

Opdrachten week 4 les 2 – Opzetten microcontroller

Tot nu toe is er geprogrammeerd in Python op de pc. Deze les wordt de overstap gemaakt naar de taal C waarbij de programma's worden uitgevoerd op een microcontroller. Je leert hoe je:

- Code Composer Studio en de MSP-EXP430G2ET LaunchPad gebruikt om de MSP430G2553 microcontroller te programmeren;
- registers instelt om de pinnen van de microcontroller aan te sturen;
- de MSP430G2553 microcontroller op een breadboard kan plaatsen en kan programmeren met behulp van de MSP-EXP430G2ET Launchpad;
- door een programma heen kunt stappen;
- leds kunt aansturen.

In de volgende les wordt hierop verder gegaan en met name gekeken naar hoe bepaalde structuren uit Python kunnen worden toegepast in de taal C.

Lees nu hoofdstuk 1 en 2 van het handboek¹.



Je gebruikt de IDE (Integrated Development Environment) Code Composer Studio om software voor de MSP430G2553 microcontroller te ontwikkelen en te testen. Op een school-pc kun je Code Composer Studio starten vanuit de Liquit Workspace. Je kunt Code Composer Studio ook gratis downloaden en installeren op je eigen pc, zie <https://www.ti.com/tool/download/CCSTUDIO>. De single file installer werkt het snelste. De installatie wijst zich grotendeels vanzelf. Kies voor een “Custom Installation” en kies bij “Processor Support” (alleen) voor “MSP430 ultra-low power MCUs”.

4.2.1 Start CCS (Code Composer Studio). Na het starten van CCS wordt er een locatie gevraagd voor de workspace, dit is de directory waarin alle nieuwe projecten worden opgeslagen. Als workspace kun je zelf een directory kiezen. Op een school-pc is de map H:\workspace_MSP430 een goede keuze gezien deze netwerkmap toegankelijk is vanaf elke school-pc. Deze workspace neemt zo'n 15 MB in beslag. Wanneer een firewall melding verschijnt, kun je toestemming geven voor de verbinding.

¹ Daniël Versluis. *Microprocessor Programmeren in C*. 2018. URL: https://bytebucket.org/HR_ELEKTRO/ems10/wiki/Handboek/EMS10_handboek_ebook.pdf

Waarschuwing

Als je CCS op een school-pc gebruikt, dan moet je **geen** updates uitvoeren. Omdat je geen rechten hebt om software te installeren zal CCS na de ‘update’ **niet** meer opgestart kunnen worden!

4.2.2 Maak nu een nieuw project aan via de menukeuze `File` `>>` `New` `>>` `CCS Project`. Pas vervolgens de projectinstellingen aan zoals is weergegeven in [figuur 1](#). Noem dit project `opdr_4.2.2`. Verbind de LaunchPad met de pc en druk hierna op . Hiermee wordt de code gecompileerd en ingeladen in de microcontroller. Mogelijk verschijnt er een vraag over low-power instellingen. Deze kun je negeren. Mogelijk verschijnt de mededeling: “A firmware update is required for the MSP430 Debug Interface”. Klik in dat geval op `Update`. Als alles goed is gegaan zie je [figuur 2](#).² Stop de debugsessie met .

Een van de grote verschillen tussen C en Python is dat elk stuk code in C binnen een functie moet staan. Elk C-programma heeft daarom een functie genaamd `main`. Dit is het begin van de programma-uitvoer en zonder deze `main`-functie zal de compiler het programma niet accepteren. Gelukkig wordt deze functie automatisch aangemaakt in de code van een nieuw project. Later leer je meer over functiedefinities in C. In deze les wordt alle code geschreven binnen de `main`-functie.

De eerste regel in de `main`-functie is misschien verwarrend. Deze regel moet *altijd* in je code staan, het schakelt namelijk een automatische reset-timer uit. Zou deze niet uitgeschakeld worden dan zou de microcontroller zich op vaste perioden resetten. Dit kan onverwacht gedrag opleveren. De reden voor deze timer zal in een latere les uitgelegd worden. Voor nu mag je de regel laten staan en verder negeren.

² Indien dit niet werkt, dan is er waarschijnlijk een driver probleem. Ga naar *apparaten en printers* binnen het Windows configuratiescherm om te zien of de LaunchPad wordt herkend. Zo niet, vraag de docent om hulp.

New CCS Project

Create a new CCS Project.

Target: MSP430Gxxx Family MSP430G2553

Connection: TI MSP430 USB1 [Default] Identify...

MSP430

Project name: opdr_4.2.2

Use default location

Location: C:\workspace_v12\opdr_4.2.2 Browse...

Compiler version: TI v21.6.1.LTS More...

Project type and tool-chain

Project templates and examples

type filter text

- Empty Projects
 - Empty Project
 - Empty Project (with main.c)
 - Empty Assembly-only Project
 - Empty RTSC Project
- Basic Examples
 - Blink The LED

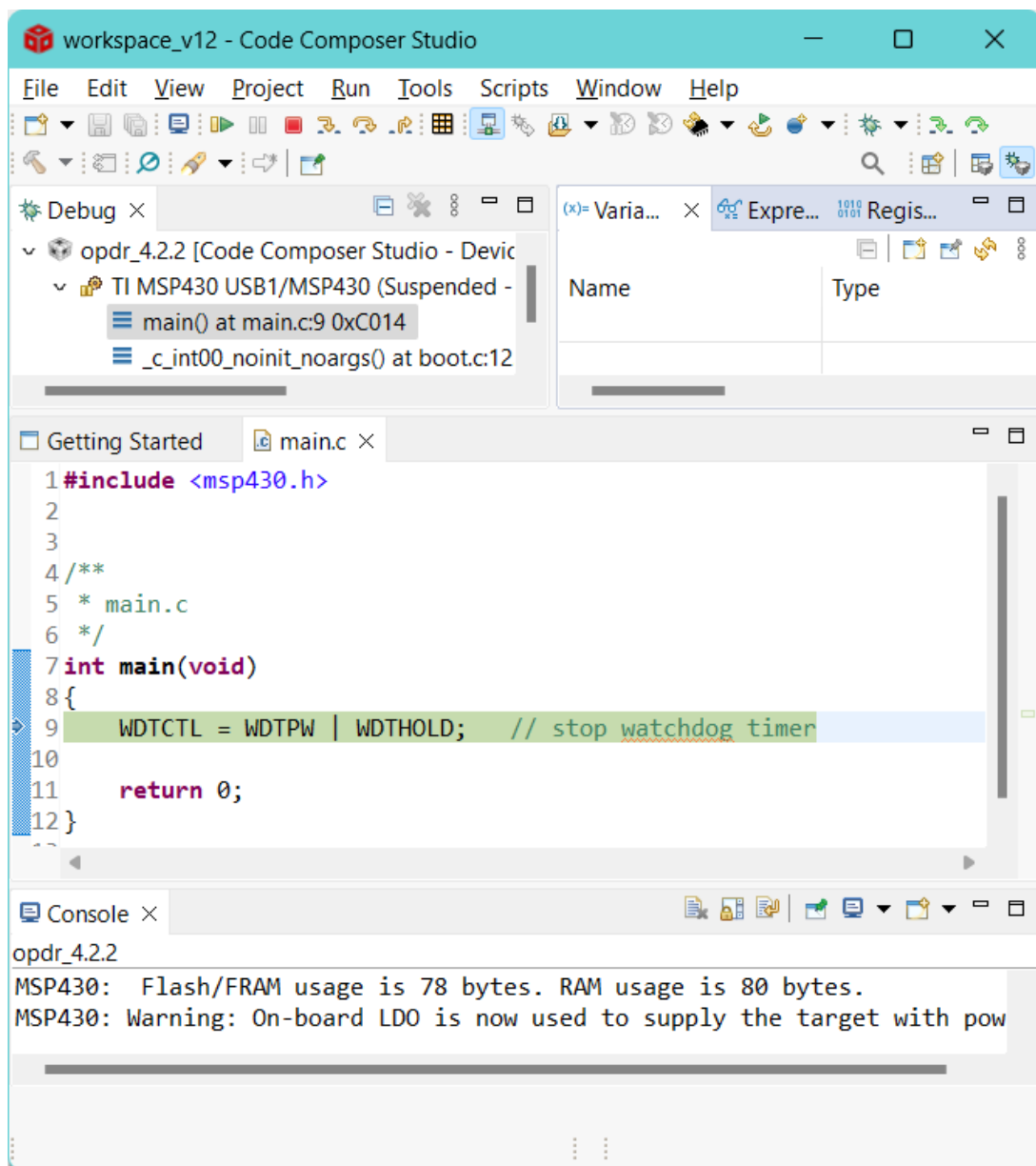
Creates an empty project initialized for the selected device. The project will contain an empty 'main.c' source-file.

Open [Resource Explorer](#) to browse a wide selection of example projects...




Open [Import Wizard](#) to find local example projects for selected device...

? < Back Next > Finish Cancel

Figuur 1: Instellingen voor een nieuw project.



Figuur 2: Venster bij het succesvol debuggen van een nieuw project.

4.2.3 Pas de projectcode aan zoals gegeven is in [listing 1](#). Druk opnieuw op  en . De groene en rode leds aan de onderkant van de LaunchPad zouden nu aan moeten gaan. Stop de debugsessie met .

```
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer

    P1DIR = 0b01000001; // P1.6 en P1.0 als output instellen
    P1OUT = 0b01000001; // Rode led op P1.6 en groene led op P1.0 aan

    return 0;
}
```

Listing 1: Code om leds aan te zetten. Zie [les42_leds_lp.c](#)

Bij het programmeren van microcontrollers is het gebruikelijk om de hexadecimale notatie toe te passen in plaats van de binaire of decimale notatie. De reden hiervoor is dat hexadecimale notatie compacter is dan de binaire notatie en dat hiermee ook snel de bitwaarden op diverse posities kunnen worden bepaald.

```
// Verschillende notaties van hetzelfde getal
// In C wordt standaard de decimale notatie gebruikt:
P2OUT = 178;
// In C wordt de prefix 0b gebruikt voor de binaire notatie:
P2OUT = 0b10110010;
// In C wordt de prefix 0x gebruikt voor de hexadecimale notatie:
P2OUT = 0xB2;
```

De decimale notatie is ook kort, maar hiermee is het weer lastig om de waarden van de verschillende bitposities te bepalen.

4.2.4 Vul [tabel 1](#) in om behendigheid te krijgen bij het omzetten van notaties. Pak indien nodig de [PowerPoint-presentatie van ELE10 \(week 1\)](#) erbij.

4.2.5 Pas de code van je programma aan door P1DIR in te stellen met een hexadecimaal getal en P1OUT met een decimaal getal. Gaan de juiste leds nog aan?

Tabel 1: Omzetten notaties.

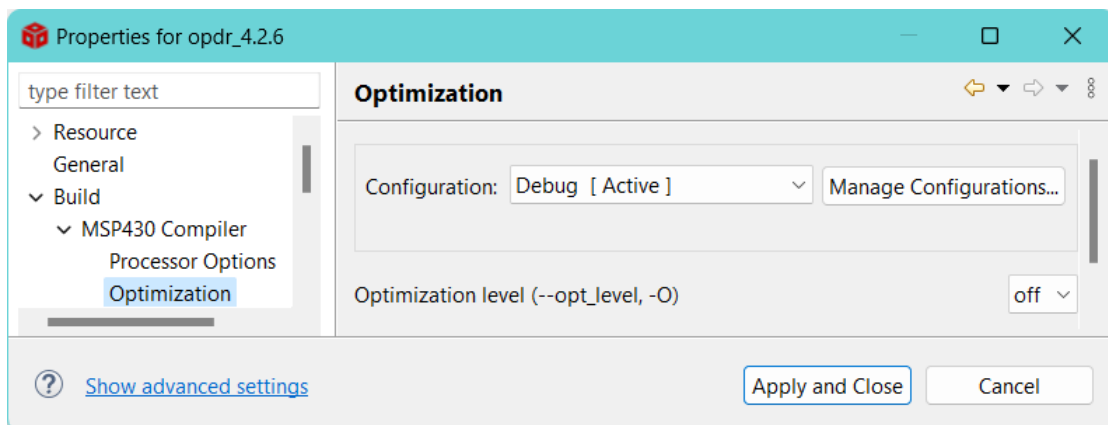
Hexadecimaal	Binair
0xBC	0b.....
0x..	0b10001101
0x20	0b.....
0x..	0b01111111
0xF0E1	0b.....
0x....	0b0001001010100101

4.2.6 Maak een nieuw project aan en noem dit opdr_4.2.6³. Plaats [listing 2](#) achter regel 9. Start de debug sessie (🐛) en stap door de regels heen met (👉). Om netjes regel voor regel door het programma heen te kunnen stappen moet de optimizer uitgezet worden. Kies `Project >> Properties >> Build >> MSP430 Compiler >> Optimization` en zet het 'Optimization level' op 'off', zie [figuur 3](#). Als het programma stap voor stap doorlopen wordt, dan zal eerst de rode en daarna de groene led aangaan. Waarom gaat de rode led weer uit na het uitvoeren van de laatste regel van [listing 2](#)?

```
P1OUT = 0x00; // Register resetten
P1DIR = 0x41; // P1.6 en P1.0 als output instellen
P1OUT = 0x40; // Rode led op P1.6 aan
P1OUT = 0x01; // Groene led op P1.0 aan
```

Listing 2: Code bij [opdracht 4.2.6](#), zie [opdr4.2.6.c](#)

³ Dit kun je het gemakkelijkst doen door het project opdr_4.2.2 te kopiëren in de Project Explorer van Code Composer Studio. Selecteer het te kopiëren project en klik op `Ctrl`+`C` en vervolgens op `Ctrl`+`V`.



Figuur 3: De optimizer moet uitgezet worden tijdens het stap voor stap debuggen.

4.2.7 Vervang de laatste twee regels uit de code van [listing 2](#) door de twee regels gegeven in [listing 3](#). Start de debug sessie (🐞) en stap door de regels heen met (🔍). De rode led blijft nu wel aan na het uitvoeren van de laatste regel van [listing 3](#). Dit komt omdat nu een zogenoemde bitwise-OR-operatie wordt uitgevoerd. De operator `|` voert een logische OR-operatie uit op de bits van de operanden P1OUT en `0x01`. Het resultaat wordt opgeslagen in P1OUT.

```
P1OUT = P1OUT | 0x40; // Rode led op P1.6 aan
P1OUT = P1OUT | 0x01; // Groene led op P1.0 aan
```

Listing 3: Code bij [opdracht 4.2.7](#), zie [opdr4.2.7.c](#)

Vul [tabel 2](#) nu in door de bitwise-OR-operatie `|` uit te voeren op de bits van het P1OUT register en de constante `0x01`. Waarvoor kun je de bitwise-OR-operatie gebruiken in een C-programma?⁴

Tabel 2: OR-operatie.

Variabele	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
P1OUT	0	1	0	0	0	0	0	0
0x01	0	0	0	0	0	0	0	1
P1OUT 0x01

⁴ Je kunt een bitwise-OR-operatie gebruiken om individuele bits één te maken.

4.2.8 Zet alleen de groene led weer uit door de regel $P1OUT = P1OUT \& \sim 0x01$; toe te voegen aan het programma. Voeg deze regel toe achter de regels uit [listing 3](#). Start de debug sessie (🔧) en stap door de regels heen met (👉). De rode led blijft aan omdat een bitwise-AND-operatie gecombineerd met een bitwise-NOT-operatie gebruikt wordt. De operator \sim voert een logische NOT-operatie uit op de bits van de operand $0x01$. De operator $\&$ voert een logische AND-operatie uit op de bits van de operanden $P1OUT$ en $\sim 0x01$. Het resultaat wordt opgeslagen in $P1OUT$.

Vul [tabel 3](#) in. Waarvoor kun je de bitwise-AND-operatie in combinatie met de bitwise-NOT-operatie gebruiken in een C-programma?⁵

Tabel 3: AND- en NOT-operatie.

Variabele	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
P1OUT	0	1	0	0	0	0	0	1
$0x01$	0	0	0	0	0	0	0	1
$\sim 0x01$
$P1OUT \& \sim 0x01$

4.2.9 Breid de bestaande code verder uit. Zet beide leds aan met $P1OUT = P1OUT | 0x41$; . Plak hierachter *twee keer* de regel $P1OUT = P1OUT \wedge 0x40$; . In deze regel wordt een bitwise-XOR-operatie uitgevoerd. De operator \wedge voert een logische XOR-operatie uit op de bits van de operanden $P1OUT$ en $0x40$.

Start de debugsessie en loop door de code heen. Wat gebeurt er? Vul [tabel 4](#) in. Waarvoor kun je de bitwise-XOR-operatie gebruiken in een C-programma?⁶

Tabel 4: XOR-operatie.

Variabele	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
P1OUT	0	1	0	0	0	0	0	1
$0x40$	0	1	0	0	0	0	0	0
$P1OUT \wedge 0x40$

⁵ Je kunt een bitwise-AND-operatie in combinatie met de bitwise-NOT-operatie gebruiken om individuele bits nul te maken.

⁶ Je kunt een bitwise-XOR-operatie gebruiken om individuele bits te invertieren.



Op poort P2 van de MSP-EXP430G2ET Launchpad zit een RGB-led aangesloten. De kleur van het led is afhankelijk van de combinatie van de drie kleuren rood, groen en blauw. De rode led kan aangestuurd worden door de pin P2.1, de groene led door P2.3 en de blauwe led door P2.5.

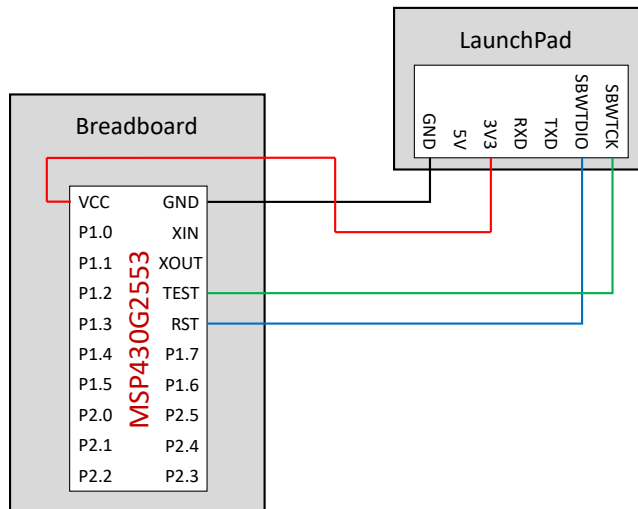
4.2.10 Maak een nieuw project aan en noem dit opdr_4.2.10. Schrijf een C-programma dat gebruik maakt van de bitwise-OR-, bitwise-AND-, bitwise-NOT- en bitwise-XOR-operator en achtereenvolgens de volgende acties uitvoert, als je stap voor stap door het programma loopt:

- stel P2.1, P2.3 en P2.5 in als uitgangen;
- zet alle leds in de RGB-led uit;
- zet de rode led in de RGB-led aan;
- zet ook de groene led in de RGB-led aan;
- zet ook de blauwe led in de RGB-led aan;
- zet de rode en groene led in de RGB-led uit;
- zet de blauwe led in de RGB-led uit;
- inverteer de rode en groene led in de RGB-led;
- inverteer alle drie leds in de RGB-led;
- inverteer de blauwe led in de RGB-led.

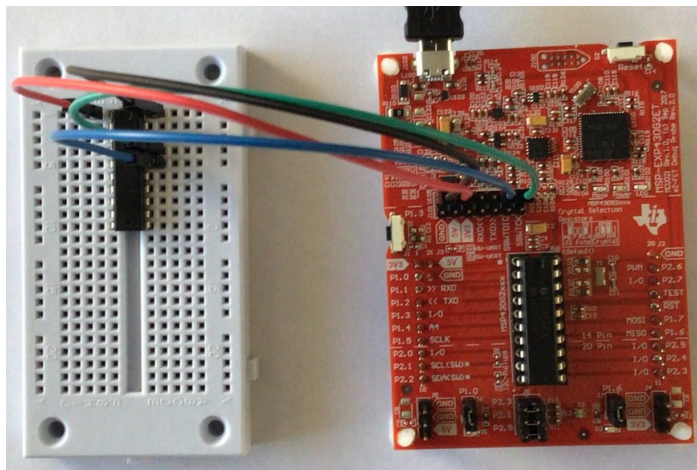
In de praktijk willen we de MSP430G2553 niet gebruiken op de MSP-EXP430G2ET Launchpad, maar in onze eigen hardware-schakeling. Daarom gaan we de MSP430G2553 nu op een breadboard plaatsen.

4.2.11 De LaunchPad bestaat uit twee delen, een hardware debugger en programmer en een target deel. Met het bovenste deel (waarop ook de USB-connector zit) kun je de MSP430G2553 microcontroller op het onderste deel (de target) programmeren en debuggen. De twee delen zijn met elkaar verbonden met behulp van zeven jumpers op een rij. Omdat we nu de microcontroller op een breadboard gaan plaatsen moeten al deze zeven jumpers verwijderd worden. Verwijder ook de MSP430G2553 voorzichtig van je LaunchPad, zonder de pootjes te beschadigen, en verhuis de microcontroller van de LaunchPad naar het breadboard. Verbind het programmer- en debuggerdeel van je LaunchPad nu met de MSP430G2553 op je breadboard, zie [figuren 4 en 5](#). **Let er**

goed op dat je de 3V3 gebruikt, de MSP430G2553 is namelijk niet bestand tegen vijf volt! Test de opzet door opnieuw door een nieuw leeg project aan te maken en op  te drukken en te controleren of de breadboardopstelling werkt. Indien je een foutmelding krijgt, dan is het belangrijk om de aansluitingen te controleren, de GND, 3V3, SBWTDIO en SBWTCK verbindingen zijn allemaal belangrijk. Stop de debugsessie met .






Figuur 4: Aansluiten microcontroller op breadboard en LaunchPad.

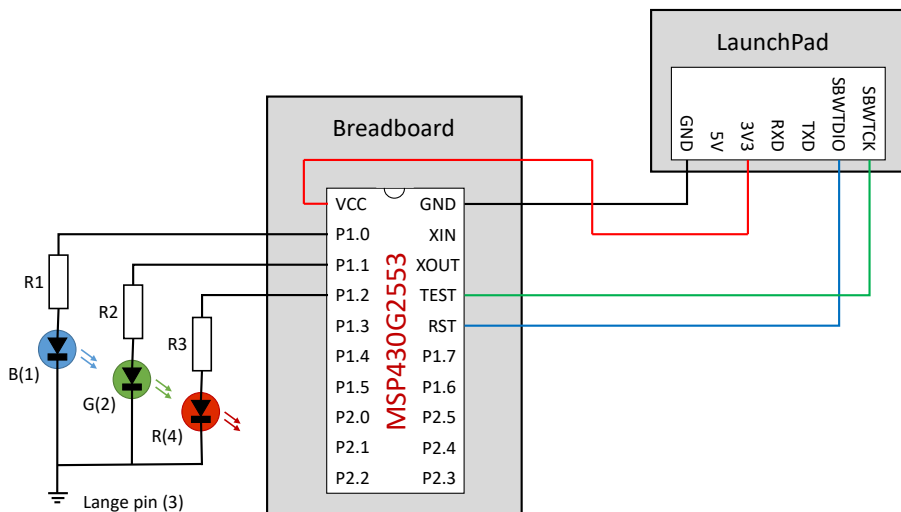


Figuur 5: Verbindingen tussen de MSP430G2553 op het breadboard en de LaunchPad.

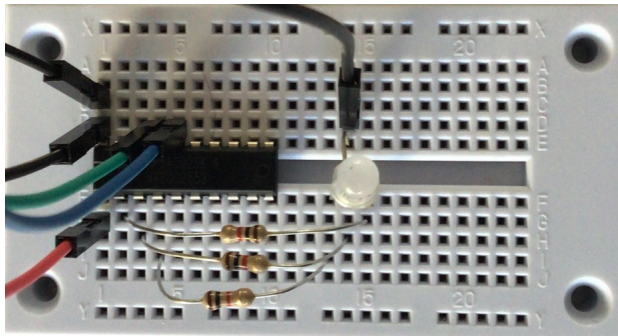
4.2.12 Nu bekend is dat de MSP430G2553 op het breadboard geprogrammeerd en gedebuggd kan worden, kan de RGB-led aangesloten worden zoals in [figuren 6 en 7](#) is weergegeven. De MSP430G2553 kan maximaal 6 mA per pin leveren, voor een totaal van 48 mA (Zoek deze waarden zelf op in de [datasheet van de MSP430G2553](#)). De uitgangsspanning van een outputpin die hoog is, bedraagt, bij een V_{CC} van 3,3V, ook ongeveer 3,3V.

De spanningsval over de leds is ongeveer 2,3V. De stromen door leds moeten begrensd worden op 1 mA. Bereken welke weerstanden nodig zijn in het schema gegeven in [figuur 6](#). Sluit de RGB-led vervolgens aan zoals in dit schema is weergegeven.

Maak een nieuw project aan en noem dit opdr_4.2.12. Pas de projectcode aan zoals gegeven is in [listing 4](#). Druk opnieuw op  en . De blauwe en groene leds zouden nu aan moeten gaan. Stop de debugsessie met .



Figuur 6: Aansluiten RGB-led op breadboard en LaunchPad.



Figuur 7: Aansluiten RGB-led op breadboard.

```
#include <msp430.h>

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;

    P1DIR = 0b00000111;
    P1OUT = 0b00000000;
    P1OUT = P1OUT | 0b00000011;

    return 0;
}
```

Listing 4: Code om leds aan te zetten. Zie [les42_leds.c](#)

4.2.13 Maak een nieuw project aan en noem dit opdr_4.2.13. Herhaal [opdracht 4.2.10](#) maar nu voor de RGB-led op het breadboard.

Lees nu, ook ter voorbereiding van de volgende les, paragraaf [1.1](#), [1.2](#), [1.3](#), [2.2](#), [2.3](#) en [2.4 van het boek](#)⁷. Lees ook [hoofdstuk 3](#) en [4 van het handboek](#)⁸.

4.2.14 Maak een nieuw project aan en noem dit opdr_4.2.14. Maak in het begin van de functie `main` een integer `i` variabele aan en zet hier de waarde `0` in. Configureer pinnen

⁷ Carl Burch. *C for Python programmers*. 2011. URL: <http://www.cburch.com/books/cpy/>.

⁸ Daniël Versluis. *Microprocessor Programmeren in C*. 2018. URL: https://bytebucket.org/HR_ELEKTRO/ems10/wiki/Handboek/EMS10_handboek_ebook.pdf

P1.0, P1.1 en P1.2 als uitgangen. Schrijf de waarde van de variabele i naar het register P1OUT. Verhoog de variabele i met 1 en maak het P1OUT register weer gelijk aan de variabele i . Blijf dit herhalen (dus nieuwe regels code toevoegen) tot de variabele 8 is geworden. Start de debug sessie (🔧) en stap door de regels heen met (👉).

Tabel 5: Waarde van de variabele weergegeven op de leds.

Waarde	Rode led	Groene led	Blauwe led
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

4.2.15 Pas het programma nu aan zodat het verhogen van de variabele i met 1 en het P1OUT register gelijk maken aan de variabele i , in een **for**-statement gebeurt.

Oefening

Je hebt deze les heel veel nieuwe dingen geleerd. Het is verstandig om een en ander uit je hoofd te leren. Flashcards kunnen je daarbij helpen. Er is een stok Anki flashcards beschikbaar zodat je ze regelmatig kunt herhalen. Deze stok kun je hier downloaden: [EMS10 Week 4 Les 2.apkg](#).

Daarnaast zijn er een aantal video's beschikbaar waarin de stof van deze les nogmaals wordt uitgelegd, zie [tabel 6](#).

Tabel 6: Video's over de stof van week 4 les 1.

Link	Tijd	Beschrijving
Code Composer Studio	06:11	Legt uit hoe je CCS kunt gebruiken, layout, views, debuggen enz.
Opbouw van een microcontroller programma	04:51	Standaard opbouw microcontrollerprogramma wordt besproken. Waar zet je welke code neer?
Output pin deel 1	05:09	In deel 1 van deze presentatie wordt uitgelegd hoe je een pin van de MSP430 als digitale output kan configureren.
Output pin deel 2	01:44	In deel 2 van deze presentatie wordt uitgelegd hoe je een output pin hoog kan maken.
Output pin deel 3	02:54	In deel 3 van deze presentatie wordt uitgelegd hoe je een output pin laag kan maken.
Output pin deel 4	02:45	In deel 4 van deze presentatie wordt uitgelegd hoe je een output pin kan inverteren.