Soft UART Challenge

Tiemon Steeghs

Inhoud

[1. Aanleiding 3](#_Toc126916067)

[2. Probleemstelling 3](#_Toc126916068)

[3. Onderzoek 4](#_Toc126916069)

[4. Proof of concept 5](#_Toc126916070)

[5. Testen 6](#_Toc126916071)

[6. Conclusie 7](#_Toc126916072)

[7. Bronvermelding 8](#_Toc126916073)

# Aanleiding

Om UART nog beter te begrijpen ga ik het in deze challenge in een moeilijkere context laten zien. Doormiddel van software matig UART toe te passen hoop ik UART beter te begrijpen en hoop ik dit ook te kunnen laten zien.

# Probleemstelling

UART communicaties maak ik al een lange tijd gebruik van. Maar ik heb eigenlijk al die tijd de standaard functies gebruikt die in de Arduino library worden meegegeven. Ik weet dus eigenlijk nog helemaal niet hoe je dit handmatig zou doen zonder gebruik te maken van libraries.

# Onderzoek

## Hoofdvraag

**Hoe zet je een softwarematige UART communicatie op tussen een Arduino en een Laptop zonder gebruik te maken van bestaande libraries?**

## Deelvraag 1

**Wat is softwarematige UART?**

Software matige UART is een implementatie van UART die embedded systemen zonder een UART module toch de mogelijkheid geeft om een UART communicatie op te zetten. Soft UART simuleert eigenlijk hard UART, doormiddel van de communicatie snelheid, de bit formatie van het bericht en de verwerking van fouten te regelen.

Software matige UART is een goede oplossing voor systemen die niet de hardware hebben die UART ondersteunt. Over het algemeen is software matige UART wel minder snel en ook minder betrouwbaar dan hardware versies, vooral bij de versturing van grote hoeveelheden data wordt dit duidelijk.

## Deelvraag 2

**Wat is het verschil tussen soft UART en hard UART?**

Software UART is een implementatie van UART die volledig software matig is opgesteld. Dit houd in dat er geen speciale hardware module wordt gebruikt, om de UART werkelijkheid te maken. Hardware UART daarentegen, maakt juist gebruik van een module die UART ondersteunt.

De grootste technische verschillen tussen de twee, zijn de betrouwbaarheid en de snelheid. Software UART is over het algemeen minder betrouwbaar en ook minder snel dan hardware UART.

## Deelvraag 3

**Wat is een statemachine?**

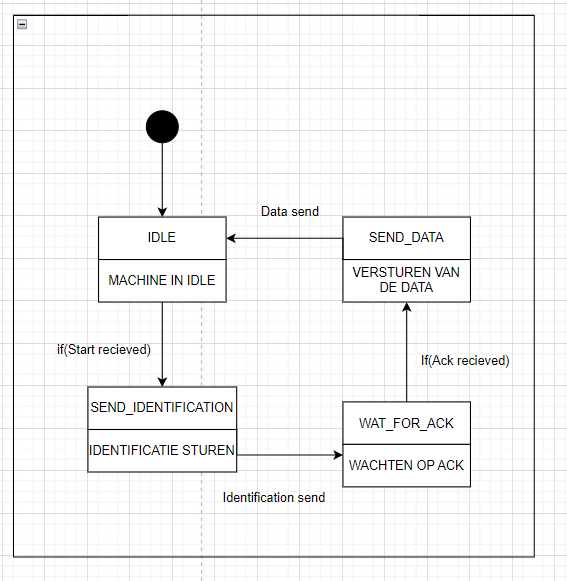
Een statemachine is een model waarin wordt beschreven hoe een machine werkt. Je kan in dit model zien wat voor processen er plaats vinden en hoe deze op elkaar in spelen. Een statemachine bestaat uit een selectie van toestanden waarin de machine zich kan bevinden. Aan de hand van “Events” of gebeurtenissen veranderd de machine van toestand.

## Deelvraag 4

**Waaruit bestaat de statemachine die voor deze challenge van toepassing zou zijn?**

Voor deze challenge maak ik een statemachine die zich in vier toestanden kan bevinden: IDLE, SEND\_IDENTIFICATION, WAIT\_FOR\_ACK en SEND\_DATA. De machine zal aan de hand van commando’s die hij van de laptop ontvangt veranderen van state.

Als eerste start de machine in de “IDLE” toestand. In deze toestand doet de machine nog niet veel, hij wacht namelijk op een start commando van de laptop. Op het moment dat hij het start commando heeft ontvangen stuurt de state machine een identificatie bericht in de SEND\_IDENTIFICATION state. Na het bericht gestuurd te hebben gaat de machine naar de WAIT\_FOR\_ACK toestand. In deze toestand wacht de machine weer op een bericht van de laptop. Op het moment dat hij dit bericht ontvangen heeft zal hij naar de SEND\_DATA toestand gaan en de data sturen. Na de data gestuurd te hebben gaat de machine weer terug naar de IDLE toestand.



## Deelvraag 5

**Hoe zet je software matige UART op?**

Software matige UART kan je opzetten doormiddel van alle functionaliteit van de hardware UART te simuleren. Bij mij bestaat de soft UART uit één transmitter (redboard) en een reciever (arduino). Hierbij beginnen we met het instellen van de baudrate. Nadat deze is ingesteld moeten we ook de bit-time bepalen, dit is steeds de tijd die de arduino moet wachten om de volgende bit te gaan sturen. Het is natuurlijk ook belangrijk dat de arduino en de redboard op dezelfde pin gaan communiceren. Na deze dingen ingesteld te hebben kan de daadwerkelijke communicatie beginnen. Deze is hieronder verder beschreven aan de hand van twee flowcharts.

Hieronder zie je de flowchart van soft UART transmitter. In deze flowchart is de volgende flow beschreven:

1. Als eerste begint het met het sturen van de start bit. Hierbij wordt de TX pin op laag gezet om aan te geven dat de data begint.
2. Hierna wordt er 104 ms gewacht op zo de juiste timing te hebben met de reciever.
3. Als deze tijd voorbij is zal de eerste data bit verzonden worden.
4. Dan wordt er weer 104 ms gewacht
5. Vervolgens wordt er gecontroleerd of het data bericht af is, zo niet terug naar (3)
6. Als tenslotte de data af is wordt er nog een stop bit gestuurd doormiddel van de pin op hoog te zetten.

Afbeelding met diagram

Automatisch gegenereerde beschrijving

De reciever flowchart werkt als volgt:

1. Als eerste is de setup van de code, hier worden alle juiste instellingen gemaakt
2. Na de setup wordt er gewacht op een start bit
3. Bij ontvangst van een start bit wordt er gewacht totdat er een volledige byte aan data is ontvangen
4. Als de volledige data is ontvangen wordt er nog gecontroleerd op een stopbit, als deze niet aanwezig is gaan we terug naar wachten op de start bit (2)
5. Tenslotte als de stopbit is ontvangen zal het karakter worden toegevoegd aan de string

Afbeelding met diagram

Automatisch gegenereerde beschrijving

# Testen

## Transmitter

De eerste versie van mijn transmitter was nog niet helemaal goed. Deze versie maakte nog gebruik van delay micros waardoor hij niet elke snelheid aan kon. De data werd in deze versie wel goed verstuurt zoals je hieronder ziet bij de grafiek van de logic analyzer.

# Afbeelding met grafiek Automatisch gegenereerde beschrijving

Na de eerste versie van de transmitter gemaakt te hebben ben ik verder gegaan met de volgende. Deze versie maakt geen gebruik van delays, maar inplaats daarvan micros. Door steeds de tijd die de code kost om uit te voeren mee te rekenen met de daadwerkelijke wachttijd, kan je een stuk preciezer versturen. Echter ging dit in het begin nog niet goed. Je ziet hieronder dat mijn logic analyzer de boel nog niet goed leest. De data wordt wel juist gestuurd maar de timing klopt nog niet, waardoor het slecht wordt gelezen.

# 

Tenslotte mijn laatste versie van de transmitter had een kleine aanpassing. Het probleem lag hem in de variabel waarin ik steeds de micros in opsloeg. Deze variabel was niet van hetzelfde type waardoor het timing-gedeelte van de code niet goed ging. Na dit aan te passen klopte de timing wel en, de logic analyzer leest de data nu perfect af zoals je hieronder kan zien.

# Afbeelding met grafiek Automatisch gegenereerde beschrijving

# Bronvermelding

Software uart:

[**https://doc.riot-os.org/group\_\_drivers\_\_soft\_\_uart.html**](https://doc.riot-os.org/group__drivers__soft__uart.html)

Software matige serial communicatie op de arduino documentatie:

<https://docs.arduino.cc/learn/built-in-libraries/software-serial>

<https://deepbluembedded.com/stm32-external-interrupt-example-lab/>