Opdrachten week 8 les 3 – Integratie

Verdieping

De opdrachten van deze les laten zien hoe de verschillende dingen die je geleerd hebt bij EMS10 met elkaar gecombineerd kunnen worden. Ze zijn bedoeld om te laten zien hoe je een en ander in de praktijk zou kunnen gebruiken. Het is niet noodzakelijk om deze opdrachten te maken voor de toets.

In week 1 tot en met 3 van EMS10 heb je geleerd om met Python te werken. In week 4 tot en met 8 heb je geleerd om met een microcontroller te werken, de MSP430G2553. Deze les gaan we een aantal dingen integreren. Je leert om:

- data van de microcontroller te versturen naar een computer en te verwerken in Python;
- de data zo energiezuinig mogelijk te vergaren en te versturen.

8.3.1 Als eerste zetten we een simpel framework op om alle functies die we willen uitvoeren te timen. Bekijk het project van les 7.3 of de documentatie en broncode ervan. Zoals in de documentatie is uitgelegd, maken we gebruik van een timer met een vaste periodetijd om de microcontroller wakker te maken. In de `while(1)` loop van de main functie wordt bijgehouden hoe vaak de microcontroller wakker is geworden. In het voorbeeld is de timer ingesteld op 20 ms, dat wil zeggen dat de `while(1)` loop elke 20 ms de code uitvoert. Door dan tot 5 te tellen weten we dat er 100 ms voorbij is gegaan. Op deze manier kunnen we 1 timer gebruiken om verschillende tijden voor verschillende functies bij te houden met een simpel `if()` statement.

Kopieer de relevante stukken code van de ems10 bibliotheek en zorg voor een periodetijd van 100 ms. Maak in de `while(1)` loop een aatal `if()` statements om:

- elke 300 ms een led te gaan togglen;
- elke 500 ms de adc te triggeren en via UART de waarde door te sturen;
- elke 1000 ms de temperatuur op de OLED te weergeven.

In de volgende opdrachten zullen we de inhoud van de `if()` statements gaan vullen.

8.3.2 Sluit een led aan op P1.1 en zorg ervoor dat deze in de 300 ms `if()` statement wordt getoggled.

8.3.3 Kopieer de relevante code van les 7.1 om elke 500 ms de ADC te starten om de LM35DZ te samplen in dit project. Maak hiervoor ook de ISR aan die de waarde in een globale variabele zet. Sluit de LM35DZ dit keer aan met de positieve voeding aangesloten op P2.2[1]. Zet deze pin op output en maak hem hoog. Controleer de gesamplede waarde in de debugger. Als deze waarde klopt, dan kun je verder met de volgende opgave.

De makkelijkste manier om data naar een pc te sturen is via de ingebouwde seriële UART-verbinding van de MSP-EXP430G2ET Launchpad. Een UART-verbinding is bovendien een goede keuze omdat veel draadloze modules ook het UART-protocol gebruiken voor de in- en output. In plaats van een pc kan dan later simpelweg een draadloze module worden gekoppeld.

8.3.4 Gebruik de code van les 8.1 om de UART peripheral te configureren op 9600 baud, 8-bit data, no parity en 1 stopbit. Test de werking via CoolTerm door bijvoorbeeld voortdurend het karakter `'A'` naar de terminal te versturen in het 1000 ms `if()` statement. Let erop dat de TXD pin van de MSP430G2553 op de juiste manier met het development board is verbonden. Wanneer de communicatie werkt kun je verder.

Om de data aan de kant van de pc makkelijk te kunnen interpreteren is het handig om tekst te versturen (ASCII-karakters) in plaats van binaire getallen. In tekst kan een 'enter' bijvoorbeeld het einde van een regel aangeven. De volgende opdracht helpt je om de waarde van `ADC10MEM` om te zetten naar tekst.

8.3.5 Listing 1 laat zien hoe je een integer omzet naar een c-string. De functie `itoa` kun je vinden in de broncode (`opdr8.3.3.c`). De array `tekst[]` is een zogenoemde C-string. Deze array bevat na aanroep van de functie `itoa` de karakters die de waarde van `ADC10MEM` beschrijven en een karakter om de einde van de string aan te geven (`'\0'`). Maak in de ADC interrupt gebruik van een loop om één voor één door de karakters in de array `tekst[]` heen te lopen tot aan het `'\0'`-karakter. In elke iteratie van de loop kun je dan een karakter versturen door te schrijven naar `UCA0TXBUF`[2]. Controleer of de waarden goed worden ontvangen op de pc met CoolTerm.

[1] Let op de oriëntatie van de LM35DZ. De LM35DZ verbruikt 50 µA dus een I/O-pin kan deze gemakkelijk voeden.

[2] Vergeet niet op het hoog worden van de bit `UCA0TXIFG` te wachten.

8.3.6 Zorg er nu voor dat je de OLED kunt gebruiken om de temperatuur te weergeven. Update de temperatuur op het schermpje in het 1000 ms `if()` statement.

```
#pragma vector = ADC10_VECTOR
__interrupt void adc10isr(void)
{
    volatile char tekst[5];
    adc_waarde = ADC10MEM;
    itoa(adc_waarde, tekst);
}
```

Listing 1: ADC10MEM naar tekst. Zie ook opdr8.3.3.c.

Nu is het tijd over te schakelen naar Python. De waarden van de seriële poort kunnen makkelijk worden opgehaald met de 'pyserial' module.

8.3.7 Open Thonny en ga naar Tools > Manage Packages..., zoek en installeer 'pyserial'.

8.3.8 Importeer de 'serial' module in een nieuw Python script.

8.3.9 In listing 2 wordt getoond hoe COM29 geopend kan worden. Gebruik dit om de COM-poort te openen van de LaunchPad. Maak een oneindige `while`-loop en roep hierin `print(int(ser.readline()))` aan. Als alles goed gaat, verschijnt nu elke halve seconde de ADC-waarde op het scherm. Pas de code aan om de temperatuur te printen in plaats van de ADC-waarde.

```
ser = serial.Serial(
    port = 'COM29',
    baudrate = 9600,
    parity = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE,
    bytesize = serial.EIGHTBITS
)

ser.isOpen()
```

Listing 2: Het openen van een seriële poort.

8.3.10 Breid het script uit om naast de huidige temperatuur ook een gemiddelde te printen. Dit gemiddelde kun je berekenen met een 'Moving Average Filter' zoals in les 8.1. Houd hiervoor een lijst bij met 100 ontvangen waarden.

8.3.11 Naast het printen van de temperatuur en het gemiddelde, kunnen we deze ook plotten in Python. Gebruik listing 3 om jouw script uit te breiden met een plot. Het verkrijgen van de temperatuurwaarde en het gemiddelde moet nog ingevuld worden. Voeg ook een titel toe en een legenda om het verschil tussen de waarde en het gemiddelde duidelijk te maken. Zie figuur 1 als voorbeeld.

```
import matplotlib.pyplot as plt
import serial

#invullen met 100 maal de eerst gemeten waarde
waarden = []
gemiddelden = []
temperatuur =  ... ser.readline() ...

for i in range(100):
    waarden.append(temperatuur)
    gemiddelden.append(temperatuur)

#verkrijg figure uit plot
figure = plt.gcf()
figure.show()
figure.canvas.draw()

while True:
    #Wachten met plt.pause() totdat voldoende
    #bytes zijn ontvangen
    while (ser.inWaiting() < 5):
        plt.pause(0.2)

    temperatuur =  ... ser.readline() ...
    gemiddelde = ...

    # nieuwe erbij
    waarden.append(temperatuur)
    gemiddelden.append(gemiddelde)
```

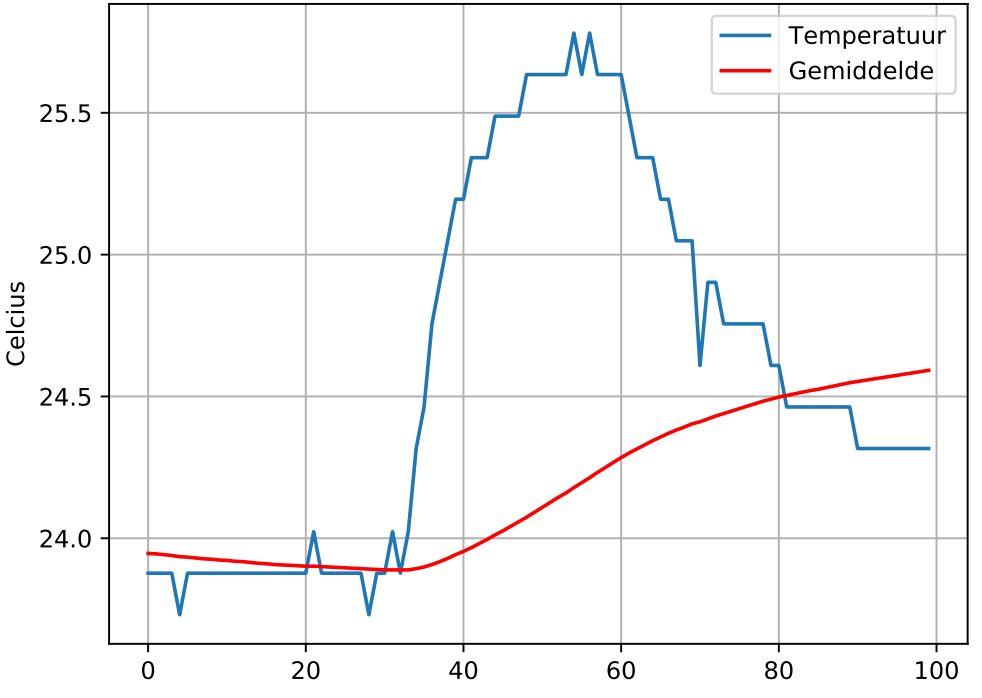
```
#oude weg
del(waarden[0])
del(gemiddelden[0])

#oude plot weg (clear figure)
plt.clf()

#nieuwe plots
plt.plot( .... )
plt.plot( .... )

#plot opnieuw tekenen
figure.canvas.draw()
```

Listing 3: Een plot bijwerken



Figuur 1: Mogelijke uitvoer bij het plotten van de temperatuur en het gemiddelde.

Zoals je waarschijnlijk hebt gemerkt is het verwerken van data gemakkelijker te doen in Python op de pc dan in C op de microcontroller. Dit is ook waar het 'Internet of Things' naar toe neigt. Veel simpele (embedded) systemen om data te meten en door te sturen. Grote servers worden gebruikt om alle data op te slaan en te verwerken en hier nuttige statistische berekeningen op uit te voeren (big data).

Een ander aspect van zulke kleine embedded systemen is zuinigheid. In de volgende opdrachten ga je het microcontrollerprogramma zo optimaliseren dat er slechts zo'n 0,5 µA verbruikt wordt buiten het meten en verzenden om.

8.3.12 Zorg ervoor dat er niets anders op je µC zit aangesloten dan de LM35DZ en de UART-connectie. De led en oled halen we dus weg. Meet het opgenomen vermogen met de Energy Trace en bereken de opgenomen stroom. Waarschijnlijk ligt dit rond de 0,6 mA. Dit moet omlaag!

8.3.13 De eerste stap tot zuinigheid is om de ADC continu te starten a.d.h.v. de VLO klok in plaats van handmatig het ADC10SC bit aan te zetten. Implementeer dit. Hoeveel invloed heeft dit op het stroomverbruik? Vergeet niet te meten zonder te debuggen.

8.3.14 Zoek nu in de MSP4302x53 datasheet[3] op hoeveel stroom de µC trekt in LPM3 en hoeveel stroom de ADC en de referentiebuffer trekken. Zoek als laatste in de datasheet van de LM35DZ[4] op hoeveel stroom deze component vraagt. Welke stroom is het grootst?

8.3.15 Als het klopt, heb je gevonden dat de ADC de meeste stroom nodig heeft. Het is dus belangrijk om de ADC alleen aan te zetten wanneer deze nodig is, maar dit aan en uitzetten kost tijd. Zoek in de MSP430x2xx Family User's Guide[5] op hoeveel tijd het kost om de referentiegenerator op te starten, zie paragraaf 22.2.3.1. Zet de timer in de up-mode als dit nog niet is gebeurd. Maak een kleine toestandsmachine in de ISR om eerst de buffer en ADC aan te schakelen (let op dat dit alleen kan met de bit `ENC` uit). Je kunt hierbij de waarde van register `TACCR0` aanpassen om ongeveer 30 µs later de

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

[4] Zie http://www.ti.com/lit/ds/symlink/lm35.pdf.

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

volgende ISR-aanroep te krijgen om de ADC-conversie te starten. Na het starten van de conversie kun je de oorspronkelijke waarde in het register `TACCR0` weer terugzetten, zodat de volgende interrupt na 4 seconden optreedt. Wat is het stroomverbruik nu? Het zou in de orde van 1 µA moeten zijn.

8.3.16 Ook kunnen de pinnen die niet verbonden zijn aan iets, nog worden ingesteld als een uitgeschakelde output pin. De zwevende signalen op een input pin zorgen namelijk voor variaties in het register `PxIN` met als gevolg extra energieverbruik.

8.3.17 Als laatste kan de DCO snelheid op 16 MHz worden gezet. Dit zorgt dat er korter een hoge stroom wordt gevraagd dan op 1 MHz, terwijl het stroomverbruik niet zoveel scheelt voor beide frequenties. Als het klopt, is nu het stroomverbruik omlaag tot onder de 1 µA en kan de timer-ISR op een langere periodetijd worden gezet.

8.3.18 Stel wij hebben een knoopcel van 3,6 V met een capaciteit van 800 mA h. Hoe lang kan de microcontroller in standby blijven bij een verbruik van 0,5 µA?