Module

Module sind Funktionssammlungen die in eigene Programme eingebunden werden können.

In anderen Programmiersprachen werden sie als **Library** oder **Bibliothek** bezeichnet.

Micropython bringt eine Reihe Standard Module mit.

M5Stack-Micropython ebenfalls.

Jedes Micropythonscript, dass Funktionen enthält kann als Modul benutzt werden.

Module

In Micropython enthaltene Module:

- Builtin functions and exceptions
- `cmath` – mathematical functions for complex numbers
- `gc` – control the garbage collector
- `math` – mathematical functions
- `sys` – system specific functions
- `uarray` – arrays of numeric data
- `ubinascii` – binary/ASCII conversions
- `ucollections` – collection and container types
- `uerrno` – system error codes
- `uhashlib` – hashing algorithms
- `uheapq` – heap queue algorithm
- `uio` – input/output streams
- `ujson` – JSON encoding and decoding
- `uos` – basic “operating system” services
- `ure` – simple regular expressions
- `uselect` – wait for events on a set of streams
- `usocket` – socket module
- `ussl` – SSL/TLS module
- `ustruct` – pack and unpack primitive data types
- `utime` – time related functions
- `uzlib` – zlib decompression
- `_thread` – multithreading support

Micropython spezifische Module:

- `btree` – simple BTree database
- `framebuf` — Frame buffer manipulation
- `machine` — functions related to the hardware
- `micropython` – access and control MicroPython internals
- `network` — network configuration
- `ubluetooth` — low-level Bluetooth
- `ucryptolib` – cryptographic ciphers
- `uctypes` – access binary data in a structured way

In Micropython enthaltene Module für ESP32:

- `esp` — functions related to the ESP8266 and ESP32
  - Functions
- `esp32` — functionality specific to the ESP32
  - Functions
  - Flash partitions
  - RMT
  - The Ultra-Low-Power co-processor
  - Constants

Module

Module in der M5Stack Firmware:

```
>>> help('modules')
IoTcloud/AWS        hmac                menu/startup       units/_ext_io2
IoTcloud/Ali        i2c_bus             menu/wifi          units/_fader
IoTcloud/Azure      inisetup            micropython        units/_fader8
IoTcloud/Tencent    libs/__init__       modules/_catm_iot  units/_fan
IoTcloud/__init__   libs/bmm150         modules/_nb_iot    units/_finger
IoTcloud/blynk      libs/bmp280         neopixel           units/_gesture
MediaTrans/Mqtt_Printer                 libs/config        network
   units/_gps
MediaTrans/TimerCam                     libs/dht12         ntptime
   units/_grove2grove
MediaTrans/__init__                     libs/easyIO        ntptime
   units/_hall
MicroWebSrv/__init__                    libs/echo          simpleOTA
   units/_heart
MicroWebSrv/microWebSocket              libs/emoji         smartconfig
   units/_imu6886
MicroWebSrv/microWebSrv                 libs/ethernet/__init__
   sys                 units/_ir
MicroWebSrv/microWebTemplate            libs/ethernet/wiznet5k
   uarray              units/_joystick
__main__               libs/ethernet/wiznet5k_dhcp        ubinascii
   units/_joystick_led
_boot                  libs/ethernet/wiznet5k_dns         ucollections
   units/_key
_flow                  libs/ethernet/wiznet5k_mqtt        ucryptolib
   units/_kmeter
```

```
_onewire          libs/ethernet/wiznet5k_ntp      uctypes           units/_laserrx
_thread           libs/ethernet/wiznet5k_socket   uerrno            units/_lasertx
_uasyncio         libs/ethernet/wiznet5k_wsgiserver uhashlib        units/_lcd
_webrepl          libs/imu        uhashlib        units/_light
apa106            libs/ir/ir_rx/__init__          uheapq            units/_limit
base64            libs/ir/ir_rx/nec uiflow        units/_makey
btree             libs/ir/ir_rx/print_error       uio               units/_microphone_AD
builtins          libs/ir/ir_tx/__init__          ujson             units/_microphone_I2S
cmath             libs/ir/ir_tx/nec umqtt/__init__ units/_mqtt_eth
collections/__init__              libs/lorabus    umqtt/robust      units/_ncir
collections/defaultdict           libs/m5_espnow  umqtt/simple      units/_oled
collections/deque libs/m5qtt      unit            units/_op
comx/LoRaWAN      libs/microcoapy/__init__        units/VFunction/__init__          units/_pahub
comx/__init__     libs/microcoapy/coap_macros     units/VFunction/_apriltag_code    units/_pbhub
deviceCfg         libs/microcoapy/coap_option     units/VFunction/_bar_code         units/_pir
display           libs/microcoapy/coap_packet     units/VFunction/_color_track      units/_relay
esp               libs/microcoapy/coap_reader     units/VFunction/_dm_code          units/_relay2
esp32             libs/microcoapy/coap_writer     units/VFunction/_face_detect      units/_relay4
espnow            libs/microcoapy/microcoapy      units/VFunction/_jpeg_transfer    units/_rfid
flashbdev         libs/mlx90640     units/VFunction/_line_tracker     units/_rgb
flow/__init__     libs/modbus/__init__            units/VFunction/_motion           units/_rgb_multi
flow/adaptation   libs/modbus/master/__init__     units/VFunction/_qr_code          units/_rotary_encoder
flow/ezdata       libs/modbus/master/uModBusConst units/VFunction/_tag_reader       units/_scales
flow/ezdata_queue libs/modbus/master/uModBusFunctions units/VFunction/_target_track units/_servo
flow/flowDeInit   libs/modbus/master/uModBusSerial units/VFunction/_v2_code_detector units/_sonic_io
flow/m5cloud      libs/modbus/master/uModBusTCP   units/VFunction/_v2_color_tracker units/_ssr
flow/m5ucloud     libs/modbus/slave/__init__      units/VFunction/_v2_face_detector units/_tcp_eth
flow/protocol     libs/modbus/slave/exceptions    units/VFunction/_v2_face_recognition              units/_thermal
framebuf          libs/modbus/slave/functions     units/VFunction/_v2_lane_line_tracker             units/_thermal_cam2
gc                libs/modbus/slave/redundancy_check units/VFunction/_v2_motion_tracker  units/_tof
hardware/__init__ libs/modbus/slave/route         units/VFunction/_v2_object_recognition            units/_tracker
hardware/_led     libs/modbus/slave/rtu           units/VFunction/_v2_online_classifier             units/_tube_pressure
hardware/axp192   libs/modbus/slave/utils         units/VFunction/_v2_shape_detector  units/_tvoc
hardware/bm8563   libs/mstate     units/VFunction/_v2_target_tracker  units/_uhf_rfid
hardware/button   libs/numbers    units/_ID       units/_ultrasonic
hardware/mpu6050  libs/nvs        units/_IR_NEC   units/_uwb
hardware/sh200q   libs/paj7620u2  units/_ISORS485 units/_v_function
hardware/speaker  libs/pca9685    units/_LoRaWAN  units/_vibrator
hat               libs/pid        units/_NBIoT    units/_vmeter
hats/_RS485       libs/power      units/_RS485    units/_watering
hats/_adc         libs/qmp6988    units/_RTC8563  units/_weight
hats/_balaC       libs/servo      units/_accel    units/_zigbee
hats/_beetlec     libs/sh1107     units/_acsocket uos
hats/_bugc        libs/sh200q     units/_adc      urandom
hats/_c_back_driver               libs/sht30      units/_ameter     ure
hats/_c_back_nbiot                libs/simcom/__init__              units/_angle      urllib/parse
hats/_cardKB      libs/simcom/common              units/_angle8     urllib/urequest
hats/_dac         libs/simcom/gps   units/_bps    uselect
hats/_dlight      libs/simcom/gsm   units/_button usocket
hats/_env         libs/simcom/lte   units/_buzzer ussl
hats/_env2        libs/simcom/nb    units/_cardKB ustruct
hats/_env3        libs/speak        units/_catm   utils
hats/_finger      libs/timeSchedule units/_catm_gnss utime
hats/_joyC        libs/time_ex      units/_co2_scd40 utimeq
hats/_joystick    libs/urequests    units/_color  uwebsocket
hats/_ncir        libs/v15310x      units/_dac    uzlib
hats/_pir         m5stack           units/_dds    v15310x
hats/_powerc      m5uart            units/_digi_clock warnings
hats/_puppy       m5ui              units/_dlight wav/chunk
hats/_roverc      machine           units/_dual_button                wav/wav_player
hats/_servo       math              units/_earth  wav/wave
hats/_servo8      max30100          units/_encoder_led                wifiCfg
hats/_servos      menu/__init__     units/_env    wifiWebCfg
hats/_speaker     menu/app          units/_env2
hats/_tof         menu/cloud        units/_env3
hats/_yun         menu/setup        units/_ext_io
Plus any modules on the filesystem
```

Module

Interessante Module:

- machine
- math
- os
- sys
- gc
- time

Auf den folgenden Folien werden wir einen Blick auf diese Module werfen.

Module

Das Modul **machine**.

Es enthält Funktionen zur Bedienung der Hardware des Microcontrollers. Es ist also maschinenspezifisch.

Dieses Module enthält auch die Klasse **Pin** die den Zugriff auf die Pins des ESP32 ermöglicht.

```
>>> 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']
```

Module

Die Methoden von **machine.Pin**:

```
>>> dir(machine.Pin)
['__class__', '__name__', 'value',
'__bases__', '__dict__', 'IN',
'IRQ_FALLING', 'IRQ_RISING',
'OPEN_DRAIN', 'OUT', 'PULL_DOWN',
'PULL_HOLD', 'PULL_UP', 'WAKE_HIGH',
'WAKE_LOW', 'init', 'irq', 'off',
'on']
```

Module

Einen GPIO-Pin ansteuern.

Im M5Stick C Plus ist die Led an Pin GPIO10 angeschlossen.

```
>>> import machine
>>> LEDintern = machine.Pin(10,machine.Pin.OPEN_DRAIN)

>>> LEDintern.value()
1
>>> LEDintern.value(0)    # LED leuchtet
>>> LEDintern.value()
0
>>> LEDintern.value(1)    # LED verlischt
>>> LEDintern.value()
1
>>>
```

Module

Die Methoden

machine.Pin.on()

und

machine.Pin.off()

```
>>> LEDintern.on()
>>> LEDintern.value()
1
>>> LEDintern.off()    # LED an
>>> LEDintern.value()
0
>>> LEDintern.on()     # LED aus
>>> LEDintern.value()
1
>>>
```

Module

Das Modul **math.**

Dieses Modul enthält weitere mathematische Funktionen.

```
>>> import math
>>> dir(math)
['__class__', '__name__', 'pow',
'acos', 'acosh', 'asin', 'asinh',
'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees',
'e', 'erf', 'erfc', 'exp', 'expm1',
'fabs', 'floor', 'fmod', 'frexp',
'gamma', 'isclose', 'isfinite',
'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log2', 'modf', 'pi',
'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'trunc']
>>>
>>> math.pi
3.141593
```

Module

Das Modul **os.**

Es enthält Funktionen für den Zugriff auf das Filesystem.

```
>>> import os
>>> dir(os)
['__class__', '__name__', 'remove',
'VfsFat', 'VfsLfs2', 'chdir',
'dupterm', 'dupterm_notify', 'getcwd',
'ilistdir', 'listdir', 'mkdir',
'mount', 'rename', 'rmdir', 'stat',
'statvfs', 'umount', 'uname',
'urandom']
>>>
```

Module

Das Modul **sys**.

Dieses Modul enthält Informationen zur Software.

```
>>> import sys
>>> dir(sys)
['__class__', '__name__', 'argv',
'byteorder', 'exit', 'implementation',
'maxsize', 'modules', 'path',
'platform', 'print_exception',
'stderr', 'stdin', 'stdout',
'version', 'version_info']
>>>
>>> sys.implementation
(name='micropython', version=(1, 12,
0), mpy=10757)
>>>
```

Module

Das Modul **gc**.

Diese Modul enthält Funktionen und Informationen zum Speicher.

```
>>> import gc
>>> dir(gc)
['__class__', '__name__', 'collect',
'disable', 'enable', 'isenabled',
'mem_alloc', 'mem_free', 'threshold']
>>>

>>> gc.mem_free()
66112

>>> x = '-'*1000
>>> gc.mem_free()
65088
>>> del(x)
>>> gc.collect()
>>> gc.mem_free()
66064
>>>
```

Module

Das Modul **time** bietet Funktionen zum Abrufen der aktuellen Uhrzeit und des Datums, zum Messen von Zeitintervallen und für Verzögerungen.

sleep(), **sleep_ms()** und **sleep_us()** lässt den Microcontroller für die angegebene Zeit schlafen.

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

Module

Ticks erzeugt einen Zähler, mit dem Zeiten gemessen werden können.

```
>>> import time
>>> start_time = time.ticks_ms()
>>> start_time
261312
>>> stop_time = time.ticks_ms()
>>> stop_time
279565
>>> time.ticks_diff(stop_time,
start_time)
18253
>>>
```

Module

Datum

- Jahr 4-stellig
- Monat 1-12
- Tag 1-31
- Stunde 0-23
- Minute 0-59
- Sekunde 0-59
- Wochentag 0-6 0=Montag
- Tag des Jahres 1-366

```
>>> time.time()
717596621
>>> time.localtime(time.time())
(2022, 9, 27, 12, 24, 29, 1, 270)
>>>
```

Module

Informationen zu den Standardmodulen von Micropython sind unter

https://docs.micropython.org/en/v1.12/library/index.html

zu finden.

Module

Module importieren.

Damit Module verwendet werden können müssen zuerst importiert werden. Dazu gibt es mehrere Methoden mit unterschiedlichen Auswirkungen.

Module haben ihren eigenen **namespace**.

Module

Die einfache Methode.

Die Funktionen des Moduls liegen im **namespace** des Moduls.

```
>>> import time
>>> a = ticks_ms()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ticks_ms' isn't defined
>>> a = time.ticks_ms()
>>> a
883706
>>>
>>> import time as t
>>> a = t.ticks_ms()
>>> a
1069923
>>>
```

Module

Die gefährliche Methode 1

Sie bindet die Funktionen des Moduls in den **namespace** des Hauptprogramms ein.

Dabei kann es zu Namenskonflikten kommen.

```
>>> from time import *
>>> a = ticks_ms()
>>> a
1314660
>>>
```

Module

Die gefährliche Methode 2

Es werden nur ausgewählte Funktionen importiert.

Es gilt das auf der vorherigen Folie dargestellte. Allerdings sind hier die Namen der importierten Funktionen bekannt.

```
>>> from time import ticks_ms
>>> a = ticks_ms()
>>> a
77908
>>> a = ticks_cpu()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ticks_cpu' isn't defined
>>>
```

Module

Die universelle Methode.

Hier kann der Name der importierten Funktion geändert werden.

```
>>> from time import ticks_cpu as t_cpu
>>> a = t_cpu()
>>> a
7187425
>>>
```

Module

Die in der Firmware enthaltenen Module stehen ohne weitere Maßnahmen zur Verfügung.

Externe Module müssen in das Filesystem des M5Stick geladen werden.

Am Besten legt man sie in den selben Ordner in dem auch das Programm liegt.

Man kann auch einen eigenen Ordner anlegen und diesen in den Suchpfad eintragen.