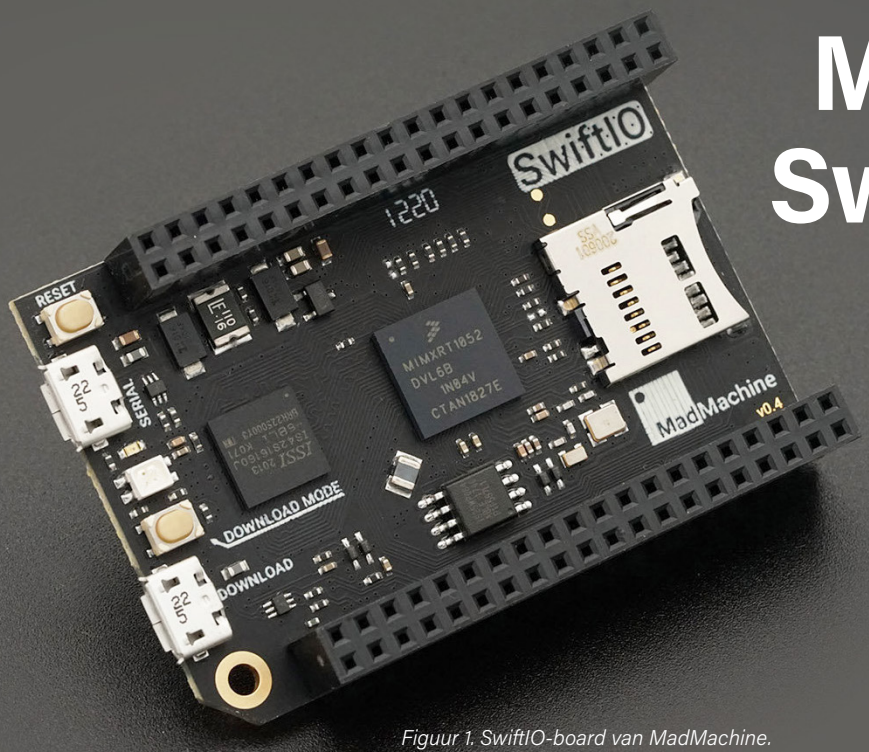
MadMachine SwiftIO Board

Moderne taal ontmoet moderne Hardware



Figuur 1. SwiftIO-board van MadMachine.

Mathias Claußen (Elektor)

De Swift programmeertaal - bekend van apparaten van Apple - kan nu ook gebruikt worden op een MCU. We zullen een blik werpen op het SwiftIO bord, dat een ARM Cortex-M7 Processor en 32 MB RAM integreert.

Het SwiftIO ontwikkelboard van MadMachine heeft een BeagleBone Black-vormfactor. Wat mij aanspreekt van het board, is het feit dat we een interessante hardware set hebben in combinatie met een ongebruikelijk software framework. Code kan worden geschreven in Swift, een programmeertaal die populair is op Apple-apparaten.

De hardware

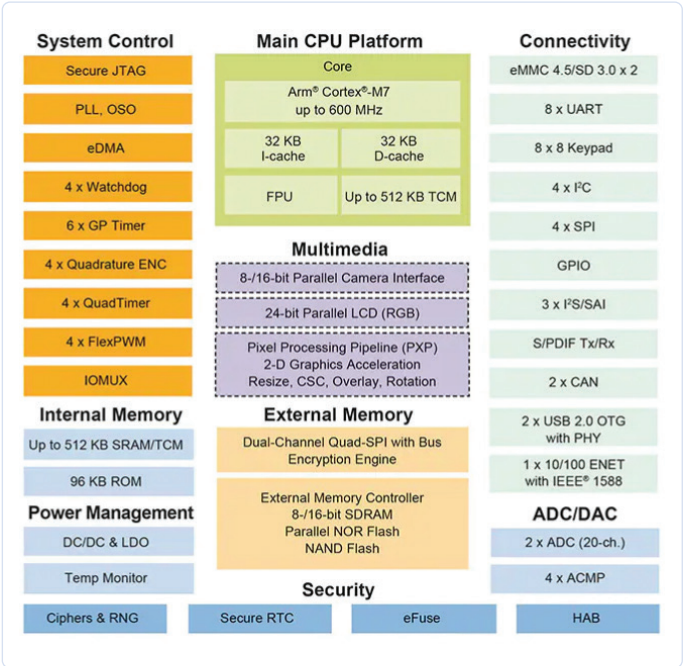
Terwijl Swift zich richt op de Apple-hardware, zal het SwiftIO-board (**figuur 1**) van MadMachine de Swift-taal naar de embedded wereld brengen.

De kerncomponenten die op het board worden gebruikt, zijn:

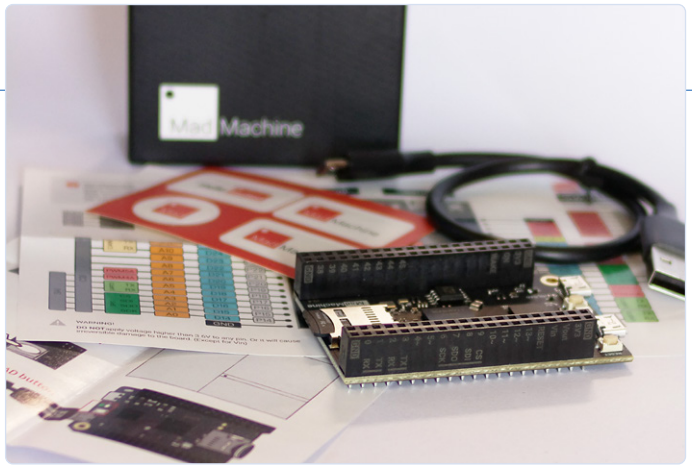
- NXP i.MX RT1052 Crossover-processor met Arm® Cortex®-M7-kern op 600 MHz
- 8 MB externe QSPI-flash
- 32 MB SDRAM
- Micro SD-card slot, ondersteunt standaard en hoge capaciteit SD-kaarten

Met de i.MX RT1052 is het board voorzien van een stevige MCU, waarvan de kenmerken te zien zijn in **figuur 2**.
Het board biedt bovendien:

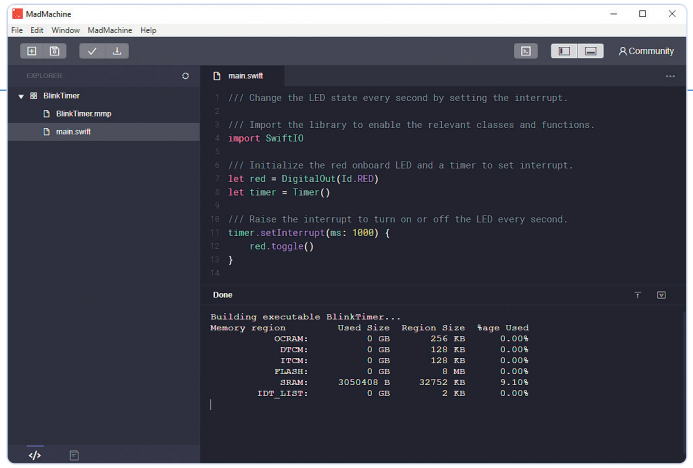
- 46 GPIO (general-purpose I/O)
- 12 ADC-kanalen (12 bit)



Figuur 2. overzicht i.MX 1052 (bron: NXP).



Figuur 3. Inhoud van de doos.



Figuur 4. IDE van MadMachine.

- 14 PWM-kanalen
- 4 UART
- 2 CAN (CAN 2.0B)
- 2 I²C (3,4 MHz max.clock)
- 2D graphic engine
- RGB-schermuitvoer (1366 x 768 WXGA max)
- Camera-ingang (CSI 8,16,24 Bit)

Dit is geen volledige lijst met functies, maar geeft je een goed idee van de hardware die je krijgt. Zelfs het complete pakket is compact van formaat. Je krijgt een board, USB-kabel, een set stickers en het SwiftIO-board met SD-kaart (**Figuur 3**). Alles wat je nodig hebt om aan de slag te gaan.

De IDE

Als het gaat om softwareontwikkeling, levert MadMachine je een IDE voor SwiftIO. Deze IDE kan van hun website worden gedownload en ziet er in principe uit als in **figuur 4** . De IDE is beschikbaar voor Windows en macOS, wat goed nieuws is voor onze Apple-gebruikers.
Als je de IDE niet prettig vindt, kun je de meegeleverde CLI-tools voor dit board gebruiken of deze combineren met XCode of Visual Studio Code voor je project. Een tutorial over het gebruik van de CLI-tool wordt geleverd door MadMachine [2].

Je hoeft niet opnieuw te flashen, je hoeft alleen maar te kopiëren

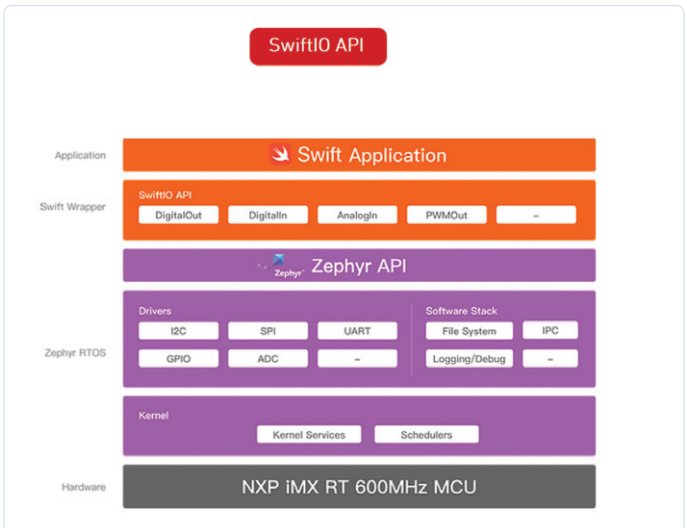
Het board bevat in de bijgevoegde SPI-flash een bootloader die je applicatie bij het opstarten naar de SDRAM kopieert en deze vanaf de SDRAM uitvoert. Dit maakt het herprogrammeren van het board eenvoudig. Met de ingebouwde programmer kun je de code naar de bijgevoegde micro-SD-kaart schrijven die verschijnt als apparaat voor massaopslag. Je kunt ook eenvoudig de SD-kaart verwijderen en je nieuwe firmware er rechtstreeks op schrijven.

Het SwiftIO Framework

Wat mijn aandacht trok, was de gebruikte software. Het SwiftIO-framework is te zien in **Figuur 5** en maakt gebruik van de Zephyr RTOS eronder. Toegang tot alle low-level hardware, zoals GPIO's, analoge functies, PWM, I²C enzovoort, wordt geabstraheerd, dus het zou gemakkelijk moeten zijn om onze eigen software te schrijven zonder rekening te houden met alle details van de i.MX 1052 MCU. Dit zorgt ook voor de SDRAM-toegang en initialisatie zoals voor de SPI-flashinstellingen.
Wat betreft de programmeertaal die voor dit board wordt gebruikt, Swift wordt vaak gebruikt op Apple-systemen en meestal gevonden in het Apple-ecosysteem. Swift is niet beperkt tot Apple-systemen en kan worden gebruikt om ook applicaties te ontwikkelen voor Linux en sinds kort ook voor Windows. Omdat Swift een programmeertaal is die moderne elementen en concepten biedt en het beschikbaar is op een MCU, is het een perfecte aanvulling om je Swift-kennis uit te breiden.
Als je je afvraagt hoe SwiftIO-code eruit ziet, laat **listing 1** zien hoe je een LED kunt laten knipperen. **Listing 2** laat zien hoe je PWM gebruikt om de helderheid van een LED te regelen. Er zijn al meer geavanceerde voorbeelden opgenomen, dus je kunt SwiftIO gaan verkennen.

Ondersteuning en verdere documentatie

Voor dit board kun je gemakkelijk ondersteuning krijgen, ga gewoon naar de MadMachine Discord-server [3] en stel je vragen in een van de chats. Gebruik zoals voor elk platform vriendelijke en beleefde bewoording; heb respect voor elkaar. Zelfs ingewikkelde technische vragen kunnen worden gesteld. Als je documenten mist, kan je hierom vragen.



Figuur 5: Software-stack (bron: MadMachine.io).

Listing 1. Knipperdemo-applicatie.

```
/// Change the LED state every second by setting the interrupt.
/// Import the library to enable the relevant classes and functions.
 import SwiftIO
/// Initialize the red onboard LED and a timer to set interrupt.
 let red = DigitalOut(Id.RED)
 let timer = Timer()
/// Raise the interrupt to turn on or off the LED every second.
 timer.setInterrupt(ms: 1000) {
     red.toggle()
 }

 while true {
}
```

Laatste gedachten

Als je op zoek bent naar een Arduino-compatibel board of iets dat direct met C / C++ kan worden gebruikt, dan is dit niets voor jou. Het board is ook veel duurder dan dan de meeste gebruikelijke ontwikkel-boards. Bij dit bordje krijg je ondersteuning en je kunt er software voor ontwikkelen onder alle belangrijke besturingssystemen. Maar uiteraard: aan jou de keuze!

210297-03

Listing 2. PWM Demo applicatie.

```
/// Brighten or dimming the LED by changing the duty cycle of PWM signal.
/// Import the library to enable the relevant classes and functions.
 import SwiftIO
/// Initialize the PWM pin the LED is connected to, with other parameters set to default.
 let led = PWMOut(Id.PWM0A)
/// Initialize a variable to store the value of duty cycle.
 var value: Float = 0.0
/// Change the brightness from on to off and off to on over and over again.
 while true {
     // Brighten the LED in two seconds.
     while value <= 1.0 {
         led.setDutycycle(value)
         sleep(ms: 20)
         value += 0.01
     }
     // Keep the duty cycle between 0.0 and 1.0.
     value = 1.0
    // Dimming the LED in two seconds.
     while value >= 0 {
         led.setDutycycle(value)
         sleep(ms: 20)
         value -= 0.01
     }
     // Keep the duty cycle between 0.0 and 1.0.
     value = 0.0
 }
```


Een bijdrage van

Ontwerp en tekst: **Mathias Claußen**
Redactie: **Jens Nickel**
Indeling: **Giel Dols**
Vertaling: **Hans Adams**

Vragen of opmerkingen?

Heb je technische vragen of opmerkingen over dit artikel? E-mail de auteur op mathias.claussen@elektor.com of neem contact op met Elektor via editor@elektor.com.

Gerelateerde producten

- **SwiftIO – Swift-based Microcontroller Board (SKU 19426)**
  www.elektor.nl/swiftio-swift-based-microcontroller-board

WEBLINKS

[1] **MadMachine**: www.madmachine.io/
[2] **MadMachine CLI How-To**: https://resources.madmachine.io/about-our-project#how-does-the-building-procedure-work
[3] **MadMachine Discord Server**: https://discord.gg/zZ9bFHK
[4] **MadMachine Resources**: https://resources.madmachine.io/