ESP32 + micropython : blink

Lesson Plan for Grade 7, micro science
Prepared by Ms. e

(11.oct.2020 18:10)

OVERVIEW & PURPOSE

Das Programmieren mit micropython hat ein lustiges (kruder Humor) Feature, die Kommandozeile am esp32. Was geht damit gut und wo macht das Probleme ?

MATERIALS NEEDED

1. Interesse
2. Thonny IDE
3. Esp32 mit installiertem micropython

Blink eins

(18:17)

Thonny starten und mit dem esp32 verbinden.

In der shell "`from machine import Pin`" engeben.

"`machine`" ist die Bibliothek die den Zugriff auf die Maschine, die Hardware enthält. Unter anderem die "`Pin`"s.

Python ist eine objektorientierte Programmiersprache. Das ist ein Problem für micropython, aber ein Vorteil für uns. Und die shell hilft auch.

Wenn wir "`Pin`" eingeben, gibt die shell "`<class 'Pin'>`" aus. Eine `class` ist der Bauplan für ein `object`. Wenn wir ein Pin-Objekt haben wollen, wird das aus der Pin-Klasse konstruiert:

```
>>> Pin()
TypeError: function missing 1 required positional arguments
```

Der python-Interpreter braucht noch etwas Information um das Objekt zu bauen … welchen Pin, die Nummer, weil der esp32 mehr als einen hat:

```
>>> Pin(2)
Pin(2)
```

… nicht sehr spannend:

```
>>> p2 = Pin(2)
>>>
```

… noch weniger … aber jetzt gibt es etwas mit dem Namen p2:

```
>>> p2
Pin(2)
```

… ahhhhh

(18:33)

Vielleicht sollten wir doch die Hilfe lesen … vorher probieren wir noch etwas :

```
>>> dir(p2)
['__class__', 'value', 'IN', 'IRQ_FALLING', 'IRQ_RISING',
'OPEN_DRAIN', 'OUT', 'PULL_DOWN', 'PULL_HOLD', 'PULL_UP',
'WAKE_HIGH', 'WAKE_LOW', 'init', 'irq', 'off', 'on']
```

Endlich etwas mehr. Das ist die Liste von Dingen, Attributen und Methoden, die der Pin hat und kann:

```
>>> p2.__class__
<class 'Pin'>
>>> p2.value
<bound_method>
```

Eine ist etwas das man ausführen, aufrufen kann, weil die “method” “bound” gebunden ist, muss man sie am Objekt aufrufen:

```
>>> p2.value()
0
```

Wir können den Pin zum Ausgang machen:

```
>>> p2.init(Pin.OUT)
```

Und dann eine angeschlossene LED zum Leuchten bringen:

```
>>> p2.value(1)
>>> p2.value(0)
```

Damit ist unser erstes manuelles Blinkprogramm fertig.

(18:45 Pause (kochen, essen, lesen 21:02)

Wir packen das ganze in ein Programm, initialisieren den Pin als OUTput:

```
from machine import Pin
p2 = Pin(2, Pin.OUT)
p2.value(1)
p2.value(0)
```

Die LED blitzt kurz auf.

Das man das überhaupt sieht bedeutet dass micropython nur bedingt für hardwarenahe Programmierung geeignet ist … zum Probieren und wenn man weiß was man tut aber schon.

Wir können den Blitz, die Dauer des Impulses mit einem Oszilloskop messen … leichter geht es in einer Schleife:

```
from machine import Pin
p2 = Pin(2, Pin.OUT)
while True:
    p2.value(1)
    p2.value(0)
```

“`while True:`” ergibt eine Endlosschleife … die man mit Strg-C abbrechen kann.

NB: `p2.on()` und `p2.off()` kann man auch verwenden.

Es blinkt zu schnell für das Auge, es braucht etwas mehr Zeit zwischen den Ein- und Auszuständen. Wir probieren:

```
>>> import time
>>>
```

… keine Fehlermeldung, es gibt ein Modul “time”. Schauen wir einmal was drin ist (micropython ist micro, die Module enthalten nicht so viel wie die am CPython):

```
>>> dir(time)
'__class__', '__name__', 'localtime', 'mktime', 'sleep',
'sleep_ms', 'sleep_us', 'ticks_add', 'ticks_cpu', 'ticks_diff',
'ticks_ms', 'ticks_us', 'time']
```

Dreimal “`sleep`” … und “`localtime`”:

```
>>> time.localtime()
(2020, 10, 11, 19, 16, 0, 6, 285)
```

Datum stimmt, die Uhrzeit nicht ganz … zurück zum Blinker:

```
from machine import Pin
import time
p2 = Pin(2, Pin.OUT)
while True:
    p2.value(1)
    time.sleep(1)
    p2.value(0)
    time.sleep(1)
```

oder kürzer … wir lesen den `value()` und setzen den invertierten Wert:

```
from machine import Pin
import time
p2 = Pin(2, Pin.OUT)
while True:
    p2.value(not p2.value())
    time.sleep(1)
```

Wenn man diesen Text in eine Datei “`main.py`” am esp32 speichert wird das Blinkprogramm beim nächsten Reset ausgeführt. Strg-C bricht das Programm zwar ab, die shell-Verbindung funktioniert aber erst nach einem Klick auf das Stoppschild im Thonny oder Strg-F2.

(21:30 genug für heute)