# Besturingssystemen WPO8: “Task scheduler” op de PIC18F2455

In dit labo maken we eens scheduler die op de microcontroller taken sequentieel zal laten uitvoeren. We maken de scheduler non-preemptive. Dit betekent dat wanneer een taak wordt uitgevoerd er geen andere taak wordt gestart. De discipline waar volgens de te ontwerpen scheduler zijn taken uitvoert is round robin. De taken worden uitgevoerd naargelang de volgorde waarin ze gedefinieerd zijn. Taak 1 wordt uitgevoerd, hierop volgt taak 2, tot de lijst van taken is uit waarop deze opnieuw start. De scheduler is logischerwijs gebaseerd op tijd. Hiervoor hebben we in WPO7 een real-time klok (RTC) gemaakt. De taken zullen met een minimaal interval van 50 ms worden uitgevoerd. Dit werd zo in WPO7 gedefinieerd.

## Definiëren van de structuur van een taak

Een taak zal een structuur zoals weergegeven in figuur 1 bevatten.

```
typedef struct task {
  unsigned long period;      // Rate at which the task should tick
  unsigned long elapsedTime; // Time since task's last tick
  void (*TickFct)(void);     // Function to call for task's tick
} task;
```

Figuur 1: Structuur van een taak

Hierin worden volgende variabelen bijgehouden:

- **task.period**: om welk tijdsinterval de taak wordt uitgevoerd
- **task.elapsedTime**: de tijd sinds de vorige uitvoering van de taak
- **task.TickFct**: de functie zelf die de taak uitvoert

Aan de hand van deze structuur kan een taak aangemaakt worden.

## Aanmaak van een taak

Het aanmaken van een taak gebeurt als weergegeven in figuur 2.

```
static const unsigned int tasksNum = 2; //aantal taken
static const unsigned long periodADC  = 20; //s
static const unsigned long periodLEDpulse = 1; // één periode: 50ms

void setupTasks(void)
{
   unsigned int i = 0;
   tasks[i].period      = periodLEDpulse; //task 1
   tasks[i].elapsedTime = periodLEDpulse;
   tasks[i].TickFct     = &LEDpulse; //aan te roepen functie
   ++i;
   tasks[i].period      = periodADC; //task 2
   tasks[i].elapsedTime = periodADC;
   tasks[i].TickFct     = &ADCRead;
}
```

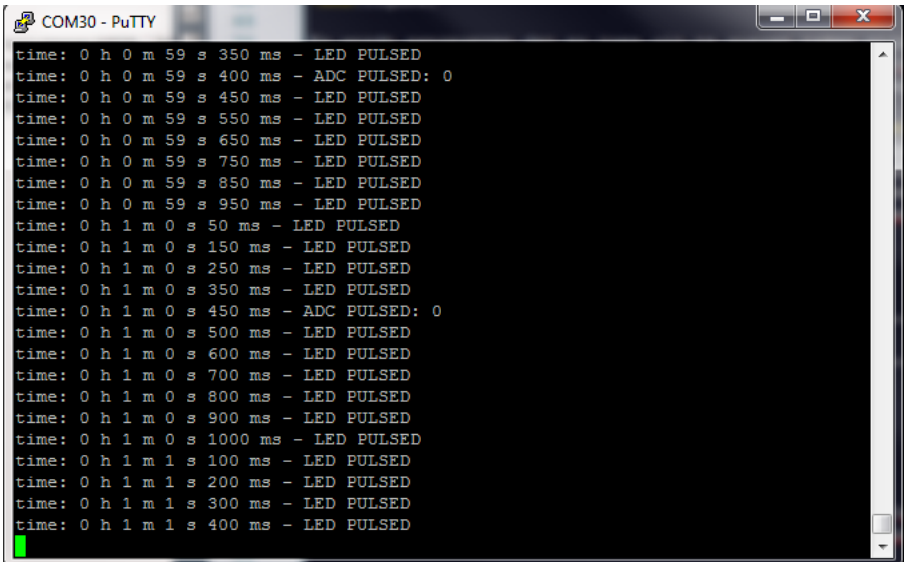
Figuur 2: Setup van de taken: LEDpulse en ADCRead

Er wordt aangegeven dat er twee uit te voeren taken zijn. Het tijdsinterval zoals reeds gezegd was 50 ms. Taak 1 zal altijd voor taak 2 worden uitgevoerd. In de variabele **tasksNum** wordt meegegeven hoeveel taken er worden uitgevoerd. In **periodADC** en **periodLEDpulse** wordt meegegeven om de hoeveel periodes de taak moet ticken (uitgevoerd worden).

## Opdracht WPO8: "Scheduler: Interrupt service routine"

De opdracht van dit labo is het schrijven van de interrupt service routine (ISR) die taken zal plannen. Deze ISR wordt aangeroepen in de while lus, maar zal enkel om de bepaalde periode (50 ms) controleren of er taken moeten worden uitgevoerd. Men kan zich hiervoor baseren op het RIOS besturingssysteem [2]. Op deze website is een voorbeeld van een scheduler gegeven van een simulatie in C en van een Atmega microcontroller.

Op het einde van het labo wordt verwacht dat een output gelijkaardig aan 3 bekomen wordt. De inhoud van de taak LEDPulse en de taak ADCPulse worden meegegeven in een extra document. De taak LEDpulse dient elke 50 ms uitgevoerd te worden en de taak ADCRead om de seconde. Dit weerspiegelt in een periode van 1 voor de taak LEDpulse en 20 voor de taak ADCpulse.




Figuur 3: PuTTy output van de scheduler

**Extra:** In de PuTTy monitor merken we de op dat de taak LEDpulse om de 100 ms wordt uitgevoerd. Wanneer men zelf de code van de scheduler uitvoert zal dit normaliter gezien ook zijn. Verklaar waarom dit is als je weet dat de baudrate van de UART module 9600 baud is. Schrijf de beredenering in commentaar.

## Referenties


1. Datasheet: PIC18F2455, http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf
2. Task scheduler: Riverside-Irvine operating system, http://www.riosscheduler.org/