

Onderzoeksverslag RTOS



Opdrachtgever: Hogeschool Utrecht

Begeleider: Joop Kaldeway

Klas: V2A

Team: MajorLazor + 2

Opgesteld door: Niels de Waal (1698041), Jasper Smienk (1700502),
Nick Bout (1709217), Eelke Feitz (1701685),
Wouter Dijkstra (1700101)

Datum: 25-10-2017

Versie: 0.1

Samenvatting

Het hele rapport is gebaseerd op de hoofdvraag en deelvragen:

Hoofdvraag:

1. Met behulp van welk opensource RTOS kunnen tasks & concurrency mechanismen (pool, channel, flag (group), clock timer & 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?
eisen:
 - a. Het RTOS moet beschikbaar zijn voor het ARM platform.
 - b. Het RTOS moet C en C++ ondersteunen.
 - c. Het RTOS moet binnen het afgelopen jaar een update hebben gekregen.
 - d. Het RTOS moet beschikken over documentatie waarin de geboden functionaliteit is beschreven.
3. 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?
4. Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?
5. 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?

Inhoudsopgave

Samenvatting	2
Inhoudsopgave	3
Inleiding	4
Deelvraag 1	5
Inleiding	5
Kern / Conclusie	5
Deelvraag 2	7
Inleiding	7
Kern	7
Conclusie	8
Deelvraag 3	9
Inleiding	9
Kern	9
Conclusie	9
Deelvraag 4	10
Inleiding	10
Kern	10
Conclusie	10
Deelvraag 5	11
Inleiding	11
Kern	11
Conclusie	11
Conclusie	12
Bronnenlijst	13

Inleiding

Als tweedejaars studenten van de opleiding Technische Informatica aan de Hogeschool Utrecht maken wij voor ons project Thema Devices een onderzoeksrapport om te kijken welke real time operating systems er beschikbaar zijn, welke functionaliteit deze bieden, en welke als vervanging zouden kunnen dienen voor het huRTOS.

Deelvraag 1

“Wat zijn de kenmerkende eigenschappen van tasks en de concurrency mechanismen van het Arduino RTOS?”

Inleiding

Deze deelvraag geeft ons informatie en inzicht over de functies die we hebben in het huRTOS. Door te weten wat we met het huidige huRTOS kunnen, weten we betere eisen stellen aan de andere huRTOS waar we ons programma naar willen overschrijven

Kern / Conclusie

Na het doorlezen van de huRTOS reader en de github source code blijkt dat het huRTOS beschikt over de volgende tasks & concurrency mechanismen.

RTOS Name	Flag	Timer	Clock	Mutex	Channel	Pool	Mailbox	Grouped waitables
huRTOS	✓	✓	✓	✓	✓	✓	✓	✓

Om beeld te krijgen wat deze mechanismen inhouden volgt voor ieder een korte uitleg.

Clock

Een clock is een timing element dat een puls afvuurt na een vaste tijdseenheid. Deze vaste tijdseenheid is gedefinieerd bij het aanmaken van de clock. Als deze eenmaal aangemaakt is zal deze door blijven lopen en pulsen afvuren om deze tijdseenheid tot het programma eindigt.

Timer

Een timer is net als een clock een timing element. Ook deze vuurt na een tijdseenheid een puls af, echter is deze een stuk flexibeler. Waar een clock oneindig door blijft lopen kan een timer stilgezet worden en opnieuw aangezet. Tijdens het aanzetten kan ook de tijdseenheid tussen de pulsen opnieuw ingesteld worden.

Flag

Een flag is een synchronisatie element wat deel is van een grotere task (task A). Deze flag is dus deel van deze task A, echter wordt hier niet door uitgevoerd. Een andere task (task B) kan de flag zetten. Task A kan wachten op de flag, wanneer deze geset wordt door task B kan task A weer verder met zijn taak en wordt de flag weer gereset.

Pool

Een pool een class template die gebruikt wordt als variabele dat gedeeld door verschillende tasks. Het doel hiervan is om voor de gebruiker deze deling tussen de verschillende tasks duidelijk te maken. Hiernaast wordt het makkelijker om de code om te zetten naar een ander platform waar er meer acties nodig zijn om data te delen tussen verschillende tasks.

Mutex

Een mutex, ofwel Mutual Exclusion Flag, is een flag die geclaimd kan worden door een task wanneer deze bij gedeelde data moet zijn. Wanneer de mutex over een deel data geclaimd is kunnen andere tasks niet bij deze data en wachten tot de mutex wordt vrijgegeven. Dit voorkomt dat er door meerdere tasks tegelijkertijd naar een gedeeld stuk data wordt gelezen en aangepast.

Channel

Een channel is een class template die wordt gebruikt als een array met data elementen. De task die eigenaar is van de channel kan hem uitlezen vanaf de voorkant. Andere tasks kunnen data schrijven naar het einde van de channel. De data die in de channel staat wordt dus uitgelezen via het FIFO systeem, First In First Out. Wanneer er data wordt uitgelezen wordt dit element ook verwijderd uit de channel, waardoor alles ook een positie hierin opschuift naar de voorkant. Wanneer er niets in de channel staat en de task toch deze probeert uit te lezen blokkeert deze task totdat er data aanwezig is. Wanneer een ander task data aan een volle channel probeert toe te voegen wordt dit genegeerd en gaat de data verloren.

Mailbox

Een mailbox is een class template en een synchronisatie mechanisme dat in principe werkt als een doorgeefluik. Een task schrijft data in deze mailboxs en wacht totdat de een andere taak, waarvoor deze data bedoeld is deze data uitleest. Net zoals bij een channel wordt de data in de mailbox vernietigd. Als de mailbox al data bevat worden andere tasks geblokkeerd om hier naar te kunnen schrijven.

Grouped waitables

Waitables zijn abstracte objecten, dus bestaan niet op zichzelf. In het huRTOS erven flag, timer, clock en channel van waitable. Deze mechanismen zijn dus waitable en kunnen door tasks op gewacht worden. De tasks in het huRTOS kunnen op één of meerdere, tot maximaal 31 waitables wachten.

Deelvraag 2

“Welke open source RTOS-en zijn er beschikbaar?”

Inleiding

Met deze deelvraag hopen we een beter beeld te krijgen van de verschillende beschikbare RTOS systemen en ook over de kwaliteiten, de mogelijkheden en de functies die deze bieden.

Voor het beantwoorden van deze deelvraag hebben we gebruik gemaakt van de onderzoeksmethode “Bieb” gekozen. Deze vorm van onderzoek paste het best in het gegeven tijdsbestek. Dit komt doordat deze werkvorm het snel en efficiënt resultaten levert. Tevens sluit deze werkvorm aan met de opdracht die we moeten verrichten, namelijk het bestuderen van al bestaande informatie.

Kern

Het RTOS moet voldoen aan de volgende eisen:

- Het RTOS moet beschikbaar zijn voor het ARM platform.
- Het RTOS moet C en C++ ondersteunen.
- Het RTOS moet binnen het afgelopen jaar een update hebben gekregen.
- Het RTOS moet beschikken over documentatie waarin de geboden functionaliteit is beschreven.

Hieronder een lijst van alle RTOS-en die mogelijk als vervanging kunnen dienen tot het huRTOS. **(bron 1)**

Name	Open Source	Links
BeRTOS	YES	https://en.wikipedia.org/wiki/BeRTOS https://github.com/develersrl/bertos
uKOS	YES	http://www.ukos.ch/
BRTOS	YES	https://code.google.com/archive/p/bRTOS/
distortos	YES	http://distortos.org/documentation/
Dnx RTOS	YES	http://www.dnx-rtos.org/
eChronos	YES	https://ts.data61.csiro.au/projects/TS/echronos/
F9 microkernel	YES	https://github.com/f9micro/f9-kernel
FreeRTOS	YES	https://en.wikipedia.org/wiki/FreeRTOS http://www.freertos.org/

silRTOS	YES	http://spanidea.com/sil.html
stratifyOS	YES	https://stratifylabs.co/Stratify-OS/

Conclusie

Voor ons onderzoek zijn veel verschillende RTOS-en die voldoen aan onze gestelde eisen. Sinds het tijdsbestek waarin wij moeten werken nogal krap is hebben wij ons onderzoek beperkt tot de bovenstaande tien real time operating systems.

Deelvraag 3

“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?”

Inleiding

Voor het beantwoorden van deze deelvraag hebben we gebruik gemaakt van de onderzoeksmethode “Bieb” gekozen. Deze vorm van onderzoek paste het best in het gegeven tijdsbestek. Dit komt doordat deze werkvorm het snel en efficiënt resultaten levert. Tevens sluit deze werkvorm aan met de opdracht die we moeten verrichten, namelijk het bestuderen van al bestaande informatie.

Kern

Om een beeld te krijgen welke RTOS-en dezelfde features bieden als het huRTOS, hebben we een tabel gemaakt waarin staat welke features deze real time operating systems bieden.

RTOS Name	Flag	Timer	Clock	Mutex	Channel	Pool	Mailbox	Grouped waitables
BeRTOS		✓		✓	✓	✓	✓	
uKOS		✓		✓	✓	✓	✓	
BRTOS		✓		✓	✓		✓	
distortos	✓	✓	✓	✓	✓			
Dnx RTOS	✓			✓	✓			
eChronos		✓		✓	✓			
F9 microkernel		✓			✓			
FreeRTOS	✓	✓		✓	✓		✓	✓
stratifyOS		✓		✓	✓			

Conclusie

Voor het bepalen welk real time operating system de meeste concurrency mechanismen zonder enige modificatie biedt is het een kwestie van vergelijken en afstrepen. Zo komen we tot de conclusie dat FreeRTOS de meeste concurrency mechanismen van het Arduino RTSO aan te bieden zonder enige modificatie.

Deelvraag 4

“Welke mechanismen van het huRTOS worden niet ondersteund door de beschikbare RTOS-en?”

Inleiding

Voor het beantwoorden van deze deelvraag hebben we gebruik gemaakt van de onderzoeksmethode “Bieb” gekozen. Deze vorm van onderzoek paste het best in het gegeven tijdsbestek. Dit komt doordat deze werkvorm het snel en efficiënt resultaten levert. Tevens sluit deze werkvorm aan met de opdracht die we moeten verrichten, namelijk het bestuderen van al bestaande informatie.

Kern

Voor het beantwoorden van deze deelvraag bouwen we voort op de conclusie van de vorige. Hier concludeerde dat FreeRTOS de meest geschikte optie zou zijn om de concurrency mechanisme zoals aangeboden door het huRTOS me zo weinig mogelijk overhead te realiseren.

Conclusie

Ookal is FreeRTOS het meest geschikte RTOS uit ons onderzoek gebleken, mist het toch nog een aantal concurrency mechanismen die het huRTOS wel heeft. Deze mechanismen zijn de clock en de pool.

Deelvraag 5

“Hoe kunnen de mechanismen van het huRTOS die niet direct worden ondersteund door de beschikbare RTOS-en worden gerealiseerd m.b.v. van deze RTOS-en?”

bij geen flag: gebruik mutex
geen channel: gebruik buffer

Inleiding

Voor het beantwoorden van deze deelvraag hebben we gebruik gemaakt van de onderzoeksmethode “Bieb” gekozen. Deze vorm van onderzoek paste het best in het gegeven tijdsbestek. Dit komt doordat deze werkvorm het snel en efficiënt resultaten levert. Tevens sluit deze werkvorm aan met de opdracht die we moeten verrichten, namelijk het bestuderen van al bestaande informatie.

Kern

Pool

Een pool zou je kunnen maken door gebruik te maken van een globale variabelen.

Clock

Een clock zou je kunnen maken door gebruik te maken van het bestaande System Tick dat beschikbaar is op ARM processoren.

Conclusie

Het blijkt dus dat we de concurrency mechanismen die het RTOS niet direct ondersteunt kunnen nabootsen met al bestaande mechanismen.

Conclusie

Met behulp van FreeRTOS zouden we de tasks & concurrency mechanismen aangeboden door het Arduino RTOS, met zo weinig mogelijk overhead kunnen realiseren. Dit komt doordat FreeRTOS van alle onderzochte RTOS-en de meest overeenkomende concurrency mechanismen heeft. Hierdoor hoeft er het minst overhead te komen bij het improviseren van de missende concurrency mechanismen.

Bronnenlijst

Bronnen:

1. https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems
2. <https://github.com/wovo/rtos>
3. <https://cursussen.sharepoint.hu.nl/fnt/35/TCTI-V2CPSE1-16/Studiemateriaal/2017-2018-V2CPSE1-reader-rtos.docx>
4. <https://sites.google.com/site/rtosmifmim/home/message-queues-mailboxes-and-pipes>
5. <https://e2e.ti.com/support/embedded/tirtos/f/355/t/351311>
6. <http://www.learnitmakeit.com/freertos-tick/>