Onderzoeksverslag RTOS



Maarten Wassenaar, Frank Tamer, Leon Zhang, Jeffrey de Waal, Arsalan Anwari

Onderzoeksverslag Thema Opdracht Devices

Thema Opdracht Devices

Begeleider: dhr Zuurbier

Utrecht, 31 november 2017

# Inhoudsopgave

[**Inhoudsopgave**](#_byfg1w40alk1) **2**

[**Inleiding**](#_wodlxytz46eb) **4**

[Hoofdvraag](#_3d21qsbvbwsu) 4

[Deelvragen](#_tl1wgqaqf0k4) 4

[**Methode**](#_mozrj6psljzm) **5**

[Onderzoeksmethode](#_zbboycpg9gx4) 5

[RTOS Criteria](#_bs61ok1iub7) 5

[**Resultaten**](#_t8u7b5v6itx0) **6**

[Deelvraag 1](#_ix7jonnwf5a4) 6

[Tasks](#_htkktk5mm3t4) 6

[Waitable](#_uepb57xx6bxn) 6

[flag](#_m1ads33s98g9) 6

[(one-shot) timer](#_79lwj7qzeixi) 6

[clock](#_b9e79fil7uzi) 6

[channel\_base](#_thaf54t4yjst) 7

[mutex](#_bs6sfd8sx7g2) 7

[mailbox\_base](#_uwbp1f522oxs) 7

[pool\_base](#_zegwg3fhxapv) 7

[Deelvraag 2](#_bfmunu2h9hhu) 7

[Deelvraag 3 & 4](#_4u7v24u2j7kg) 8

[Atomthreads](#_jh85i2qxjv6) 8

[ChibiOS/RT](#_fqvo6orumu0f) 8

[Concurrency mechanismen](#_54izwgzvky6) 8

[DistortOS](#_415vqwkghztk) 8

[Concurrency mechanismen](#_o48bgyovmbji) 8

[ECOS](#_ecnbw6c6yydx) 9

[Concurrency mechanismen](#_jbmnv69333g) 9

[Mbed-rtos](#_lkubz4y7nsgl) 9

[Concurrency mechanismen](#_49nbgrevppu3) 10

[NuttX](#_5mmnzoaem6hq) 10

[Concurrency mechanismen](#_9ycfvjmzab8) 10

[StateOS](#_va929m99ce47) 10

[**Conclusie en aanbeveling**](#_ve8i0b9po5gy) **11**

[**Literatuur**](#_18q0ms66lice) **12**

# 

# Inleiding

Voor het thema opdracht Devices maken wij gebruik van Arduino RTOS dat beschikbaar is gesteld door de Hogeschool Utrecht. In ons onderzoek gaan we het Arduino RTOS vergelijken met andere RTOS-en. Uiteindelijk gaan we aan de hand van een aantal criteria een keuze maken welk RTOS het beste alternatief is en in hoeverre deze de mechanismen bevat van het Arduino RTOS. In het geval dat een aantal mechanismen niet aanwezig zijn bij het gekozen RTOS, gaan we ook kijken hoe we deze wel kunnen realiseren.

## Hoofdvraag

*Met behulp van welke open source real-time operating system kunnen tasks en de concurrency mechanismen, pool, channel, flag(group), clock timer en mutex, zoals aangeboden door het Arduino RTOS, met zo weinig mogelijk overhead worden gerealiseerd?*

## Deelvragen

1. Wat zijn de kenmerkende eigenschappen van tasks en de concurrency mechanismen van het Arduino RTOS?
2. Welke open source RTOS-en zijn beschikbaar?
3. Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?
4. Hoe kunnen de mechanismen van het Arduino RTOS die niet direct worden ondersteund door de beschikbare RTOS-en worden gerealiseerd m.b.v. van deze RTOS-en?
5. Welk van de beschikbare RTOS-en biedt de meeste van de concurrency mechanismen van het Arduino RTOS aan zonder enige modificatie en biedt dezelfde functionaliteit om taken te realiseren?

# Methode

Bij ons onderzoek gaan wij gebruik maken van literatuurstudie als onderzoeksmethode. Hiervoor maken wij voornamelijk gebruik van de documentatie geleverd door de ontwikkelaars van de verschillende RTOS-en en verscheidene wikipedia pagina’s over de RTOS’en. Wij gaan ieder RTOS verder uitlichten in de resultaten. Wij gaan de RTOS-en beoordelen op hun updatebeleid, documentatie en concurrency mechanismen.

## Onderzoeksmethode

Wij hebben gekozen voor een literatuurstudie, omdat daarbij de antwoorden op de hoofd- en deelvragen binnen de beschikbare tijd te vinden zijn. Hierdoor kunnen wij vlot een conclusie trekken. Sneller dan bij veel andere onderzoeksmethodes.

## RTOS Criteria

Een RTOS, welke uitgekozen is als vergelijking, moet voldoen aan de volgende eisen:

* De RTOS moet werken op een Cortex m3 platform.
* De RTOS moet Open-source zijn.
* De RTOS moet C of C++ ondersteuning bieden.
* De RTOS moet afgelopen jaar minimaal eenmaal geupdate zijn en actief ontwikkeld worden.
* De RTOS moet over een uitgebreide documentatie beschikken.

# Resultaten

## Deelvraag 1

*Arduino RTOS is het RTOS dat door Wouter van Ooijen is ontwikkeld voor gebruik in Thema Opdracht Devices en voor gebruik in de CSPE1 lessen van Wouter. Dit RTOS zal de basis vormen voor de vergelijking met de andere Open-Source RTOS-en. Het RTOS dat hier ter sprake komt is pre-emptive.*

### Tasks

De gebruiker stelt een lijst van taken op. RTOS werkt aan de hand van de opgegeven taken, iedere taak heeft een ingestelde timer om bij te houden hoelang het moet wachten. Wanneer de timer afgelopen is, wordt deze gemarkeerd als ready en wordt verdergegaan met deze taak. Taken kunnen ook prioriteiten hebben (hoe lager het getal, hoe hoger de prioriteit) en de taken met de hoogste prioriteiten worden zo snel mogelijk uitgevoerd wanneer deze daartoe de beschikking krijgt. Verder kan een taak pas uitgevoerd worden wanneer deze zich in de status van *Resumed* en *non-waiting* bevindt*.*

### Waitable

Abstract object waar een taak op kan wachten. Bij RTOS zijn *flag*, *clock*, *timer* en *channel* concrete klassen die overerven van de abstracte klasse *Waitables*. Een *Waitable* is altijd gemaakt voor een bepaalde taak.Verder kan de *Waitable* zich in twee statussen bevinden, *cleared* en *set* (standaard op clear).

#### flag

Basisch synchronisatie mechanisme. Meerdere taken kunnen hiermee aan een enkele taak doorgeven dat een bepaald event is opgetreden, zonder dat daarbij overdracht plaatsvindt van data. Een *flag* is initieel *clear*. Wanneer een taak wacht op een *flag* en die *flag* wordt *set*, zal de *wait call* de *flag clearen* en zal die taak verder gaan.

#### (one-shot) timer

De *timer* die hier wordt gebruikt is een speciale type van een *flag.* De *timer* kan namelijk instructies krijgen om na een bepaalde tijd zichzelf op *set* te zetten. Wordt overschreven door een nieuwe *timer* wanneer deze op *set* gezet wordt. De timer kan ook onderbroken worden door de *cancel call.*

#### clock

De *clock* is een mechanisme dat altijd loopt en deze zet zichzelf op *set* na een bepaalde tijd die hij heeft meegekregen.

#### channel\_base

De *channel\_base* is een *waitable data queue*. Een *channel\_base* is bedoeld voor één taak. Deze taak wacht bij een *channel\_base* tot deze *non-empty* is en voert hier een *read* op uit. Alleen een *read* dat resulteert in een lege queue zal de *channel\_base clearen*.

### mutex

Een mutual exclusion semaphore is een synchronisatie mechanisme dat ervoor zorgt dat een taak exclusief toegang verkrijgt tot een bepaalde resource, waarop gegarandeerd wordt dat andere taken geen toegang hebben tot deze resource wanneer deze in gebruik is door een taak. Initieel is een *mutex free.*

### mailbox\_base

Een *mailbox* kan gebruikt worden om waarden tussen taken uit te wisselen, deze zijn niet gebonden aan een specifieke taak. Wanneer taak 1 schrijft naar een mailbox, zal deze wachten tot taak 2 deze uitleest. Als taak 2 uit de mailbox leest en niks kan lezen, zal deze wachten tot taak 1 iets in de mailbox schrijft.

### pool\_base

Een pool is een mechanisme om data uit te wisselen, zonder enige vorm van synchronisatie. Pool kent twee methoden, read() en write(), die garanderen dat ze *atomic* zijn.

## Deelvraag 2

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Naam** | **ARM** | **C of C++** | **Up-to- date** | **Documentatie** | **flag** | **timer** | **Clock** | **Mutex** | **Channel** | **Mailbox** | **Pool** |
| Arduino RTOS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Atomthreads | ✓ | ✓ | ✓ | ✓ | ✘ | ✓ | ✘ | ✓ | ✓ | ✘ | ✘ |
| [ChibiOS/RT](https://en.wikipedia.org/wiki/ChibiOS/RT) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| distortos | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| eCOS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✘ | ✓ | ✘ |
| [mbed-rtos](https://en.wikipedia.org/wiki/Mbed) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✘ | ✓ | ✓ | ✓ | ✓ |
| [NuttX](https://en.wikipedia.org/wiki/NuttX) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✘ | ✓ |
| StateOS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

## 

## 

## Deelvraag 3 & 4

### Atomthreads

Het Atomthreads RTOS is geschreven door Kelvin Lawson. Het systeem ondersteund het ARM Platform maar Atomthread is eigenlijk bedoeld voor een 8-bit cpu architectuur. ook zijn er een aantal concurrency mechanismen die ontbreken. Verder is de documentatie voldoende maar niet erg uitgebreid of overzichtelijk. De RTOS wordt ongeveer eens per 3 maanden geupdate.

### ChibiOS/RT

Het ChibiOS/RT developed door Giovanni Di Sirio is een RTOS die compact en snel werkt. De RTOS is ontworpen om te werken met embedded systemen op basis van een 8,16 en 32 bit microcontroller en is ook geschikt voor het Cortex-M3 platform. Met ChibiOS kan in C en Assembly geprogrammeerd worden. ChibiOS maakt gebruik Pre-emptive scheduling en lijkt mede hierdoor veel op het Arduino RTOS. ChibiOS bevat een uitgebreide documentatie die alle functies goed en duidelijk beschrijft en wordt door middel van GitHub goed up to date gehouden.

#### Concurrency mechanismen

Binnen ChibiOS worden eigenlijk alle concurrency mechanismen ondersteund die ook Arduino RTOS worden ondersteund. Soms worden alleen andere namen gebruikt, maar de functionaliteit is identiek. Hierdoor kunnen wij zonder veel mechanismen om te bouwen deze RTOS gebruiken.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Naam: | Flag | Timer | Clock | Mutex | Channel | Mailbox | Pool |
| ChibiOS/RT | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

### DistortOS

DisortOS is een object-oriented c++ RTOS voor microcontrollers geschreven in c++11. DisortOS is momenteel geschikt voor de Armv6-M en ARMv7-M architectuur, wat betekent dat deze RTOS erg veelzijdig is. Verder is deze RTOS geheel preemptive. De RTOS probeert de functionaliteiten zo efficiënt en universeel te implementeren. daarom probeert deze RTOS ook niet de snelste of kleinste RTOS te zijn, waardoor er eventuele overhead kan ontstaan. DisortOS richt zich ook op het implementeren van de beste functionaliteiten van andere open source RTOS en haalt alle de minder goede weg.

#### Concurrency mechanismen

DisortOS ondersteund alle concurrency mechanismen die ook door Arduino RTOS gebruikt worden. Echter is het voor gebruikers moeilijk dit uit de documentatie te halen, je moet erg lang zoeken tot je erachter komt dat DisortOS dezelfde mechanismen biedt wat natuurlijk voor de gebruiker erg onwenselijk is.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Naam: | Flag | Timer | Clock | Mutex | Channel | Mailbox | Pool |
| DisortOS | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |

### ECOS

ECOS is een opensource RTOS dat gebruikt wordt voor applicaties die heel diep in het embedded systeem zitten. Deze RTOS wordt gebruikt door hele grote internationale bedrijven zoals SONY, NETGEAR, etc. Dit RTOS gebruikt een HAL gebaseerd op C en heeft een C/C++ toolset. Het ondersteund bijna alle 32bits architecturen, wat er voor zorgt dat de flexibiliteit heel hoog ligt. Deze flexibiliteit is echter niet toepasselijk voor ARM based architecturen.

Het update beleid van deze RTOS is matig, omdat het een in ze zoveel maanden een andere platform ondersteund of een nieuwe versie komt uit. Verder is de documentatie van deze RTOS onvoldoende, omdat er een geen handleiding aanwezig is waar de functies goed uitgelegd zijn. Er is een enkele pagina die verteld dat de mechanismen bestaan.

#### Concurrency mechanismen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Naam: | Flag | Timer | Clock | Mutex | Channel | Mailbox | Pool |
| ECOS | ✔ | ✔ | ✔ | ✔ | ✘ | ✔ | ✘ |

Om de stabiliteit van deze RTOS te garanderen, worden niet alle concurrecy mechanismen ondersteund. Een Channel een Pool zijn hele intensieve mechanismen die moeilijk te implementeren zijn voor kleine microcontrollers. In principe kan je dit probleem oplossen door een array of een globale variabel te gebruiken, maar dit is niet aangeraden, omdat het voor on-stabiliteit zorgt.

### Mbed-rtos

Mbed-rtos is een Real TIme Operating System gebaseerd op de RTX-implementatie van de CMSIS-RTOS API. Deze RTOS is geschreven in C++ en kan gebruikt worden door alle Cortex-M varianten. Ook kun je SVD(XML) files gebruiken van chipmakers en deze gebruiken als een header-file. Dit is de meest uitbreidbare API in dit verslag omdat het door verschillende RTOS-vendors ondersteund wordt. Dit zorgt er echter wel voor dat het niet de meest efficiënte RTOS is, wel is het de meeste flexibele RTOS omdat je gegevens van meerdere chipmakers kan verwerken via de API.

Mbed-rtos maakt gebruik van priority based sheduling wat betekent dat een taak met met hoogste prioriteit eerst door de Processor wordt uitgevoerd.

Het update beleid van deze RTOS is zeer goed, ARM-Mbed heeft een actieve community met een Q&A forum om problemen en vragen te beantwoorden. Verder bezit ARM-Mbed over meer Hardware API’s en Libraries dan alleen Mbed-rtos.

#### Concurrency mechanismen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Naam: | Flag | Timer | Clock | Mutex | Channel | Mailbox | Pool |
| Mbed-rtos | ✔ | ✔ | ✘ | ✔ | ✔ | ✔ | ✔ |

Het enige nadeel van deze RTOS is dat het geen clock mechanisme bevat. In principe kan je dit heel simpel zelf realiseren door een timer een maal te setten een constant in een loop te wachten op de timer. Ook is het mogelijk om een methode toe te voegen aan de thread / taak die ervoor zorgt dat hij oneindig wacht.

### NuttX

NuttX is

#### Concurrency mechanismen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Naam: | Flag | Timer | Clock | Mutex | Channel | Mailbox | Pool |
| NuttX | ✘ | ✔ | ✔ | ✘ | ✘ | ✘ | ✔ |

## 

### StateOS

StateOS is een simpele RTOS waardoor het een snel systeem is. De documentatie is matig maar wel ondersteunt met voorbeelden. Verder wordt StateOS nog regelmatig bijgewerkt. Het concurrency mechanisme bevat alles wat verwacht wordt volgens onze criteria. Het is gebouwd uit het concept van de state machine.

# 

# 

# Conclusie en aanbeveling

# Literatuur

<https://en.wikipedia.org/wiki/ChibiOS/RT>

<http://www.chibios.org/dokuwiki/lib/exe/fetch.php?media=documents:manuals:17.6:chibios_17.6.2_chibios_rt_rm.pdf>

<http://distortos.org/>

<https://github.com/DISTORTEC>

<http://ecos.sourceware.org/>

http://atomthreads.com/

<https://github.com/stateos/StateOS>

<http://www.ecoscentric.com/news/latest_news.shtml>

<https://os.mbed.com/handbook/>

<https://os.mbed.com/forum/news-announcements/topic/3215/>

<https://www.arm.com/about/newsroom/arm-extends-cmsis-with-rtos-api-and-system-view-description.php>

https://os.mbed.com/questions/53993/Which-kind-of-scheduler-is-used-in-mbed-/

<https://www.osrtos.com/>

<https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems>

<http://www.embeddedcraft.org/listrtos.html>

http://nuttx.org/

<https://www.osrtos.com/rtos/stateos>

<https://github.com/stateos/StateOS>