|  |
| --- |
| Technisch ontwerp\_  EINDOPDRACHT: VERKEERSSIMULATIE |

Sonny Selten 654954

29 maart 2021

INHOUDSOPGAVE

[1 aansluitschema 4](#_Toc68006151)

[2 software ontwerp 5](#_Toc68006152)

[3 STATEMACHINE 6](#_Toc68006153)

[4 hardware ontwerp 7](#_Toc68006154)

[4.1 Hardware Componenten 8](#_Toc68006155)

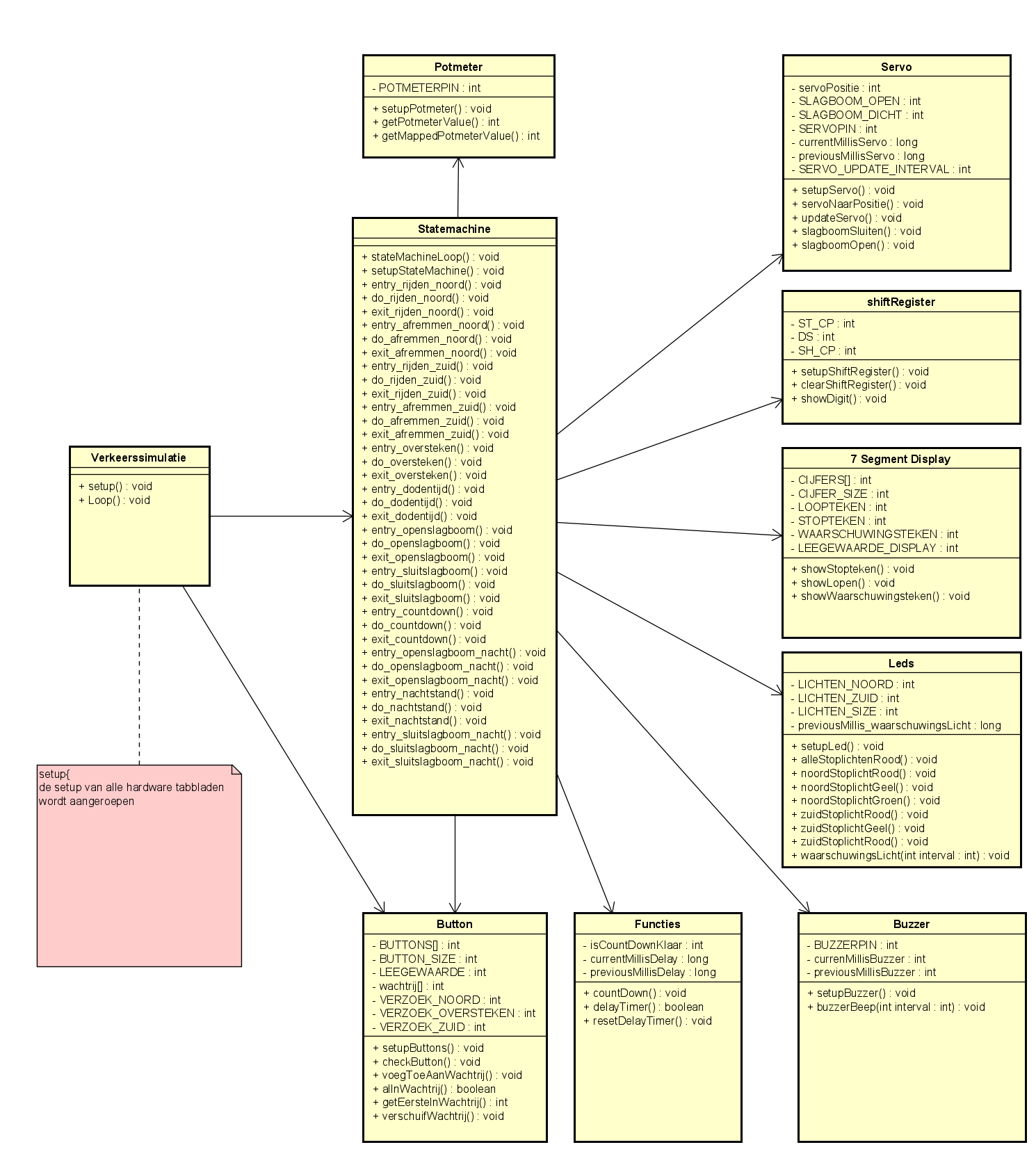
[5 functie omschrijving 10](#_Toc68006156)

[5.1 Functies 10](#_Toc68006157)

INLEIDING

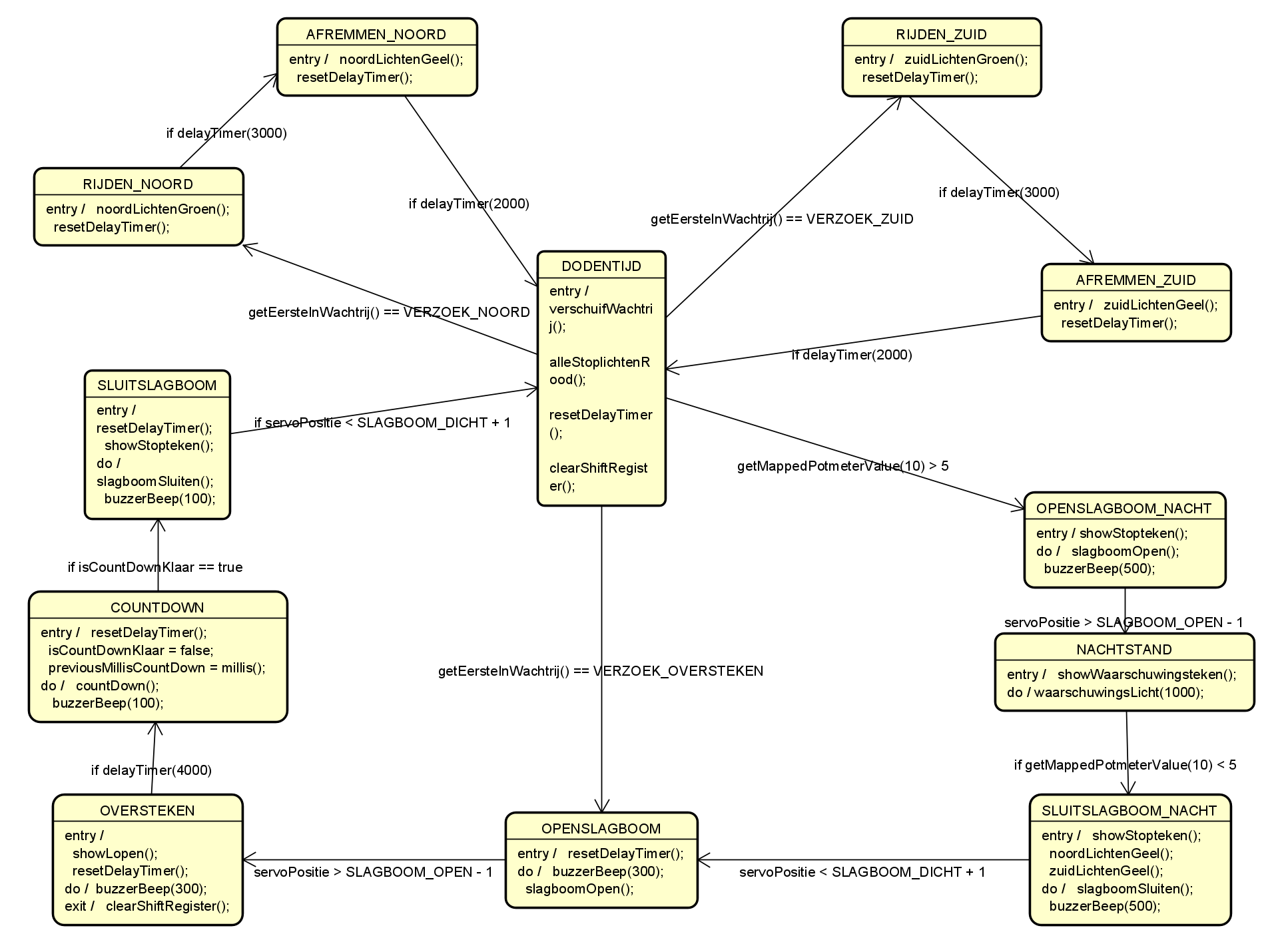
In dit document worden de technische specificaties besproken omtrent de demo verkeer simulatie naar aanleiding van een verzoek van een school in Gendt om de snelheid van het verkeer te reduceren.

# software ontwerp

Het software ontwerp voor de Verkeer Simulatie is zoals onderstaand afgebeeld.

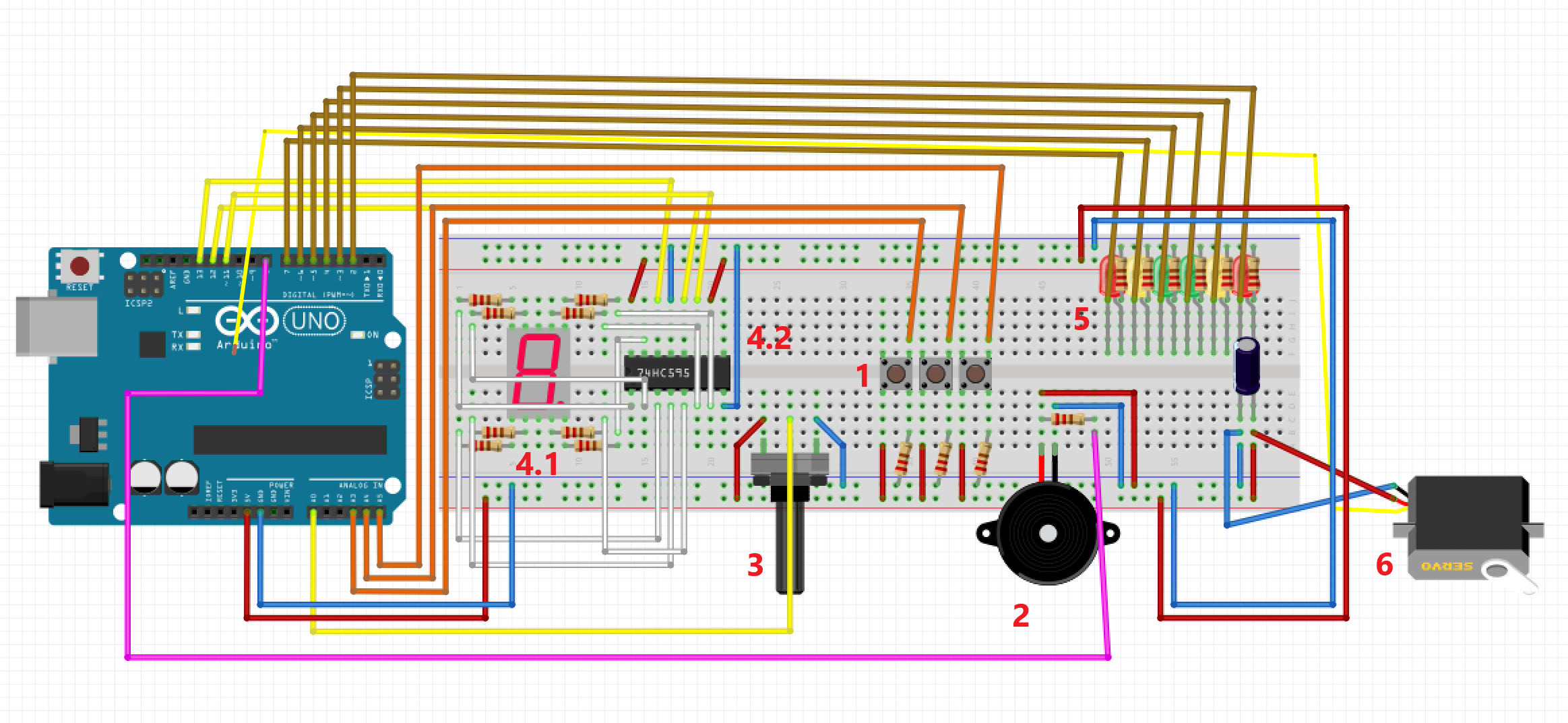
Figuur

# STATEMACHINE

De state machine voor de Verkeer Simulatie is zoals onderstaand afgebeeld.

Figuur

# hardware ontwerp

Het hardware ontwerp voor de Verkeer Simulatie is zoals onderstaand afgebeeld.

Figuur

## Hardware Componenten

Er is afgestemd met de opdrachtgever dat er verschillende hardware zal worden gebruikt voor de demo zoals ook kort beschreven in het opdracht verzoek. In dit hoofdstuk wordt de hardware beschreven en welke functie dit hardware onderdeel heeft in de demo

1. Drukknop

Er zal een 3-tal knoppen worden aangesloten in de demo. Twee knoppen dienen ter simulatie van een beweging sensor voor het auto verkeer dat nadert van de noord- en zuid-positie.

De derde knop dient voor het oversteek verzoek van de voetganger.

1. Buzzer

De buzzer zorgt ervoor dat er een toon wordt gegenereerd op een bepaald frequentie. Doormiddel van een geluidspartroon te creëren met de buzzer, maken we het duidelijk wanneer er mag worden overgestoken. Dit maakt het ook makkelijker voor slechtziende.

1. Potentiometer

Dit is een hardware onderdeel dat ons een bepaalde waarde kan geven gebaseerd of deze naar links of naar rechts is gedraaid. Met dit onderdeel simuleren we of de verkeer simulatie zich in de nachtstand of in de dag-stand bevindt.

1. 7 segment display & shiftregister

Het segment display (4.1) is een display dat bestaat uit 7 balken die individueel licht kunnen geven. Met dit onderdeel tonen we voor de voetgangers een loop teken om aan te geven dat de voetganger mag oversteken, en een countdown timer dat aangeeft hoe lang de voetganger nog heeft voordat de slagboom weer dicht gaat.

Ook wordt er op dit display een waarschuwing teken getoond als de verkeer simulatie zich in de nachtstand bevindt. Het shiftregister (4.2) zal ons ondersteunen in het gebruik van het segment display.

1. Ledlampen

Er zijn een 6-tal ledlampen aangesloten in de kleuren groen, geel en rood om het verkeerslicht te simuleren voor autoverkeer.

1. Servomotor

De servomotor is een motor die over zijn eigen as kan draaien. De servo in de verkeerssimulatie zal de rol van de slagboom spelen.

# functie omschrijving

In dit hoofdstuk zal kort worden beschreven van welke functies er gebruik wordt gemaakt en wat deze functie precies doet.

## Functies

* **showStopteken()**

Deze functie leegt het huidige segment display en toont het stop teken

* **showLopen()**

Deze functie leegt het huidige segment display en toont het loop teken

* **showWaarschuwingsteken()**

Deze functie leegt het huidige segment display en toont het waarschuwing teken

* **steupButtons()**

Deze functie zet de pin mode van de buttons op input zodat de actie van het indrukken van de knop wordt waargenomen

* **checkButton()**

Deze functie checkt of de button is ingedrukt, als dit waar is zal de button worden toegevoegd aan de wachtrij

* **voegToeAanWachtrij()**

Deze functie krijgt een int ‘waarde’ mee en zet deze waarde in de array wachtrij op de plek van een lege waarde, als de meegegeven waarde niet al in de wachtrij staat

* **alInWachtrij()**

Deze functie krijgt een int ‘waarde’ mee en check of de waarde al aanwezig is in de array wachtrij

* **getEersteInWachtij()**

Deze functie returned de index positie 0 van de array wachtrij, dus het eerstvolgende verzoek dat moet worden afgehandeld

* **verschuifWachtrij()**

Deze functie schuift alle waardes in de wachtrij met een plek op. De index positie 0 in de wachtrij wordt verwijderd en op de hoogste index positie in de array wachtrij wordt een lege waarde geplaats

* **showWachtrij()**

Dit is een debug functie om te kijken welke waardes aanwezig zijn in de wachtrij en wanneer deze wordt geleegd

* **setupBuzzer()**

Deze functie zet de pin mode van de buzzer op output er een toon kan worden gegenereerd met de buzzer

* **buzzerBeep()**

Deze functie laat de buzzer een toon generen gebaseerd op de int ‘interval’ die wordt meegegeven

* **countDown()**

Deze functie telt af van 6 naar 0 om de 1000ms, zodra get cijfer kleiner is dan 0 wordt de teller weer op 6 gezet en de boolean waarde van isCountDownKlaar veranderd naar true

* **delayTimer()**

Deze functie krijgt een int ‘interval in milliseconden’ en checkt of de waarde van de ingebouwde functie Mills() groter is dan de waarde van previousMillisDelay plus het meegegeven interval

* **resetDelayTimer()**

Deze functie zet de waarde van preciousMillisDelay naar de huidige waarde van Mills()

* **setupLed()**

Deze functie zet de pin mode van de led op output zodat we de ledlampen licht kunnen laten gegeven. Dit gebeurt twee keer omdat het rode, gele en groene licht per stoplicht in een aparte array groep zitten.

* **stoplichtRood()**

Deze functie krijgt een int array i.e. stoplicht mee, vervolgens zet deze functie de rode lamp van het betreffende stoplicht aan en de andere lampen uit

* **stoplichtGeel()**

Deze functie krijgt een int array i.e. stoplicht mee, vervolgens zet deze functie de gele lamp van het betreffende stoplicht aan en de andere lampen uit

* **stoplichtGroen()**

Deze functie krijgt een int array i.e. stoplicht mee, vervolgens zet deze functie de groene lamp van het betreffende stoplicht aan en de andere lampen uit.

* **zuidStoplichtRood()**

Deze functie zet de rode lamp van het zuid stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **zuidStoplichtGeel()**

Deze functie zet de gele lamp van het zuid stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **zuidSstoplichtGroen()**

Deze functie zet de groene lamp van het zuid stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **noordStoplichtRood()**

Deze functie zet de rode lamp van het noord stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **zuidStoplichtGeel()**

Deze functie zet de gele lamp van het noord stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **noordStoplichtGroen()**

Deze functie zet de groene lamp van het noord stoplicht aan en de rest van de lampen op het zuid stoplicht uit

* **alleStoplichtGroen()**

Deze functie zet alle stoplichten op rood

* **waarschuwingsLicht()**

Deze functie krijgt een int ‘interval’ me, de gele lampen op alle stoplichten gaan knipperen gebaseerd op het ‘interval’ dat wordt meegegeven in milliseconden

* **setupPotmeter()**

Deze functie zet de pin mode van de potmeter op input zodat er waargenomen wat de huidige waarde van de potentiometer is.

* **getPotmeterValue()**

Deze functie geeft ons de analoge waarde van de potentiometer terug.

* **getMappedPotmeterValue()**

Deze functie krijgt een int ‘max’ mee en vertaald de standaardwaarde van de potmeter naar waardes binnen een bereik van 1 en de meegegeven int ‘max’.

* **setupServo()**

Deze functie zorgt ervoor dat we de servo motor kunnen bewegen. Tegelijkertijd wordt de huidige servo waarde bijgewerkt naar de dicht positie

* **servoNaarPositie()**

Deze functie krijgt een int ‘positie’ mee en zet de servo motor op de positie van de meegegeven int ‘positie’

* **updateServo()**

Deze functie krijgt een int ‘targetPositie mee en zet de servo motor geleidelijk op de positie van de meegegeven int ‘targetPositie’ met een interval van 50 milliseconden

* **slagboomOpen()**

Deze functie opent de slagboom.

* **slagboomSluiten()**

Deze functie sluit de slagboom.

* **setupShiftRegister()**

Deze functie zet de pin mode van de ST\_CP, SH\_CP en de DS-pin naar output zodat we hier bytes naartoe kunnen schrijven.

* **clearShiftRegister()**

Deze functie schrijft een lege waarde naar het schiftregister.

* **clearShiftRegister()**

Deze functie schrijft een lege waarde naar het schiftregister.

* **loopStateMachine()**

Deze functie zorgt ervoor dat de state machine zoals [**hier**](#_STATEMACHINE)beschreven intact blijft.

* **setupStateMachine()**

Deze functie zorgt ervoor dat de state machine zoals [**hier**](#_STATEMACHINE)overgaat naar de start fase.

