Einführung in MicroPython und CircuitPython

Mattias Schlenker

Wilhelm-Ostwald-Gymnasium

14. Juli 2021

Herausforderung Hardware

- Ziel war es, einen Python-Interpreter auf 32 Bit Microcontroller zu bringen
- Microcontroller sind oft Fingernagel kleine Computer mit geringstem Energiebedarf
- Einsatzgebiete: Steuerungs- und Regelungs-Aufgaben, Messwerte erfassen
- Kosten: von unter 1 Euro bis ca. 15 Euro
- Problem: Wenig Festspeicher (oft unter 1MB) und wenig Arbeitsspeicher

Die Lösung

- ,,Baremetal Python" – der Python-Interpreter läuft ohne Betriebssystem direkt auf dem Microcontroller
- Abgespeckte Standardbibliothek

Ergebnis: Micropython läuft bereits auf 32 Bit Microcontrollern mit 32kB RAM und 256kB Festspeicher. Mehr ist natürlich immer besser...

MicroPython oder CircuitPython?

- MicroPython ist das ältere Projekt mit der breiteren Hardwareunterstützung
- CircuitPython basiert auf Micropython, hat aber eine leichter verständliche Klassenbibliothek für Interaktion mit dem Microcontroller und ist intuitiver nutzbar

Empfehlung: Steht der Neukauf eines Microcontroller-Boards an, ein paar Euro mehr investieren und ein von CircuitPython unterstütztes Board kaufen, ist ein Board vorhanden, das nur MicroPython unterstützt, dann halt MicroPython.

Installation am Beispiel Xiao (CircuitPython auf SAMD21)

- Download der UF2-Datei von circuitpython.org
- Reset-Pins zweimal hintereinander kurzschließen
- USB-Massenspeicher-Laufwerk wird angezeigt
- UF2-Datei dorthin verschieben
- Microcontroller rebootet

Installation ist ohne zusätzliche Tools möglich. Programmierung auch. Dazu gleich...

Installation am Beispiel ESP32 Devkit (MicroPython)

- Download der BIN-Datei von micropython.org
- Installation der Python-Programme esptool und ampy
- esptool –chip esp32 –port COM4 write_flash -z 0x1000 esp32. . . .bin
- Auf serieller Konsole zugreifen und mit ,,import webrepl_setup" die Webkonsole aktivieren
- Microcontroller rebooten

Bei aktiviertem WebREPL sind keine zusätzlichen Tools nötig. Die Programmierung kann an jedem PC stattfinden. Dazu gleich...

Programmierung am Beispiel Xiao (CircuitPython)

- Beim Anschließen an den PC öffnet sich ein USB-Laufwerk
- Hier kann mit jedem guten Editor die ,,main.py" verändert werden
- Speichert man die Datei, startet der Interpreter neu
- Serielle Konsole kann bspw. mit PuTTY zugegriffen werden
- Adafruit empfiehlt den Editor ,,https://codewith.mu/"

Blink-Beispiel CircuitPython

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

(Beispiel mit Auslesen eines Pins wird vorgeführt)

Blink-Beispiel MicroPython

In Mu speichern und per WebREPL hochladen...

```
import machine
import time

led = machine.Pin(5, machine.Pin.OUT)

while True:
    led.on()
    time.sleep(0.5)
    led.off()
    time.sleep(0.5)
```

Achtung: Pin-Mapping entspricht oft nicht den Aufdrucken am Board. Pinout-Diagramme heranziehen!

Links und Fazit

- Ich habe 3 ESP ausprobiert, bis ich einen gefunden habe, bei dem serielle Konsole und WIFI zuverlässig funktionierten
- CircuitPython ist von Handhabung und Stabilitär zu bevorzugen
- CircuitPython
- MicroPython
- WebREPL
- EditorMu
- https://github.com/mschlenker/GtaRoboter2020 - Einzelveranstalung CircuitPython - diese Folien