**Free RTOS**

Free RTOS is een realtime operating system die al 12 jaar lang gedeveloped wordt door het bedrijf Nabto. Dit leek ons een geschikt RTOS om te onderzoeken aangezien het nog onderhouden wordt en het vele functies bevat. De functies die we willen hebben moeten vergelijkbaar zijn met die van de hwlib RTOS.

Als we kijken naar de pool in de hwlib RTOS is deze niet aanwezig in deze RTOS. Er is geen specifieke functie voor aanwezig. In principe is een pool een const pointer naar data die je steeds weer kan wijzigen. Of ik nou één integer naar de pool stuur of een hele array met integers, ze worden allebei opgeslagen, maar als er weer nieuwe data gestuurd wordt dan wordt de oude data vergeten. Dit is dus prima te implementeren in het Free RTOS systeem, ondanks dat het niet aanwezig is.

Een channel lijkt heel erg op de pool, alleen is dit een array van const pointers waar ook nog eens een mutex op staat. Een array van const pointers bestaat in c++ en mutexen worden ondersteund door het Free RTOS, dus is dit ook niet al te moeilijk om te implementeren.

Flags worden ondersteund door Free RTOS, maar worden event bits of event groups genoemd. Een event bit kan 1 worden gezet wanneer er data klaar is en er dus een flag gezet moet worden. In event groups zitten meerdere bits, wat dus inhoud dat een task op meerdere flags tegelijkertijd kan wachten. Dit zou in ons geval van pas kunnen komen voor bijvoorbeeld de schietknop, die het signaal moet versturen en dan het geluidje moet afspelen voordat hij weer een nieuw schot kan versturen.

De timer functie is aanwezig, maar dit is dan ook het enige wat tijd gerelateerde functies aanwezig is. De timer functie in Free RTOS bevat periodieke taken, taken waarvan de timer gereset wordt zodat hij na een periodieke interval nog een keer wordt uitgevoerd, en taken die maar een enkele keer worden uitgevoerd na een bepaalde tijd. Andere timers zoals de clock en de sleep zijn er niet en zijn volgens hun website lastig te implementeren.

**Zephyr**

Zephyr is een open source RTOS systeem, die erg lichtgewicht is en continue ontwikkeld wordt door de community om zo ondersteuning te bieden voor verschillende hardware en sensoren. Zephyr heeft een apache 2.0 licentie en wordt uitgegeven door de linux foundation.

Ook in Zephyr zoeken we naar een pool. Zephyr ondersteund een soortgelijke functie met ook een soortgelijke naam, namelijk een memory pool. In deze pool kan je memory blocks aanmaken die variabel in grote zijn. Hierdoor zal deze pool minder ruimte innemen dan een andere functie in Zephyr die ook voor data sharing is genaamd memory maps. Echter wordt deze data niet automatisch overschreven als er naar deze pool geschreven wordt, maar net zoals bij een channel het in een ander memory block opgeslagen. Eigenlijk hebben we hier dus meer te maken met een channel dan een pool. Als dan toch echt beter een pool gebruikt kan worden kan deze worden aangemaakt door een variabele pointer te maken en die aan te roepen en te herschrijven als er nieuwe data naar geschreven wordt.

Zephyr heeft ook mutexen die of voor een bepaalde tijd, of totdat ze worden vrijgegeven een datablock bezetten. Ook kan er een mutex gebruikt worden zonder dat deze gelockt wordt. Zo kan je dus een taak laten runnen, totdat ergens anders deze nodig is en deze dan daar gelockt wordt. Dit is een betere functionaliteit dan in de hwlib RTOS, dus gunstig.

In de synchronisation services van Zephyr zitten de flags, semaphores en mutexen. De flags worden ook hier events genoemd. Events kunnen geset worden of gecleared, dus in essentie true en false gezet worden. Ook kan in Zephyr een event handler functie gebruikt worden. Deze functie wordt aangeroepen op het moment dan de event is gebeurd.

De timer functie kan ook hier alleen ingezet worden als clock, wat dus inhoud dat er tegelijkertijd nog wel code uitgevoerd wordt en een event triggert als de timer is afgelopen.