Verkeerslichten

2 Protocol 3

Voorbeelden 3

3 Een basis versie 4

3.2 State diagram 4

3.3 Opdracht 5

3.4 Test de opstelling 5

4 Een veiligere versie 6

4.2 State diagram 6

4.3 Opdracht 7

5 Optioneel: Een non-blocking versie 8

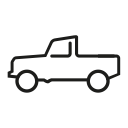
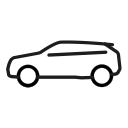
5.1 Opdracht 8

1 Inleiding

Voor de slave mag je de voorbeeld-code gebruiken die is te vinden via: <https://git.fhict.nl/technology/t-sem2-code/-/tree/master/es2/StartPointForExercises/Assignments/TrafficLight/slave>

Tijdens weg werkzaamheden komt het voor dat een rijstrook afgesloten moet worden. Om het verkeer in goede banen te leiden moet er een verkeerslicht installatie ontworpen worden.

Zie situatie hier beneden:



Elk verkeerslicht bestaat uit een Arduino, een rode en groene led. De twee verkeerslichten zullen met elkaar moeten communiceren via een serieel protocol om het verkeer op een veilige manier te regelen. De verkeerslichten wisselen na een vaste tijd.

Omdat de meeste Arduino’s geen 2de seriële poort bezitten maak je gebruik van de [SoftwareSerial Library](https://www.arduino.cc/en/Reference/SoftwareSerial).

# 2 Protocol

De master stuurt berichten naar de slave. De slave mag alleen een bericht sturen als reply op een ontvangen bericht.

Voor master en slave zien de berichten als volgt opgebouwd:

**<begin-karakter><bericht><eind-karakter>**

In tabel 2.1 staat een overzicht van de speciale karakters. De request- en reply-berichten van de master en slave zijn de te vinden in tabel 2.2 en 2.3.

|  |  |
| --- | --- |
| Speciale karakters | Waarde |
| begin-karakter | # |
| eind-karakter | ; |

Tabel 2.1: speciale karakters

|  |  |
| --- | --- |
| Bericht | Betekenis |
| RED | Zet het stoplicht op rood |
| GREEN | Zet het stoplicht op groen |
| BEAT | De master stuurt dit commando naar de slave om te kijken de communicatie nog correct is. |

Tabel 2.2: berichten van de master.

|  |  |
| --- | --- |
| Bericht | Betekenis |
| ACK | Het bericht is bekend |
| NACK | Het bericht is niet bekend |

Tabel 2.3: berichten van de slave.

## Voorbeelden

In tabel 2.4 staan een aantal voorbeelden.

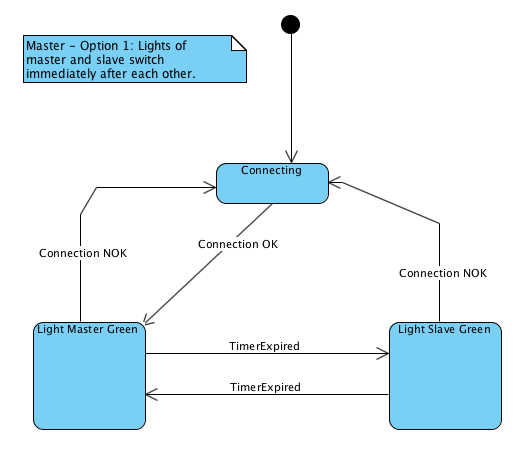
|  |  |
| --- | --- |
| Master request bericht | Slave reply bericht |
| #GREEN; | #ACK; |
| #ORANGE; | #NACK; |

Tabel 2.4: voorbeelden van communicatie tussen master en slave.

# 3 Een basis versie

## 3.2 State diagram

Voor deze versie van de verkeersregelaar op de master gaan we uit van het onderstaande state diagram (zie ook tabel 3.1 en 3.2).



Figuur 3.1: state diagram voor de master

|  |  |  |
| --- | --- | --- |
| State | Toelichting | Verkeerslichten |
| Light Master Green | Normale operatie | Master verkeerslicht = Green Slave verkeerslicht = Rood |
| Light Slave Green | Normale operatie | Master verkeerslicht = Rood Slave verkeerslicht = Green |
| Connecting | De master probeert herhalend een verbinding te maken met de slave. | Master verkeerslicht = Rood Slave verkeerslicht = Rood |

Tabel 3.1: states van de master

|  |  |
| --- | --- |
| Event | Toelichting |
| Connection OK | De master heeft een bericht gestuurd naar de slave en een correct antwoord ontvangen. |
| Connection NOK | De master heeft een bericht gestuurd naar de slave en een incorrect of gewenantwoord ontvangen. |
| TimerExpired | De tijd die een stoplicht op groen mag staan is verlopen. |

Tabel 3.2: events

## 3.3 Opdracht

De implementatie moet aan de volgende voorwaarden voldoen:

* + Je maakt gebruik van de gegeven code voor de slave (zie lesmaterialen).
  + De code voor de master maak je helemaal zelf.
  + In de code moeten de gegeven states en events duidelijk herkenbaar terugkomen.
  + In de code moet het gegeven protocol duidelijk herkenbaar terugkomen.
  + Laat zien dat je de algemene leerdoelen van deze course beheerst. Maak bijvoorbeeld zo min mogelijk gebruik van globale- en static-variabelen.
  + Het is verboden gebruik te maken van de Arduino **S**tring.h (grote **S**) library. Maak voor het opslaan van strings gebruik van ‘char’ arrays. Het is wel toegestaan om functies uit de **s**tring.h (klieine **s**) library te gebruiken.

* + Als de master een bericht naar de slave stuurt dan wacht deze meteen op een antwoord van de slave. Als er na 500 ms geen correct is ontvangen dan treedt een ‘Connection NOK’ event op. Omdat de tijd tussen zenden en het ontvangen relatief kort is, hoeft er geen extra state te worden geïmplementeerd waarin de Arduino aan het wachten is. (Wees je ervan bewust dat de Arduino hierdoor wel ‘[blocked](https://www.beningo.com/embedded-basics-blocking-vs-non-blocking-drivers/)’ is en gedurende deze tijd niet meer op andere events zal reageren).

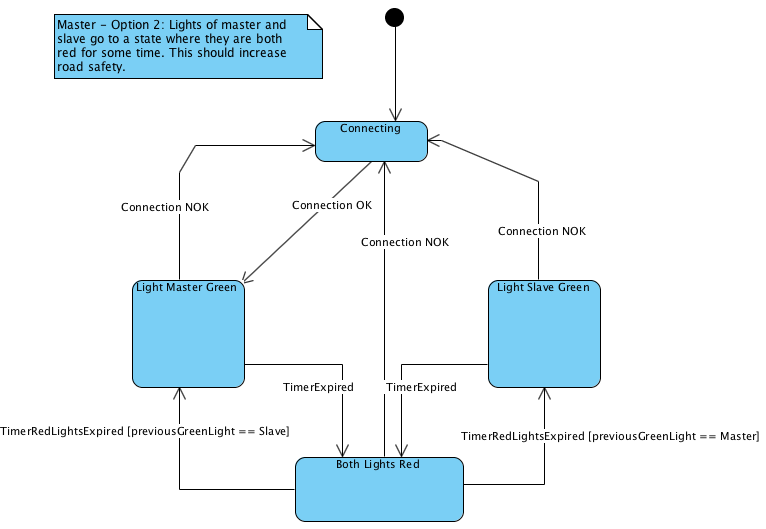
## 3.4 Test de opstelling

* Sluit op elke Arduino een groene en rode led aan voor de demo.
* Laat de werking zien:
  + onder normale omstandigheden.
  + als er geen communicatie tussen de arduino’s meer is.
  + als de verbinding tussen de arduino’s weer hersteld is.

# 4 Een veiligere versie

## 4.2 State diagram

Als er geen tussentijd is tussen de tijd dat een stoplicht naar groen schakelt dan kan tegenmoet komend verkeer op elkaar botsen. Voor een veiligere versie van de verkeersregelaar op de master gaan we uit van het onderstaande state diagram (zie ook tabel 4.1 en 4.2).



Figuur 4.1: state diagram voor de master

|  |  |  |
| --- | --- | --- |
| State | Toelichting | Verkeerslichten |
| Both Lights Red | Het nog reeds op de rijbaan aanwezige verkeer krijgt tijd om plaats te maken voor opkomend vekeer. | Master verkeerslicht = Rood Slave verkeerslicht = Rood |

Tabel 4.1: states van de master

|  |  |
| --- | --- |
| Event | Toelichting |
| TimerRedLightsExpired | De tijd die beide stoplichten op rood staan om botsingen te voorkomen. |

Tabel 4.2: events

## 4.3 Opdracht

Implementeer een veiligere versie van de master code.

# 5 Optioneel: Een non-blocking versie

## 5.1 Opdracht

In de vorige versie is de Arduino ‘blocked’ op het wachten op een reply van de Arduino. Pas het state diagram en code aan zodat de Arduino na het sturen van het bericht niet meteen gaat wachten op een reply van de slave.

Tip: de code van de slave mag je aanpassen zodat deze even wacht met het versturen van een reply. Nog uitdagender wordt het als je tussen het versturen van elke karakter van de slave-reply een delay inbouwt. De master zal hierdoor wat geduldiger moeten zijn met het teruglezen van de reply.