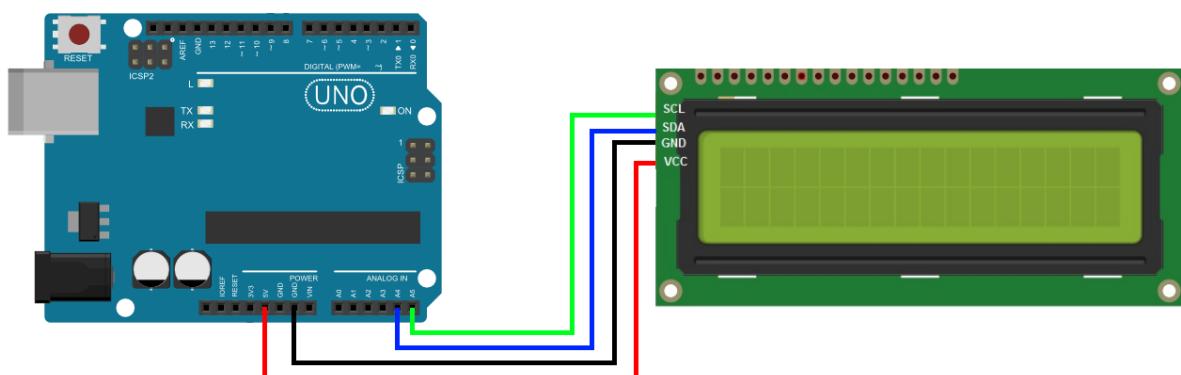
Dit type LCD is ideaal voor het weergeven van tekst en cijfers, vandaar dat het ook ‘character LCD’ wordt genoemd. De LCD die ik gebruik, heeft achteraan een I2C module gemonteerd. Deze module is voorzien van een PCF8574-chip (voor I2C-communicatie) en een potentiometer om de led-achtergrondverlichting aan te passen. Het voordeel van een I2C LCD is dat de bedrading heel eenvoudig is. Er zijn slechts 4 pinnen nodig om het LCD-scherm te bedienen, 2 data pinnen en de VCC en GND.



Figuur 23. LCD met I2C module gemonteerd.

### 2.4.2 Aansluiting

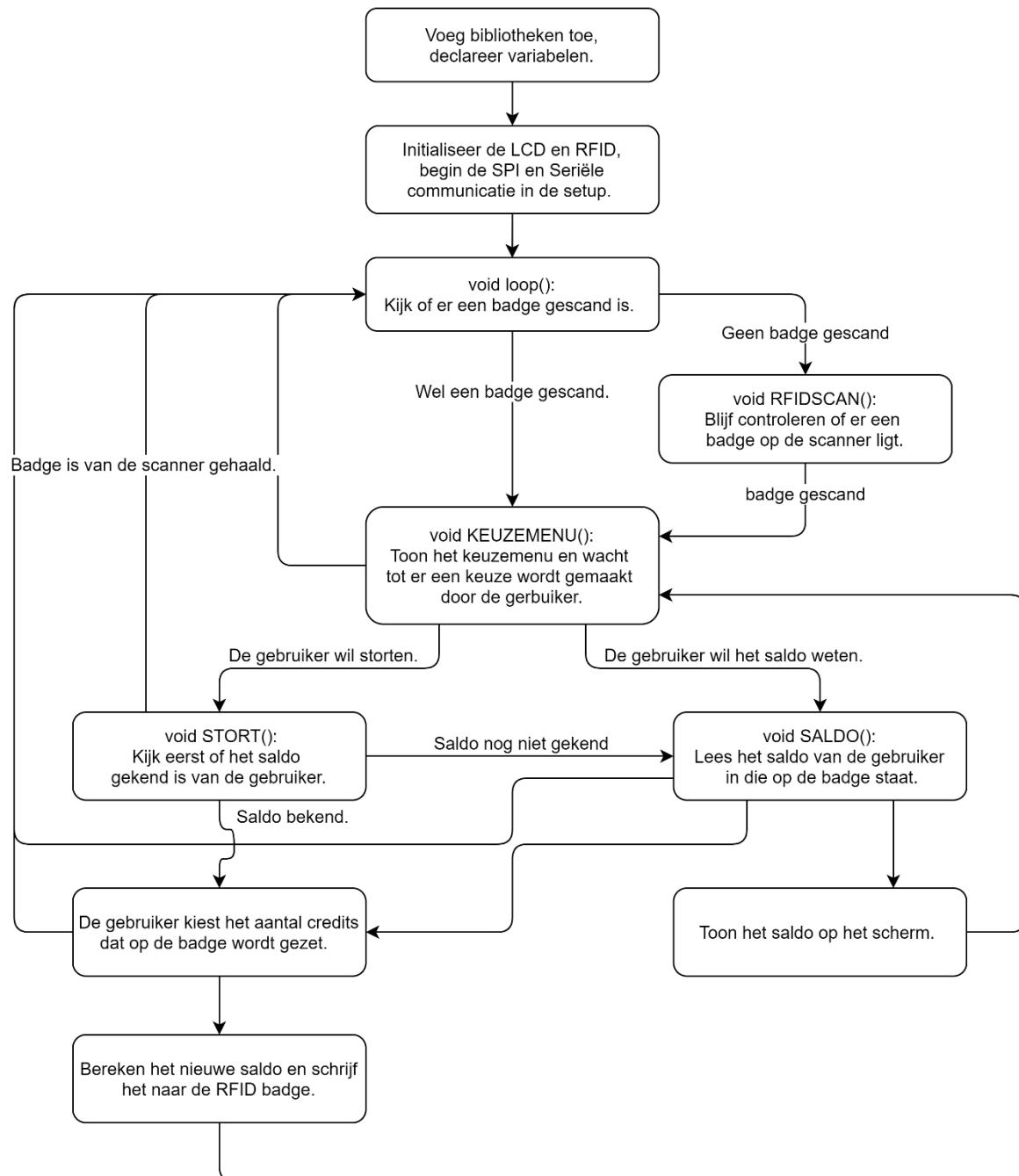
Zoals eerder gezegd is dankzij de I2C module maar 4 aansluitingen nodig i.p.v. 11. De 2 datapinnen (SDA en SCL) worden respectievelijk aangesloten op de A4 en A5 pin van de Arduino Uno. Zoals te zien op figuur 24.



Figuur 24. Aansluitingen voor LCD.

## 2.5 Software

### 2.5.1 Diagram



## 2.5.2 Uitleg

Het programma start met het initialiseren van bibliotheken, declaratie van de nodige variabelen, het aanmaken van de LCD en RFID objecten. Daarna volgt de setup, deze wordt maar 1 maal doorlopen in het programma. Hier begint de seriële communicatie en de SPI bus. De LCD en RFID worden hier geïnitialiseerd. Daarna komt het programma in de loop. Dan wordt er gecontroleerd of de variabele ‘CorrectRFID’ true of false is. Deze variabele houdt bij of er een badge met een UID die in het geheugen zit, op de scanner ligt. Dit zal de eerste keer altijd false zijn en dus naar de RFIDSCAN functie gaan.

```
void loop()
{
    Serial.println(F("Nu in Void Loop"));
    if (CorrectRFID == false) RFIDSCAN();
    if (CorrectRFID == true) KEUZEMENU();
}
```

Het programma komt in de RFIDSCAN functie. Hier wordt eerst de uitleg op de LCD gezet. Daarna blijft de Arduino de while lus doorlopen. Deze while lus wacht tot er een nieuwe badge op de scanner ligt, dan wordt de lokale ‘card’ variabele true en gaat het programma verder.

```
void RFIDSCAN()
{
    Serial.println(F("Nu in Void RFIDSCAN"));
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Leg je badge"));
    lcd.setCursor(0, 1);
    lcd.print(F("op de scanner"));
    bool Card = false; // lokale var voor check of er een kaard
    while (Card == false) { // aanwezig is, dit houd de code tegen
        if (mfrc522.PICC_IsNewCardPresent()) { // als er geen badge op
            if (mfrc522.PICC_ReadCardSerial()) { // de scanner ligt.
                Card = true; // Zo wordt niet heel de RFIDSCAN()
            } // doorlopen. Dit bakje staat dan
        } // ook vaak inactief.
    }
}
```

De badge is nu nog niet ingelezen en gecontroleerd. Dit gebeurt nu. Door gebruik te maken van de eerder genoemde bibliotheek, is het inlezen van de UID niet moeilijk. Deze UID wordt dan in een for lus gecontroleerd of die overeenkomt met een UID die gekend is in de ‘UIDtags’ variabele. Met de UID kan de naam van de gebruiker ook bepaald worden. De ‘CorrectRFID’ variabele wordt true gemaakt en het programma gaat verder naar het keuzemenu. Als dit echter niet het geval is en de badge een UID bezit die niet gekend is, dan zal de LCD een error geven en naar de loop terugkeren.

```

while (CorrectRFID == false)
{
    tag = "";
    Serial.println(tag);
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
        tag.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
        tag.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
    tag.toUpperCase();
    Serial.println(tag);
    for (byte i = 0; i < 12; i++)
    {
        if (tag == UIDtags[i])           // Als 1 van de UIDtags gelijk
        {
            CurrentPlayer = AllPlayers[i]; // is aan de ingelezen tag
            CorrectRFID = true;          // Zet CorrectRFID true
            lcd.clear();                // En verwelkom de speler.
            lcd.print(F("Welkom "));
            lcd.print(CurrentPlayer);
            Is_Ingelezen = false;
            delay(500);
            KeuzeMenu = true;           // Zo geraakt het programma in
            KEUZEMENU();               // KEUZEMENU()
        }
    }
    if (CorrectRFID == false)
    {
        Serial.println(F("Deze tag zit niet in de database"));
        lcd.clear();
        lcd.print(F("Afgewezen"));
        delay(500);
        return;
    }
}
}

```

Indien de RFID badge gekend is, weet het programma nu wie de gebruiker is. Het keuzemenu wordt weergegeven op de LCD en wacht nu op de keuze van de gebruiker. De Arduino blijft in een while lus controleren met if en else if functies of de gebruiker op een toets drukt. Als dit een '#' is, dan zal het programma in de STORT functie gaan. Als het een '\*' is, zal het programma in de SALDO functie gaan. Als het echter een andere knop is, zal er een melding op LCD scherm komen, die laat weten dat je op '#' of '\*' moet drukken. Ook zit er nog een blokje code in de while lus, die blijft controleren of er nog een badge op de reader ligt. Ditzelfde stukje komt voor in de keuzemenu, saldo, stort en inlees functie. Als er nog een badge aanwezig is, gaat het programma gewoon verder, maar als er geen meer is, gaat het terug naar de RFIDSCAN functie.

```

while (KeuzeMenu == true)
{
    MFRC522::StatusCode status; //check of tag nog aanwezig is, zo niet,
    MFRC522::MIFARE_Key key; //zal programma terug in RFIDSCAN() gaan.
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
    Block, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        Serial.print(F("Authentication failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        CorrectRFID = false;
        RFIDSCAN();
    }

    char keypressed = keypad.getKey();
    if (keypressed == '#')
    {
        KeuzeMenu = false;
        STORT();
    }
    else if (keypressed == '*') // * voor naar saldo menu te gaan.
    {
        KeuzeMenu = false;
        SALDO();
    }
    else if (keypressed != NO_KEY)
    {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(F("Druk op # of *"));
        delay(1000);
        lcd.clear();
        lcd.print(F("#: Stort Credits"));
        lcd.setCursor(0, 1);
        lcd.print(F("*: Bekijk saldo"));
    }
}
}

```

Als de gebruiker beslist heeft om het saldo te bekijken, zal het programma eerst kijken of het saldo al niet eerder is ingelezen. Zo hoeft het niet een 2de keer ingelezen te worden als het niet nodig is. Als het saldo echter nog niet bekend is, zal het programma in de INLEES functie gaan. Er wordt ook een parameter meegegeven die aangeeft dat op het einde van de inlees functie het programma moet terugkeren naar de saldo functie. Dit is nodig omdat de STORT functie dit ook gebruikt. De inlees functie start met het resetten van de variabele die het aantal Credits bijhoudt, daarna wordt de sleutel aangemaakt die nodig is om te lezen en te schrijven van en naar een RFID badge. Deze sleutel kan je aanpassen en zo kan je het beveiligen. Standaard is deze sleutel: FFFFFFFFFFFFFh. Daarna wordt er gekeken of de badge klaar is om uitgelezen te worden. Als dit niet het geval is, zal er naar de seriële monitor een error gestuurd worden. En gaat het programma terug naar de RFIDSCAN functie om de badge opnieuw te scannen. Als alles in orde is en de data van de badge ingelezen is naar een variabele, dan kan deze gebruikt worden. Er wordt teruggekeerd naar de saldo functie en het aantal Credits wordt op het scherm aangegeven. Als de gebruiker nu op ‘#’ drukt, zal er teruggekeerd worden naar het keuzemenu.

```

void SALDO()
{
    Serial.println(F("Nu in Void SALDO"));
    if (Is_Ingelezen == false) INLEZEN(false);
    Serial.println(NCredits); // Credits in,
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Credits: "));
    lcd.print(NCredits);
    lcd.setCursor(0, 2);
    lcd.print(F("#: Keer terug"));
    char KeyPressed = ' ';
    while (KeyPressed != '#')
    {
        // code dat controleert of er nog een badge aanwezig is.
    }
    KeuzeMenu = true;
    loop();
}

```

```

void INLEZEN(bool Stort)
{
    NCredits = 0; // Reset Credits var
    Serial.println(F("Nu in Void INLEZEN"));
    MFRC522::MIFARE_Key key; // Maak key klaar
    for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
    MFRC522::StatusCode status;
    status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
    Block, &key, &(mfrc522.uid));
    if (status != MFRC522::STATUS_OK) {
        lcd.clear();
        lcd.setCursor(5, 0);
        lcd.print(F("Error 404"));
        Serial.print(F("Authentication failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        CorrectRFID = false;
        return;
    }
    status = mfrc522.MIFARE_Read(Block, ReadBuffer, &BufferSize);
    if (status != MFRC522::STATUS_OK) {
        lcd.clear();
        lcd.setCursor(5, 0);
        lcd.print(F("Error 400"));
        Serial.print(F("Reading failed: "));
        Serial.println(mfrc522.GetStatusCodeName(status));
        CorrectRFID = false;
        return;
    }
    Is_Ingelezen = true; // Hierdoor lees je maar 1 keer in
    for (byte i = 0; i < 16; i++) {
        if (ReadBuffer[i] != 32) {
            Serial.println(ReadBuffer[i]);
            NCredits = NCredits + ReadBuffer[i];
        }
    }
    memset(ReadBuffer, 0, sizeof(ReadBuffer)); // zet de buffer terug leeg
    Serial.print("Bool Stort = "); // voor volgende keer.
    Serial.println(Stort);
    if (Stort == false) SALDO();
    if (Stort == true) STORT();
}

```

Als de gebruiker echter gekozen heeft om Credits te storten op de badge, dan wordt er ook eerst gekeken of het saldo dat op de badge staat, bekend is. Dit is nodig zodat het saldo kan opgeteld worden. Eens het saldo bekend is, wordt er aan de gebruiker gevraagd om het gewenste aantal credits in te geven. Het maximum is 999. Als het gewenste getal is ingegeven, wordt dit opgeteld bij het aantal Credits die al op de badge stonden. Dan wordt er gevraagd om te bevestigen. Na bevestiging wordt het decimale getal omgezet naar hexadecimale getallen die naar de badge kunnen geschreven worden.

```

void STORT()
{
    Serial.println(F("Nu in Void STORT"));
    if (Is_Ingelezen == false) INLEZEN(true);
    memset(WriteBuffer, 0, sizeof(WriteBuffer));
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Bedrag: "));
    lcd.setCursor(0, 1);
    lcd.print(F("Bevestig met *"));
    char PressedKeys[3] = {" "};
    byte n = 0;
    bool AsteriskPressed = false;           // check voor '*',
    while (AsteriskPressed == false) {       // bevestig de storting.
        char PressedKey = ' ';
        PressedKey = keypad.waitForKey();
        if (PressedKey == '*') {
            AsteriskPressed = true;
        }
        else if (n < 3 && PressedKey != '#') {
            PressedKeys[n] = PressedKey;
            Serial.println(PressedKeys[n]);
            lcd.setCursor(8 + n, 0);
            lcd.print(PressedKey);
            n = n + 1;
        }
    }
    if (n == 0)
    {
        lcd.clear();
        lcd.print(F(" Annuleren? "));
        lcd.setCursor(0, 1);
        lcd.print(F(" Druk # "));
        char KeyPressed = ' ';
        while (KeyPressed != '#') {
            KeyPressed = keypad.waitForKey();
        }
        KeuzeMenu = true;
        KEUZEMENU();
    }
    NCredits = NCredits + atoi(&PressedKeys[0]); //bereken totaal credits
    bool bevestigd = false;
    while (bevestigd == false) {
        lcd.clear();                                // Nog éénmaal bevestigen.
        lcd.print(F("Saldo:"));
        lcd.setCursor(7, 0);
        lcd.print(NCredits);
        lcd.setCursor(0, 1);
    }
}

```

```

lcd.print(F("Bevestig met *"));
char KeyPressed = ' ';
KeyPressed = keypad.waitForKey();
if (KeyPressed != '*')
{
    lcd.clear();
    lcd.print(F(" Annuleren? "));
    lcd.setCursor(0, 1);
    lcd.print(F(" Druk # "));
    char KeyPressed = ' ';
    KeyPressed = keypad.waitForKey();
    delay(100);
    if (KeyPressed == '#')
    {
        Is_Ingelezen = false;
        KeuzeMenu = true;
        KEUZEMENU();
    }
}
else if (KeyPressed == '*')
{
    bevestigd = true;
}
}

Is_Ingelezen = false;

for (byte i; i < 16; i++)
{
    if (NCredits > 255) {
        WriteBuffer[i] = 255;
        NCredits = NCredits - 255;
    }
    else if (NCredits < 255)
    {
        WriteBuffer[i] = NCredits;
        NCredits = 0;
    }
    else if (NCredits = 0)
    {
        WriteBuffer[i] = 0;
    }
}

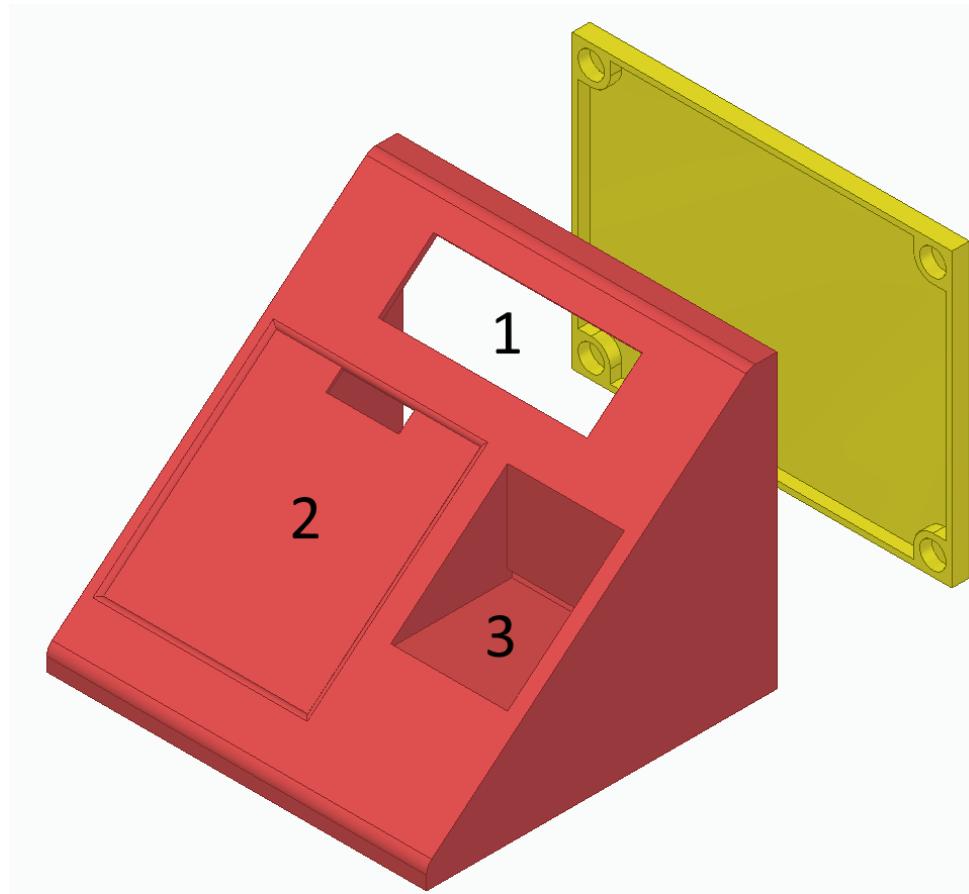
// hier komt code die key klaarmaakt en controleert of de badge klaar
// is om data te ontvangen. Bij een error, ga terug naar RFIDSCAN

status = mfrc522.MIFARE_Write(Block, WriteBuffer, 16);
// code die bij een error dit aangeeft op de LCD/seriele monitor
if (status == MFRC522::STATUS_OK) {
    lcd.clear();
    lcd.setCursor(4, 0);
    lcd.print(F("Startende"));
    lcd.setCursor(4, 1);
    lcd.print(F("geslaagd"));
    delay(2000);
    KeuzeMenu = true;
    KEUZEMENU();
}
}

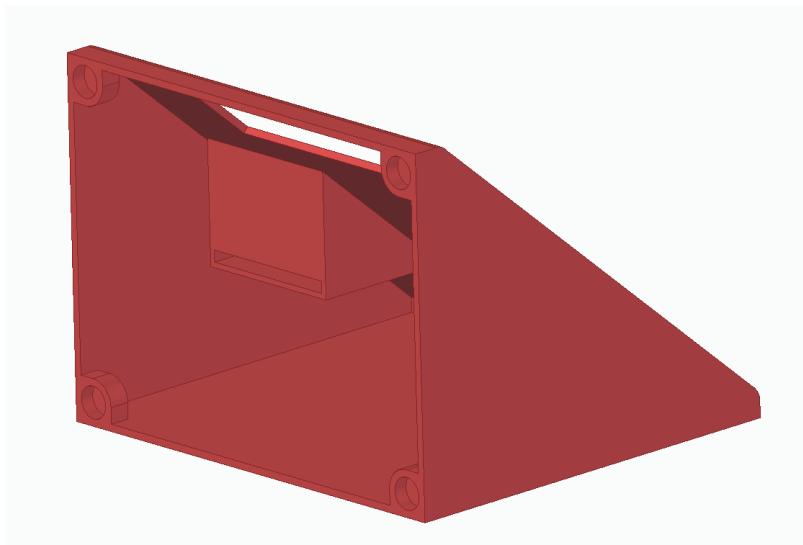
```

## 2.6 Omhulsel

Het omhulsel bestaat uit 2 delen, het stuk waarin alle componenten en de Arduino zich bevinden. En het 2<sup>de</sup> stuk, dat alles afsluit met behulp van magneten die in de 4 gaten in de hoeken werden gelijmd. Beide stukken zijn getekend in Solid Edge en ge-3D-print uit PLA, een soort kunststof. De LCD (1) zit vast in de opening. Het toetsenbord heeft een zelfklevende achterkant maar dit bleef niet goed plakken aan de PLA. Daarom wordt het toetsenbord (2) met dubbelzijdige tape vastgemaakt en de draden voor de verbindingen met de Arduino gaan via de opening. Spijtig genoeg heb ik hier een fout gemaakt en moet de opening aan de onderkant i.p.v. de bovenkant. Ik wou dit niet opnieuw laten printen want door zelf een opening te maken met een warme soldeerbout is het gemakkelijk opgelost (zie figuur 28). De RFID-module (3) schuift door de gleuf en wordt ook met warme lijm vastgemaakt. Omdat er heel wat verbindingen gemaakt worden, en er niet veel plaats is (zie figuur 27), heb ik de draden ook met warme lijm vastgemaakt aan het doosje.



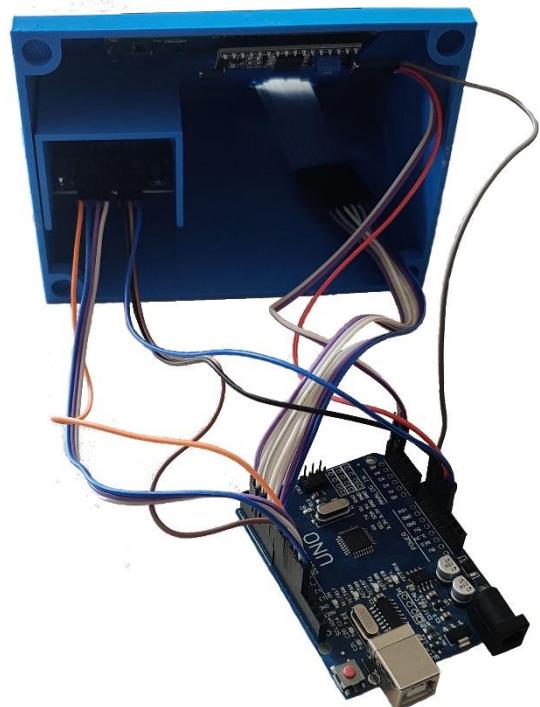
Figuur 25. Solid Edge assembly tekening .



Figuur 26. Achterkant van de betaalmodule.



Figuur 28. Extra gat gemaakt voor bedrading van toetsenbord.



Figuur 27. Veel draden en weinig plaats.

## 2.7 Tussenbesluit

Als ik de betaalmodule nog eens zou moeten maken, zou ik enkele dingen aanpassen. Eerst en vooral, het omhulsel zou ik iets groter maken met meer plaats voor de componenten, of andere draden gebruiken die het makkelijker maken om alles er in te krijgen. De gaten voor de magneetjes moeten iets groter en de opening voor de draden van het toetsenbord moet op een andere plaats. Ik zou ook een andere methode gebruiken om het toetsenbord vast te maken. Seconde lijm plakte niet goed op de achterkant van het toetsenbord en warme lijm is nogal dik. Ik had geëxperimenteerd met een manier om het toetsenbord er in te laten schuiven maar omdat dit problemen opleverde met het ondersteunend materiaal die ik niet uit de print kreeg, heb ik toch geopteerd om het er aan te lijmen. Achteraf gezien zou de optie van het ‘inschuiven’ misschien de betere piste zijn geweest. Uiteindelijk heb ik toch mijn doel bereikt. Verder zou ik ook nog eens willen uitrekenen hoelang deze module kan werken met één 9V batterij.



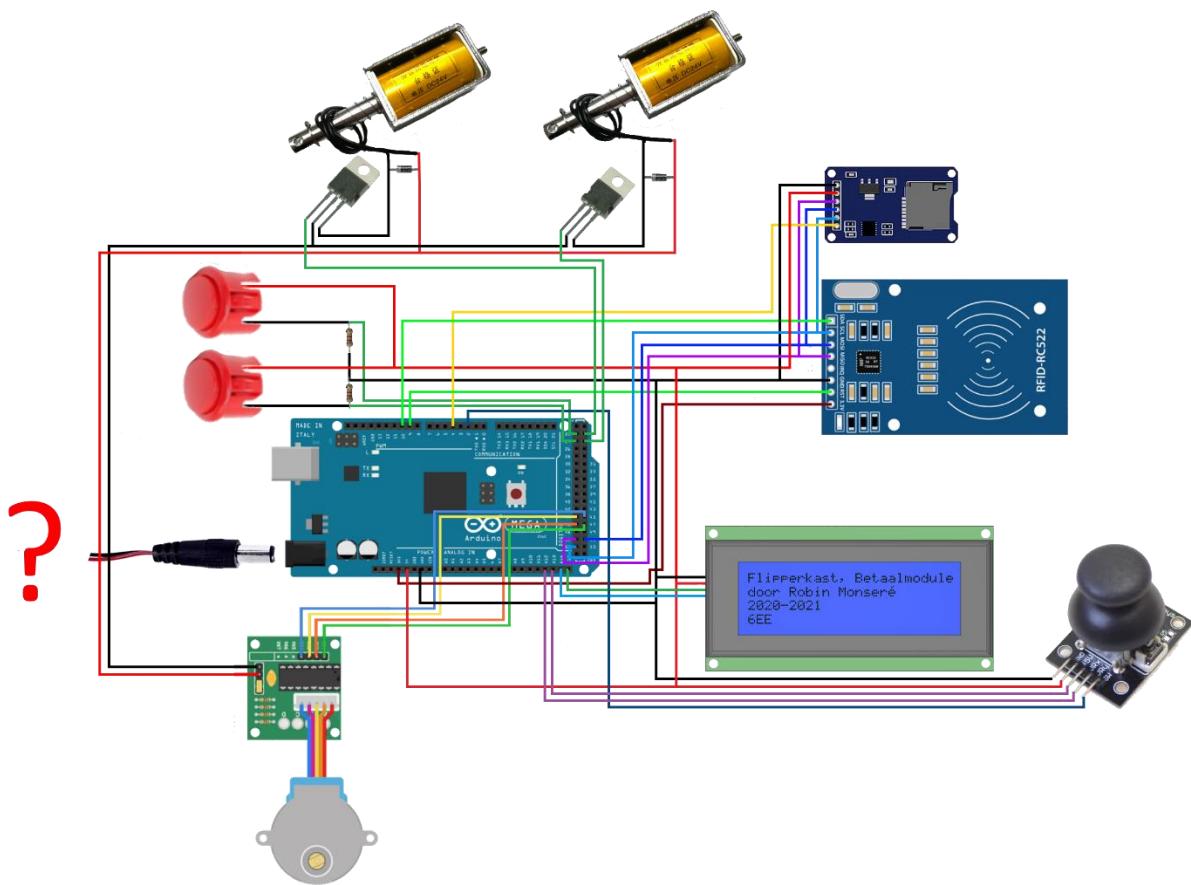
Figuur 30. Solid Edge tekening voor toetsenbord case.



Figuur 29. Toetsenbord kon er niet volledig in.

### 3 Flipperkast

In de flipperkast zit nog veel meer hardware en software. Ook hier zijn een LCD scherm en de RC522 aanwezig. De LCD is hier wel een stuk groter, 16 karakters per rij, en 4 rijen. Daarnaast is er nog een micro SD kaard lezer, hierop komt het klassemement te staan. Ook zijn er 3 kleine printplaten aanwezig, 1 voor de stuurschakeling van de solenoïdes, 1 voor de drukknoppen en 1 voor de SPI bus. De solenoïdes zijn het belangrijkste in de flipperkast, hiermee worden de flippers aangestuurd. Aan de zijkant van de flipperkast zijn er 2 knoppen geïnstalleerd Daarnaast is er nog een joystick aanwezig die de gebruiker kan gebruiken om op de LCD te navigeren.



Figuur 31. Schematische voorstelling van alle hardware verbonden met de Arduino Mega.

## 3.1 Constructie

De flipperkast zelf is gemaakt uit hout. Door ook eens een prototype te maken kon ik verschillende dingen testen en ook fouten ontdekken. Zo heb ik beslist om de uiteindelijke versie iets groter te maken.

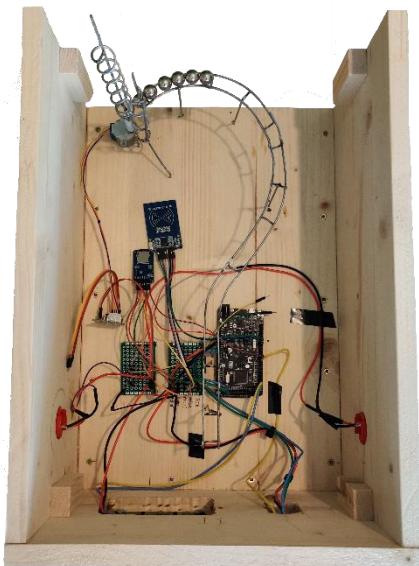


Figuur 33. Prototype.



Figuur 32. Uiteindelijk design.

In de flipperkast zit een baan waar de ballen over rollen om van de voorkant naar de achterkant te gaan, waar de bal via een spiraal weer het speelveld op kan. Deze spiraal is aangedreven door een stappenzetmotor. Deze baan en spiraal zijn gemaakt uit verzinkte staaldraad van 2mm. De verbindingen zijn gesoldeerd. Hier is er een fout bij het maken van de spiraal, die start namelijk te hoog. Als die iets lager was gestart was er meer ruimte om de baan iets steiler te maken en was er meer ruimte over. Uiteindelijk was er geen probleem, maar het had alles wel makkelijker gemaakt.



Figuur 35. binnenkant flipperkast met baan.



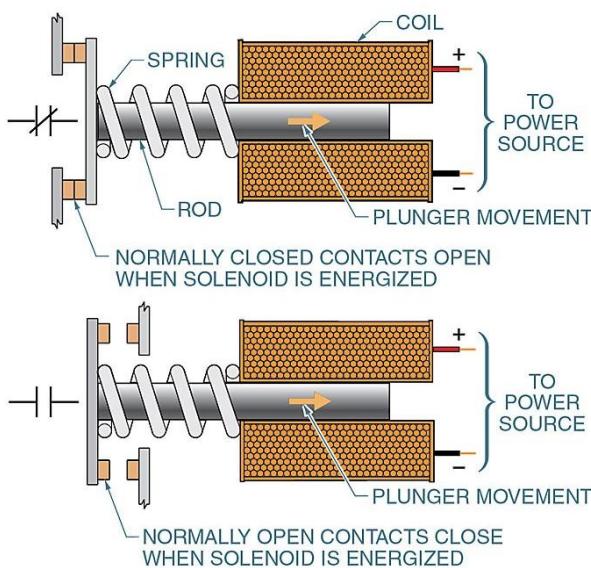
Figuur 34. Spiraal.

## 3.2 Solenoïdes

Om te de flippers te laten flipperen, maak ik gebruik van solenoïdes. In het begin heb ik een solenoïde getest van 6V, deze was niet krachtig genoeg en heb ik beslist om 2 solenoïdes te bestellen van 24V. Bij nader inzien was dit zeer krachtig en leek het mij genoeg om te werken op 12V. Dit had ook als voordeel dat ik zo met 1 bron kon werken aangezien de Arduino op 12V gevoed wordt.

### 3.2.1 Werking

De solenoïde is een spoel met een ijzeren staaf in. Als er stroom door de spoel vloeit, zal dit een magnetisch veld opwekken. Dit magnetisch veld trekt de staaf aan. Als er geen stroom meer door de spoel vloeit, zal het magnetisch veld verdwijnen en de ijzeren staaf zal door een veer terugkeren. Dit is snel en krachtig genoeg om een balletje ver weg te slaan.



Figuur 37. Principe werking solenoïde.



Figuur 36. Het type solenoïde die in de flipperkast zit.

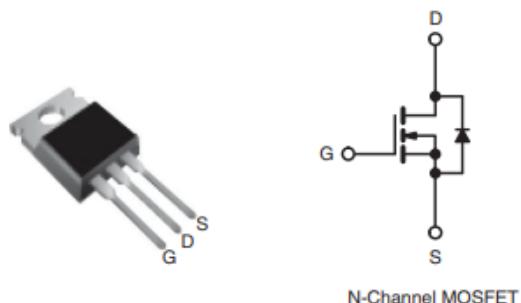
### 3.2.2 Aansturing

1 solenoïde vraagt 8A bij 12V, als beide spoelen op hetzelfde moment, parallel, gestuurd worden, zal er dus 16A gevraagd worden van de bron. De Arduino kan amper 20mA per pin sturen. Dit is ver van genoeg om de solenoïdes te laten werken, daarom wordt er gebruik gemaakt van twee MOSFETS.

### 3.2.2.1 MOSFET

Door een MOSFET te gebruiken los je dit probleem op. Eerst had ik een MOSFET van het type IRF540, maar deze heb ik vervangen door de IRL540. Wanneer de IRF540 geschakeld werd met 5V, was die nog niet volledig in doorlaat. De IRL540 deed dit wel, de L staat namelijk voor ‘logic’. Dus wanneer de IRL540 aangestuurd wordt, is de MOSFET volledig in doorlaat. De MOSFET wordt geschakeld met de Arduino Mega. De datasheet van de IRL540 is te vinden op de volgende pagina.

- $U_{DS} = 100V$  max
- $I_D = 28A$  max
- Fast switching
- Logic-Level Gate Drive

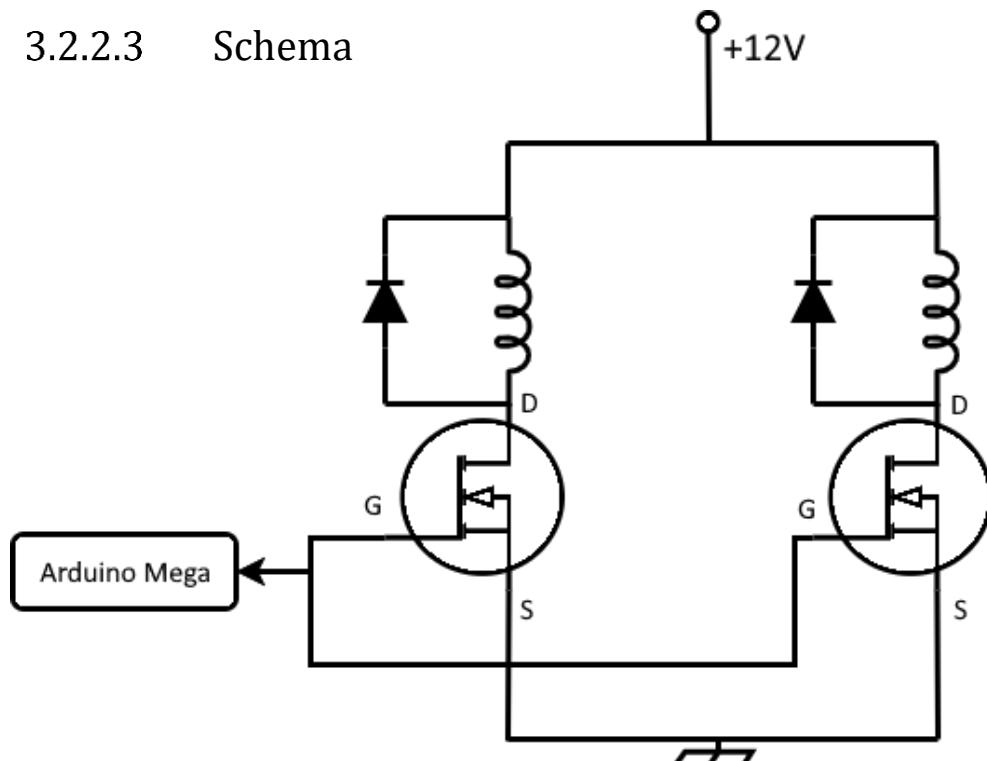


Figuur 38. De IRL540 en schema.

### 3.2.2.2 Blusdiode

Als er door een spoel (of solenoïde) een stroom vloeit, zal deze een tegen elektromotorische kracht opwekken. De blusdiode wordt gebruikt om de tegen EMK kort te sluiten en het vernietigen van de MOSFET tegen te gaan.

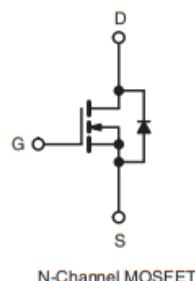
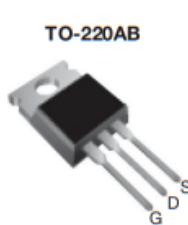
### 3.2.2.3 Schema



Figuur 39. Schema schakeling solenoïdes.

## Power MOSFET

PRODUCT SUMMARY	
$V_{DS}$ (V)	100
$R_{DS(on)}$ ( $\Omega$ )	$V_{GS} = 5.0$ V    0.077
$Q_g$ (Max.) (nC)	64
$Q_{gs}$ (nC)	9.4
$Q_{gd}$ (nC)	27
Configuration	Single



### FEATURES

- Dynamic dV/dt Rating
- Repetitive Avalanche Rated
- Logic-Level Gate Drive
- $R_{DS(on)}$  Specified at  $V_{GS} = 4$  V and 5 V
- 175 °C Operating Temperature
- Fast Switching
- Ease of Paralleling
- Compliant to RoHS Directive 2002/95/EC



### DESCRIPTION

Third generation Power MOSFETs from Vishay provide the designer with the best combination of fast switching, ruggedized device design, low on-resistance and cost-effectiveness.

The TO-220AB package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 W. The low thermal resistance and low package cost of the TO-220AB contribute to its wide acceptance throughout the industry.

ORDERING INFORMATION			
Package	TO-220AB		
Lead (Pb)-free	IRL540PbF		
SnPb	SiHL540-E3	IRL540	SiHL540

### ABSOLUTE MAXIMUM RATINGS ( $T_C = 25$ °C, unless otherwise noted)

PARAMETER	SYMBOL	LIMIT	UNIT
Drain-Source Voltage	$V_{DS}$	100	V
Gate-Source Voltage	$V_{GS}$	$\pm 10$	
Continuous Drain Current	$I_D$	28	A
		20	
Pulsed Drain Current <sup>a</sup>	$I_{DM}$	110	
Linear Derating Factor		1.0	W/°C
Single Pulse Avalanche Energy <sup>b</sup>	$E_{AS}$	440	mJ
Avalanche Current <sup>a</sup>	$I_{AR}$	28	A
Repetitive Avalanche Energy <sup>c</sup>	$E_{AR}$	15	mJ
Maximum Power Dissipation	$P_D$	150	W
Peak Diode Recovery dV/dt <sup>c</sup>	dV/dt	5.5	V/ns
Operating Junction and Storage Temperature Range	$T_J, T_{stg}$	- 55 to + 175	°C
Soldering Recommendations (Peak Temperature)	for 10 s	300 <sup>d</sup>	
Mounting Torque	6-32 or M3 screw	10	lbf · in
		1.1	N · m

#### Notes

a. Repetitive rating; pulse width limited by maximum junction temperature (see fig. 11).

b.  $V_{DD} = 25$  V, starting  $T_J = 25$  °C,  $L = 841 \mu\text{H}$ ,  $R_g = 25 \Omega$ ,  $I_{AS} = 28$  A (see fig. 12c).

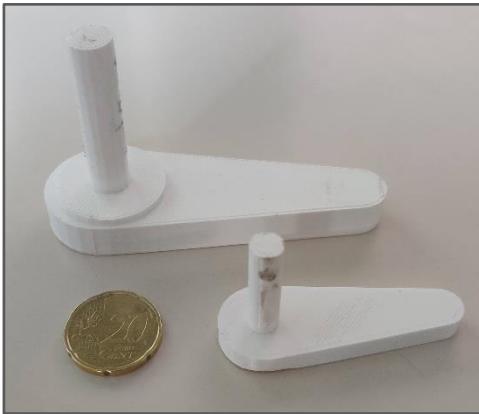
c.  $I_{SD} \leq 28$  A,  $dI/dt \leq 170$  A/μs,  $V_{DD} \leq V_{DS}$ ,  $T_J \leq 175$  °C.

d. 1.6 mm from case.

\* Pb containing terminations are not RoHS compliant, exemptions may apply

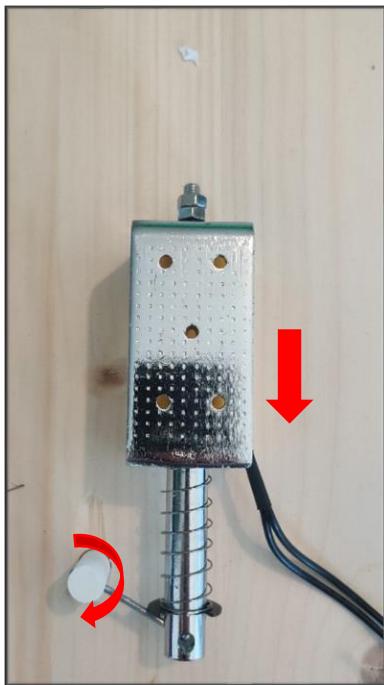
### 3.3 Flippers

De flippers zijn ge-3D-print, en getekend in Solid Edge. De eerste die ik had laten printen was veel te klein in vergelijking met de afmetingen van de flipperkast. De tweede versie was groot genoeg. Hier is er ook nog een klein randje bij de flipper, zo sleept de flipper niet over de flipperkast en is er dus minder wrijving.

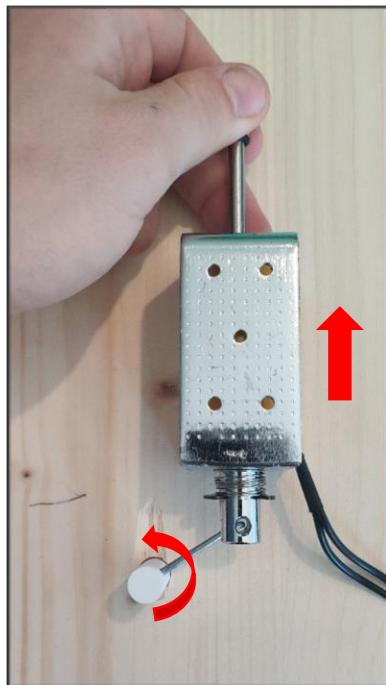


Figuur 40. Prototypes flippers.

Door de solenoïdes met een ijzeren staafje aan de flippers te verbinden, zullen die bewegen wanneer de speler op de knop drukt.



Figuur 42. Solenoïde niet ingetrokken.



Figuur 41. Solenoïde ingetrokken.

## 3.4 Drukknoppen

Als de speler een flipper wil bedienen, moet die de corresponderende drukknop indruwen. De grote rode knopen zijn op de zijkant van de flipperkast gemonteerd. Je kan de drukknoppen blijven inhouden, maar om de solenoïdes niet te overbeladen, is het onmogelijk om de flippers omhoog te houden. Als de solenoïdes voor een te lange tijd worden bekrachtigd, zal de stroom in de spoelen te groot worden, en zullen die doorbranden. Deze beveiliging is software matig aangebracht.



Figuur 43. Knop buitenkant.



Figuur 44. Knop binnenkant.

### 3.4.1 Software

```
void FLIPPEREN()
{
    if(digitalRead(drukknop_1)==HIGH && stateF1==LOW && isUsableF1==true)
    {
        stateF1 = HIGH;
        previousMillisF1 = millis();
        digitalWrite(flipper_1, HIGH);
    }
    if (digitalRead(drukknop_1) == LOW and stateF1 == HIGH)
    {
        digitalWrite(flipper_1, LOW);
        millisHTL1 = millis();
        isUsableF1 = false;
        stateF1 = LOW;
    }
    currentMillis = millis();
    if (currentMillis - previousMillisF1 >= timerFlipper and stateF1==HIGH)
    {
        digitalWrite(flipper_1, LOW);
        millisHTL1 = millis();
        isUsableF1 = false;
        stateF1 = LOW;
    }
    currentMillis = millis();
    if (currentMillis - millisHTL1 >= timeBetweenUses and stateF1 == LOW)
    {
        isUsableF1 = true;
    }
}
```

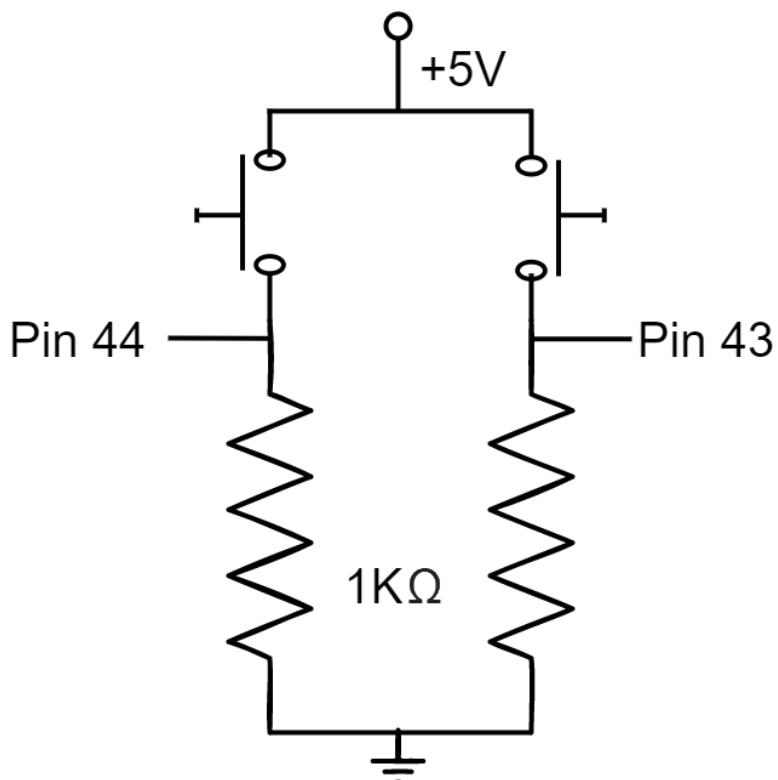
De functie ‘`millis()`’ retourneert het aantal milliseconden dat is verstreken sinds de Arduino het huidige programma begon uit te voeren. Als je dit getal opslaat in een variabele wanneer je de flipper aanstuurt, kan je makkelijk weten wanneer je opnieuw de flipper kan bedienen. Dit getal wordt opgeslagen in een variabele van het type ‘unsigned long’. In zo’n variabele kunnen er 32 bits worden opgeslagen. Oftewel een getal van 0 tot 4,294,967,295. Dit getal in seconden omgezet naar maanden, is 1,63 maand. Dus het zal geen probleem opleveren dat deze variabele volzit.

Om de flipper te bedienen moet er aan enkele voorwaarden worden voldaan. Ten eerste moet de drukknop natuurlijk worden ingedrukt, ten tweede mag de flipper niet al in gebruik zijn op dat moment en ten derde moet de ‘isUsable’ variabele ‘true’ zijn. Pas dan zal de MOSFET de solenoïde kunnen aansturen.

Als de solenoïde geactiveerd is en er zijn 500ms verstreken, zal in de derde if statement de solenoïde gedeactiveerd worden.

### 3.4.1.1 Schema

De drukknoppen zijn simpelweg een pull down circuit verbonden met pin 43 en 44 van de Arduino Mega



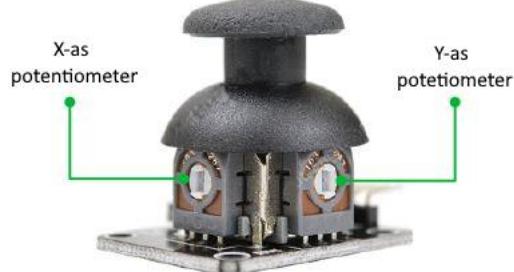
Figuur 45. Elektrisch schema voor beide drukknopen.

## 3.5 Joystick en LCD

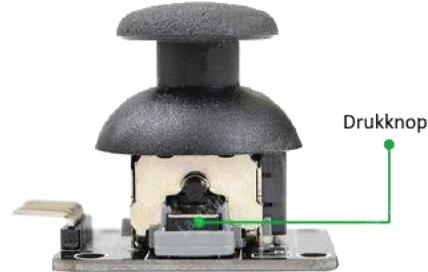
Om te navigeren op het LCD scherm kan de gebruiker gebruik maken van de joystick. Hiermee kan je dan scrollen door het klassement of het spel starten/stoppen door op de joystick te drukken. De lcd is groter dan deze die gebruikt wordt in de betaalmodule. Op de LCD zijn er 4 lijnen beschikbaar met 20 karakters per lijn. De I<sup>2</sup>C LCD werd eerder al uitvoerig besproken op pagina 14.

### 3.5.1 Werking

Het doel van een joystick is om een beweging in 2D naar een arduino te communiceren. Dit wordt bereikt door 2 onafhankelijke 10k ohm potentiometers (1 per as). Deze functioneren als spanningsdelen en de output hiervan kan via een analoge poort ingelezen worden op de Arduino. Deze joystick bevat ook een schakelaar die wordt geactiveerd wanneer er op de knop wordt gedrukt. Deze werkt ongeacht in welke positie de joystick zich bevindt.



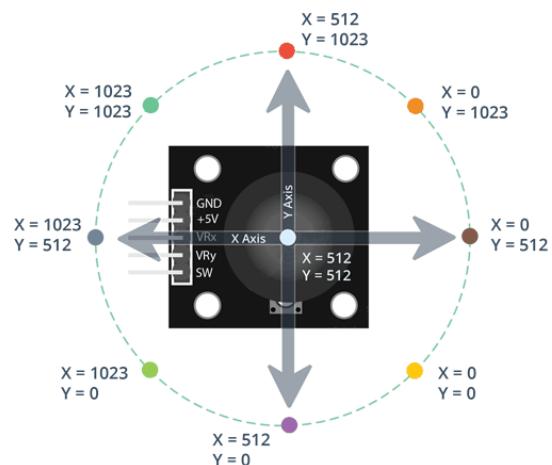
Figuur 47. X en Y as potentiometers.



Figuur 46. Drukknop van joystick.

Om de fysieke positie van de joystick te kunnen aflezen, moeten we de spanning van de potentiometer inlezen op een analoge Arduino-pin. Omdat het Arduino-bord een ADC-resolutie (analoog-digitaalomzetter) van 10 bits heeft, zullen de ingelezen waarden tussen 0 en 1023 liggen. Dus als de joystick over de X-as van het ene uiteinde naar het andere wordt verplaatst, veranderen de X-waarden van 0 tot 1023. Hetzelfde met de Y-as.

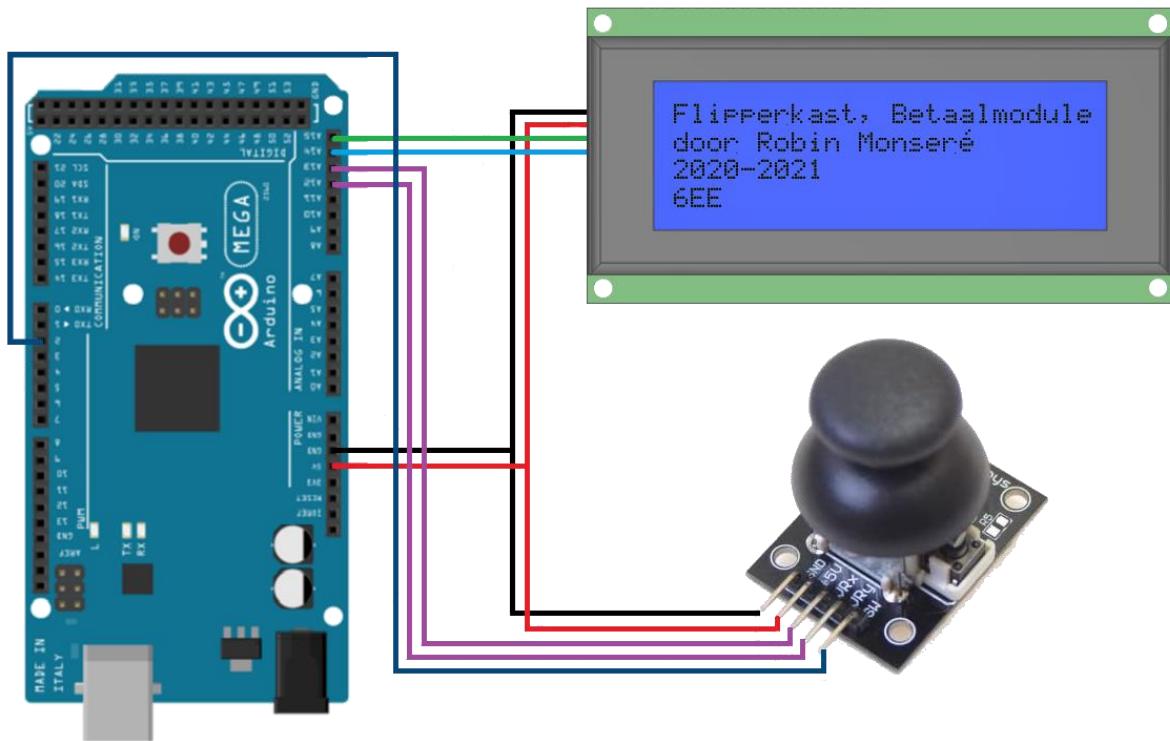
Als de joystick in zijn middenpositie blijft, ligt de waarde rond de 512.



Figuur 48. Positiebepaling joystick.

### 3.5.2 Aansluitingen

De aansluitingen voor de lcd zijn hetzelfde als op de Arduino Uno. De joystick wordt gevoed met 5V, de drukknop lees je in via een digitale pin, en de X en Y-as zijn verbonden met 2 analoge pinnen.



Figuur 49. Aansluitschema LCD + joystick.

### 3.5.3 Software

Net zoals bij de LCD van de betaalmodule, wordt er gebruik gemaakt van de I2C bibliotheek. Eerst wordt de lcd geinitialiseerd, je geeft het programma het I2C adres mee, hier 0x27. Zo weet de master welke slave hij moet aansturen. Het aantal karakters per lijn (20) en het aantal lijnen (4). Als je op de eerste plaats van de tweede lijn een ‘5’ wil printen, dan gebruik je ‘lcd.setCursor(1, 2);’ om aan te geven waar je iets wil printen op de lcd en ‘lcd.print (“5”);’ om dit dan effectief te tonen. Bij de joystick worden de waarden continu uitgelezen om te weten wanneer de gebruiker de joystick bedient en als er iets moet gebeuren op het scherm.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

const byte joystickSW = 2;
const byte joystickX = A0;
const byte joystickY = A1;

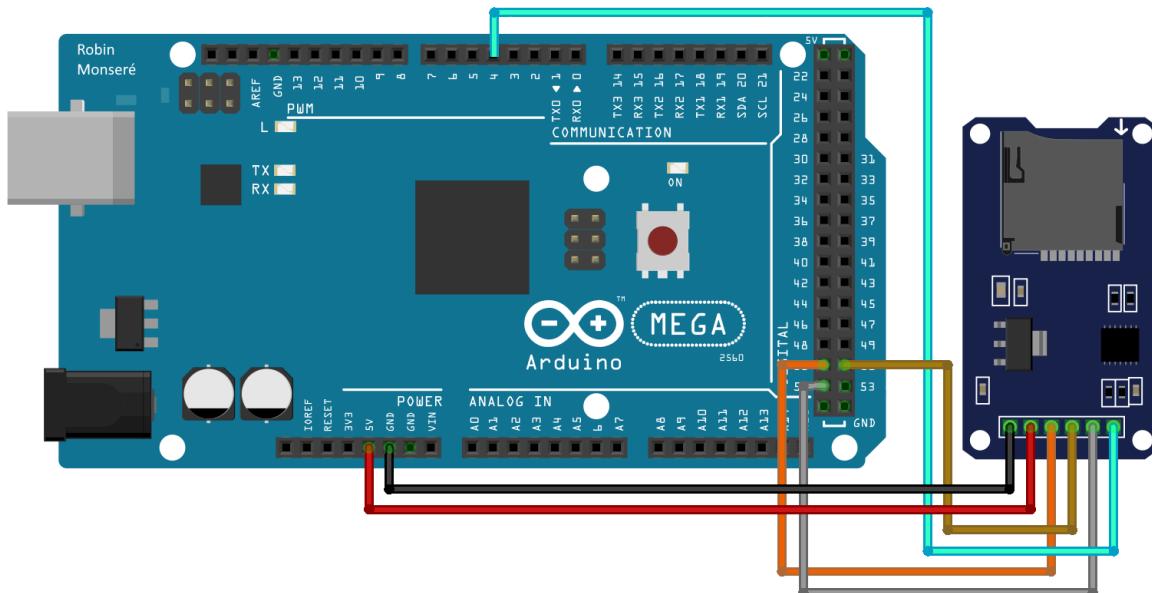
int yValue = analogRead(joystickY);
int xValue = analogRead(joystickX);
```

## 3.6 Micro SD module

De SD kaard module wordt gebruikt om de punten van het klassemement op te slaan. Ook was het oorspronkelijk de bedoeling om geluidjes af te spelen via een box. Deze geluiden staan opgeslagen op de SD kaart. Dit heb ik er echter niet ingestoken want het geluid kon niet luid genoeg. De oplossing voor dit probleem is een amplifier, maar die heb ik nog niet besteld.

### 3.6.1 Aansluitingen

De SD kaard module gebruikt net zoals de RFID module de SPI bus. (Meer uitleg over de SPI bus op pagina 11.) De MISO, MOSI en SCK zijn verbonden met de standaard SPI pinnen op de Arduino Mega. Dit zijn pin 50, 51 en 52 respectievelijk. Daarnaast is de SS verbonden met pin 4, en wordt gevoed met 5V.



Figuur 50. Aansluitingen SD kaard module.

### 3.6.2 Software

Eerst was het de bedoeling om met een JSON bibliotheek de data van het klassemement in een JSON formaat om te zetten en die op de SD kaard te schrijven en dan terug uit te lezen. Maar omdat de SD kaard bibliotheek die ik gebruik enkel .txt bestanden ondersteund was dit niet mogelijk. Ik kon wel de JSON string schrijven naar een .txt bestand maar dit ging alles moeilijker maken. Daarom heb ik beslist om de data in 2 te splitsen en naar 2 bestanden te sturen. 1 bestand voor de volgorde van de namen en 1 voor de punten.

In beide tekstbestanden staan de namen en punten in respectievelijke volgorde, als die ingelezen worden in 2 verschillende arrays, kunnen deze dan met elkaar vergeleken worden en zo weet de Arduino wie de meeste punten heeft. Op figuur 51 en 52 kan je zien wat er op het scherm komt als je naar het klassemement kijkt en met de joystick door de plaatsen scrolt.

	VbNaam.txt -	VbPunten.txt
	Bestand	Bewerk
<b>Oranje</b>	<b>51</b>	
<b>Groen</b>	<b>45</b>	
<b>Blauw</b>	<b>41</b>	
<b>Rood</b>	<b>37</b>	
<b>Paars</b>	<b>20</b>	
<b>Wit</b>	<b>19</b>	

Figuur 53. Tekstbestanden met namen en punten.



Figuur 51. Klassemement plaats 1 en 2.



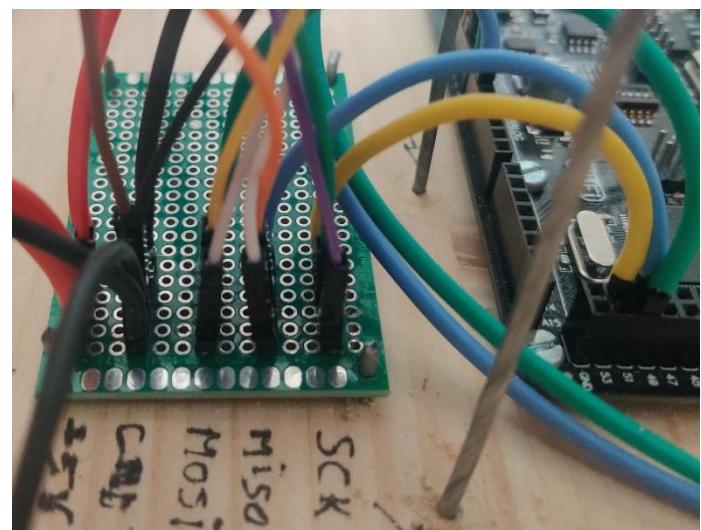
Figuur 52. Klassemement plaats 5 en 6.

Het programma dat bij het inlezen/schrijven en omzetten van de punten hoort, is meer dan 400 lijnen lang. Om deze portfolio niet vol te steken met code heb ik beslist om die hier dan ook niet in te verwerken. De code is te vinden op:

[https://github.com/xdNiBoR/GIP\\_RM\\_2020-2021/blob/main/GIP\\_RM\\_Arduino/GIP\\_RM\\_Flipperkast/GIP\\_RM\\_Flipperkast/HIHGSORES.ino](https://github.com/xdNiBoR/GIP_RM_2020-2021/blob/main/GIP_RM_Arduino/GIP_RM_Flipperkast/GIP_RM_Flipperkast/HIHGSORES.ino)

## 3.7 RFID Module

In de flipperkast wordt de RFID module op exact dezelfde manier gebruikt als in de betaalmodule. Eerst wordt de UID ingelezen en daarna het aantal Credits die op de badge staan. Wanneer de speler een spel start, zal er een aantal Credits van af gaan. Samen met de micro SD module gebruikt de RFID module de SPI bus. Dit betekent dat deze beide aan dezelfde pinnen verbonden worden. Om deze te verbinden heb ik gebruik gemaakt van gaatjes PCB's.



Figuur 54. PCB met de nodige bus lijnen.

## 3.8 Stappenmotor

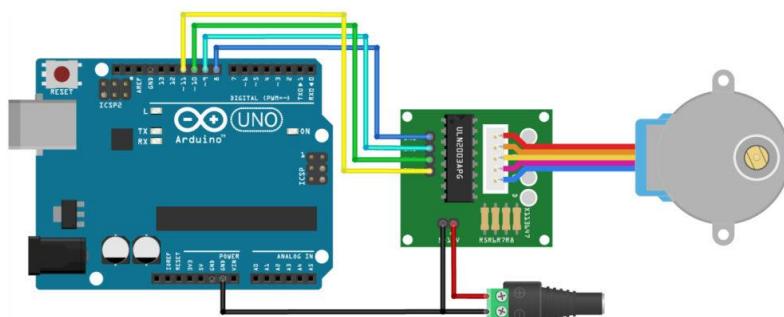
Voor de spiraal die de ballen naar boven brengt, wordt er gebruik gemaakt van een stappenmotor. De 28BYJ-48 met de ULN2003 driver. Deze worden veel gebruikt omdat ze zeer goedkoop zijn en makkelijk aan te sturen. De versie die ik gebruik, is voor 12V, zo kan ik die aansluiten aan dezelfde bron als de solenoïdes en voeding van de Arduino. Met een stappenmotor kan je heel precies draaien, al was dit hier niet zo van belang, het maakte het toch wel makkelijk om precies en zonder error 1 volledige omwenteling te maken.



Figuur 55. 28BYJ-48 in werking.

### 3.8.1 Aansluitingen

De stappenmotor aansluiten met de ULN2003 driver is zeer eenvoudig. De 4 pinnen van de driver moeten rechtstreeks met eender welke uitgang van de Arduino verbonden worden. Het is wel belangrijk om de motor niet via de Arduino maar via een externe voeding te voeden. De motor vraagt namelijk te veel stroom om door de Arduino gevoed te worden.



Figuur 56. Aansluitingen 28BYJ-48 en ULN2003.

### 3.8.2 Software

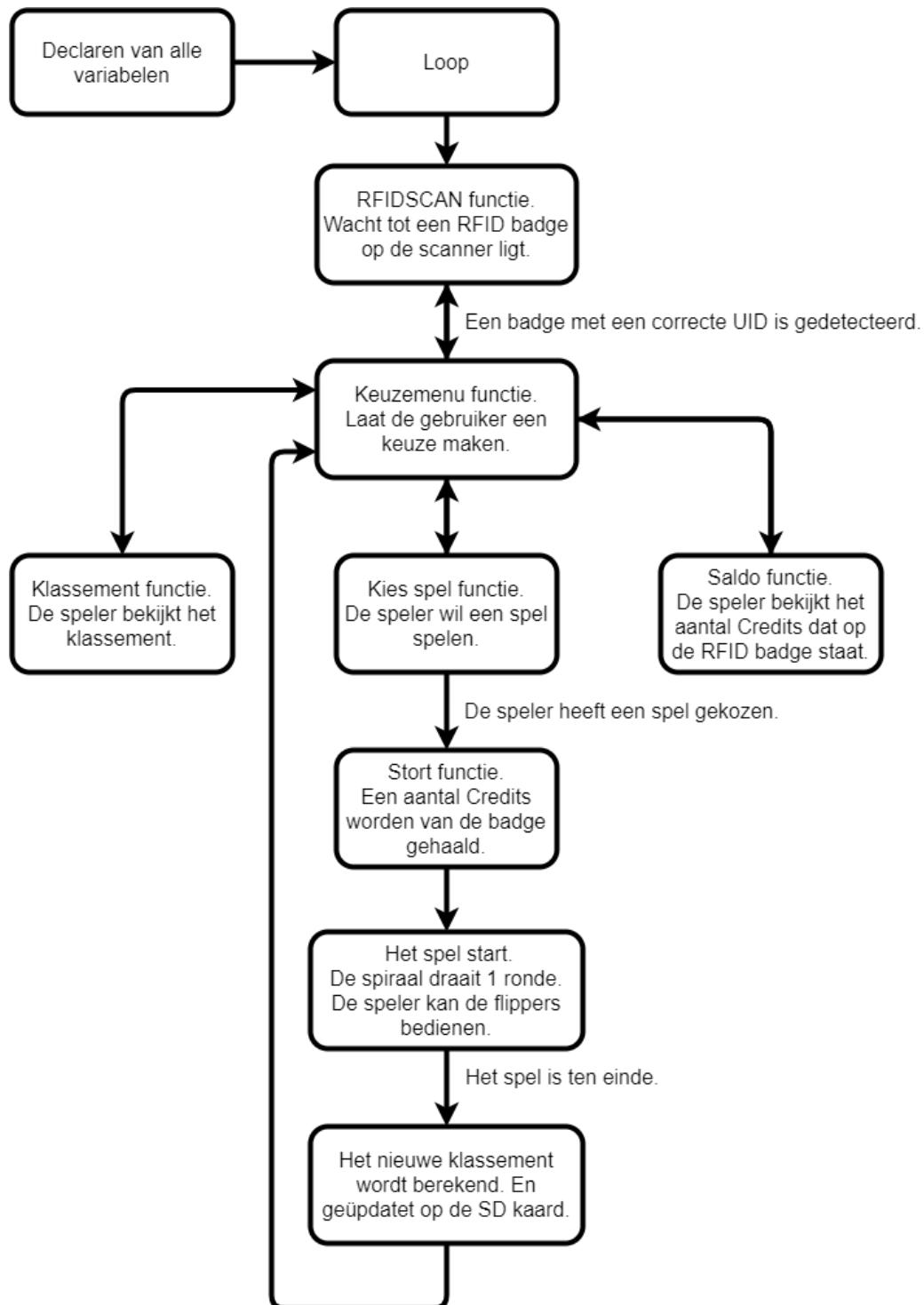
Door gebruik te maken van een bibliotheek en de uln2003 is het simpel om deze aan te sturen. Je begint met een stappenmotor object aan te maken. Hier geef je ook mee met welke pinnen de stappenmotor is verbonden. Daarna kan je per stap de motor sturen. De 28BYJ-48 heeft 2048 stappen per revolutie, dus om de motor 1 omwenteling te laten maken, step je 2048 keer.

```
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
myStepper.step(2048);
```

## 3.9 Software

Alle software van alle componenten samen in de portfolio toelichten zou onnodig veel plaats in beslag nemen. De code alleen zou tientallen pagina's in beslag nemen, het totaal komt neer op meer dan 1200 lijnen. Alle code is terug te vinden op de GitHub pagina:

[https://github.com/xdNiBoR/GIP\\_RM\\_2020-2021](https://github.com/xdNiBoR/GIP_RM_2020-2021)



Figuur 57. Schematische voorstelling van de software voor de flipperkast.

## 3.10 Bron

De bron die op het einde van het jaar gebruikt wordt om de flipperkast aan te sturen, is een bron van school. Omdat het eerst de bedoeling was om de solenoïdes op 24V te voeden en de rest op 12V, had ik een geschakelde bron nodig. Ook was ik niet zeker hoeveel stroom beide solenoïdes zouden trekken. Ik had ook geen duidelijk zicht op wat de rest van de schakelingen zouden nodig hebben. Daarom heb ik beslist om tot op het einde van het jaar een bron van school te gebruiken en dan op te meten wat voor bron er precies geschikt is. Zover ben ik niet geraakt en gebruik nog steeds een bron van school.

## 4 Besluit

Dit eindwerk was zeer interessant. Ik heb op alle vlakken dingen bijgeleerd. Zoals hoe je het best veel software schrijft zodat die ook voor jezelf te lezen is. Ik kwam vele bugs tegen en heb vele uren gespendeerd aan het oplossen daarvan. Er kunnen veel dingen verbeterd worden aan mijn project, en dit zal ik dan ook nog doen. Het schrijven van deze portfolio was zeer interessant, omdat het de eerste keer was dat ik effectief zo'n groot project heb gedocumenteerd. Ik ben blij dat ik in het begin van het jaar besloten heb om alles vanaf het begin te documenteren en op te schrijven. Bij het maken van deze portfolio heeft me dat goed geholpen. Ik heb alle documentatie/software/schema's op een GitHub pagina geüpload. Deze is te vinden op: [https://github.com/xdNiBoR/GIP\\_RM\\_2020-2021](https://github.com/xdNiBoR/GIP_RM_2020-2021) Met meer dan 80 commits heb ik ook veel geleerd over GitHub en hoe dit werkt. Dit zal zeker handig zijn voor volgend jaar en later.

Ik ben blij dat ik voor de draden binnen in de flipperkast geen PCB op maat heb gemaakt. Dit zou teveel tijd in beslag nemen en met de gaatjes PCB's ging het perfect. Het was ook zeer makkelijk om deze te veranderen als ik fouten ontdekte.

## 4.1 Fouten

Mijn grootste fout is om in het midden van het jaar niet genoeg aan de GIP te werken. Ik was in het begin goed begonnen met de betaalmodule, ik had die al snel af. Ik was begonnen aan de flipperkast, maar de inspiratie en interesse viel wat weg midden het jaar. Dit kwam waarschijnlijk door het feit dat we op school nauwelijks met de GIP bezig waren. We hadden ook maar halftijds school door Covid-19 en dit speelde een grote rol. Uiteindelijk ben ik dan niet geraakt waar ik wou, maar dit betekent dat ik er later nog verder aan ga werken.

Bij de spiraal die de ballen omhoog brengt zitten er ook enkele foutjes, deze is eerst en vooral niet simpel te maken omdat die mooi recht moet zijn en perfect rond. Om het makkelijker te maken voor de ballen om op de spiraal te geraken zou de draairichting anders moeten. En die moet ook iets lager bij de motor gemonteerd worden. Dit zou ervoor zorgen dat er meer ruimte over is voor de baan die de ballen ernaar toe brengt. Dit zou de montage daarvan iets makkelijker maken.

Een fout die ik gemaakt heb is om eerst een prototype van de flipperkast te maken. Dit heeft enkele weken in beslag genomen die verloren zijn gegaan.

In het design van de betaalmodule zit ook een kleine fout, het gat voor de kabels van het toetsenbord zit op de verkeerde plaats, dit was makkelijk op te lossen door zelf een gat bij te maken.

Om de solenoïdes te sturen wordt er gebruik gemaakt van MOSFETS, hiervoor had ik de IRF540. Deze schakelde echter niet 100% bij 5V. Dit betekent dat die niet 100% in doorlaat kwamen als die met de Arduin werden gestuurd. Om dit op te lossen heb ik de IRL540 besteld. Deze werkten zonder problemen.

## 4.2 To do

De flipperkast is zeker nog niet volledig klaar. Ook zijn er een paar dingen die beter moeten. De belangrijkste staan hier opgenoemd.

- De spiraal opnieuw maken.
- Extra obstakels/manieren om punten te verdienen toevoegen.
- De flippers steviger verbinden met de solenoldes.
- Meerdere gamemodes programmeren.
- Een administrator mode toevoegen.

# 5 Componentenlijst

Component	Aantal	Prijs
Arduino Uno	1	€20
Arduino Mega	1	€12,45
20*4 I2C LCD	1	€3,65
16*4 I2C LCD	1	€3,19
4*4 membraam toetsenbord	1	€1,95
MFRC-522 RFID module	2	€2,12
10 RFID badges	1	€2,58
8 drukknoppen	1	€6,99
Solenoïde 24V DC	2	€34,53
Micro SD card module	1	€1,94
Joystick	1	€3,99
Gaatjes PCB	5	€3,25
12V 28BYJ-48 + ULN2003 stappenmotor	1	€3,85
Flexibele koperdraad	NA	€12,37
Verzinkte ijzerdraad 2mm 10m	1	€1,89
Verzinkte ijzerdraad 1,3mm 40m	1	€5,89
IRL540 MOSFET	5	€10,66
30 stalen ballen 10mm	30	€10,59
MM/FF/MF jumper wires	NA	NA
Weerstanden	NA	NA
Totaal:		€141,89

# 6 Bronnenlijst

<https://www.arduino.cc/>

<https://www.getpaint.net/>

<https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/>

<https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/>

<https://playground.arduino.cc/Code/Keypad/>

[https://nl.wikipedia.org/wiki/Serial\\_Peripheral\\_Interface](https://nl.wikipedia.org/wiki/Serial_Peripheral_Interface)

<https://nl.wikipedia.org/wiki/I%C2%B2C-bus>

<https://en.wikipedia.org/wiki/MIFARE>

[https://en.wikipedia.org/wiki/Unique\\_identifier](https://en.wikipedia.org/wiki/Unique_identifier)

Github:

[https://github.com/xdNiBoR/GIP\\_RM\\_2020-2021](https://github.com/xdNiBoR/GIP_RM_2020-2021)

<https://github.com/miguelbalboa/rfid>

[https://github.com/johnrickman/LiquidCrystal\\_I2C](https://github.com/johnrickman/LiquidCrystal_I2C)

Datasheets:

<https://www.vishay.com/docs/91300/sihl540.pdf>

<https://www.digikey.be/nl/datasheets/mikroelektronika/mikroelektronika-step-motor-5v-28byj48-datasheet>

<https://diy.waziup.io/assets/src/sketch/libraries/MFRC522/doc/rdmifare.pdf>