Jochem van Kranenburg (1033795)

27-12-2021

Eindopdracht CSC10

Een reactiemeter in verschillende uitvoeringen

Abstract

In dit verslag zal de voor CSC10 uitgevoerde eindopdracht met de bijbehorende deelopdrachten worden toegelicht.

Inhoud

[Samenvatting 1](#_Toc91508820)

[Inleiding 2](#_Toc91508821)

[Applicatie 2](#_Toc91508822)

[Deelsystemen 2](#_Toc91508823)

[Real-time operating system 2](#_Toc91508824)

[Platform designer 3](#_Toc91508825)

[Linux 3](#_Toc91508826)

[Conclusie 4](#_Toc91508827)

[Bijlagen 5](#_Toc91508828)

[RTOS timings 5](#_Toc91508829)

[Platform designer timings 5](#_Toc91508830)

[Linux timings 6](#_Toc91508831)

# Samenvatting

In dit document wordt er ingegaan op de gerealiseerde reactietijd metende applicaties en de bij ieder platform behorende mogelijkheden en sterke en zwakke punten. Uit dit verslag komt dan ook naar voren dat het RTOS gebaseerde systeem erg accuraat presteert en in bijzonder weinig tijd uit de grond kan worden gestampt. Het platform designer component in combinatie met het RTOS komt slechter uit de bus door de grotere ontwikkeltijd en de beperkte toename in accuraatheid en mogelijkheden. Het platform designer component in combinatie met Linux levert een enorme hoeveelheid aan extra mogelijke functionaliteiten op zonder dat er daarbij accuraatheid verloren gaat en de ontwikkeltijd al te veel toeneemt.

# Inleiding

In de slotperiode van de cursus Configurable System on Chip van de minor Embedded Systems aan de Hogeschool Rotterdam is er gewerkt aan een reactiemeter. Ondanks dat een reactiemeter in essentie erg simpel is, zijn er veel uitbreidingsmogelijkheden die het tot een concreet en uitdagend systeem kunnen maken. Daarbij komt ook kijken dat het realiseren van meerdere – functioneel gezien identieke – applicaties het behalen van de leerdoelen van de cursus vereenvoudigd.

Het doel betreft dan ook het implementeren van een reactiemeter op een zodanige manier dat alle leerdoelen worden aangetoond. Om al deze leerdoelen aan te tonen, moet er worden gewerkt met drie deelsystemen op drie platformen; een RTOS op de NIOS-II softcore, Linux dat draait op de HPS en een Platform Designer[[1]](#footnote-1) component dat ter compleetheid wordt aangevuld door een RTOS.

## Applicatie

Deze applicaties bieden in de kern exact dezelfde functionaliteit aan. Bij het opstarten – en daarna iedere 3 rondes – dient de gebruiker de naam van de gebruiker in te voeren in de terminal.

Zodra er op KEY 0 wordt gedrukt, treed een willekeurige vertraging in werking. Nadat deze willekeurige vertraging is afgelopen, zullen de leds stuk voor stuk aan gaan om de gebruiker ervan op de hoogte te stellen dat het zaak is zo snel mogelijk op KEY 1 te drukken. Vanaf het moment dat de gebruiker KEY 1 indrukt tot het moment dat deze wordt losgelaten, zal de reactietijd worden weergegeven op de zeven segment displays. Indien er een high-score is behaald, zullen de leds cirkelen ter versterking van de euforie van de gebruiker.

Nadat KEY 1 wordt losgelaten, komt de applicatie in de rust toestand. In deze toestand bevind zich op de linker displays het aantal pogingen en op de vier rechter displays de tot op heden behaalde highscore.

Deze functionaliteit kan door de deelsystemen worden aangevuld met extra functionaliteiten.

# Deelsystemen

## Real-time operating system

De RTOS gebaseerde reactiemeter is bewonderenswaardig snel te realiseren. Binnen de volledige vijf uur was het systeem up-and-running. Deze korte ontwikkeltijd resulteert echter niet in een belabberde accuraatheid, in tegendeel; er wordt een reactietijd van 336ms weergegeven op de displays terwijl de logic analyzer een reactietijd van 334.89 aangeeft.

Het systeem werkt als volgt. Bij het opstarten wordt de high-score en het aantal pogingen, net zoals bij alle andere systemen, met behulp van de custom platform designer seven segment display controller component op de displays weergegeven. De knoppen genereren interrupts die door de irq handler worden afgevangen. In deze handler worden semaphores ge-post die in de state machine worden gepend.

Zodra de start semaphore wordt gepost, gaat de state machine implementerende taak wachten op de stop semaphore. Zodra deze kan worden gepend, wordt de system time gebruikt als seed voor *rand()*. Als de random delay is afgelopen en de knop al wordt ingedrukt, treed de random delay nogmaals op. Dit gaat door totdat de knop niet is ingedrukt als de delay afloopt.

In de volgende toestand gaan de leds stuk voor stuk aan en wordt er met een timeout van 1 milliseconde gewacht op de stop sempahore die wordt gezet als de reactieknop wordt ingedrukt. Indien de reactietijd te kort is, wordt de meting genegeerd. Anders wordt deze op de displays weergegeven. Na het loslaten begint het verhaal van voor af aan.

Afgezien van de state-machine implementerende taak, is er ook een console taak die commando’s van de gebruiker doorsluist naar de state-machine taak. De communicatie tussen deze twee taken is geïmplementeerd in de vorm van een messsage queue die door de state-machine om de 3 beurten wordt gepend; zo kan de naam worden ingesteld.

## Platform designer

Het platform designer gebaseerde systeem lijkt in alle aspecten op het RTOS gebaseerde systeem met als uitzondering het feit dat er niet direct met knoppen wordt ge-interfaced en dat de reactie-metende state-machine is geïmplementeerd in het custom platform designer component.

Er is dan ook een irq handler[[2]](#footnote-2) van het response time meter platform designer component. In die handler kan uit register 0 de toestand worden gelezen. Op basis van deze toestand kunnen semaphores worden gepost. Het afhandelen van knoppen gebeurt binnen het component. De applicatie dient dus enkel de leds en displays aan te sturen op de bijbehorende momenten die worden aangeduid door gepende semaphores.

Het component heeft 2 leesbare en een enkel schrijfbaar register. De leesbare registers bevatten de reactietijd en de huidige toestand. Met het schrijfbare register kan de (dan dus niet meer random) delay worden ingesteld.

In de code valt dan ook op dat de state machine van het RTOS gebaseerde systeem is opgesplitst in twee nieuwe state-machines; een state-machine in VHDL van het component en een high-level state-machine in de applicatie. In het component vind dus ook de detectie van invalide reactietijden en het al indrukken van de knop bij het aflopen van de delay plaats.

Het realiseren van het component kost meer tijd en is foutgevoeliger, maar resulteert wel in een perfectie reactietijdmeting die exact overeenkomt met de meting van de logic analyzer.

## Linux

Aangezien de platform designer components voor het aansturen van zeven segment displays en uitvoeren van reactie-metingen reeds zijn geïmplementeerd en getest, dienen er enkel nog kernel modules van gemaakt te worden. Met behulp van de aangeboden tutorial is dat een fluitje van een cent. Gezien er sprake is van dezelfde splitsing in state machines als bij het hoofdstuk platform designer, is de software van de user-space applicatie erg overeenkomend.

Het verschil is echter dat het op Linux eenvoudig is om voor embedded systems complexe dingen te doen. Het is dan ook een kleine moeite een post request te versturen naar een remote server, of high-scores bij te houden in een bestand. Dit bestand houdt bij wie hoe laat en op welke dag hoe snel heeft gereageerd.

# Conclusie

De RTOS gebaseerde implementatie is snel te realiseren en heeft een afwijking van gemiddeld 1ms. De RTOS en platform designer gebaseerde implementatie vereist een grotere ontwikkeltijd, maar resulteert in een meting zonder afwijking. De Linux gebaseerde implementatie vereist nog een extra software-laag, maar opent een hele reeks aan nieuwe functionaliteiten.

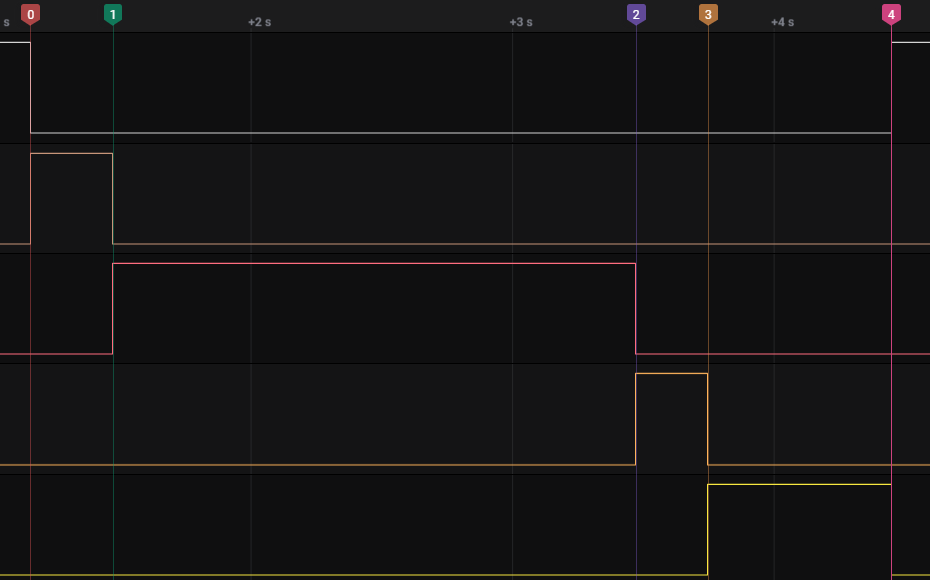
Het beste platform is dan ook afhankelijk van het doel en de reeks van klantenwensen. Als een afwijking van 1ms acceptabel is en zeker is dat de reactiemeter (ook in de toekomst) een op zichzelf staand, niet met de buitenwereld interfacend, apparaat is, ligt de keuze gezien de verhouding tussen ontwikkeltijd en prestatie voor de hand. Als een afwijking niet is toegestaan, maar het apparaat ten alle tijden niet uitgebreid hoeft te kunnen worden, is een RTOS en platform designer gebaseerde implementatie een goede keuze.

Bij twijfel over de noodzaak van flexibiliteit of de wens tot interfacing met de buitenwereld via bijvoorbeeld Wi-Fi, is Linux een goede optie. Linux vereist een initiele grote effort, maar is daarna eenvoudig uit te breiden.

# Bijlagen

## Afbeelding met tekst Automatisch gegenereerde beschrijvingAfbeelding met licht, kleurrijk, helder Automatisch gegenereerde beschrijvingRTOS timings

## Afbeelding met tekst Automatisch gegenereerde beschrijvingPlatform designer timings



## Afbeelding met tekst Automatisch gegenereerde beschrijvingAfbeelding met tekst, binnen Automatisch gegenereerde beschrijvingLinux timings

1. Voorheen bekend als Qsys. [↑](#footnote-ref-1)
2. Het reactie-metende component genereerd bij iedere toestandsovergang een interrupt. [↑](#footnote-ref-2)