ESP Weektaak 3: Task Priorities en Interrupts

# FreeRTOS Hoofdstuk 7 en 9

# Inleveren

* Het week3.txt met jullie antwoorden voor opdrachten 1, 2, 3, 4 en 5.
* Het week3-2.ino bestand met de aangepaste code voor opdracht 2.
* Het week3-5.ino bestand met de aangepaste code voor opdracht 5.
* Het week3-6.ino bestand met de code voor opdracht 6.
* Een link naar je GitHub Repo in de beschrijving. (Met je Practicum docent hierin toegevoegd.)

# Algemene regels

1. **Te laat inleveren:** is een automatische onvoldoende voor de eerste kans.

* Deadlines zijn onafhankelijk van het moment dat de eerste practicum les plaatsvindt. Zorg dus dat je zelfstandig al een begin maakt met de practicumopdracht van die week!

1. Alle programmeeropdrachten worden bijgehouden in [GitHub](https://www.github.com/). Zorg dat je repository privé is en dat je je docent toegang geeft.
2. Er wordt gewerkt in duo's met **Pair Programming** (d.w.z. dat beide studenten aan een laptop tegelijk werken aan dezelfde code, spreek onderling af wanneer je het typen af wisselt. Terwijl de een typt kan de ander al nadenken/opzoeken hoe het volgende deel geschreven moet worden).
   * Als één van de twee groepsleden onvoldoende kennis van het gemaakte werk kan tonen, resulteert dit in een onvoldoende voor beide groepsleden
   * Alleen werken is bij uitzondering toegestaan. Overleg dit altijd met je practicumdocent.  Werken in drietallen is niet toegestaan.
   * In het geval van een onevenredige verdeling van de werklast mag ervoor gekozen worden om een duo, in overleg met de practicumdocent, op te breken. Dit duo mag dan niet op een later moment weer bijeenkomen.
3. Code die is overgenomen van het internet (of sterk is gebaseerd op code van het internet) dient te worden voorzien van bronvermelding. Een gebrek aan bronvermelding geldt als plagiaat en zal worden gemeld bij de examencommissie.
4. Kennis uitwisselen en van elkaar leren is toegestaan en wordt aangemoedigd! Het overnemen van code geldt echter als **plagiaat** en zal worden gemeld bij de examencommissie.
5. Namen van variabelen, functies en alle andere namespaces in de scripts zijn origineel en dus zelfbedacht.
6. Alle scripts dienen te worden voorzien van **zelfgeschreven commentaar**. Bij voorkeur in het Engels, maar Nederlands is evengoed toegestaan.
7. **Herkansingen/2e gelegenheid:** De student heeft recht om elke weektaak te herkansen.
8. **Beoordeling**: Elke opdracht wordt beoordeeld met een (O)nvoldoende of (V)oldoende.
   * Voor een voldoende dienen alle weekopdrachten te zijn afgerond met een Voldoende.

Vak docenten: Roy van Leeuwen (LERO) en Jan-Wiepke Knobbe (KNJA)

## Opdrachten

1. **Prioritering van taken**

Lees hoofdstuk 7 van het lesboek ”freertos for esp32”.

1. Maak de opdrachten 1 t/m 13 op bladzijde 150 van het lesboek ”freertos for esp32”.

Lever de gemaakte opdrachten in: “week3.txt”.

1. **Hydraulische pers met correct tijdstempels in de audit\_queue**

Vorige week hebben we er voor gezorgd dat er tijdstempels worden opgeslagen in een queue als er tegelijk op beide knoppen wordt gedrukt. De pers kan echter al geactiveerd worden voordat de juiste tijd is verkregen van de NTP-server. Door prioriteiten aan de verschillende taken toe te kennen, gaan we ervoor zorgen dat de pers pas geactiveerd wordt als de juiste tijd bekend is. We willen dat de taken in de volgende volgorde worden uitgevoerd: allereerst de taak die de Wifi verbinding opzet, daarna de taak die ervoor dat de tijd door de NTP-server wordt gezet en tot slot de taken die ervoor zorgen dat een druk op de knoppen wordt verwerkt

Pas het programma “week2-3.ino“, dat jullie hebben geschreven voor het huiswerk van les 2 (opdracht 3) op de volgende manier aan:

* 1. Zorg dat de functie “WiFi\_connect()” wordt omgezet in een taak die wordt gecreëerd in de setup functie m.b.v. de functie xTaskCreatePinnedToCore(..). Geef deze taak hierbij als prioriteit 3.
  2. Zorg dat de functie “SNTP\_connect()” wordt omgezet in een taak die wordt gecreëerd in de setup functie m.b.v. de functie xTaskCreatePinnedToCore(..). Geef deze taak hierbij als prioriteit 2.
  3. Zorg dat aan de taken “debounceL”, “debounceR” en “PressTask” de prioriteit 1 wordt toegekend.
  4. Voeg onderstaande globale variabelen, waarmee wordt geteld hoe vaak een taak is uitgevoerd, toe aan de code:

volatile int wifi\_delay\_counter = 0;  
volatile int ntp\_delay\_counter = 0;  
volatile int press\_task\_counter = 0;  
volatile int debouncel\_left\_task counter = 0;  
volatile int debounce\_right\_task counter = 0;  
static int MAX\_COUNTER\_VALUE = 5;

* 1. Voeg in de taak WiFi\_connect een aantal print statements toe om aan te geven welk deel van de taak is uitgevoerd:
     + Vlak voor het statement Wifi.begin(ssid, password)
     + Direct na het statement Wifi.begin(ssid, password)
     + Vlak voor het statement delay(500)
     + Direct na het statement delay(500)
  2. Voeg in de taak SNTP\_connect een aantal print statements toe om aan te geven welk deel van de taak is uitgevoerd:
     + Vlak voor het statement delay(500)
     + Direct na het statement delay(500)
  3. Voeg onderstaande code toe aan het begin ***en*** aan het einde van de taak “debounce\_task”.

if ( button\_gpio== GPIO\_BUTTONR ) {

debounce right task counter++;

if ( debounce\_right\_task\_counter < MAX\_COUNTER\_VALUE ) {

printf("\t\tin debounce\_task(): RIGHT start %d\n", debounce\_right\_task\_counter );

}

}

else if ( button\_gpio== GPIO\_BUTTONL ) {

debounce left task counter++;

if ( debounce\_left\_task\_counter < MAX\_COUNTER\_VALUE ) {

printf("\t\tin debounce\_task(): LEFT start %d\n", debounce\_left\_task\_counter );

}

}

* 1. Voeg onderstaande functie toe aan het begin ***en*** aan het einde van de taak “press\_task”.

if ( press\_task\_counter < MAX\_COUNTER\_VALUE ) {

printf("\t\tin press\_task(): end %d\n", press\_task\_counter );

}

* 1. Voeg in de functie setup() een aantal print statements toe om aan te geven welk deel van de functie is uitgevoerd:
     + vlak voor de creatie van elke taak
     + direct na de creatie van elke taak
  2. Voer het programma uit en leg aan de hand van de output van het programma uit waarom de pers, ondanks de aangegeven prioritering van de taken, al geactiveerd kan worden voordat de juiste tijd van de NTP-server is ontvangen.
  3. Zorg dat de pers pas geactiveerd wordt als de juiste tijd bekend is door bepaalde taken te pauzeren en te herstarten nadat de verbinding met de NTP-server tot stand is gekomen.

Lever het bestand met de code (van opdracht k) in als “week3-2.ino” samen met het antwoord op vraag j in “week3.txt”.

1. **Interrupts**

Lees hoofdstuk 9 tot het onderdeel ”Running the demo” op pagina 177 van het lesboek ”freertos for esp32”.

1. Maak de opdrachten 2, 3, 4, 5, 8 en 15 op bladzijde 190 van het lesboek ”freertos for esp32”.

Lever de gemaakte opdrachten in: “week3.txt”.

1. **Interrupts: timer met 1 drukknop en 1 LED**

We willen dat door het indrukken van een knop (of het omzetten van een schakelaar) een interrupt wordt gegenereerd, dat een lampje afwisselend aan of uit zet. De status van de drukknop (of schakelaar) moet periodiek worden gecontroleerd door middel van een hardware timer. Hierbij moet uiteraard ook rekening worden gehouden met het “debounce” effect.

a. Importeer de library ”ESP32TimerInterrupt.h” in de Arduino IDE om de HW timer van de ESP32 te gebruiken: ”Tools” −→ ”Manage Libraries” −→ zoek ”ESP32TimerInterrupt” en installeer **versie 1.5.0**).

b. Download “ch9-int\_hwtimer\_std.ino” van blackboard, run de code en bestudeer de werking.

c. Leg uit hoe men de schakelaars ontdendert (debounce).

d. Leg uit wat de taak van de timer interrupt is

Lever de antwoorden op vragen c. en d. in: “week3.txt”.

1. **Interrupts: timer met 2 drukknoppen en 1 LED**

We willen de code uit opgave 4 zo aanpassen dat het lampje ook met een 2e drukknop aan of uit gezet kan worden (net zoals we dat in week 2 zonder interrupts hebben gedaan). De twee knoppen hebben dus precies hetzelfde effect: het lampje wordt aan- of uitgezet.

a. Download “ch9-int\_hwtimer\_2buttons\_std.ino” van blackboard.

b. Breid de code uit zodat ook het indrukken van de 2e knop ervoor zorgt dat er een interrupt wordt gegenereerd, dat een lampje afwisselend aan of uit zet.

c. Leg uit wat de taak van de timer interrupt is in week3.txt

Lever het bestand met de code in als “week3-5.ino”.

1. **Interrupts: binary Semaphore**

In opgave 5 van week 2 hebben we een semaphore gebruikt om te zorgen dat een LED met 2 verschillende snelheden tien keer knippert. In de code worden de drukknoppen (schakelaars) ontdenderd door een debounce\_task (één voor elke knop). Deze taken draaien continue.

a. De opdracht is om de debounce taken te vervangen door interrupts, zodat er zo weinig mogelijk cpu tijd wordt besteed aan het ontdenderen..

Lever het bestand met de code in als: “week3-6.ino”.

Feedback verwerk van Coen Hegeman (22/12/2023)