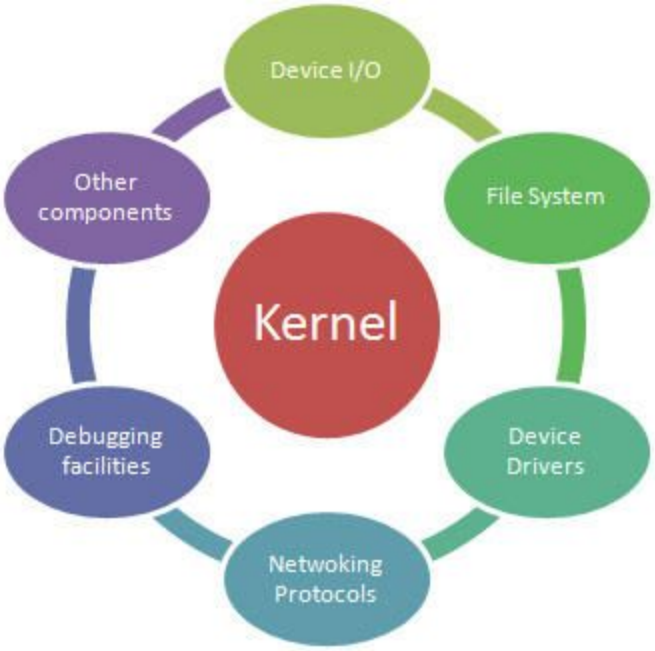
RTOS ONDERZOEK

implementatie concurrency mechanismen door open source RTOS



B1 BAZEN

Team:

Bob Thomas
Robbie Valkenburg
René de Kluis

Begeleider:

Jan Zuurbier

INLEIDING

Tijdens een project op school moesten we een laser tag systeem schrijven met gebruik van het Arduino RTOS die beschikt over tasks en de concurrency mechanismen zoals pool, channel, flag(group), clock timer en mutex. Het doel van dit onderzoeksverslag is uit te vinden of er meerdere open source RTOS systemen bestaan die over dezelfde functionaliteit beschikken als de Arduino RTOS. Om deze vraag te kunnen beantwoorden hebben we een hoofdvraag opgesteld met daarbij een aantal deelvragen om via de onderzoeksresultaten te kunnen aantonen of er meerdere RTOS systemen bestaan zoals de Arduino RTOS. Om hierdoor erachter te komen welk RTOS systeem de beste vervanger kan zijn.

Hoofdvraag

Met behulp van welk ‘open source real time operating system’ (RTOS) 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. 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 worden gerealiseerd die niet direct worden ondersteund door de beschikbare RTOS-en m.b.v. van deze RTOS-en?

GEBRUIKTE ONDERZOEKSMETHODEN

De methoden die zijn gevolgd om elk van de deelvragen te beantwoorden.

Deelvraag 1

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

Om deelvraag 1 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we hier literatuurstudie kunnen toepassen door gebruik te maken van de documentatie van het Arduino RTOS.

Deelvraag 2

Welke open source RTOS-en zijn er beschikbaar?

Om deelvraag 2 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” methode omdat we verschillende RTOS systemen moeten vergelijken en we alle informatie voornamelijk op het internet kunnen vinden.

Hoe we deelvraag 2 aanpakken is door een lijstje op te stellen van de minimale eisen voor het RTOS systeem.

- 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.

Met dit lijstje gaan we het internet op en zullen wij een inventarisatie opstellen. Hiervan zullen we een top 3 maken en deze vergelijken tegenover het Arduino RTOS.
Hierbij maken we gebruik van de zoekmachine google met de zoekterm “opensource arm rtos c++”

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?
(OSRTOS Open source real-time operating systems)

Om deelvraag 3 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we het Arduino RTOS gaan vergelijken tegenover de beschikbare rtos systemen.

Deelvraag 4

Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?

Om deelvraag 4 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we het Arduino RTOS gaan vergelijken tegenover de beschikbare rtos systemen.

Deelvraag 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?
(OSRTOS Open source real-time operating systems)

Om deelvraag 5 te kunnen beantwoorden kunnen we de methode “werkplaats” gebruiken door bijv vertical prototyping te gebruiken kunnen we besluiten of het mogelijk is om de niet ondersteunde mechanismen er in te bouwen.

DEELVRAAG 1

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

Om deelvraag 1 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we hier literatuurstudie kunnen toepassen door gebruik te maken van de documentatie van het Arduino RTOS.

Het Library aspect

Voor het gebruiken van de Arduino RTOS is in principe slechts één library nodig. De library is **91.5 KB** en de header file van de library zal eenmalig in de main moeten ge-include worden. Om een class de mogelijkheden van het RTOS systeem te bieden, dient de class wel een interface te hebben van de RTOS library. De library stelt verder geen eisen aan de programma architectuur (geen verdere frameworks nodig).

- 91.5 KB library
- Een interface voor de klasse die Arduino RTOS wilt gebruiken.

Eigenschappen van de Arduino RTOS task

Een eigenschap van de RTOS task is dat het eenvoudig te definiëren is bij een class door slechts een void functie genaamd ‘main’ toe te voegen. Bij het compileren van de code zal de Arduino RTOS deze functie gaan herkennen als een task. Bij het gebruiken van de ‘wait’ functie uit de Arduino RTOS library, zal tijdens het wachten gekeken worden of een andere task actief is en deze verder mag gaan. Ook is het mogelijk om prioriteiten te geven aan de RTOS task. De RTOS task kan gebruik maken van alle concurrency mechanismen die geboden worden in de Arduino RTOS. Op de volgende pagina is een samenvatting te vinden van de concurrency mechanismen met daarbij de eigenschappen.

RTOS Waitables

(max 32 waitables per task, 1 meestal gebruikt voor de wait)

Je kan via een wait() wachten op een waitable als de waitable als de waitable waar op de wait() geset word gaat de task weer verder en word die unblocked

RTOS Flag

Een eigenschap van de Arduino RTOS flag is dat het een functie bevat genaamd set. De functie biedt de mogelijkheid om dit voor andere classen ook te zien. Belangrijke eigenschap is dat de flag niet meer '*enabled*' is bij de volgende check.

RTOS Pool

In de Arduino RTOS wordt bij de pool niet rekening gehouden met enige vorm van synchronisatie. Er kan slechts één data type worden opgeslagen per pool. Wel is het mogelijk om elke beschikbare data type mee te geven. De pool kan een write en een read operation uitvoeren op het data type.

RTOS Channel

De channel in Arduino RTOS kan waardes zetten in een 'rij', nieuwe waardes worden achteraan geplaatst en kan alleen van vooraan uitgelezen door de taak die de eigenaar is. Zolang er geen waarden in de channel zijn kan deze niet 'gelezen' worden. De waardes worden pas gecleared als de channel volledig gelezen is. De 'read' zal de channel blocken tot er ruimte over is en deze dan weer unblocken.

RTOS Mutex

De Mutex van het Arduino RTOS wordt gebruikt als semafoor. Zo kan er voorkomen worden dat het systeem over data gaat schrijven of lezen wat momenteel bewerkt wordt door een andere taak.

Deze mutex kent twee belangrijke operaties als eigenschappen genaamd 'wait' en 'signal'. Bij het aanroepen van de wait operatie, wordt er aangegeven dat andere taken die dezelfde mutex delen moeten wachten tot de operatie signal weer is vrijgegeven.

RTOS Mailbox

Een task kan een enkele waarde naar een mailbox worden geschreven en een andere task deze uitlezen. De ‘write’ en ‘read’ calls kunnen elkaar een seintje geven en op elkaar wachten voordat ze hun gang mogen gaan. De mailbox is niet gemaakt voor een bepaalde task en is geen waitable.

RTOS Timer

Een waitable die je kan starten door de set (unsigned long int time) methode te gebruiken. Als deze methode word aangeroepen gaat de timer automatisch lopen en kan je er op wachten via een wait() met als parameter de timer. Ook tijdens een suspend in de task zelf zal de timer door lopen en zich zichzelf setten.

RTOS Clock

De RTOS Clock is een waitable die tijdens de initialisatie een interval mee krijgt.
De clock “set” zichzelf als het interval bereikt is en zal daarna weer opnieuw beginnen met lopen. De clock blijft lopen tijdens een suspend net zoals de timer.

Deelvraag 2

Welke open source RTOS-en zijn er beschikbaar?

Om deelvraag 2 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” methode omdat we verschillende RTOS systemen moeten vergelijken en we alle informatie voornamelijk op het internet kunnen vinden.

Beschikbare RTOS systemen

Bij het zoeken naar beschikbare RTOS systemen (OSRTOS Open source real-time operating systems) is voldaan aan de eisen die vastgelegd waren aan de onderzoeksmethoden. Doormiddel van deze werkwijze zijn we op de volgende resultaten gekomen:

FreeRTOS

Developed in partnership with the world's leading chip companies over a 12 year period, FreeRTOS is the market leading real time operating system (or RTOS), and the de-facto standard solution for microcontrollers and small microprocessors.

Real Time Engineers ltd.. (2014).

Chibios

ChibiOS is a complete development environment for embedded applications including RTOS, an HAL, peripheral drivers, support files and tools.

Di Sirio, G. D. S. (2016, 2 juli).

NuttX

NuttX is a real-time operating system (RTOS) with an emphasis on standards compliance and small footprint. Scalable from 8-bit to 32-bit microcontroller environments, the primary governing standards in NuttX are Posix and ANSI standards.

Nutt, G. N. (2015, 24 juli)

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?

Om deelvraag 3 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we het Arduino RTOS gaan vergelijken tegenover de beschikbare rtos systemen

Voor dit onderzoek is een tabel gemaakt om een duidelijk overzicht te krijgen over de concurrency mechanismen die worden ondersteund door de gevonden beschikbare RTOS-en.

Beschikbare RTOS	flag	pool	channel	mutex	mailbox	Clock, timers	preem ptive
FreeRTOS	✓	?	✓	✓	✓	✓	✓
ChibiOS	✓	✓	✓	✓	✓	✓	✓
NuttX	✓	✓	✓	✓	?	✓	✓

Bij het uitlezen van de tabel blijkt dat *ChibiOS* alle genoemde concurrency mechanismen bevat die in het Arduino RTOS ook aangeboden worden.

Deelvraag 4

Welke mechanismen van het Arduino RTOS worden niet ondersteund door de beschikbare RTOS-en?

Om deelvraag 4 te kunnen beantwoorden gebruiken wij de “bieb” uit de verschillende onderzoeksmethodes. We hebben gekozen voor de “bieb” omdat we het Arduino RTOS gaan vergelijken tegenover de beschikbare rtos systemen.

Uit nader onderzoek is gebleken dat de beschikbare RTOS-en die gevonden zijn over het algemeen de meeste mechanismen wel bevatten.

NuttX

Bij NuttX was niet zo gauw in de documentatie of beschrijving een uitleg te vinden over het mailbox mechanismen. Het kan zijn dat NuttX het wellicht wel biedt, maar een andere naamgeving hier verzonnen voor heeft.

FreeRTOS

De originele versie van FreeRTOS gebruikte wel pools. Omdat er verschillende meningen waren over het gebruik van pools (geheugengebruik), zijn pools niet meer terug te vinden in de huidige versie. Wel is het mogelijk om zelf een pool te schrijven voor freeRTOS.

Chibios

Chibios biedt gegarandeerd wel alle mogelijkheden voor het gebruiken van de concurrency mechanismen die Arduino RTOS ook biedt. Dit komt vanwege de duidelijke documentatie waarin dit te vinden is.

Deelvraag 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?

Om deelvraag 5 te kunnen beantwoorden kunnen we de methode “werkplaats” gebruiken door bijv vertical prototyping te gebruiken kunnen we besluiten of het mogelijk is om de niet ondersteunde mechanismen er in te bouwen.

Mailbox NuttX rtos

Zover het onderzoek is er gebleken dat NuttX geen concurrency mechanismen had van de mailbox. Wel was er ondersteuning voor de andere concurrency mechanismen.
Doormiddel van het uitzoeken hoe een task word gemaakt in NuttX en een klasse maken die bij een writer en een reader koppelt aan tasks.
Het zou een template Klasse met als parameter voor het type dat die opslaat.
De mailbox classe krijg 2 functies read en write.
Read zou er ongeveer zo uitzien

```
void write( const T item ) {
   data = item;
   if ( reader != nullptr ) {
      task * tmp = reader;
      reader = nullptr;
      sched_unlock(reader);
   }
   else {
      if (writer != nullptr)
         //thorw an error
         throw("Mailbox error");
      writer = this_task();
      sched_lock(writer);

   }
}
T read (void) {
   if ( writer != nullptr ) {
      task * tmp = writer;
      writer = nullptr;
      sched_unlock(tmp);
   }
   else {
      if (reader != nullptr)
         //throw error
         throw("Mailbox error");
      reader = this_task();
      sched_lock(reader);
   }
   return data;
}
```

Hier mee zou dus gemakkelijk een mailbox systeem in Nuttx kunnen worden gemaakt.
Die de zelfde eisen en over de zelfde mechanismen beschikt als de Arduino RTOS.

Pool free-RTOS

Richard. (2016, 1 juni) Free-RTOS biedt zover bekend, niet meer de pool als concurrency mechanismen in zijn huidige versie bij de source files. Echter bestaat er nog wel een api voor. De code van deze api voor memory allocation staat opgeslagen onder de map “portable layer”.

Resultaten

Door het beantwoorden van alle deelvragen, is uit het onderzoek gebleken dat er meerdere open source rtos systemen beschikbaar zijn die voldoen aan de gewenste zoekcriteria. In deelvraag 2 is grondig onderzoek gedaan naar de ondersteuning van alle concurrency mechanismen die gewenste zijn bij het project. In de tabel is nogmaals een duidelijk overzicht van de geboden concurrency mechanismen per rtos.

	flag	pool	channel	mutex	mailbox	Clock, timers	preem ptive
FreeRTOS		?					
ChibiOS							
NuttX					?		

Aangezien er wel oplossingen zijn voor het gebrek aan de concurrency mechanismen bij Free-RTOS en NuttX, zal je als resultaat krijgen dat alle 3 de rtos-en mogelijk zijn voor gebruik. Echter neemt het niet weg dat de oplossingen enig sinds technische kennis vereisen. Als resultaat kunnen we concluderen dat ChibiOS gelijk de mogelijkheid biedt op het gewenste resultaat in dit onderzoek.

CONCLUSIES EN AANBEVELINGEN

Dus ons antwoord op de hoofdvraag luid.

Met behulp van **ChibiOS** 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?

Wij zijn op dit antwoord gekomen door de deelvragen te beantwoorden en een conclusie te trekken.

Voor deelvraag 1 hebben we uitgezocht hoe de Arduino RTOS in elkaar zit en over welke mechanismen het systeem beschikt.
Met deze antwoorden konden we deelvraag 2 makkelijker beantwoorden door gericht te zoeken op systemen die bijna identiek zijn aan de Arduino RTOS.

Hierbij hebben wij een top 3 kunnen maken van de 3 verschillende RTOS systemen die het meest leken op het Arduino RTOS. Met deze 3 RTOS systestemen konden wij makkelijk de systemen vergelijken met het Arduino RTOS systeem en daaruit een tabel maken met de overeenkomsten. Met deze tabel kunnen wij makkelijk de resultaten vergelijken.

Van uit de top 3 hebben we deelvraag 4 ook kunnen beantwoorden door uit te leggen welke mechanismen ontbreken van de RTOS systemen.

Voor deelvraag 5 hebben we de RTOS systemen die niet over alle mechanismen beschikken van de Arduino RTOS bestudeerd. Hierna hebben wij de oplossingen gezocht hoe je deze zou kunnen implementeren in de verschillende RTOS systemen om zo toch aan de eisen te voldoen vergeleken met Arduino RTOS.

Aanbeveling

Als aanbeveling raden wij aan om ChibiOs naast Arduino RTOS te draaien en testen uit te voeren. Met de data van dit onderzoek zouden we de resultaten kunnen vergelijken en hieruit beslissen welk RTOS systeem beter is. En als vervolgonderzoek zouden meerdere RTOS systemen kunnen toegevoegd aan de onderzoekslijst.

APA bronvermelding

1. Di Sirio, G. D. S. (2016, 2 juli). ChibiOS/RT. Geraadpleegd op 30 oktober, 2016, van http://chibios.sourceforge.net/docs3/rt/modules.html
2. Hahm, O. H., Baccelli, E. B., Petersen, H. P., & Tsiftes, N. T. (2015). Operating Systems for Low-End Devices in the Internet of Things: a Survey. Geraadpleegd van https://hal.inria.fr/hal-01245551/file/IoT-OS-survey.pdf
3. Nutt, G. N. (2015, 24 juli). NuttX Operating System User's Manual. Geraadpleegd op 30 oktober, 2016, van http://NuttX.org/doku.php?id=documentation:userguide
4. Ooijen, W. O. van. (z.j.). Bare Metal Programming Tool Kit. Geraadpleegd op 31 oktober, 2016, van http://www.voti.nl/bmptk/docs/namespacebmptk_1_1rtos.html
5. OSRTOS Open source real-time operating systems [Dataset]. (z.j.). Geraadpleegd van http://www.osrtos.com/
6. Real Time Engineers ltd.. (2014). The FreeRTOS™ Reference Manual. Geraadpleegd van http://www.freertos.org/Documentation/FreeRTOS-Reference-Manual-Table-of-Contents.pdf
7. Real Time Engineers Ltd.. (2014). Memory Management. Geraadpleegd van http://www.freertos.org/a00111.html
8. Richard. (2016, 1 juni). Memory Pool in FreeRTOS like in uC/OS II [Forumpost]. Geraadpleegd op 30 oktober, 2016, van http://stackoverflow.com/a/37562638/3094496