RISC-V IoT-ontwikkeling in AWS

met FreeRTOS-bibliotheken

Avra Saslow (USA)

De open source-filosofie heeft de afgelopen decennia bewezen enorm belangrijk te zijn in de software- en wetenschappelijke community. Het idee is dat iedereen kan bijdragen aan het maken van een beter project of product, waardoor dat uiteindelijk toegankelijker en betrouwbaarder wordt. Open source-technologie moedigt experts aan om samen aan één project te werken, wat kwaliteit en veiligheid garandeert en uiteindelijk een mooi product oplevert zonder copyright-beperkingen. Dit artikel is een gids voor het implementeren van deze concepten in een echte real time-applicatie.

Achtergrond van RISC-V en FreeRTOS

Na jaren van succes in de software-industrie, heeft nu ook de hardware-industrie eindelijk toegang tot een open source-technologie (RISC-V) die de microcontroller-toekomst volledig zou kunnen veranderen. Traditioneel beschrijft een ISA (*Instruction Set Architecture*) hoe software en hardware op een board met elkaar communiceren, en gewoonlijk is zo'n ISA vastgelegd met licenties, royalty's en NDA's (*Non-Disclosure Agreements*).

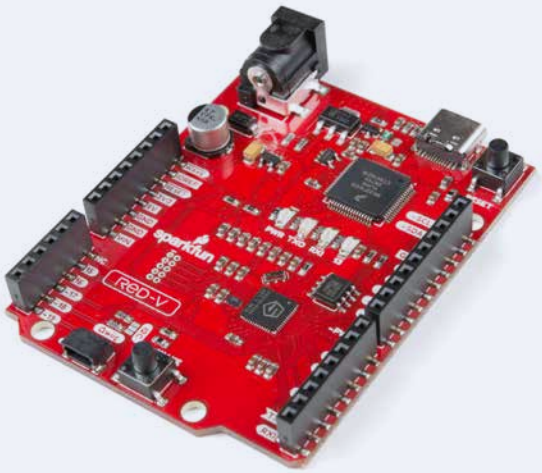
Daarentegen wordt de RISC-V ISA aangeboden onder open-source licenties. Daarom lijkt de RISC-V-architectuur klaar te zijn om het bedrijfsmodel voor technologie volledig te veranderen. Een bedrijf hoeft dus geen ISA te kiezen en zich daarmee vast te pinnen op de bibliotheek van die leverancier; RISC-V stelt bedrijven in staat de processorkern aan te passen, te vereenvoudigen en op te schalen om in hun specifieke behoeften te voorzien.

Het hier beschreven project maakt gebruik van de open source-hardware door FreeRTOS te implementeren, een real-time besturingssysteem voor embedded apparaten. FreeRTOS maakt het mogelijk om flexibele multithreading-oplossingen te implementeren. In plaats van losse stand alone-bibliotheken te gebruiken, maakt FreeRTOS het mogelijk om code te onderhouden en in de loop van de tijd te vertalen naar verschillende applicaties.

U zou dit als volgt kunnen zien: bij gebruik van een gewone microcontroller, zoals een Arduino, bestaat de traditionele programmastructuur uit een oneindige lus, waarin alle taken in het systeem worden uitgevoerd. Wanneer het programma start, voert het een setup-functie uit en daarna voert het programma een vaste reeks taken uit in die oneindige lus totdat er een interrupt komt.

Die taken kunnen sensoren inlezen, naar displays schrijven of getallen kraken; maar wat de taken ook doen, ze worden sequentieel uitgevoerd. Een voordeel van die programmastructuur is dat er weinig energie bij verbruikt wordt.

Figuur 1. Single-threading versus multithreading.



Figuur 2. Het SparkFun RED-V-board is gebaseerd op een krachtige RISC-V-processor.

Maar stel nou dat u verschillende taken tegelijkertijd moet uitvoeren. De hierboven geschetste structuur kan snel draaien, maar de taken draaien niet parallel. Daar is een real time-besturingssysteem zoals FreeRTOS op zijn plaats, omdat het een minimale interrupt- en thread switching-latentie heeft.

Minimale interruptlatentie betekent dat het besturingssysteem (OS) is geoptimaliseerd voor een zo kort mogelijke tijd tussen het uitvoeren van taken of subroutines. Minimale thread switching-latentie betekent dat er zo weinig mogelijk tijd verloren gaat als het besturingssysteem de CPU (één enkele CPU) moet omschakelen van de ene taak naar de andere. Deze minimale latenties zijn in feite wat een RTOS tot een multitasking-besturingssysteem maakt, zodat het taken gelijktijdig in plaats van sequentieel kan uitvoeren, zoals afgebeeld in **figuur 1**.

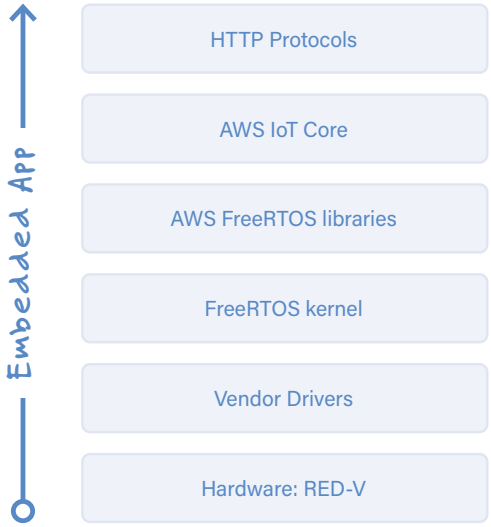
Misschien hebt u geen RTOS nodig voor programma's die sensoren uitlezen, berekeningen op die meetwaarden uitvoeren en die naar een LCD sturen, maar multithreading kan uiterst nuttig worden bij toepassingen zoals interactie met de draadloze stacks die op heel korte termijn een reactie op gebeurtenissen vereisen. FreeRTOS is specifiek gebouwd voor embedded apparaten, dus het is perfect geschikt om zijn OS-kernel op een RISC-V-board te laden en te communiceren via draadloze bibliotheken in AWS (*Amazon Web Services*™).

Opzetten van de RED-V-microcontroller met FreeRTOS-kernel, FreeRTOS-bibliotheken, en AWS-cloud

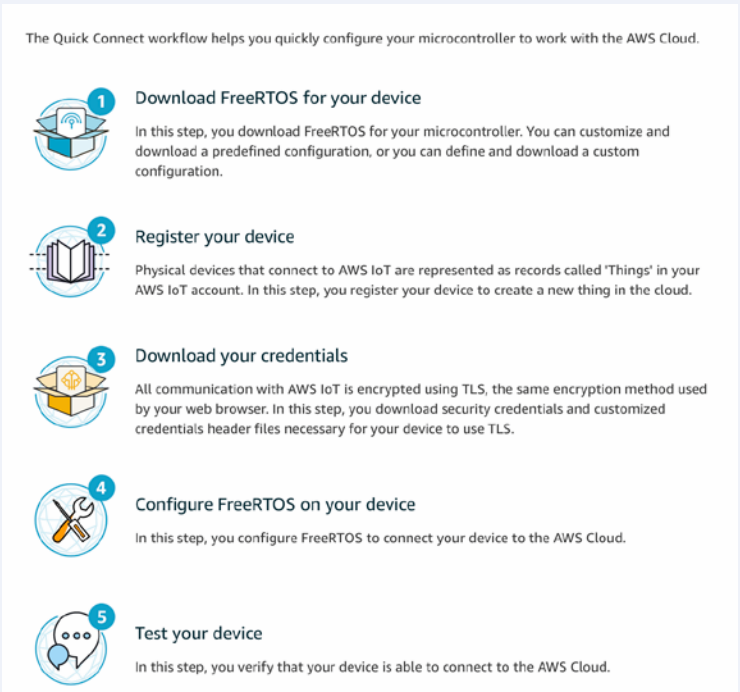
Om in deze moderne tijd wat voor nuttige IoT-toepassing dan ook te bouwen, is het noodzakelijk om een basistemplate te maken dat een IoT-apparaat koppelt aan een veilig clouddienst-platform. Dit project breidt de FreeRTOS-kernel uit en maakt gebruik van applicatieprotocol-bibliotheken die connectiviteit bieden voor het bouwen van IoT-toepassingen vanuit rond een microcontroller opgebouwde apparaten, zoals het SparkFun RED-V-board (**figuur 2**) dat is gebouwd met RISC-V. De structuur waaraan we ons moeten houden voor onze applicatie is te zien in **figuur 3**.

Gelukkig kan AWS eenvoudig met FreeRTOS worden geïmplementeerd! Om een HTTP-serververbinding via AWS IoT te configureren, moet u een gebruiker aanmaken in de sectie "Identity and Access Management (IAM)" van AWS. Deze gebruiker zal rechten hebben om toegang te krijgen tot zowel AmazonFreeRTOSFullAccess als AWSIoTFullAccess (beide zijn AWS-policies). Het RED-V-board moet ook geregistreerd zijn bij AWS IoT, wat inhoudt dat u moet zorgen voor:

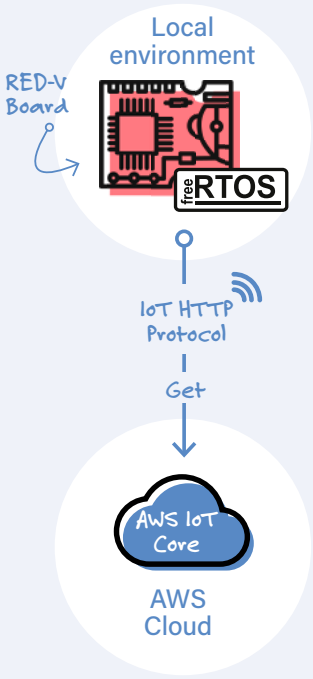
- een AWS IoT-policy, die uw apparaat rechten geeft om toegang te krijgen tot AWS IoT-resources;
- een AWS IoT-thing, waarmee u uw apparaten kunt beheren in AWS IoT;
- een private sleutel en certificaat waarmee uw apparaat zich kan authentiseren bij AWS IoT.



Figuur 3. Lagen-structuur van een embedded toepassing.



Figuur 4. De Quick Connect-workflow (bron: AWS).



Figuur 5. Schematische voorstelling van het proces dat onze IoT-on-AWS toepassing doorloopt.

Ik weet dat dat allemaal behoorlijk lastig klinkt, maar het is snel te realiseren door de Quick Connect-workflow (**figuur 4**) te volgen in de FreeRTOS-console. Als u dat wilt, kunt u alle stappen ook handmatig uitvoeren via de opdrachtregel. Wat daarna nog nodig is om het RED-V-board te configureren als een IoT-apparaat in AWS is het downloaden van FreeRTOS als besturingssysteem voor de RISC-V-architectuur.

RED-V als IoT-apparaat verbinden met de AWS IoT-server

Nu de setup geregeld is, wordt het pas echt interessant... we draaien een http-demo op het RED-V-board dat via AWS IoT een enkele applicatietaak creëert.

HTTP staat voor *Hypertext Transfer Protocol*, en HTTPS is versleuteld met *Transport Layer Security* of TLS. HTTP en HTTPS zijn protocollen die worden gebruikt om gegevens over te dragen via het web en zij maken gebruik van specifieke request-methoden om verschillende taken uit te voeren. Twee algemeen bekende methoden zijn GET, waarmee gegevens van een gespecificeerde bron worden opgevraagd, en POST, waarmee gegevens naar een server worden gezonden om een resource aan te maken of bij te werken. Dat zijn de basisbouwstenen achter elke IoT-toepassing, en deze demo laat zien dat dit allemaal via AWS kan worden gerealiseerd met behulp van open source-hardware en -software.

Zoals te zien is in **figuur 5**, maakt de demo verbinding met de AWS IoT HTTP-server, maakt een HTTP-request aan en verstuurt die, ontvangt een HTTP-response en verbreekt dan de verbinding met de server. Dankzij de FreeRTOS-documentatie kunnen we de structuur van de applicatie volgen via de – vrij complexe – AWS-code [1].

Door de FreeRTOS HTTP Client-bibliotheek te gebruiken, is de RED-V nu een IoT-apparaat dat HTTP-requests kan verzenden en HTTP-responses kan ontvangen met de meeste HTTP-servers (niet alleen AWS-servers). Daardoor is het gemakkelijk te vertalen naar andere dynamische toepassingen, omdat het beschikt over:

- zowel volledig asynchrone als synchrone API-functies;
- door de applicatie beheerd geheugen voor interne context en HTTP-headers;
- thread-aware en parallelle verbindingen.

Onbegrensde mogelijkheden

Het is eenvoudig om een IoT-toepassing te bouwen op de RISC-V-architectuur in combinatie met de FreeRTOS-kernel en -bibliotheken. Daarmee kunt u embedded apparaten aanpassen en opschalen voor elke toepassing. Het opzetten van HTTP-requests met de cliëntbibliotheek is het topje van de ijsberg bij het maken van IoT-toepassingen. De mogelijkheden met deze open source-technologieën zijn onbegrensd!

200699-04

Elektor online-pagina voor dit artikel
www.elektormagazine.nl/esfe-en-redv

Gerelateerde Producten

Bent u op zoek naar de belangrijkste artikelen die in dit artikel genoemd zijn? Elektor en SparkFun staan voor u klaar!

- **SparkFun RED-V RedBoard - SiFive RISC-V FE310 SoC**
  www.elektormagazine.nl/esfe-en-redv1

- **SparkFun RED-V Thing Plus - SiFive RISC-V FE310 SoC**
  www.elektormagazine.nl/esfe-en-redv2

WEBLINKS

[1] AWS-code: https://www.freertos.org/http/http-demo-with-tls-mutual-authentication.html