Opdrachten week 8 les 1 – UART (Universal Asynchronous Receiver-Transmitter)

In de opdrachten van week 4 en 5 heb je geleerd om digitale signalen parallel in te lezen of uit te sturen. Zo heb je bijvoorbeeld de RGB-led met drie parallelle signalen aangestuurd. Als er veel data overgedragen moet worden tussen twee systemen is het vaak eenvoudiger om gebruik te maken van een seriële verbinding. In een parallelle verbinding worden de verschillende bits parallel aan elkaar over verschillende draden getransporteerd. Bij een seriële verbinding worden de verschillende bits na elkaar over dezelfde draad getransporteerd.

Als de seriële verbinding twee systemen met elkaar verbindt, dan wordt dit een point-to-point verbinding genoemd. Bij een busverbinding kunnen meer dan twee systemen met elkaar communiceren. Degene die bepaalt wie er dan mag zenden wordt de busmaster genoemd. De overige systemen worden slaves genoemd[1]. Bij een zogenaamd multi-master bus kunnen verschillende systemen de rol van master vervullen. Als de communicatie tegelijkertijd in twee richtingen kan plaatsvinden, dan wordt dit een full duplex verbinding genoemd. Er zijn dan twee draden voor de overdracht van data nodig. Al het kloksignaal met de data meegestuurd wordt, dan wordt dit een synchrone verbinding genoemd. Bij een asynchrone verbinding wordt het kloksignaal niet met de data meegestuurd, maar worden afspraken gemaakt over de snelheid waarmee de bits getransporteerd worden.

Er zijn zeer veel standaarden voor seriële verbindingen beschikbaar die alle hun specifieke voor- en nadelen hebben. Om er een paar te noemen: CAN, DMX512, Ethernet, I^2C, IrDA, MIDI, PCI Express, Profibus, RS-232, RS-422, RS-485, SATA, SPI en USB. In het verleden was elke pc uitgerust met een RS-232 poort (COM-poort genoemd). Tegenwoordig is elke pc voorzien van USB-poorten. Het is mogelijk om berichten die volgens het RS-232-protocol zijn gecodeerd over USB te transporteren. Er is dan geen fysieke COM-poort op de pc nodig maar er wordt gebruik gemaakt van een zogenoemde virtuele COM-poort. We maken in dit geval geen gebruik van de in de RS-232 gedefinieerde signaalniveaus, maar wel van de overige zaken die in de RS-232-standaard zijn vastgelegd. Deze manier van het gebruik van de RS-232-standaard wordt vaak UART-communicatie genoemd. UART staat voor Universal Asynchronous Receiver-Transmitter. Een UART-verbinding is een asynchrone, full duplex, point-to-point verbinding. Communicatie via RS-232 is veel eenvoudiger dan communicatie

[1] Deze terminologie wordt tegenwoordig door velen als ongepast ervaren omdat het verwijst naar het slavernijverleden. Zie https://en.wikipedia.org/wiki/Master/slave_(technology)#Terminology_concerns. Omdat de documentatie van Texas Instruments deze terminologie gebruikt, hebben wij besloten om de begrippen master en slave bij deze cursus toch te gebruiken, om verwarring te voorkomen.

via USB. Om deze reden maken veel microcontrollers gebruik van UART-communicatie in plaats van USB. De microcontroller die wij gebruiken, de MSP430G2553, heeft hardware support voor I^2C-, IrDA-, SPI- en UART-communicatie. Om deze protocollen te ondersteunen beschikt de MSP430G2553 over twee zogenoemde USCI (Universal Serial Communication Interface) peripherals. USCI_A0 ondersteunt UART-, IrDA- of SPI-communicatie. USCI_B0 ondersteunt I^2C- of SPI-communicatie.

In deze les gaan we een UART-verbinding opzetten tussen de MSP430G2553 en een pc. Op de pc gebruiken we daarbij een virtuele COM-poort. Tijdens deze les leer je:

- wat een UART-verbinding is;
- hoe je een UART-verbinding kunt opzetten tussen de MSP430G2553 en een pc;
- hoe je karakters en strings kunt gebruiken in een C-programma.

Een prima introductie in UART-communicatie is te vinden op Wikipedia. **Lees nu https://-en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter tot het kopje 'History'.**

Deze week gaan we de USCI_A0 van de MSP430G2553 gebruiken. In hoofdstuk 15 van de MSP430x2xx Family User's Guide[2] wordt beschreven hoe je deze peripheral kunt gebruiken voor UART-communicatie.

Lees nu paragraaf 15.1, 15.2 en 15.3 tot 15.3.3 van de MSP430x2xx Family User's Guide.

8.1.1 Maak een nieuw project aan in CCS en noem dit `echoput`. Zet de klokfrequentie van de DCO op 1 MHz. In paragraaf 15.3.1 van de MSP430x2xx Family User's Guide wordt uitgelegd dat je de USCI in een aantal specifieke stappen moet initialiseren.

De eerste stap is het setten van de bit `UCSWRST` in het register `UCA0CTL1`. In de User's Guide is de machinecode voor deze operatie gegeven. Wat is de C-code om de bit `UCSWRST` in het register `UCA0CTL1` één te maken? Plaats deze code in je `main`-functie.

De tweede stap is het initialiseren van alle USCI-registers. Deze stap vullen we later in. Zet commentaar in je `main`-functie zodat je niet vergeet om deze stap later in te vullen.

De derde stap is het configureren van de poorten. Bij UART-communicatie maken we gebruik van een pin waarmee we zenden, TXD (Transmit eXchange Data) genoemd en een pin waarmee we ontvangen, RXD (Receive eXchange Data) genoemd. Zoek

[2] Deze kun je vinden in de Resource Explorer van CCS of online op: `http://www.ti.com/lit/ug/slau144k/slau144k.pdf`.

in de MSP4302x53 datasheet[3] op met welke pinnen de TXD en RXD van USCI_A0 verbonden kunnen worden. Zoek uit hoe de betreffende poort geïnitialiseerd moet worden[4]. Bepaal de code die nodig is om de TXD en RXD pinnen van USCI_A0 correct te configureren en plaats deze code in je `main`-functie.

De vierde stap is het clearen van de bit `UCSWRST` in het register `UCA0CTL1`. In de User's Guide is de machinecode voor deze operatie gegeven. Wat is de C-code om de bit `UCSWRST` in het register `UCA0CTL1` nul te maken? Plaats deze code in je `main`-functie.

Omdat wij een verbinding willen opzetten kun je paragrafen 15.3.3 tot en met 15.3.5 van de User's Guide overslaan. Paragrafen 15.3.6 tot en met 15.3.15 van de User's Guide zijn relevant voor deze opdracht, maar bevatten wel heel veel gedetailleerde informatie.

8.1.2 We gaan nu de tweede stap van het initialiseren van de USCI invullen waarbij de volgende USCI-registers geïnitialiseerd moeten worden:

- `UCA0CTL0`: USCI_A0 control register 0
- `UCA0CTL1`: USCI_A0 control register 1
- `UCA0BR0`: USCI_A0 Baud rate control register 0
- `UCA0BR1`: USCI_A0 baud rate control register 1
- `UCA0MCTL`: USCI_A0 modulation control register

De betekenis van de verschillende bits in deze registers wordt beschreven in paragraaf 15.4 van de MSP430x2xx Family User's Guide. Bepaal de waarde die in het register `UCA0CTL0` moet worden gezet als gegeven is dat we geen pariteit[5] willen gebruiken, het LSB eerst willen verzenden en ontvangen, 8-bits data en één stop bit willen gebruiken en de USCI in Asynchronous UART mode willen gebruiken.

Bepaal de waarde die in het register `UCA0CTL1` moet worden gezet als gegeven is dat we SMCLK willen gebruiken en *geen* gebruik willen maken van interrupts, dormant mode, adressen en breaks. Let erop dat de bit `UCSWRST` geset moet blijven zoals gespecificeerd is in paragraaf 15.3.1van de MSP430x2xx Family User's Guide.

[3] Deze kun je vinden in de Resource Explorer van CCS of online op: http://www.ti.com/lit/ds/symlink/msp430g2553.pdf.

[4] Deze informatie kun je vinden op pagina 43 van de MSP4302x53 datasheet.

[5] Een pariteitsbit kan gebruikt worden om fouten in de dataoverdracht te detecteren, zie eventueel https://nl.wikipedia.org/wiki/Pariteitsbit.

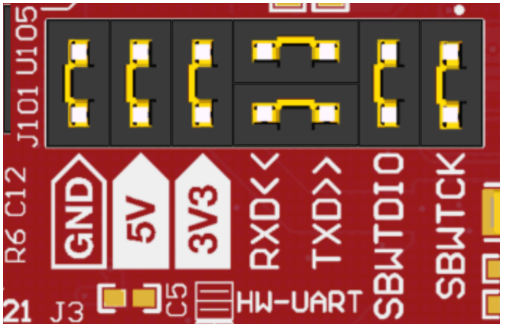
Omdat we gebruik maken van een hoge klokfrequentie (SMCLK = 1 MHz) maken we gebruik van de oversampling mode. Bepaal welke waarde we in de registers `UCA0BR0` en `UCA0BR1` moeten zetten als we een baudrate van 9600 baud willen gebruiken. Je kunt dit bepalen met behulp van de formule die gegeven is in paragraaf 15.3.10.2 of met behulp van tabel 15-5 uit de MSP430x2xx Family User's Guide.

Bepaal de waarde die in het register `UCA0MCTL` moet worden gezet als we een baudrate van 9600 baud in oversampling mode willen gebruiken. De benodigde waarden van `UCBRFx` en `UCBRSx` kun je bepalen met behulp van de formule die gegeven is in paragraaf 15.3.10.2 of met behulp van tabel 15-5 uit de MSP430x2xx Family User's Guide.

Schrijf de benodigde C-code om de USCI-registers te initialiseren en plaats deze code op de juiste plaats in je `main`-functie[6].

8.1.3 Voordat we het programma gaan voltooien, gaan we eerst de hardwarematige UART-verbinding aanbrengen. Er zijn twee mogelijkheden:

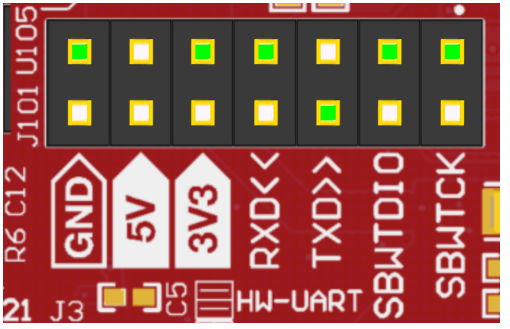
- We plaatsen de MSP430G2553 op het MSP-EXP430G2ET LaunchPad ontwikkelbord en zetten de vier jumpers voor GND, 3V3, SBWTDIO en SBWTCK terug. Daarnaast moeten we dan twee jumpers gebruiken om de TXD van de (virtuele) COM-poort van de pc te verbinden met de RXD van de MSP430G2553 en om de RXD van de (virtuele) COM-poort van de pc te verbinden met de TXD van de MSP430G2553. *Let op!* Dit doe je door de jumpers in de HW UART stand te plaatsen (haaks op de andere jumpers), zie figuur 1.



Figuur 1: Benodigde jumperinstellingen op de MSP-EXP430G2ET LaunchPad om de USCI peripheral in de MSP430G2553 als hardware UART te kunnen gebruiken.

[6] Je kunt hier controleren of je dit correct hebt gedaan: `opdr8.1.2.c`.

- We plaatsen de MSP430G2553 op een breadboard en verbinden de GND, 3V3, SBWTDIO en SBWTCK op de bekende wijze. Daarnaast moeten we de TXD van de connector J101 verbinden met de TXD van de MSP430G2553 (P1.2) en moeten we de RXD van de connector J101 verbinden met de RXD van de MSP430G2553 (P1.1). *Let op!* Dit doe je door de pinnen te gebruiken die groen gemarkeerd zijn in figuur 2.



Figuur 2: Benodigde pinnen op de MSP-EXP430G2ET LaunchPad om de USCI peripheral in de MSP430G2553 op het breadboard als UART te kunnen gebruiken.

8.1.4 We gaan nu de `while(1)`-lus in de `main`-functie invullen. We willen een echoput creëren die elk, via RXD ontvangen, karakter weer terugstuurt via TXD. Zodra er een karakter ontvangen is wordt de bit `UCA0RXIFG` in het register `IFG2` geset. Het ontvangen karakter bevindt zich dan in het register `UCA0RXBUF`. Als de USCI klaar is om een karakter te verzenden, dan is de bit `UCA0TXIFG` in het register `IFG2` geset. Het te versturen karakter kan dan in het register `UCA0TXBUF` geschreven worden. Karakters kunnen in de programmeertaal C opgeslagen worden in een variabele van het type `char`[7]. De code in de `while(1)`-loop moet de volgende stappen uitvoeren:

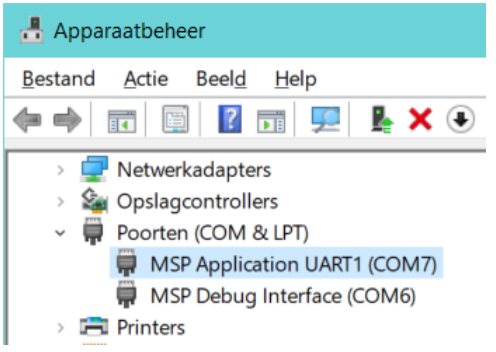
- Wacht tot de bit `UCA0RXIFG` in het register `IFG2` geset is.
- Maak de `char`-variabele c gelijk aan register `UCA0RXBUF`.
- Wacht tot de bit `UCA0TXIFG` in het register `IFG2` geset is.
- Maak het register `UCA0TXBUF` gelijk aan de variabele c.

8.1.5 Nu het programma op de MSP430G2553 compleet is moeten we nog een programma hebben op de pc dat in staat is via een (virtuele) COM-poort te communiceren volgens

[7] Zie eventueel http://en.cppreference.com/w/c/language/arithmetic_types#Character_types.

het UART-protocol. Code Composer Studio heeft een ingebouwde terminal die dit kan doen maar wij maken bij deze opdracht gebruik van het programma 'CoolTerm' dat je kunt vinden op: https://freeware.the-meiers.org/CoolTermWin64Bit.zip.

Je moet nu eerst uitzoeken met welke virtuele COM-poort je moet verbinden. Dit kun je vinden (als je bordje is aangesloten via de USB-kabel) in Apparaatbeheer (Engels: Device Manager) van Windows. Open Apparaatbeheer[8] en kijk onder 'Poorten (COM & LPT)' aan welke COM poort de 'MSP Application UART1' is gekoppeld, zie figuur 3. In mijn geval is dat COM7, bij jou kan dat een ander nummer zijn.



Figuur 3: Het te gebruiken COM-poortnummer kun je vinden in Apparaatbeheer.

Pak het bovenstaande zip-bestand uit en start het programma `CoolTerm.exe` en klik op 'Options'. Kies vervolgens de juiste COM-poort en stel deze als volgt in: Baudrate: 9600, Data Bits: 8, Parity: None en Stop Bits: 1.

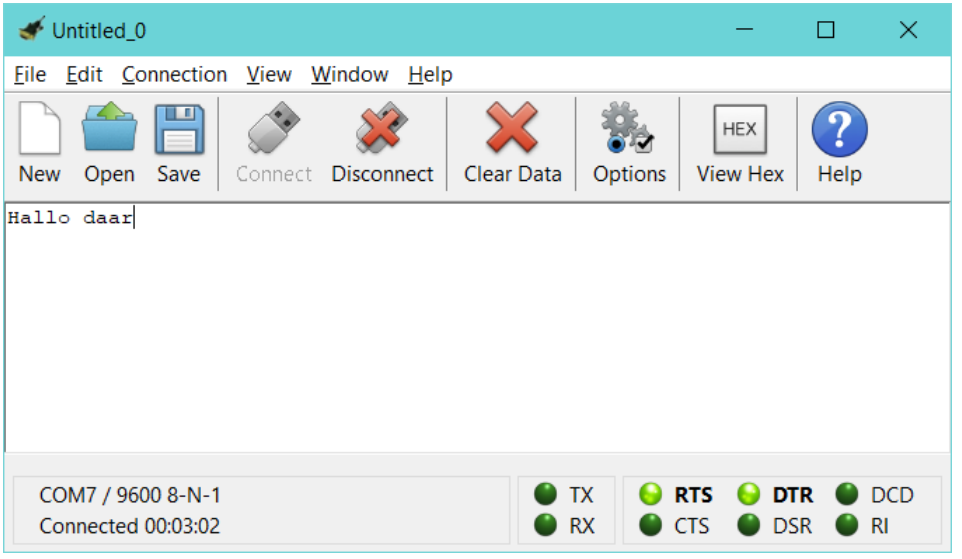
Klik op 'OK' en vervolgens op 'Connect'. Als je nu het programma `echoput` uitvoert op de MSP430G2553 en iets intypt in CoolTerm dan zal elk karakter dat je intypt teruggestuurd worden door de MSP430G2553 en zichtbaar worden in CoolTerm[9], zie figuur 4.

Traditioneel worden karakters in computerprogramma's gecodeerd in ASCII-code[10]. De ASCII-code is een 7-bits code die bestaat uit afdrukbare en niet-afdrukbare karakters. De niet-afdrukbare ASCII-karakters worden ook wel besturingskarakters genoemd. Enkele

[8] Klik op Start en type Apparaatbeheer of Device Manager om dit programma te openen.

[9] Als het niet werkt zoals verwacht, dan kun je hier controleren of je opdracht 8.1.4 correct hebt uitgevoerd: `opdr8.1.4.c`.

[10] ASCII staat voor American Standard Code for Information Interchange, zie eventueel https://en.wikipedia.org/wiki/ASCII.



Figuur 4: Alles wat verstuurd wordt vanuit de terminal naar de MSP430G2553 wordt weer teruggestuurd naar de terminal.

voorbeelden zijn CR (Cariage Return: verplaatst de cursor naar het begin van de regel) en BS (Backspace: verplaatst de cursor één positie terug.) Zoek in de ASCII-tabel van de afdrukbare karakters (https://en.wikipedia.org/wiki/ASCII#Printable_characters) de ASCII-code op voor de karakters 'A' en '1'. Zoek in de ASCII-tabel van de besturingskarakters (https://en.wikipedia.org/wiki/ASCII#Control_characters) de ASCII-code op voor de CR (Cariage Return). De 7-bits ASCII-code bevat alleen de karakters die in het (Amerikaans) Engels gebruikt worden. Karakters zoals ë, î en ç ontbreken in de ASCII-code. Om deze reden zijn er veel 8-bits uitbreidingen van de ASCII-code bedacht. Een veelgebruikte 8-bits karaktercode is ISO/IEC 8859-1[11]. Natuurlijk kunnen nog steeds niet alle wereldwijd gebruikte karakters gecodeerd worden met een 8-bits codering. Er zijn bij een 8-bits code immers maar $2^8 = 256$ verschillende karakters mogelijk. Om deze reden wordt heden ten dage veel gebruik gemaakt van de UTF-8 karaktercodering[12]. Dit is een karaktercode met een variabele breedte (1, 2, 3 of 4 bytes) waarin alle 143 859 Unicode[13] karakters gecodeerd kunnen worden. De UTF-8-codering is backwards compatibel met de 7-bits ASCII-codering.

[11] Zie https://en.wikipedia.org/wiki/ISO/IEC_8859-1.

[12] Zie eventueel https://en.wikipedia.org/wiki/UTF-8.

[13] Zie eventueel https://en.wikipedia.org/wiki/Unicode.

In de programmeertaal C kun je een karakter opslaan in een variabele van het type **char**. Een constant karakter kun je in C gebruiken door het betreffende karakter tussen enkele aanhalingstekens te plaatsen. Dus bijvoorbeeld `char c = 'A';`. Besturingskarakters kun je in C coderen met behulp van een zogenoemde escape sequence. Zie http://en.cppreference.com/w/c/language/escape. Bijvoorbeeld: een CR (Cariage Return) karakter kan in C-code gebruikt worden als `'\r'`.

Als het programma `echoput` draait op je microcontroller en je typt in de terminal enkele karakters en vervolgens een 'enter' in, dan verstuurt de terminal een CR (Cariage Return) gevolgd door een LF (Line Feed). Het `echoput` programma stuurt deze twee karakter weer terug naar de terminal waardoor de cursor naar het begin van de volgende regel gaat. Je kunt de ASCII-codes van deze karakters zien door in CoolTerm op 'View Hex' te klikken.

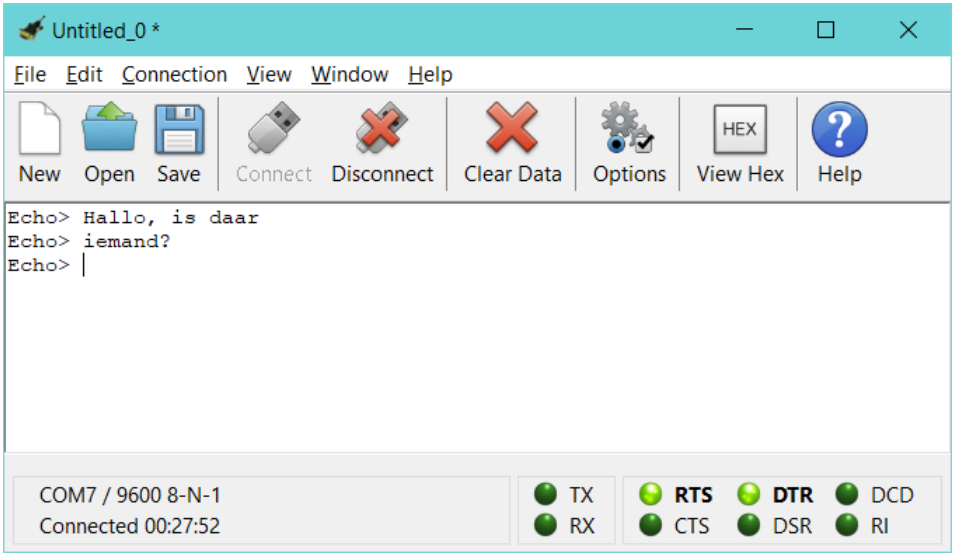
Een string (stukje tekst) wordt in C opgeslagen in een array van het type **char**. De programmeertaal C kent *geen* apart type voor strings. Een array heeft in C, nadat hij gedefinieerd is, een vaste grootte. Om het nu toch mogelijk te maken om strings van verschillende lengtes in deze array op te kunnen slaan, hebben de bedenkers van C gedefinieerd dat een string afgesloten moet worden met het ASCII karakter NUL (`'\0'`). Een constante string kan in C worden gebruikt door de betreffende tekst tussen dubbele aanhalingstekens te plaatsen. Bijvoorbeeld: `zend("Hallo");`.

8.1.6 Schrijf een functie met als prototype `void zend(char str[])` waarmee een als argument meegegeven string verzonden wordt naar de terminal. Gebruik deze functie om in het programma `echoput` aan het begin van elke regel de prompt[14] `Echo>` af te drukken in de terminal[15], zie figuur 5.

8.1.7 Maak een nieuw project aan genaamd `UARTled`. Sluit een led aan op pin P1.0 (via een weerstand). Schrijf een programma voor de MSP430G2553 waarbij de led wordt aangezet als via de UART-verbinding het karakter `'0'` wordt ontvangen en de led wordt uitgezet als via de UART het karakter `'1'` wordt ontvangen. Maak gebruik van polling zoals in het programma `echoput`.

[14] Een prompt is een tekenreeks op een beeldscherm die aangeeft dat de gebruiker iets moet doen.

[15] Als het niet werkt zoals verwacht, dan kun je hier controleren of je de functie zend correct hebt geïmplementeerd: `opdr8.1.6.c`.



Figuur 5: De gewenste output van het programma echoput.

Verdieping

De volgende opdrachten zijn bedoeld als extra oefening met de UART en interrupts.

Lees nu paragraaf 15.3.14 en 15.3.15 van de MSP430x2xx Family User's Guide.

8.1.8 Pas het programma uit opdracht 8.1.7 aan zodat in plaats van polling een interrupt gebruikt wordt. Maak daarbij gebruik van de interruptvector `USCIAB0RX_VECTOR`. Zet de microcontroller in een zo energiezuinige slaapmode[16].

8.1.9 Pas het programma uit opdracht 8.1.4 aan zodat het helemaal op interruptbasis werkt. Let erop dat de transmit interrupt pas aangezet moet worden als er iets te versturen valt. Als het laatste karakter dat verstuurd moet worden in het register `UCA0TXBUF` is geplaatst, dan moet de transmit interrupt weer uitgezet worden. Zet de microcontroller in een zo energiezuinige slaapmode[17].

[16] De uitwerking van deze opdracht vind je hier: `opdr8.1.8.c`.

[17] De uitwerking van deze opdracht vind je hier: `opdr8.1.9.c`.