Datum & Uhrzeit

Zum Umgang mit Datum und Uhrzeit verwenden Python und Micropython das selbe Konzept wie Unix und Linux. Als **timestamp** wird die Anzahl der Sekunden seit dem Beginn der Epoche benutzt.

Bei Unix, Linux und Python beginnt die Epoche am 01.01.1970. 00:00:00 Bei **Micropython** ist es der **01.01.2000 00:00:00**.

Datum & Uhrzeit

Datum und Uhrzeit in Micropython.org

Datum & Uhrzeit

Die Funktionen zum Umgang mit Datum und Uhrzeit befinden sich in Micropython in dem Modul **time** bzw. **utime**.

Beide Module sind identisch. Wobei **Micropython 1.12** nur **utime** und **Micropython 1.19** nur **time** enthält. Im M5-Micropython lässt sich beides aufrufen und ist identisch.

In CPython gibt es andere Module und mehr Funktionen.

Datum & Uhrzeit

Die Module **time** und **utime** im M5-Micropython:

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

>>> time == utime
True

>>> time is utime
True

>>> id(time)
1061756836

>>> id(utime)
1061756836
```

Datum & Uhrzeit

Der Inhalt von **utime**:

Es gibt 3 Funktionen:
localtime(),
mktime(),
time(),
die für Datum und Uhrzeit zuständig sind.

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

Datum & Uhrzeit

Aktuelle Zeit ermitteln:
utime.time()

```
>>> from m5import import *

>>> import utime

>>> utime.time()
727390341
```

Datum & Uhrzeit

Timestamp in lesbares Datum umwandeln:

utime.localtime()

Jahr	
Monat	(1-12)
Tag	(1-31)
Stunde	(0-23)
Minute	(0-59)
Sekunde	(0-59)
Wochentag	(0-6)
Tag des Jahres	(0-366)

```
>>> utime.localtime(_)
(2023, 1, 18, 20, 52, 21, 2, 18)

>>> datum = utime.time()
>>> dat_lesbar = utime.localtime(datum)
>>> dat_lesbar
(2023, 1, 18, 21, 3, 33, 2, 18)

>>> dat_formatiert = str(dat_lesbar[2]) + '.' +
str(dat_lesbar[1]) + '.' +str(dat_lesbar[0])

>>> dat_formatiert
'18.1.2023'
```

Datum & Uhrzeit

```
>>> WOCHENTAG = ( 'Montag',
                  'Dienstag',
                  'Mittwoch',
                  'Donnerstag',
                  'Freitag',
                  'Samstag',
                  'Sonntag')

>>> dat_formatiert = WOCHENTAG[dat_lesbar[6]] + ', den ' + str(dat_lesbar[2]) +
'.' + str(dat_lesbar[1]) + '.' + str(dat_lesbar[0])

>>> dat_formatiert
'Mittwoch, den 18.1.2023'
```

Datum & Uhrzeit

Timestamp aus lesbarem Datum erzeugen.
utime.mktime()
Diese Funktion erwartet einen Tupel, so wie von localtime() erstellt wird.

```
>>> timestamp = utime.time()
>>> datum = utime.localtime(timestamp)
>>> datum
(2023, 1, 18, 21, 29, 41, 2, 18)
>>> new_timestamp = utime.mktime(datum)
>>> new_timestamp
727392581

>>> timestamp == new_timestamp
True
>>> timestamp is new_timestamp
True
>>> id(timestamp)
1454785163
>>> id(new_timestamp)
1454785163
```

Datum & Uhrzeit

Mit Timestamps rechnen

Datum & Uhrzeit

Die interne Uhr

RTC im Micropython

Datum & Uhrzeit

Auch das Micropython von **Micropython.org** enthält Funktionen zur Nutzung der **RTC**. Diese sind als Klasse in dem Module **machine** implementiert:

```
>>> import machine
>>> dir(machine)
['__class__', '__name__', 'ADC', 'CAN', 'DAC',
'DEEPSLEEP', 'DEEPSLEEP_RESET', 'EXT0_WAKE',
'EXT1_WAKE', 'HARD_RESET', 'I2C', 'I2S',
'Modbus', 'ModbusSlave', 'Neopixel', 'PIN_WAKE',
'PWM', 'PWRON_RESET', 'Pin', 'RTC', 'SDCard',
'SLEEP', 'SOFT_RESET', 'SPI', 'Signal',
'TIMER_WAKE', 'TOUCHPAD_WAKE', 'Timer',
'TouchPad', 'UART', 'ULP_WAKE', 'WDT',
'WDT_RESET', 'deepsleep', 'disable_irq',
'enable_irq', 'freq', 'idle', 'lightsleep',
'mem16', 'mem32', 'mem8', 'reset',
'reset_cause', 'sleep', 'soft_reset',
'time_pulse_us', 'unique_id', 'wake_reason']
>>> dir(machine.RTC)
['__class__', '__name__', '__bases__',
'__dict__', 'datetime', 'init', 'memory']
```

Datum & Uhrzeit

Da **RTC** im Module **machine** groß geschrieben ist handelt es sich um eine **Klasse**. Deshalb müssen wir zuerst eine **Instanz** der Klasse RTC erzeugen.

RTC.datetime() dient zum setzen und lesen der RTC. Daten müssen als Tupel oder Liste übergeben werden.

```
>>> uhr = machine.RTC()

# RTC auslesen
>>> uhr.datetime()
(2023, 1, 18, 2, 21, 52, 10, 398136)

# RTC stellen
>>> uhr.datetime((2024, 1, 18, 3, 21,
52, 26, 0))
>>>

>>> uhr.datetime()
(2024, 1, 18, 3, 21, 56, 6, 271975)
```

Datum & Uhrzeit

RTC.init() funktioniert wie das Stellen der RTC mit RTC.datetime()

RTC.memory() dient zum beschreiben und lesen des internen Speicher der RTC. Dieser Inhalt bleibt erhalten, wenn nur noch die RTC mit Strom versorgt wird.

```
>>> uhr.init((2020, 1, 18, 3, 21, 52, 26, 0))

>>> uhr.datetime()
(2020, 1, 18, 5, 21, 52, 31, 715816)


>>> uhr.memory()
b''
```

Datum & Uhrzeit

Im Micropython von Micropython.org ist die Reihenfolge der Daten im Tuple für die RTC etwas anders:

Jahr
Monat
Tag
Wochentag
Stunde
Minute
Sekunde
Timestamp ?

Datum & Uhrzeit

RTC in
M5Stack Micropython

Datum & Uhrzeit

Real Time Clock (RTC)

Die RTC des M5Stick C Plus.
Der M5Stick C/Plus enthält eine RTC. Diese kann man stellen und auslesen.
Hier die M5-Micropython Funktionen dafür:

```
from m5import import *
# RTC stellen
rtc.setTime(Jahr,
            Monat,
            Tag,
            Stunde,
            Minute,
            Sekunde)
# RTC lesen
Datum = rtc.now()
(2023, 2, 7, 14, 49, 27)
jahr = rtc.now()[0]
monat = rtc.now()[1]
tag = rtc.now()[2]
stunde = rtc.now()[3]
minute = rtc.now()[4]
sekunde  = rtc.now()[5]
```

Die Uhrzeit selber holen

NTP

Datum & Uhrzeit

NTP steht für **Network Time Protocol** und regelt die Übertragung von Datum und Zeit über das Internet.

Um auf diese Daten zugreifen zu können müssen wir den **M5Stack C Plus mit dem Internet verbinden**. Auf Einzelheiten gehen wir später ein.

```
from m5import import *

import network
import wifiCfg

ssid = 'Attraktor'
pw = 'blafablafa'

wlan = network.WLAN(network.STA_IF)
wlan.active(True)

wifiCfg.doConnect(ssid, pw)

while not (wifiCfg.wlan_sta.isconnected()):
    pass

print(wlan.ifconfig()[0])
```

Datum & Uhrzeit

Die Uhr über das Internet stellen

NTP mit Micropython
von Micropython.org

Datum & Uhrzeit

Das **ntp.py** Modul von Micropython.org ist im M5Stack Micropython **nicht enthalten**. Dort gibt es nur **ntptime.py**.

```
>>> import ntp

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: no module named 'ntp'
```

Datum & Uhrzeit

NTP im
M5Stack Micropython

Datum & Uhrzeit

Die Funktionen von **ntptime.py** im M5Stack Micropython.

Auch hier muss erst eine Instanz der Klasse **ntptime.client** erzeugt werden.

```
import ntptime

ntp = ntptime.client(host='de.pool.ntp.org', timezone=1)

temp_var = ntp.getTimestamp()

temp_var = ntp.formatDate('-')
temp_var = ntp.formatTime(':')
temp_var = ntp.formatDatetime('-', ':')

temp_var = ntp.year()
temp_var = ntp.month()
temp_var = ntp.day()
temp_var = ntp.weekday()
temp_var = ntp.hour()
temp_var = ntp.minute()
temp_var = ntp.second()
```

Datum & Uhrzeit

In der Klasse **ntptime.client** finden sich eine ganze Reihe Methoden.
Bis auf **printtime** und **updateTime** sind alle auch in der **UIFlow-IDE** enthalten und selbsterklärend.

```
>>> dir(ntptime)
['__class__', '__name__', '__file__',
'RTC', 'socket', 'struct', 'time',
'wifiCfg', 'NTP_DELTA', 'WEEKS',
'client']

>>> dir(ntptime.client)
['__class__', '__init__',
'__module__', '__name__',
'__qualname__', '__bases__',
'__dict__', 'printtime', 'day',
'weekday', 'getTimestamp',
'updateTime', 'formatDate',
'formatTime', 'formatDatetime',
'year', 'month', 'hour', 'minute',
'second']
```

Datum & Uhrzeit

Es muss eine **Instanz** von **ntptime.client** erzeugt werden und dieser die **URL** des **Timeservers** übergeben werden.

```
>>> import ntptime

>>> ntp = ntptime.client(host='de.pool.ntp.org', timezone=1)

>>> ntp.formatDatetime('-', ':')
'2023-02-06 Mon 15:34:56'
```

Datum & Uhrzeit

Nun muß nur noch das Geheimnis um **printtime** und **updateTime** gelüftet werden:

printtime() gibt Datum und Zeit aud der seriellen Schnittstelle aus und **updateTime()** holt die aktuelle Zeit.

Also 2 Funktionen, die das leben erleichtern.

```
>>> ntp.printtime()
2023-02-06 Mon 15:43:04

>>> ntp.updateTime()
>>> ntp.printtime()
2023-02-06 Mon 15:44:21
>>>
```

Datum & Uhrzeit

ENDE