Master-Master communicatie met I2C

Door: Lucas van Lippen

Klas: Fontys ICT T3-DB

Datum: 9-1-22

Versie 0.2

Diagram

Description automatically generated

# Samenvatting

In dit document documenteer ik hoe ik mijn master master I2C communicatie systeem implementatie heb gemaakt. Eerst begin ik met een ontwerp te maken, en daarna laat ik zien wat voor protocol ik daarbij heb bedacht.   
Als dat gedaan is maak ik mijn eigen master master I2c implementatie met het ontwerp en de protocol implementatie in mijn design.

Contents

[Samenvatting 2](#_Toc123134043)

[Inleiding 4](#_Toc123134044)

[Aanleiding 4](#_Toc123134045)

[Onderwerp 4](#_Toc123134046)

[Ontwerp 5](#_Toc123134047)

[Het protocol 6](#_Toc123134048)

[Beschrijving 6](#_Toc123134049)

[Schema 6](#_Toc123134050)

[Protocol Implementatie 7](#_Toc123134051)

[Menu arduino 7](#_Toc123134052)

[Game arduino 8](#_Toc123134053)

[Bibliografie 9](#_Toc123134054)

# Inleiding

## Aanleiding

Om een beter begrip te krijgen van hoe master communicatie werkt met het I2C protocol hebben we de opdracht gekregen om een toepassing te maken waar we werken met I2C-apparaten.

## Onderwerp

Ik heb de opdracht gekregen om een toepassing te maken die gebruik maakt van 2 arduino borden die dienen als de master over een schermpje en een joystick die dienen als slave apparaten. Mijn idee is nu om een arduino te laten werken als menu voor het spel en een andere arduino werkt als de game. Het is nu dus belangrijk dat ik de arduino ’s goed met elke coördineer zodat ze niet tegelijk informatie gaan vragen of sturen naar de slave apparaten. Hiervoor ga ik dus een protocol maken.

# Ontwerp

Ik heb de opdracht gekregen om een systeem te maken met meerdere I2C apparaten waarvan er 2 dienen als master en 2 dienen als slave.   
Het grote verschil tussen een master en een slave is dat een slave altijd een adres heeft en een master niet. Een master apparaat kan commando’s sturen naar een slave en op elk moment gegevens opvragen van een slave apparaat.

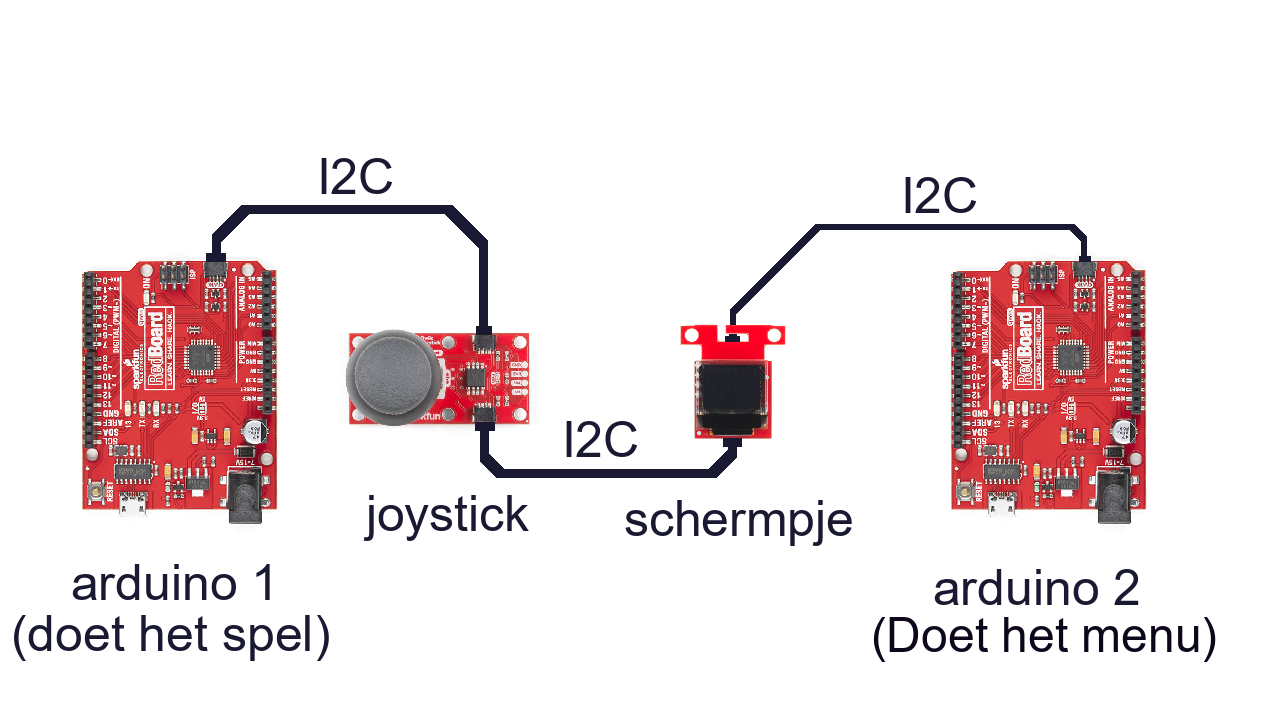
Mijn plan is nu om een systeem te maken met 2 arduino bordjes als masters en een joystick module en een OLED-module als slave apparaten.  
Hierbij wil de twee arduino apparaten gebruiken als de master die het schermpje aansturen en de inputs van de joystick ontvangen.

De eerste arduino bestuurt dan een soort Pong-achtig spel wat je dan dus bestuurt met je joystick module en kan zien op het schermpje.  
Het tweede master device wordt als eerste gestart en geeft je een menu van het spel waar je het moeilijkheidsgraad kan instellen. Zodra je in het menu je moeilijkheidsgraad hebt geselecteerd stuurt de tweede arduino een signaal naar de eerste arduino zodat hij weet dat hij kan starten en welk moeilijkheidsgraad hij moet aanhouden.

Dus als je het spel start dan heeft de tweede arduino eerst controle over het schermpje en laat hij het menu zien. Zodra er dan een moeilijkheidsgraad is geselecteerd start het spel en bestuurt de eerste arduino het scherm.

Zo geeft de eerste arduino dan dus een signaal naar de tweede arduino dat hij nu de baas is en waar de bal zich bevind op het scherm en hierop reageert hij dan met de nieuwe positie van waar zijn staafje moet staan en geeft hierbij dus weer controle terug naar de eerste arduino.

Hierbij zal ik dus een opstelling krijgen die er dus zo uit zal zien:



(robot-electronics, 2022)

# Het protocol

## Beschrijving

Het protocol is op te delen in commando’s die de menu arduino naar de game arduino stuurt en commando’s die de game arduino naar de menu arduino stuurt.

Om een bericht te sturen naar de andere arduino wordt er eerst een byte gestuurd die dient als een soort van commando byte. Daarna wordt er in het geval van tijdens het spel wordt er een extra byte daarna gestuurd die dient als een parameter die de andere controller kan gebruiken om een juist antwoord terug te sturen.

De menu arduino polt bij de game arduino voor een antwoord om er zeker van te zijn dat de game voorbij is.

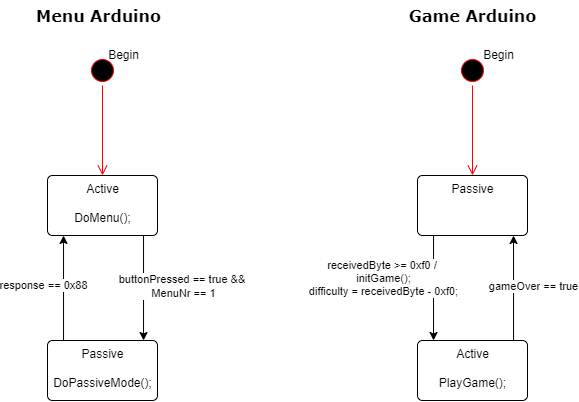
**Menu naar game byte protocol**

|  |  |  |
| --- | --- | --- |
| **commando** | **Eerste nibble** | **Tweede nibble** |
| Start Game | 0xF… | 0: easy mode  1: hard mode  2: silly mode |

**Game naar menu byte protocol**

|  |  |  |
| --- | --- | --- |
| **commando** | **Eerste nibble** | **Tweede nibble** |
| End game | 0x8… | 8: game is beëindigd en de master arduino kan weer de controle overnemen. |

## Schema



Beide masters hebben hun eigen statemachine die er voor zorgt dat ze op het juiste moment de master zijn over de slaves.

# Protocol Implementatie

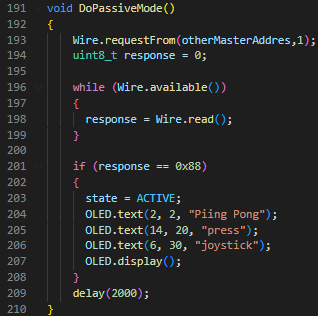
Voor de protocol implementatie zijn er een paar belangrijke code blokken die er voor zorgen dat de masters met elkaar samen kunnen werken.

## Menu arduino

Text

Description automatically generated

Met deze code gaat hij van active mode naar pasive mode en stuurt hij een seintje naar de andere master dat hij mag beginnen met controle te nemen over de slave devices.



Hier vraagt de menu arduino bij de game arduino of het spel al afgelopen is elke 2 seconden om het I2C signaal niet te vaak laten interfereren met het verloop van het spel.

## Game arduino

A screenshot of a computer

Description automatically generated with medium confidence

Als de state Pasive is en de game arduino ontvangt een byte die begint met 0xf\_ dan zal de game starten. Het moeilijkheidsgraad wordt hier ook gelijk ingesteld aan de hand van de tweede nibble.

Graphical user interface, text, application

Description automatically generated Text

Description automatically generated

Als De game over is dan zal de state naar pasive gaan en zal hij zodra er naar gevragen wordt een antwoord terug geven dat de game over is met een 0x88.

# Bibliografie

robot-electronics. (2022). *i2c-tutorial*. Retrieved from www.robot-electronics.co.uk: https://www.robot-electronics.co.uk/i2c-tutorial