ESP Weektaak 2: Queues en Semaphores

FreeRTOS Hoofdstuk 3 en 5

Inleveren

- Het week2-1.ino bestand met de aangepaste code voor opdracht 1.
- Het week2-1.png bestand met de screenshot van de console output voor opdracht 1.
- Het week2-2.txt bestand met jullie uitleg voor opdracht 2.
- Het week2-3.ino bestand met de aangepaste code voor opdracht 3. Het week2-4.txt bestand met jullie anwoorden voor opdracht 4.
- Het week2-5.ino bestand met de aangepaste code voor opdracht 5.
- 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!

2. Alle programmeeropdrachten worden bijgehouden in GitHub. Zorg dat je repository privé is en dat je je docent toegang geeft.
3. 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.

4. 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.
5. 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.
6. Namen van variabelen, functies en alle andere namespaces in de scripts zijn origineel en dus zelfbedacht.
7. Alle scripts dienen te worden voorzien van zelfgeschreven commentaar. Bij voorkeur in het Engels, maar Nederlands is evengoed toegestaan.
8. Herkansingen/2e gelegenheid: De student heeft recht om elke weektaak te herkansen.
9. 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. Real time clock

   We gaan nu code toevoegen om de Real Time Clock op de ESP32 in te stellen met de juiste tijd die we ophalen van een NTP server.

   Voeg aan het programma “week1-3.ino“, dat jullie voor het huiswerk van les 1 hebben gebruikt om met wifi te verbinden, de volgende code blokken toe:

   a. Voeg onderstaand include toe aan de code:

```
// add header file for SNTP client
#include "esp_sntp.h"
//
```

   b. Voeg onderstaande constante toe aan de code:

```
// constants for SNTP client
const char* ntpServer = "nl.pool.ntp.org";
```

   c. Voeg onderstaande globale variabelen toe aan de code:

```
// global variables for timestamp
const int glob_buf_size = (64 * sizeof(char));
char *glob_time_buf;
```

   d. Vervang in onderstaande functie de regel ‘sntp_serservername(0, “pool.ntp.org”);’ door ‘sntp_serservername(0, ntpServer);’ en voeg deze functie vervolgens toe aan de code:

```
// connect with NTP server
void SNTP_connect(){
  const time_t old_past = 1577836800;  // 2020-01-01T00:00:00Z
  printf("\r\nConnect to SNTP server\n");

  // init time protocol sync
  sntp_setoperatingmode(SNTP_OPMODE_POLL);
  sntp_setservername(0, "pool.ntp.org");
  sntp_init();
  //https://knowledgebase.progress.com/articles/Article/P129473
  setenv("TZ", "CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00", 1);
  tzset();

  bool connected = false ;
  while ( not connected ) {
    delay(500);
    time_t now;
    if (time(&now) < old_past ) {   // if there is no connection with NTP server
than local time returns Unix epoch (1970 date)
      printf(".");
    }
    else {
      printf(" CONNECTED\r\n");
      connected = true;
    }
  }
}
```

   e. Bestudeer de documentatie van de strftime functie en voeg onderstaande functie toe aan de code:

```
// get current time
//https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/syste
m/system_time.html
void getLocalTime(char * time_buf, int time_buf_size) {
  time_t now;
  struct tm timeinfo;
  time(&now);
  localtime_r(&now, &timeinfo);
  strftime(time_buf, time_buf_size, "%c", &timeinfo);
  return ;
}
```

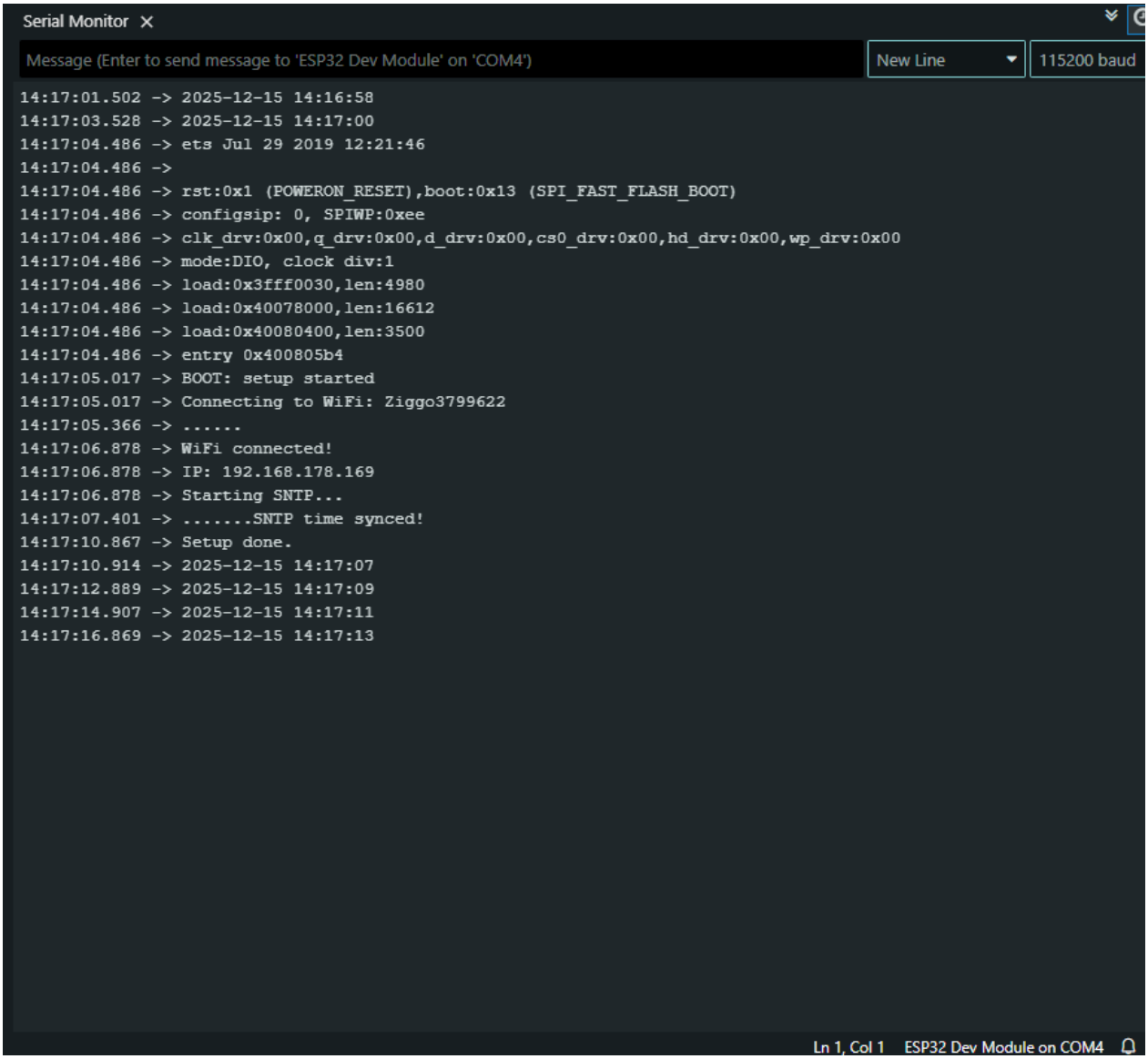
f. Zorg dat functie SNTP_connect() wordt aangeroepen in de setup functie en voeg onderstaande code toe aan de setup functie:

```
  // allocate heap memory to store 1 timestamp
  glob_time_buf = (char*)malloc(glob_buf_size);

  // check if memory is allocated
  assert( glob_time_buf != NULL );
```

g. Pas de loop functie aan zodat de tijd periodiek wordt geprint door de functie getLocalTime(…) aan te roepen (met de juiste globale variabelen) en vervolgens de tijd te printen

Lever het bestand in als “week2-1.ino” samen met een screenshot van de console output als “week2-1.png”.



2. Debouncing
   Lees hoofdstuk 3 van het lesboek ”freertos for esp32”. Besteed extra aandacht aan het onderdeel ”Demonstration” vanaf pagina 59.
   a. Bouw het schema van figuur3-2 na (Tip laat de Ledjes van de vorige opdracht zitten).

Ik heb het schema uit figuur3-2 opgebouwd op het breadboard. De LED is aangesloten via een weerstand oop GPIO23 en gaat naar GND. De twee schakelaars zijn aangelsoten op GPIO18 en GPIO! en de andere kant van elke knop is aangesloten op GND. In de code gebruik ik INPUT_PULLUP, dus los = HIGH en ingedrukt = LOW

b. Implementeer listing 3-3 (”press2.ino” staat beschikbaar op blackboard).

c.

Ik heb listing 3-3 (press2.ino) geïmplementeerd en geüpload naar de ESP32. De LED gaat alleen aan wanneer beide knoppen tegelijk worden ingedrukt. Door debouncing wordt één druk niet als meerdere drukken gezien en werkt het stabiel.

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

Een medhanische drukknop stuitert bij indrukken en loslaten. In de eerste milliseconden maakt hij meerdere snelle aan / uit overgangen, waardoor de microcontroller een druk als meerdere drukken kan zien. Debooucing betekent dat je en verandering pas accepteert als het singaal een korte tijd zoals 20 ms stabiel blijft. Dit kan met software (wachten en opnieuw meten) of met hardware.

Ik heb de schakelaars getest door GPIO18 en GPIO19 kort met GND te verbinden (als simulatie van een knop).

Lever de uitgeschreven uitleg in als week2-2.txt

3. Hydraulische pers met een audit_queue
   Stel dat de code van listing 3-3(“press2.ino”) bij een hydraulische pers wordt gebruikt. De hydraulische pers wordt geactiveerd wanneer de twee knoppen tegelijk worden ingedrukt.

   De eigenaar van de pers wil weten op welke momenten de pers recent geactiveerd is geweest.

   Dat betekent dat elke keer wanneer de pers geactiveerd wordt er een tijdstempel moet worden opgeslagen. Men wil de afgelopen 200 keer dat de pers is geactiveerd opslaan in een queue. Voor nu hoeft er verder nog niks gedaan te worden met deze tijdstempels.

   a. Je zult de uitbreiding die de functionaliteit voor het verbinden met wifi en het synchroniseren van de Real Time clock met de NTP server moeten toevoegen aan “press2.ino”.

   b. Maak een queue aan genaamd "audit_queue" waar jullie de tijdstempels van getLocalTime() in opslaan. Als de queue vol is moet de oudste tijdstempel verwijderd worden en moet de nieuwe tijdstempel aan de queue worden toegevoegd.

```
20:00:43.972 -> WiFi FAILED (timeout)
20:00:43.972 -> Stop: WiFi not connected.
20:01:56.364 -> .....
20:01:57.595 -> WiFi connected!
20:01:57.595 -> IP: 192.168.178.169
20:01:57.595 -> Starting SNTP...
20:01:58.079 -> ......SNTP time synced!
20:02:01.079 -> Setup done.
20:02:01.111 -> PRESS ACTIVATED at: 2025-12-18 20:01:58
20:02:01.111 -> Audit queue count = 1
20:02:01.111 -> Oldest = 2025-12-18 20:01:58
```

   Lever het bestand in als “week2-3.ino”.

4. Semaphores
   Lees hoofdstuk 5 van het lesboek ”freertos for esp32”.
   a. Maak de opdrachten 5, 6 en 7 op bladzijde 115 van het lesboek ”freertos for esp32”.

   Lever de gemaakte opdrachten in als: “week2-4.txt”.

   1- What is the initial state of a binary semaphore? Empty or given?

   Een binary semaphore start nomraal gezien in de empty / taken toestand (waarde 0).

Dat betekent niemand kan hem take() totdat een taak eerst give() doet.

2- What is the initial state of a counting semaphore?

Een counting semaphore heeft bij het aanmaken een startwaarde(intital count) die je zelf kiest.

- Vaak is dat 0 als je events wil tellen
- Of het is gelijk aan aantal beschikbare resources

3- How does the counting semaphore prevent a deadlock in the dining philosophers problem?

Bij dining philosophers ontstaat deadlock als idereen tegelijk een vork pakt en wacht op de tweede.

Een counting semaphore voorkomt dat door het aantal filosofen dat “tegelijk mag proberen te eten” te beperken (toegang limiteren)

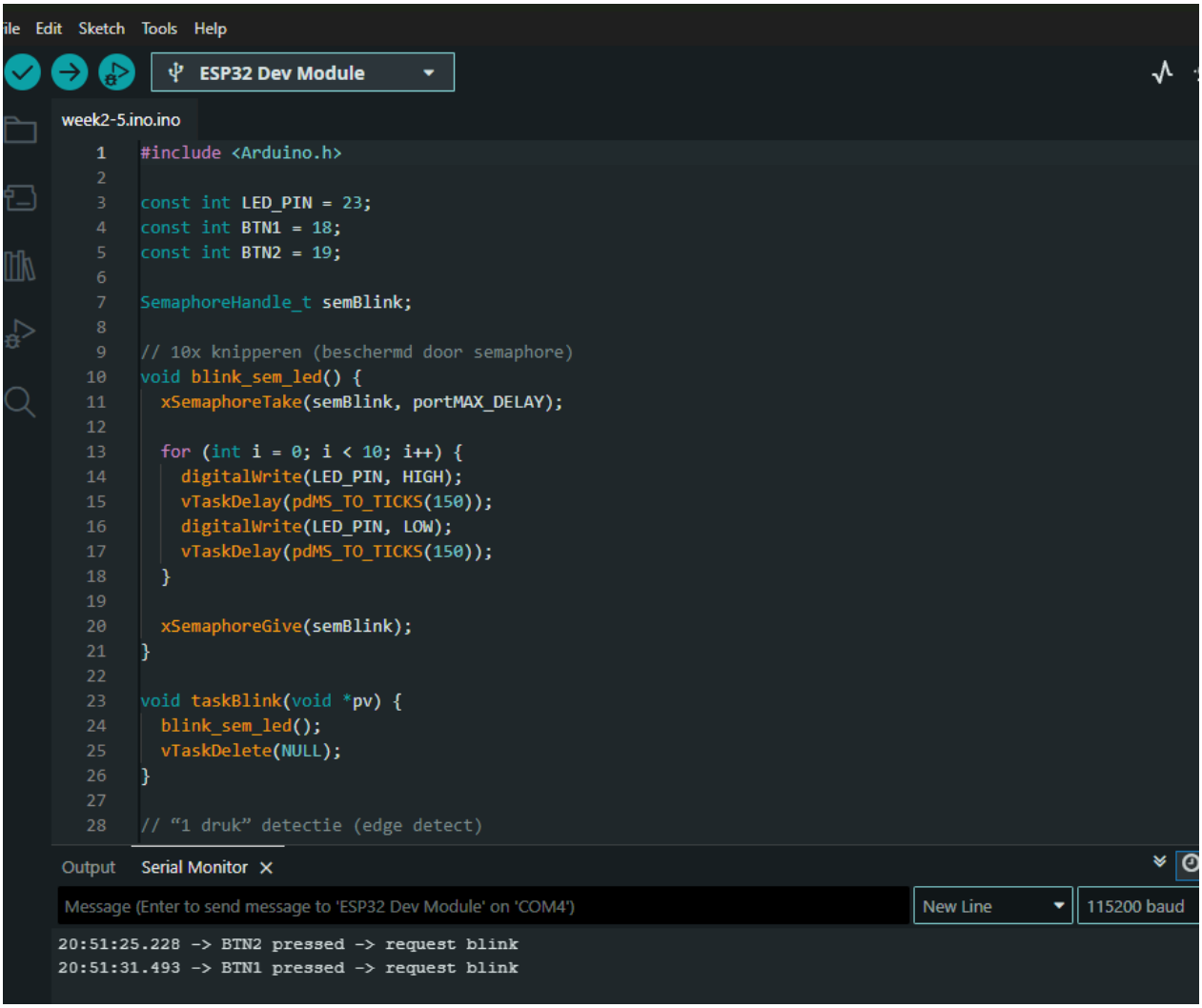
5. Binary Semaphore

Sluit twee schakelaar en 1 LED aan op de ESP32 en download ”ch5_2button _semaphore _std.ino” van blackboard.

a. Zorg dat de LED en knoppen correct aan GPIO-poorten toegewezen worden. Dit kunnen (en mogen) jullie ook aanpassen in de code als dit beter uit zou komen met jullie ESP en/of breadboard.

Run de code. Als je op een van de knoppen drukt dan moet het LEDje tien keer gaan knipperen. Als je op de andere knop drukt dan zal het LEDje nog eens tien keer knipperen. Druk je op een van de knoppen terwijl de “blink_sem_led” functie al is aangeroepen dan zal je zien dat het LEDje niet meer op hetzelfde tempo knippert.

b. Om te voorkomen dat de blink sem led door twee of om meerdere taken tegelijk wordt aangeroepen gaan we een binaire semaphore gebruiken. Maak in het begin van de code een handler aan die in de setup de wordt geïnitialiseerd/gekoppeld aan een binaire semaphore. In de “blink_sem_led” functie dien je de semaphore te activeren en nadat de functie klaar is geef je de semaphore vrij zodat een andere taak (knop die ingedrukt wordt) de functie kan gebruiken. Als alles naar behoren werkt dan zal de tweede taak niet meer uitgevoerd worden zolang de eerste taak nog niet klaar is met de functie.



Lever het aangepaste bestand in als: “week2-5.ino”.

Het Githube linkje omdat ik niet in de comment kan plakken

https://github.com/Muhand-lab/ESP-week-2