MicroPython betreedt de wereld van **Arduino**

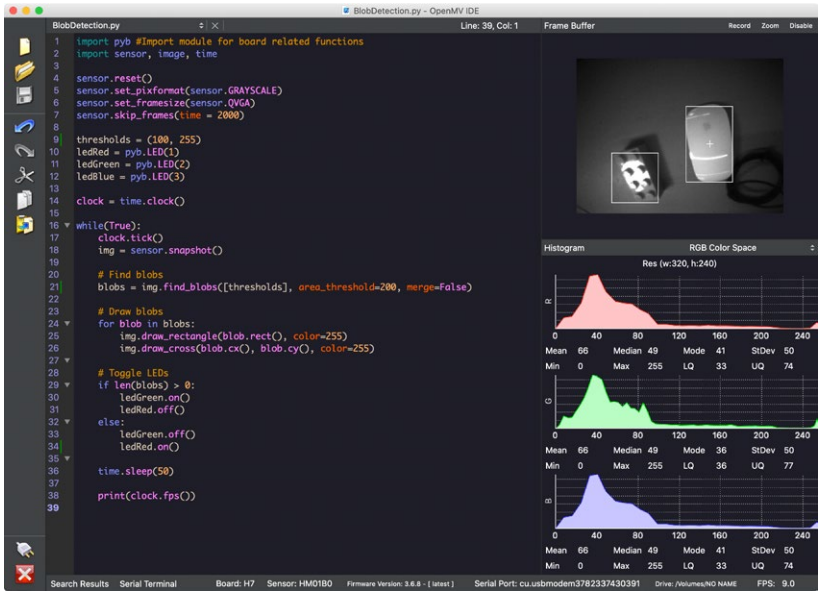
Stuart Cording (Elektor)

MicroPython is doorgedrongen tot de wereld van Arduino en biedt het eerste belangrijke alternatief voor programmeren in C en C++. Dus, wat is al die ophef, hoe gemakkelijk is het te gebruiken, en wie kan profiteren van programmeren in deze, voor microcontrollers, relatief nieuwe taal? Elektor sprak met Sebastian Romero (Head of Content, Arduino) om meer te weten te komen.

Sebastian Romero (Head of Content @Arduino)

Sebastian Romero, Head of Content bij Arduino, is een interactie-designer, pedagoog en creatief technoloog met een zwak voor mensen. Met zijn team is hij verantwoordelijk voor het creëren van boeiende leerervaringen om talloze technici, ontwerpers, kunstenaars, hobbyisten en studenten te helpen innoveren.

C en C++ zijn sinds de geboorte van Arduino in het begin van de jaren 2000 de basis geweest voor de ontwikkeling van Arduino-software. Dankzij een voorgedefinieerde programmastructuur met een setup()- en een loop()-functie, werden beginners in de wereld van embedded software-ontwikkeling begeleid bij het opzetten van hun board en het uitvoeren van hun toepassing in een loop. Nu is er een nieuwe taal beschikbaar. MicroPython is een ietwat uitgeklede versie van Python, een geïnterpreteerde programmeertaal voor algemeen gebruik, gericht op microcontrollers. De vraag is waarom we een geïnterpreteerde taal zouden gebruiken op hardware die gebruikt wordt voor real-time toepassingen?



Blobdetectie op Arduino Portenta H7 in OpenMV.

"Omdat MicroPython door zijn eenvoud zeer geschikt is voor beginners, behoorden docenten tot de eersten die ons ernaar vroegen," legt Sebastian Romero, Hoofd Content bij Arduino, uit.

In C is de omgang met microcontroller-registers gemakkelijker dan in assembler, en object-georiënteerd programmeren (OOP) in C++ zorgt voor een meer beknopte code met minder fouten. Het ontleden van strings is echter een uitdaging, en er is geen native ondersteuning in de taal voor het verwerken van de hedendaagse web-dataformaten zoals HTTP, JSON [1] of RegEx [2] (reguliere expressie). Omdat het hedendaagse onderwijs draait om interactie met het internet en webdiensten, is C/C++ op een zijspoor gezet ten gunste van talen als Python, die het coderen van zulke toepassingen eenvoudiger maken.

"Als gevolg daarvan blijf je als docent liever bij Python als het onderwerp microcontrollers ter sprake komt," zegt Sebastian.

Natuurlijk zijn het niet alleen docenten. Makers hebben een reeks boards van andere leveranciers die MicroPython ondesteunen, zoals de ESP32, Raspberry

Pi Pico en pyboard. En ook de industrie kijkt steeds meer naar MicroPython. De snelle groei van machine learning (ML) is deels te danken aan het feit dat er bibliotheken voor Python bestaan. Met teams van technici die vertrouwd zijn met Python, willen weinigen overstappen naar C/C++ wanneer ze hun ML-model en -toepassing overbrengen naar een microcontroller, en blijven ze liever bij één ontwikkelstack. Het andere probleem is het personeelsbestand – het wordt steeds moeilijker om C/C++ programmeurs te vinden, terwijl de academische wereld volop Python-competente technici voortbrengt.

MicroPython versus Python: wat is het verschil?

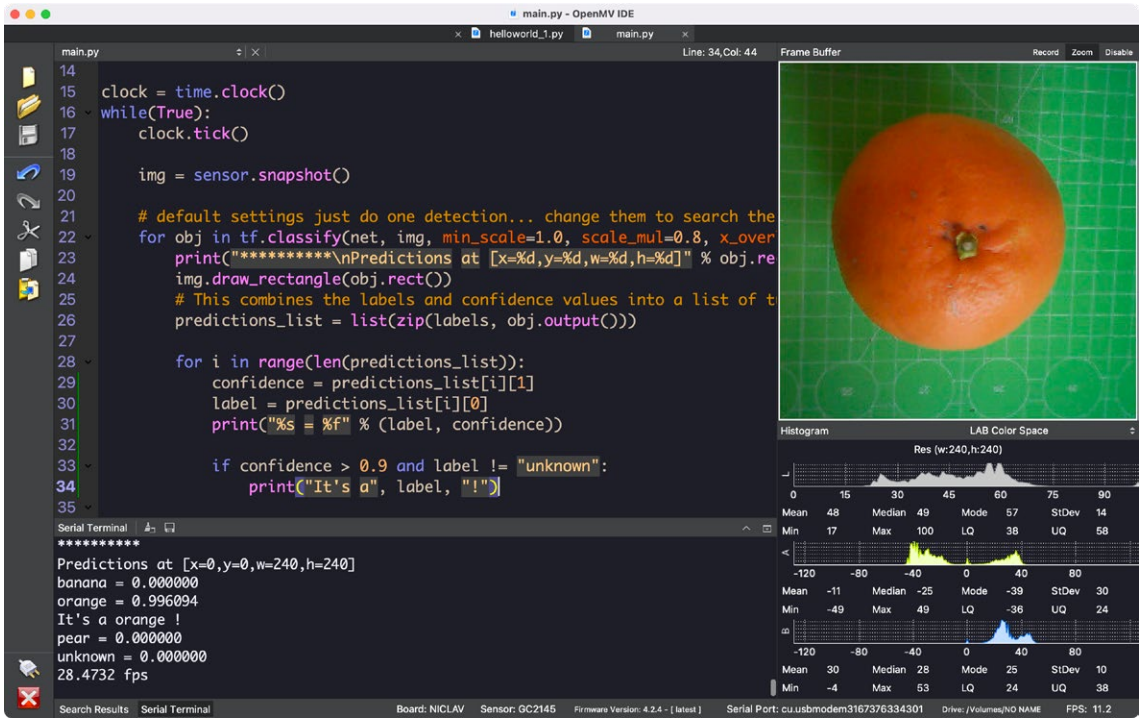
Python zag het licht tegen het eind van de jaren tachtig, ontworpen door Guido van Rossum [3]. Het is ontworpen voor een prettig gebruik, en wil ook expliciet zijn in plaats van impliciet, eenvoudig, en leesbare code opleveren. In 2013 lanceerde Damien George met succes een Kickstarter-campagne [4] voor een vanaf de grond af ontworpen versie voor microcontrollers, samen met de pyboard-hardware om het programma te draaien. Micro Python, zoals het toen heette, beloofde een scripttaal waarmee je 'moeiteloos LED's kunt laten knipperen, spanningen kunt aflezen' en meer. Microcontrollers met een USB-aansluiting zouden als een USB-stick verschijnen waar code naar toe kon worden geschreven. Het systeem kon ook dienst doen als een serieel apparaat, met

In tegenstelling tot C/C++ sketches die moeten worden gecompileerd en geladen, kan MicroPython direct na elke wijziging worden uitgevoerd.

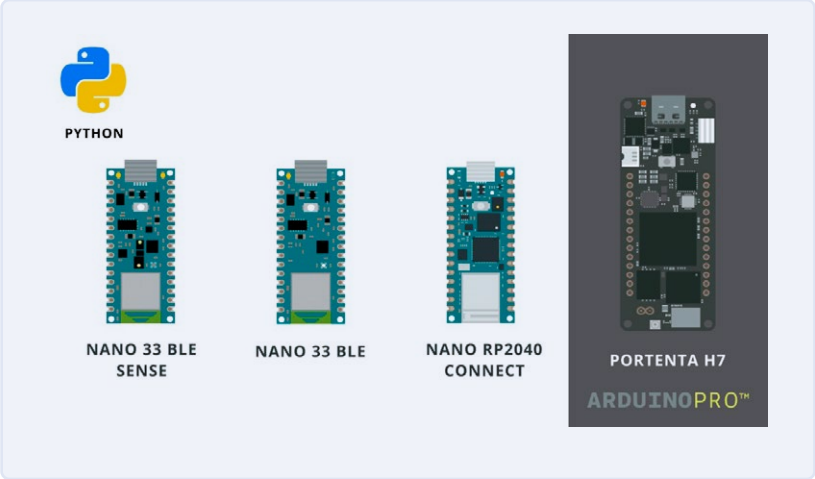
een commandoregel die bekend staat als REPL (read, evaluate, print, loop).

Zoals te verwachten, heeft MicroPython een redelijke hoeveelheid geheugen nodig om de geladen code en de interpretatie daarvan tijdens de uitvoering te ondersteunen. Hoewel 128 kB flash en 8 kB SRAM in theorie volstaan, zijn de mogelijkheden dan zo beperkt dat het niet prettig werkt. Daarom gebruiken de meeste MicroPython-boards een microcontroller met tenminste 256 kB flash en 16 kB SRAM. Dat heeft als neveneffect dat een systeem is gekozen met een relatief krachtige processor die op ongeveer 50 MHz of meer draait en een respectabele hoeveelheid mogelijkheden biedt.

"Het verbaast me nog altijd wat er met niet meer dan 16 kB SRAM mogelijk is," citeert Sebastian – het citaat is van Jim Mussared, een embedded engineer bij micropython.org. "De meeste studenten hebben, tenminste bij beginnersprojecten, geen grote heap nodig. De complexiteit van hun projecten neemt meestal toe door meer code."



Beeldclassificatie op Arduino Nicla Vision in OpenMV.



Zulke MicroPython-projecten implementeren voornamelijk toestandsmachines en evalueren sensorgegevens.
Maar het SRAM wordt niet alleen gebruikt om variabelen in op te slaan. Het slaat ook de gecompileerde bytecode op, zoals geïmporteerde modules, voor uitvoering door de MicroPython virtuele machine (VM). Dat kan problemen opleveren bij het verwerken van grote hoeveelheden data, zoals strings, of het maken en verwijderen van veel objecten waardoor er onvoldoende SRAM overblijft om de compiler uit te voeren. Zodra de code echter volwassen is geworden, kan de bytecode worden voorgecompileerd en opgeslagen in flash (in het bestandssysteem) of worden geïmplementeerd als frozen bytecode om nog meer SRAM te besparen [5].
MicroPython is ook kieskeuriger dan Python bij het invoeren van code, en eist correct gebruik van spaties. Gebruikers leren ook snel dat sommige standaardfuncties, zoals een volledige implementatie van de standaardbibliotheek, niet beschikbaar zijn vanwege de beperkte hardwaremogelijkheden.

Ontwikkelomgeving voor Arduino MicroPython

Het Arduino-team selecteerde de versie van MicroPython die door OpenMV wordt onderhouden, omdat de nieuwe met camera's uitgeruste boards van Arduino profiteren van de machine vision-functies en de ingebouwde ondersteuning voor Tensor Flow Lite die OpenMV biedt. [6] Het resultaat is dat gebruikers een goed ondersteund, volwassen platform en ontwikkelomgeving hebben. OpenMV is gemaakt om machine vision-toepassingen op microcontrollers te ondersteunen, wat goed aansluit bij de wens van veel gebruikers om afbeeldinggebaseerde ML-toepassingen te bouwen.

OpenMV IDE (Integrated Development Environment) biedt het werkterrein voor MicroPython-coders in plaats van de traditionele Arduino IDE. Het biedt de basisfuncties die ontwikkelaars verwachten, zoals een venster voor het ontwikkelen van de code en een seriële terminal. Bovendien is er ondersteuning voor machine vision toepassingen, zoals visualisatie van een framebuffer en een histogramtool om kleur- en helderheidsbereiken visueel te analyseren. Verder heeft het een ingebouwde tool om camerabeelden direct te uploaden naar Edge Impulse Studio om eenvoudig een machine learning-model te trainen.
Maar misschien ligt de grootste verandering in de manier waarop code wordt ontwikkeld en toegepast. "In tegenstelling tot C/C++ sketches die moeten worden gecompileerd en geladen, kan MicroPython direct na elke wijziging worden uitgevoerd. Dat versnelt de ontwikkeling aanzienlijk en brengt de codeerervaring dichter bij die van Python," aldus Sebastian.
Een andere geweldige functie is REPL, waarmee korte scripts kunnen worden uitgevoerd of afzonderlijke functies direct op de controller kunnen worden getest.

Arduino hardware-ondersteuning voor MicroPython

In totaal ondersteunen momenteel vijf Arduino-boards MicroPython: de Nano 33 BLE en Nano 33 BLE Sense; de Nano RP2040 Connect; de Portenta H7 en de Nicla Vision. De meeste boards vereisen een firmware-update om de MicroPython-runtime in flash te laden voordat ze aan de slag gaan. Zoals we gewend zijn, is dit proces niet alleen eenvoudig, maar ook goed gedocumenteerd [7]. Boards als de Nano 33 hebben een voorbereidingsstap die de Arduino IDE gebruikt, terwijl de andere direct door OpenMV worden herkend en geprogrammeerd met de benodigde firmware.
De toepassing wordt geschreven als een Python-script in OpenMV dat wordt overgebracht naar het board. Een enkele klik op de Play-knop is alles wat tussen de programmeur en de uitvoering van de code staat.

Wat kunnen we verwachten?

Wat is de toekomst voor Arduino nu MicroPython er is? Natuurlijk ben je bezorgd dat met de introductie van MicroPython de traditionele C/C++ sketch een antiquiteit wordt. Maar dat is niet gewenst en ook niet de bedoeling. In scenario's waar real time-uitvoering een vereiste is, zal C/C++ het aangewezen middel blijven.
"We zien een groei in de vraag naar Python-ondersteuning op microcontrollers, vooral bij pioniers in de industrie die werken aan de ontwikkeling van ML-toepassingen, die al een Python-stack gebruiken," zegt Sebastian.

In feite breidt de ondersteuning van MicroPython de beschikbare opties voor Arduino-gebruikers eerder uit dan dat het ze vervangt. Op lange termijn zouden ontwikkelaars van C/C++ sketches geschikte boards in dezelfde vormfactor moeten vinden die ook MicroPython ondersteunen.

"Voor veel van de klassieke Arduino-boards zou een MicroPython-implementatie zeer beperkte voordelen bieden en dus geen redelijke optie zijn," voegt Sebastian eraan toe.

Gebruikers kunnen ook blijven bijdragen aan het succes van Arduino [8] met MicroPython zoals ze in het verleden hebben gedaan. Er is 15 jaar aan C/C++ code die nog steeds wordt onderhouden en gebruikt als driver in combinatie met MicroPython. Dan worden bindings gebruikt om MicroPython aan deze basiscode te koppelen. Voor degenen die betrokken willen raken bij de ontwikkeling van MicroPython [10] wordt OpenMV ook gehost op GitHub [9], een project waaraan het Arduino team bijdraagt.

MicroPython kan worden gezien als een aanvulling op het huidige Arduino-ecosysteem, waarvan de toepassing wordt gedreven door het succes van Python als voorkeurstaal voor ML-toepassingen en interactie met cloud-services. Nu microcontrollers steeds vaker honderden MHz halen en bergen geheugen hebben, zal de overstap naar een geïnterpreteerde taal in veel gevallen als irrelevant worden gezien. Natuurlijk zijn er uitzonderingen waar real-time nauwkeurigheid en precisie een must zijn, en C/C++ zal er altijd zijn voor wie dat nodig heeft. Maar vooralsnog profiteren docenten en studenten in sterke mate, omdat kennis van Python kan worden overgedragen op microcontrollers (bovendien profiteren ze van de vereenvoudigde syntaxis en de leesbaarheid van de code). Tegelijkertijd kunnen industriële ontwikkelaars zich houden aan één enkele taal voor de ontwikkeling van toepassingen.

220415-03 (vertaling: Hans Adams)

Over de auteur

Stuart Cording is technicus en journalist met meer dan 25 jaar ervaring in de elektronica-industrie. Je kunt veel van zijn recente Elektor-artikelen vinden op www.elektormagazine.com/cording. Naast het schrijven voor Elektor oresenteert de maandelijkse livestream Elektor Engineering Insights (www.elektormagazine.com/eei), en geeft hij Elektor Academy-cursussen (www.elektormagazine.com/elektor-academy).

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via stuart.cording@elektor.com of naar de redactie van Elektor via redactie@elektor.com.

Gerelateerde producten

Op zoek naar de belangrijkste producten die in dit artikel worden genoemd? Arduino en Elektor hebben ze voor je klaargezet!

- **Arduino Nano 33 BLE Sense**
  elektormagazine.nl/arduino-nano33sense
- **Arduino Nano RP2040 Connect**
  elektormagazine.nl/arduino-nano-rp2040-connect
- **Arduino Portenta H7**
  www.elektormagazine.nl/arduino-portenta-h7
- **Arduino Nicla Vision**
  www.elektormagazine.nl/arduino-nicla-vision

WEBLINKS

[1] JSON (Wikipedia): https://en.wikipedia.org/wiki/JSON
[2] Regular Expression (Wikipedia): https://en.wikipedia.org/wiki/Regular_expression
[3] Python (Wikipedia): https://en.wikipedia.org/wiki/Python_(programming_language)
[4] D. George, "Micro Python: Python for microcontrollers," Kickstarter, 2016: www.kickstarter.com/projects/214379695/micro-python-python-for-microcontrollers
[5] "MicroPython on Microcontrollers": https://docs.micropython.org/en/latest/reference/constrained.html
[6] OpenMV: https://openmv.io/
[7] K. Soderby, "Python with Arduino Boards," Arduino, 2022: https://docs.arduino.cc/learn/programming/arduino-and-python
[8] Arduino (GitHub): https://github.com/arduino
[9] OpenMV (GitHub): https://github.com/openmv
[10] Official uPython Repo: https://github.com/micropython/micropython