|  |  |  |
| --- | --- | --- |
|  | |  |
| TMTI-OSLNX  Groeidocument | | |
| 6-3-2021  Rik Vos  2155867 |  | |

# Inleiding

In dit groeidocument gaat u lezen over wat er per practicum geleerd wordt en door welke opgaves. Het groeidocument is opgedeeld uit drie practicums over FreeRTOS. Alle code is te vinden op de [github repository](https://github.com/rpvos/OSLinux).

# Inhoudsopgave

[Inleiding 2](#_Toc65318904)

[Inhoudsopgave 3](#_Toc65318905)

[Practicum 1 4](#_Toc65318906)

[Practicum 2 5](#_Toc65318907)

[Conclusie 6](#_Toc65318908)

# Practicum 1

## Opdracht 1: installatie van de tool-chain

Zoals duidelijk deze opdracht was het installeren van de software die nodig zal zijn voor de rest van de practica.

## Opdracht 2: Experimenteren met \ Hello World

In deze opdracht heb ik gebruik gemaakt van de code die gegeven was. Wel moest ik in beide gevallen de functie nvs\_flash\_init() naar ets\_timer\_init() veranderen. De eerste hello world code zorgt ervoor dat de volgende dingen in de console komen.

Hello world!

Restarting in 10 seconds...

Restarting in 9 seconds...

Restarting in 8 seconds...

Restarting in 7 seconds...

Restarting in 6 seconds...

Restarting in 5 seconds...

Restarting in 4 seconds...

Restarting in 3 seconds...

Restarting in 2 seconds...

Restarting in 1 seconds...

Restarting in 0 seconds...

Restarting now.

Met de tweede code moest ik ook de pin die gebruikt wordt om het lampje te laten knipperen naar 2 zetten en dan krijg je een knipperend blauw lampje om de seconde. Ook wordt er in de console “Hello world!” om de 0.1 seconde geprint.

# Practicum 2

## Opdracht 1: Lifecycle van een task

In deze opdracht heb ik de volgende code geschreven:

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "esp\_system.h"

#include "driver/gpio.h"

void lifecycle\_task(void \*pvParameter){

    printf("task started\n");

    int timer = 0;

    for(;;){

        vTaskDelay(1000/portTICK\_RATE\_MS);

        timer++;

        if (timer%3 == 0){

            printf("weer 3 seconden\n");

        }

        if (timer%6==0){

            vTaskPrioritySet(NULL,uxTaskPriorityGet(NULL)-1);

            printf("task priority set to %d\n",uxTaskPriorityGet(NULL));

            if (uxTaskPriorityGet(NULL) == 1){

                printf("Task is being deleted!\n");

                vTaskDelete(NULL);

            }

        }

    }

}

void app\_main()

{

    ets\_timer\_init();

    xTaskCreate(&lifecycle\_task, "lifecycle\_task", 2048, NULL, 5, NULL);

    for(;;){

        vTaskDelay(1000/portTICK\_PERIOD\_MS);

    }

}

Deze code resulteert in de volgende console output:

task started

weer 3 seconden

weer 3 seconden

task priority set to 4

weer 3 seconden

weer 3 seconden

task priority set to 3

weer 3 seconden

weer 3 seconden

task priority set to 2

weer 3 seconden

weer 3 seconden

task priority set to 1

Task is being deleted!

Hieruit is dus te halen dat de prioriteit van de task met een verlaag wordt totdat deze een wordt en dan wordt deze verwijderd.

## Opdracht 2: Periodieke tasks

Voor deze opdracht heb ik twee verschillende tasks gedefinieerd, een voor het rode lampje en een voor het groene lampje.

#define ROOD\_GPIO 12

#define GROEN\_GPIO 13

void rood\_task(void \*pvParameters){

    double hertz = 1;

    gpio\_pad\_select\_gpio(ROOD\_GPIO);

    gpio\_set\_direction(ROOD\_GPIO, GPIO\_MODE\_OUTPUT);

    for(;;){

        vTaskDelay((1000/hertz)/portTICK\_PERIOD\_MS);

        gpio\_set\_level(ROOD\_GPIO,1);

        vTaskDelay((1000/hertz)/portTICK\_PERIOD\_MS);

        gpio\_set\_level(ROOD\_GPIO,0);

    }

}

void groen\_task(void \*pvParameters){

    double hertz = 0.2;

    gpio\_pad\_select\_gpio(GROEN\_GPIO);

    gpio\_set\_direction(GROEN\_GPIO, GPIO\_MODE\_OUTPUT);

    for(;;){

        vTaskDelay((1000/hertz)/portTICK\_PERIOD\_MS);

        gpio\_set\_level(GROEN\_GPIO,1);

        vTaskDelay((1000/hertz)/portTICK\_PERIOD\_MS);

        gpio\_set\_level(GROEN\_GPIO,0);

    }

}

Deze twee tasks start ik vanaf de app\_main. Ik heb de lampjes achter GPIO 12 en 13 gezet met achter elk een 230 ohm weerstand. In de volgende link kan je het circuit in actie zien.

<https://mymedia.avans.nl/media/t/1_kh1k9ver>

## Opdracht 3

In deze opdracht heb ik de volgende code gebruikt:

for(int i = 1; i<=20;i++){

        char name[] = "task20";

        name[4] = i/10+'0';

        name[5] = i%10+'0';

        if (i<10)

        {

        xTaskCreate(&name\_task, name, 2048,NULL,5,NULL );

        }

        else{

        xTaskCreate(&name\_task, name, 2048,NULL,3,NULL );

        }

    }

Deze code bevat 9 tasks met prioriteit 5 en de andere 11 hebben prioriteit 3. Deze tasks bevatten alleen deze code:

for(;;){

        printf("%s\n",pcTaskGetTaskName(NULL));

    }

De console ziet er uit dat alle tasks die voorkomen hebben een 0 na task dus weet je dat alleen de eerste 9 tasks aan de beurt komen en de andere 11 aan starvation leiden.

# Practicum 3

## Opdracht 1: Dining Philosophers Problem

Deze opdracht was bedoeld om de semaphores te gebruiken. Dit heb ik niet kunnen doen doordat ik de semaphores op geen manier werkend kreeg. Wel heb ik de opdracht gemaakt met een boolean array die de semaphores zouden voor moeten stellen.

In deze opdracht heb ik gebruik gemaakt van 4 hulp methodes naast de main en de task. Deze hulp methodes waren gemaakt om de semaphores te pakken en terug te leggen en de leds te initialiseren en de status en leds te zetten wanneer de leds aan staan.

Voor de leds heb ik pinnen 12, 13, 14, 26 en 27 gebruikt en met een weerstand van 220 ohm ze naar ground geleid.

Voor een filmpje van de werking zie de volgende link:

<https://mymedia.avans.nl/media/t/1_1znvt9ia>

Voor de code zie de github of de volgende link:

<https://github.com/rpvos/OSLinux/blob/main/practicum3opdracht1/main/main.c>

## Opdracht 2: Readers-Writers Problem