Micropython-Stubs

Release 1.0

Jos Verlinde

Jan 01, 2023

CONTENTS:

CHAPTER

ONE

MICROPYTHON-STUBS

This repo stores stubs generated by the MicroPython-Stubber tool. Currently over 3.000 stubfiles of multiple MicroPython versionas and modules are available to help you :

- write code quicker
- with less errors,
- get help from code completion,
- use static type checking,
- and improve the overall development experience while writing MicroPython.

The stubs are packaged and published to PyPi as Pep 561 stub-only packages to allow them to be installed and used with ease.

Demo using VSCode:

1.1 Installable stub packages

The installable stub packages are created by merging and assembling different types of stubs that are generated using different means. Each method of generation has its own advantages, and by combining and merging these in a mostly automated manner allows the creation of accurate and up-to-date stub packages for the different ports and board supported by MicroPython. As the recent versions MicroPython has over ??38?? different boards, maintaining the stubs manually is not really an option.

The goal is to publish and maintain stub packages for all ports of the last 4 published MicroPython versions on PyPi, and provide packages for the pre-release version (MicroPython master branch) as a pre-release. Currently the pre-release packages are not (yet) published to PyPi

See the list of *packages* for an overview of the available packages.

1.2 Stub Types

The blow stub types are referenced in the documentation and can be considered half-products that are used to generate the stub packages.

1. **Board stubs:** This type of stubs that is generated on a MicroPython board with a specific firmware, and therefore **very closely matches** the capabilities of your board and firmware. However board stubs do no provide information on the expected parameters or the types returned by functions and methods. While they have they have a low level of detail, they do contain a comprehensive overview of the modules, classes and functions available even on custom and one-off firmwares.
   Location:`'stubs/micropython-{Version}-{Port}[-{Board}]'`
2. **Frozen stubs:**
   The firmware for most boards has a number of Python modules Frozen (included) as part of the firmware. For each port and bord there is a manifest file that specified which modules should be included in the firmware. These modules have been collected, and stubs have been generated for these modules using mypy stubgen. Frozen stubs provider better information regarding the expected parameters, but If you select the correct **Port** and **Board** you should get results that exactly match your firmware.
   Location:`'stubs/micropython-{Version}-Frozen/{Port}/{Board}'`
3. **Doc stubs:**
   Documentation stubs are generated based on the MicroPython formal documentation `.rst` source files. The `.rst` files are parsed and used to build stubs that match the documentation for that specific version. This type of stub is very rich in parameter and class descriptions, but as they are generic by definition, they may/will not follow the specifics of a port or board firmware.
   Location:`'stubs/micropython-{Version}-docstubs'`
4. **Merged stubs:** In order to combine the precision of the board stubs with the richness of the docstubs, the two are merged. The merge process add the 'doc strings' parameters and return types to functions, classes and methods. The resulting stubs are both precise and rich, however may still lack some details, or contain errors due to lacking or incorrect documentation.
   Location:`'stubs/micropython-{Version}-{Port}[-{Board}]-merged'`
5. **MicroPython core stubs:** In some cases it is needed to provide an override for classes and functions that cannot yet be automatically generated reliably from the documentation, and need to be augmented manually. Currently this contains a single module stub `micropython.pyi`
   Location:`'stubs/micropython[-{Version}]-core'`
6. **MicroPython stdlib packages** MicroPython implements a number of stdlib packages different functionality and functions. Examples of this are:
   - `gc` has additional functions such as `gc.mem_free()`
   - `time` has additional functions such as `time.sleep_ms()`
   While on one hand these are just modules, tools and linters often use a different logic to locate stubs for these stdlib modules, or have a built-in copy of the mypy stdlib stubs that they use by default. in order to allow such tools to locate these, a copy of these 'MicropyPython stdlib' modules is included in the stub packages in the 'stdlib' module/folder.

CHAPTER

TWO

USING THE MICROPYTHON STUBS

There are a few different options in which you can install the stubs,

The logical steps are:

1. Determine the MicroPython version, port and board you will be using the stubs for.
2. Create and activate a virtual environment
3. Install or copy the stubs to your system
4. Configure your IDE (or other tools) where the stubs are located
5. Add configuration to suppress false positives and unneeded warnings

At minimum you will need to specify the **port** of the stubs. If you do not specify a version, the stubs for the last published version will be used. (Note that this is different from the ***latest version***)

2.1 Determine the version and port

If you do not know the exact version and port, run the below command in MicroPython `import sys; print( "version:", sys.version, "port:", sys.platform)` In the documentation these will be referred to as **version**, **port** and **board**

2.2 Create and activate a venv

A Python virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use. To create and activate a virtual environment in your project directory `bash # linux / mac python3 -m venv .venv source .venv/bin/activate bash # windows python -m venv .venv .venv\Scripts\Activate.ps1` *While it is possible to install the MicroPython stubs in your global environment it is better to use virtual environments or to a* `typings` *folder.* If you install the MicroPython stubs in your general or global python environment, then please note that this may/will cause regular Python code to also be validated against the MicroPython stubs, and this will almost certainly result in false-positive errors being flagged in your CPython code.

2.3 Install the stub packages to your system

- Install a stub-package into a virtual environment
- Install a stub package into a `typings` folder
- Legacy method using a copy or clone

2.3.1 Install into a `venv`

Last published version

The package naming convention is: `micropython-<port>[-<board>]-stubs` where port is the port of the MicroPython firmware. (stm32, eps32,rp2, samd, . . .)

To install the stubs for the last published version of MicroPython:

```
pip install -U  micropython-<port>-stubs
pip install -U  micropython-stm32-stubs
```

Install stubs for a specific version.

To install the stubs for an older version, such as MicroPython 1.18:
specify the version as follows ** `micropython-<port>-stubs==<version>.*` **

```
pip install -U  micropython-<port>-stubs==<version>.*
pip install -U  micropython-esp32-stubs==1.18.*
```

Install stubs for a specific board.

To install the stubs for a specific board, such as the `ESP32 UM-TinyPico`:
specify both the port and the board

```
pip install -U micropython-<port>-<board>-stubs
pip install -U micropython-esp32-um-tinypico-stubs
```

Notes:

- PyPi transforms all names of the ports and boards to small-caps and kebab-case, (not snake_case).
- Not all possible ports/boards are published as I do not have access to hardware to create board-stubs for all ports and boards.
- Newly published stubs may show as ‘not found’, please check PyPi directly

2.3.2 Install into a `typings` folder.

In some cases a single project my need to make use of stubs for different ports or boards at the same time. In a venv it is possible to install only **one** stub package.

Some tools such as Pylance and Pyright can also make use of stubs that are located in a folder, usually a folder named `typings` Another advantage of using a typings folder is that it can be checked in to a source code repo

In order to install the stubs into a typings folder append `--target <folder> --no-user` to the pip install commands listed in the above sections. `--target` specifies the destination folder `--no-user` is only needed to avoid conflicts with an active venv

To install the stubs for the last published version of MicroPython:

```
pip install -U  micropython-<port>-stubs --target <folder> --no-user
pip install -U  micropython-stm32-stubs --target ./typings --no-user
```

or

```
pip install -U  micropython-esp32-stubs==1.18.* --target ./typings --no-user
```

2.4 Configure your IDE (or other tools) where the stubs are located

the configuration for your IDE or tool set is specific to that IDE or tool, however there will be some commonalities. Most tools will :

- Be able to use the stubs if they are located in the active virtual environment
- If not using a `venv` you will need to configure the tool with the location, unless it uses `typings` as the default location.
- can be configured with a python language version. MicroPython is based on Python 3.5 with some features from later versions.

CHAPTER

THREE

VSCODE AND PYLANCE

VSCode uses Pylance, and optionally a linter such as pylint or mypy.

3.1 Install the stubs from PyPi.

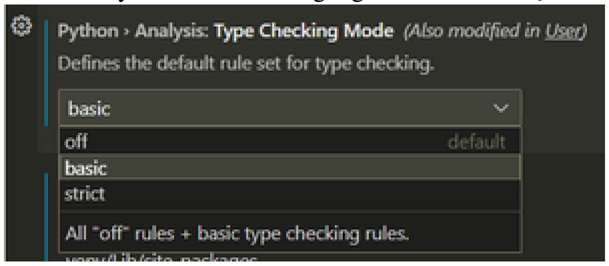
`pip install -U micropython-<port>-stubs` For details see *Using stubs*

3.2 Configure VSCode & Pylance.

VSCode allows the configuration to be set on ***workspace*** , folder or *user* level. I prefer setting it per workspace or folder as that allows different settings for different projects, but you could do either.

3.2.1 Install the Python and Pylance extensions.

1. Install the Python extension from the marketplace. Pylance will be installed as an optional extension.
2. By default the Pylance checking is sset to `Off` and the language server is set to `Default` I recommend you set the language server to `Pylance` and the checking to `basic` (or `strict`)



Python › Analysis: Type Checking Mode (Also modified in User)
Defines the default rule set for type checking.
basic
off default
basic
strict
All "off" rules + basic type checking rules.

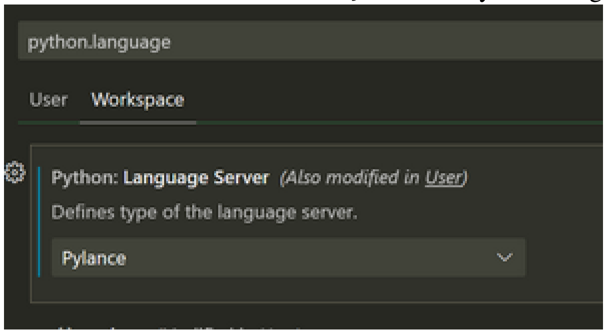
3. Open a Python (.py) file and the Pylance extension will activate.

3.2.2 Select the correct Python environment.

If you have created a `venv` make sure to also select it in VSCode using `F1, >Python: select interpreter` or the UX

3.2.3 Set Pylance as the language Server.

Note: If you've previously set a language server and want to try Pylance, make sure you've set `"python.languageServer": "Default" or "Pylance"` in your settings.json file using the text editor, or using the Settings Editor UI.



python.language

User Workspace

Python: Language Server *(Also modified in User)*

Defines type of the language server.

Pylance

Example from `.vscode/settings.json`

```
{
    "python.languageServer": "Pylance",
    "python.analysis.typeCheckingMode": "basic",
}
```

3.3 Add configuration to suppress unneeded warnings.

After installing the stubs you may see some warnings that the source code to referenced modules is not found.

```
Import "machine" could not be resolved from source
Import "time" could not be resolved from source
Import "urequests" could not be resolved from source
```

This is because the packages do not include the source code, as it are stub-only packages.

To supress these warnings add the following to youryour VSCode configuration.

`.vscode/settings.json`

```
    "python.analysis.diagnosticSeverityOverrides": {
        "reportMissingModuleSource": "none"
    },
```

3.4 Configure Pylance to read MicroPython stdlib stubs.

Pylance and Pyright do not by default allow you to override the stdlib stubs. This is possible but needs to be configured explicitly in the settings.

the VSCode configuration for this is shown below.

`.vscode/settings.json`

```
{
    "python.analysis.typeshedPaths": [
        ".venv/Lib/site-packages",
        "typings"
    ],
}
```

The diagram below shows the sequence of checks that Pylance/Pyright does to resolve the stubs for a module. with the above configuration it will first check the `venv` or the `typings` folder and then look for the stdlib stubs in these folders. Without the configuration it will only look for the stdlib stubs in the typeshed stubs that are shipped with Pyright.

3.5 Sample VSCode configuration file.

The below configuration combines the above settings.

- Enable Pylance and set basic checking
- Suppress warnings about missing source code
- Configure Pylance to read MicroPython stdlib stubs

`.vscode/settings.json`

```
{
    "python.languageServer": "Pylance",
    "python.analysis.typeCheckingMode": "basic",
    "python.analysis.diagnosticSeverityOverrides": {
        "reportMissingModuleSource": "none"
    },
    "python.analysis.typeshedPaths": [
        ".venv/Lib/site-packages"
    ],
    "python.linting.enabled": true,
    "python.linting.pylintEnabled": true,
}
```

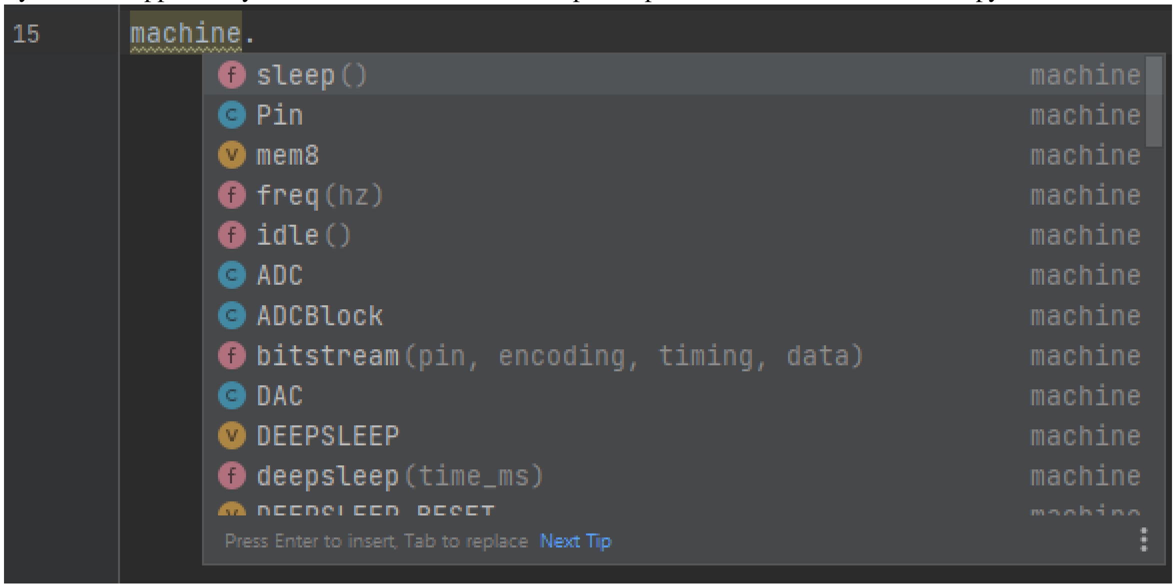
CHAPTER FOUR

PYCHARM

4.1 Configure PyCharm to use the selected stub folders

ref: https://www.jetbrains.com/help/pycharm/type-hinting-in-product.html#stub-type-hints

PyCharm supports Python stub files, so the simples option is to install the micropython-stubs from PyPi.



```
machine.
```

ref: https://www.jetbrains.com/help/pycharm/type-hinting-in-product.html#stub-type-hints

4.2 Install the stubs from PyPi.

Install the stubs as documented in *using the thubs*

Example: `pip install -U micropython-stm32-stubs==1.19.1.*`

if you have a requirements.txt file you can add the stubs to it, and PyCharm will offer to install them automatically.

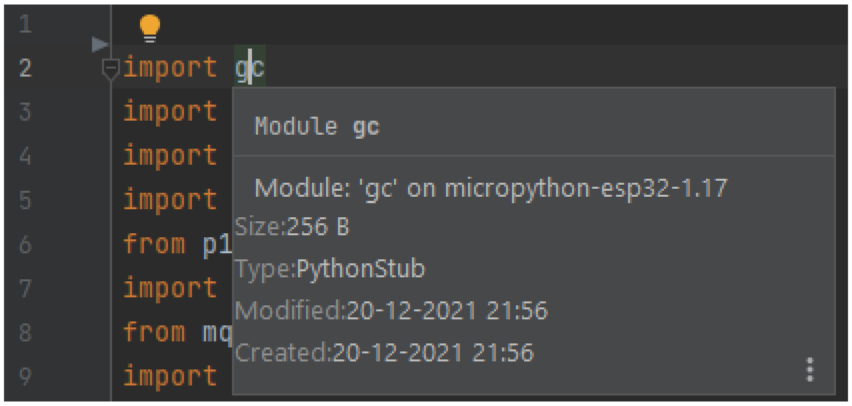
```
micropython-stm32-stubs==1.19.1.*
```

After this

4.3 legacy installation for PyCharm

4.4 Check library imports

To check if the correct types are used for your imports you can 'hover' the mouse over the module of an import statement. Pycharm will show the module's docstring that will allow you to identify which stub is being used.



```
import gc
import
import
import
from p1
import
from mq
import
```

Module gc

Module: 'gc' on micropython-esp32-1.17
Size:256 B
Type:PythonStub
Modified:20-12-2021 21:56
Created:20-12-2021 21:56

4.4.1 Disable Pycharm warnings for RP2 PIO code

As the RP2 PIO code is not valid python code, PyCharm will show muliple warning for the code. To disable these warnings, add the following line to the top of the file or to the top of the function:

```
# noinspection PyStatementEffect,PyArgumentList
@rp2.asm_pio(set_init=rp2.PIO.OUT_LOW)
def blink_1hz():
    # Cycles: 1 + 7 + 32 * (30 + 1) = 1000
    set(pins, 1)
    set(x, 31)[6]
    label("delay_high")
    nop()[29]
    jmp(x_dec, "delay_high")

    # Cycles: 1 + 7 + 32 * (30 + 1) = 1000
    set(pins, 0)
    set(x, 31)[6]
    label("delay_low")
    nop()[29]
    jmp(x_dec, "delay_low")
```

4.5 Legacy Option: Clone the stubs repo

Older method of installation

To use stubs from the micropython-stubs repository , follow these steps:

Copy some or all the stubs into a directory in your project, or use a symlink to a clone of the stubs.

- Mark the relevant directories as a source root by choosing **Mark Directory as | Sources Root** from the context menu of the directory.
  For example:
  - all-stubs/cpython_core-pycopy
  - all-stubs/micropython-v1_17-frozen/esp32/GENERIC
  - all-stubs/micropython-v1_17-esp32

You should now be able to use code completion and typechecking for your micropython code in PyCharm

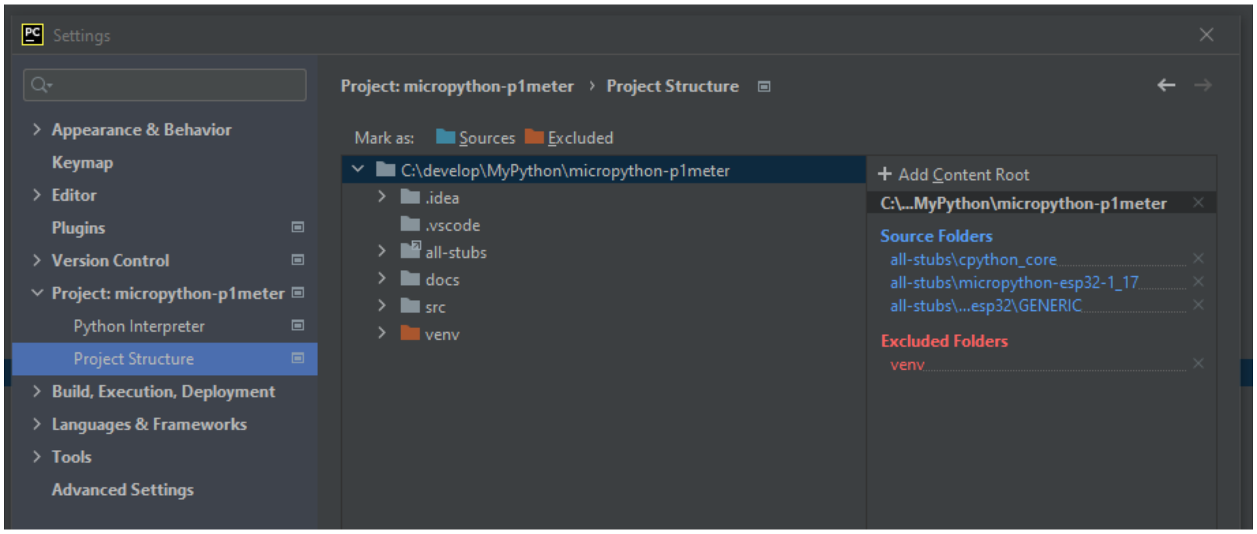
4.5.1 Verify the paths

You can verify the paths used in your project by

File > Settings > Project Settings

  Project Structure

This should list the selected folders with stubs as Source Folders



C:\develop\MyPython\micropython-p1meter
.idea
.vscode
all-stubs
docs
src
venv

C:\...MyPython\micropython-p1meter

Source Folders
all-stubs\cpython_core
all-stubs\micropython-esp32-1_17
all-stubs\...esp32\GENERIC

Excluded Folders
venv

CHAPTER

FIVE

PYLINT

5.1 Pylint dos not support stub-only packages.

Unfortunatly Pylint does not (yet) support the use of stubs int the `.pyi` format. Please refer to Being able to use stub pyi files in pylint for more information.

5.2 Pylint:Legacy configuration.

Note: The below configuration for pylint will only work if the stub folders you are using contain `.py` files.

- Configure pylint to use the selected stub folders

This instructs pylint to insert the list of paths into `sys.path` before performing linting, thus allowing it to find the stubs and use them to better validate your code.

- use the `.pylintcr sample file`.
- edit the line that starts with `init-hook=`
  ```
  init-hook='import sys;sys.path[1:1] = ["src/lib", "folder1","folder2", "folder3",];
  '
  ```
- replace the folders with your selection of stub folders.
  Note that in `.pylintrc` the list of folders MUST be on a single line
- the result should look like:
  ```
  init-hook='import sys;sys.path[1:1] = ["src/lib", "all-stubs/cpython_core-pycopy",
  "all-stubs/micropython-v1_17-frozen/esp32/GENERIC", "all-stubs/
  micropython-v1_17-esp32",];'
  ```

CHAPTER

SIX

MYPY - LIMITED FUNCTIONALITY

This section is preliminary and needs to be expanded. While MyPy used to work quite well with the .PY stubs, it is now not working anymore.

6.1 MyPy and stub-only packages installed in a virtual environment

MyPy should automatically detect and use all Micropython packages installed in a virtual environment as they follow the PEP-561 standard for stub-only packages. However MyPy fails to detect the stubs installed in a `venv` :-(

This will need to be investigated in more detail, and possibly an issue to be raised with MyPy. In addition it has not been tested if MyPy is able to detect and allow an override by the MicroPython stdlib packages.

6.2 MyPy and stub-only packages installed in a `typings` folder

MyPy can be configured to use stubs located in a folder, usually a folder named `typings` for details see: https://mypy.readthedocs.io/en/stable/stubs.html

Although the documentation also states that stub-only packages cannot be located though a provided path this seems to work on intitial testing, with a few workarounds neede. (possibly the MyPy detecttion or definiton of a stub-only package does not flag the microspython-stub packages as stub-only packages)

Therefore after installing the stubs into a `typings` folder, MyPy can be configured to use the stubs by setting the `MYPYPATH` environment variable to the path of the `typings` folder.

Linux/MacOS:

```
export MYPYPATH=./typings
```

Windows:

```
$env:MYPYPATH="./typings"
```

6.2.1 Workarounds for some mypy warnings and errors

```
mypy: "typings\sys.pyi" shadows library module "sys"
note: A user-defined top-level module with name "sys" is not supported
```

Partial workaround:

```
del typings/os.pyi
del typings/sys.pyi
```

CHAPTER

SEVEN

LEGACY INSTALLATION

Prior to publish the stub packes to PyPi, the only option was to copy or clone the entire repository to a local drive and reference the different stub types from there. that method has been depricated, but I wanted to keep the documentation just in case that is still useful for someone.

However there are still a number of stubs that are in the repo and that are not published on PyPI. For these cases please follow the below instructions, and if you would like to see them on PyPI as well, please let me know in the [Discussions][].

Note however that I will not be activly maintaining these pages.

7.1 Clone or download a copy of the Micropython-Stubs repo

1. Download a copy of this repo , either via `git clone` or by download a zip file with it's contents

    - store this in a folder, for example 'next to' your software projects such as in `c:\develop\micropython-stubs`
      this contains a `stubs` folder that contains all stubs

      ```
      git clone https://github.com/Josverl/micropython-stubs.git
      ```

2. Over time you may want to periodically update this folder using

   ```
   git fetch && git pull
   ```

7.2 Project configuration

For each project where you want to use the stubs you need to configure vscode and any linters that you use to use the correct stubs. This is not as complex as it seems initially, and once you have configured your first project, you can copy /paste that structure for other projects.

7.3 Create a symlink to the stubs folder

Create a symlink to the `c:\develop\micropython-stubs\stubs` from inside your project. This will allow you to reference the same stub files from multiple projects, and limit the space needed. This a recommendation, and things work equally well if you copy or clone the `stubs` folder into your project.
For details on how to create a symlink, please see : *Create a symbolic link*

7.4 Quick start:

Copy the [samples][] folder to your project
This contains the template files you need to improve syntax highlighting with Pylance and linting with pylint.

7.5 Select which stub folders you need to reference

- The order will influence results. place the 'higher quality' folders first.
- Use forward slashes `/` rather than backslashes, also on Windows.
- for example for micropython 1.17 on an ESP32 select:
  1. "./src/lib",
  2. "all-stubs/cpython_core-pycopy",
  3. "all-stubs/micropython-v1_17-frozen/esp32/GENERIC",
  4. "all-stubs/micropython-v1_17-esp32",

7.5.1 Clone the stubs repo

Note is the 'olde way' of installing the stubs. and it is mostly useful if you are activly developing / updating the stubs.

7.6 Configure VSCode & Pylance to use the selected stub folders

This instructs the VSCode Pylance extension to consider your libs folder and the stubs for code completion and static type-checking.

VSCode allows this configuration to be set on ***workspace*** , folder or *user* level. I prefer setting it per workspace or folder as that allows different settings for different projects, but you could do either.

The below configuration is [Pylance][] specific

- use the `samples/.VSCode/settings.json` located in the sample folder
- you can open this file in VSCode itself, or use the settings menu
- add the folders to the `python.analysis.extraPaths` section.
- it can be on a single line or split across lines.
  - make sure it is a valid json array

Example from `.vscode/settings.json`

```
    "python.languageServer": "Pylance",
    "python.analysis.autoSearchPaths": true,
    "python.analysis.extraPaths": [
        "src/lib",
        "all-stubs/cpython_core-pycopy",
        "all-stubs/micropyton-1_17-frozen/esp32/GENERIC",
        "all-stubs/micropyton-1_17-esp32",
    ],
    "python.linting.enabled": true,
    "python.linting.pylintEnabled": true,
```

7.7 legacy resolving order

7.8 Restart VSCode

VSCode must be restated for so that Pylance and linters such as Pylint to read the updated configuration.

You can press `F1` and select

- `Python:  Restart language server`
- or `Developer:  Reload Window` command.
- or stop / start the editor

Note: Pymakr: Update pymakr.conf

Depending on the tools and configuration you are using it may be needed to exclude the To avoid the "all-stubs" folder to be uploaded to your Micropython MCU

- add "all-stubs" to the "py_ignore" section

```
{
   "address": "192.168.4.1",
   "username": "micro",
   "password": "python",
   "sync_folder": "",
   "open_on_start": true,
   "safe_boot_on_upload": false,
   "py_ignore": [
       "all-stubs",
       "pymakr.conf",
       ".vscode",
       ".gitignore",
       ".git",
       "project.pymakr",
       "env",
       "venv",
       ".venv"
   ],
   "fast_upload": false
}
```

7.9 Order of the stub folders

The stubs can be used by different components in your development environment.

1. the VSCode Pylance Language Server
2. the VSCode Python add-in
3. and optionally by an additional Python linter such as pylint or mypy.

These tools work together to provide code completion/prediction, type checking and all the other good things. For this the order in which these tools use the stub folders is significant, and best results are when they use the same order. (Note that the different tools will not always agree, MyPy might show a warning where PyLance understands the intent of your code, and vice-versa)

In most cases the best results are achieved by the below setup:

CHAPTER

EIGHT

CREATE A SYMBOLIC LINK

To create the symbolic link to the `micropython-stubs/stubs` folder the instructions differ slightly for each OS/ The below examples assume that the micropython-stubs repo is cloned 'next-to' your project folder. please adjust as needed.

8.1 Windows 10 / 11

Windows 10 and later requires **Developer Mode** to create symlinks as a regular user, or you can run the below commands from an elevated PowerShell prompt (Run as Admin) .

Settings > Developer Settings > Developer Mode : On

Please see Activate Developer Mode for instruction on how to activate this. Or read the blogpost on the rationale and see some examples.

You can create a symlink using PowerShell or with the mklink cmdline tool. Both methods achieve exactly the same result.

8.1.1 Powershell

```
New-Item -ItemType SymbolicLink -Path "all-stubs" -Target (Resolve-Path -Path ../
↪micropython-stubs/stubs)
```

Note: The target must be an absolute path.
Therefore `Resolve-Path` is used to resolve the relative path to an absolute path.

8.1.2 mklink

or use mklink in an (elevated) command prompt.

```
mklink /d all-stubs c:\develop\micropython-stubs\stubs
```

Note: The `mklink` target (last path) must be an absolute path

8.2 Linux/Macos/Unix

```
# target must be an absolute path
ln -s /path/to/micropython-stubs/stubs all-stubs
```

Note: Teh target must be an absolute path

CHAPTER

NINE

LIST OF AVAILABLE PACKAGES

9.1 Packages on PyPi

Overview of the published ports and boards.

-
-
-
-
-

Notes:

- PyPi transforms all names of the ports and boards to small-caps and kebab-case, (not snake_case).
- Not all possible ports/boards are published as I do not have access to hardware to create board-stubs for all ports and boards.
- Newly published stubs may show as 'not found', please check PyPi directly

TODO:

- currently not yet automatically updated
- add links to PyPi

9.2 Packages on GitHub

- list of (latest) packages that can be installed from GitHub
- add commands to install TODO:
- currently not yet automatically updated

9.3 List of current firmwares and stubs

This includes stubs from the following MicroPython families:

- *MicroPython*
- *Pycopy*
- M5Stack
- EV3/Lego
- Loboris port (ESP32) - Available but no longer maintained

and Micropython Modules:

- All frozen MicroPython modules
- the LVGL GUI libraries
- *ulab native modules*

An up-to-date list of all current Firmwares, ports and boards is listed on the ***Firmwares page***

CHAPTER

TEN

STUBS BY FAMILY AND VERSION

10.1 Micropython

family	version	type	port	board	count	stubber version
micropython	latest	documentation	-	micropython	50	1.8.0
micropython	latest	frozen	esp32	GENERIC	24	1.9.12
micropython	latest	frozen	esp32	LILYGO_TTGO_LORA32	27	1.9.12
micropython	latest	frozen	esp32	LOLIN_C3_MINI	25	1.9.12
micropython	latest	frozen	esp32	LOLIN_S2_MINI	25	1.9.12
micropython	latest	frozen	esp32	LOLIN_S2_PICO	27	1.9.12
micropython	latest	frozen	esp32	M5STACK_ATOM	25	1.9.12
micropython	latest	frozen	esp32	UM_FEATHERS2	26	1.9.12
micropython	latest	frozen	esp32	UM_FEATHERS2NEO	25	1.9.12
micropython	latest	frozen	esp32	UM_FEATHERS3	25	1.9.12
micropython	latest	frozen	esp32	UM_PROS3	25	1.9.12
micropython	latest	frozen	esp32	UM_TINYPICO	26	1.9.12
micropython	latest	frozen	esp32	UM_TINYS2	25	1.9.12
micropython	latest	frozen	esp32	UM_TINYS3	25	1.9.12
micropython	latest	frozen	esp8266	GENERIC	27	1.9.12
micropython	latest	frozen	esp8266	GENERIC_512K	10	1.9.12
micropython	latest	frozen	mimxrt	GENERIC	11	1.9.12
micropython	latest	frozen	nrf	arduino_nano_33_ble_sense	11	1.9.12
micropython	latest	frozen	nrf	GENERIC	8	1.9.12
micropython	latest	frozen	renesas-ra	GENERIC	10	1.9.12
micropython	latest	frozen	renesas-ra	RA4M1_CLICKER	1	1.9.12
micropython	latest	frozen	renesas-ra	RA4M1_EK	1	1.9.12
micropython	latest	frozen	rp2	ARDUINO_NANO_RP2040_CONNECT	30	1.9.12
micropython	latest	frozen	rp2	GENERIC	14	1.9.12
micropython	latest	frozen	rp2	PICO_W	17	1.9.12
micropython	latest	frozen	rp2	WEACTSTUDIO	15	1.9.12
micropython	latest	frozen	samd	GENERIC	11	1.9.12
micropython	latest	frozen	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
micropython	latest	frozen	stm32	GARATRONIC_PYBSTICK26_F411	1	1.9.12
micropython	latest	frozen	stm32	GENERIC	9	1.9.12
micropython	latest	frozen	stm32	LEGO_HUB_NO6	12	1.9.12
micropython	latest	frozen	stm32	LEGO_HUB_NO7	12	1.9.12
micropython	latest	frozen	stm32	PYBD_SF2	12	1.9.12
micropython	latest	frozen	stm32	PYBV10	10	1.9.12

Table 1 – continued from previous page

family	version	type	port	board	count	stubber version
micropython	latest	frozen	teensy	GENERIC	2	1.9.12
micropython	latest	frozen	unix	GENERIC	11	1.9.12
micropython	latest	frozen	windows	GENERIC	7	1.9.12
micropython	v1.10	board	esp32	ESP32 module with ESP32	64	1.5.4
micropython	v1.10	frozen	esp32	micropython	16	1.8.0
micropython	v1.10	frozen	esp8266	micropython	15	1.8.0
micropython	v1.10	board	pyboard	PYBv1.1 with STM32F405RG	50	1.5.4
micropython	v1.10	frozen	stm32	micropython	4	1.8.0
micropython	v1.10	frozen	unix	micropython	2	1.8.0
micropython	v1.10-La	board	-	ESP module with ESP8266	67	1.1.0
micropython	v1.11	board	esp32	ESP32 module with ESP32	64	1.5.4
micropython	v1.11	frozen	esp32	micropython	16	1.8.0
micropython	v1.11	frozen	esp8266	micropython	15	1.8.0
micropython	v1.11	frozen	stm32	micropython	4	1.8.0
micropython	v1.11	frozen	unix	micropython	2	1.8.0
micropython	v1.11-La	board	-	ESP module with ESP8266	65	1.1.0
micropython	v1.12	frozen	common	GENERIC	2	1.7.4
micropython	v1.12	board	esp32	ESP32 module (spiram) with ESP32	66	1.5.4
micropython	v1.12	frozen	esp32	GENERIC	14	1.7.4
micropython	v1.12	frozen	esp32	RELEASE	18	1.7.4
micropython	v1.12	frozen	esp32	TINYPICO	16	1.7.4
micropython	v1.12	frozen	esp8266	GENERIC	15	1.7.4
micropython	v1.12	frozen	esp8266	RELEASE	29	1.7.4
micropython	v1.12	board	pyboard	PYBv1.1 with STM32F405RG	51	1.5.4
micropython	v1.12	frozen	stm32	GENERIC	4	1.7.4
micropython	v1.12	frozen	stm32	PYBD_SF2	7	1.7.4
micropython	v1.13	board	-	PYBv1.1 with STM32F405RG	47	1.3.4
micropython	v1.13	board	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
micropython	v1.13	board	esp32	ESP32 module (spiram) with ESP32	73	1.5.0
micropython	v1.13	frozen	esp32	GENERIC	21	1.8.0
micropython	v1.13	frozen	esp32	RELEASE	25	1.8.0
micropython	v1.13	frozen	esp32	TINYPICO	23	1.8.0
micropython	v1.13	frozen	esp8266	GENERIC	22	1.8.0
micropython	v1.13	frozen	esp8266	RELEASE	27	1.8.0
micropython	v1.13	frozen	stm32	GENERIC	11	1.8.0
micropython	v1.13	frozen	stm32	PYBD_SF2	14	1.8.0
micropython	v1.14	board	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
micropython	v1.14	frozen	esp32	GENERIC	21	1.8.0
micropython	v1.14	frozen	esp32	RELEASE	25	1.8.0
micropython	v1.14	frozen	esp32	TINYPICO	23	1.8.0
micropython	v1.14	board	esp8266	ESP module with ESP8266	73	1.5.4
micropython	v1.14	frozen	esp8266	GENERIC	28	1.8.0
micropython	v1.14	frozen	stm32	GENERIC	11	1.8.0
micropython	v1.14	frozen	stm32	PYBD_SF2	14	1.8.0
micropython	v1.15	board	esp32	ESP32 module (spiram) with ESP32	72	1.5.4
micropython	v1.15	frozen	esp32	GENERIC	21	1.8.0
micropython	v1.15	frozen	esp32	RELEASE	25	1.8.0
micropython	v1.15	frozen	esp32	TINYPICO	23	1.8.0
micropython	v1.15	board	esp8266	ESP module with ESP8266	73	1.5.4

Table 1 – continued from previous page

family	version	type	port	board	count	stubber version
micropython	v1.15	frozen	esp8266	GENERIC	28	1.8.0
micropython	v1.15	board	pyboard	PYBv1.1 with STM32F405RG	58	1.5.4
micropython	v1.15	frozen	stm32	GENERIC	11	1.8.0
micropython	v1.15	frozen	stm32	PYBD_SF2	14	1.8.0
micropython	v1.16	board	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
micropython	v1.16	frozen	esp32	GENERIC	21	1.8.0
micropython	v1.16	frozen	esp32	M5STACK_ATOM	22	1.8.0
micropython	v1.16	frozen	esp32	RELEASE	25	1.8.0
micropython	v1.16	frozen	esp32	UM_FEATHERS2	23	1.8.0
micropython	v1.16	frozen	esp32	UM_TINYPICO	23	1.8.0
micropython	v1.16	frozen	esp32	UM_TINYS2	22	1.8.0
micropython	v1.16	board	esp8266	ESP module with ESP8266	73	1.5.4
micropython	v1.16	frozen	esp8266	GENERIC	28	1.8.0
micropython	v1.16	frozen	esp8266	GENERIC_512K	11	1.8.0
micropython	v1.16	frozen	mimxrt	GENERIC	10	1.8.0
micropython	v1.16	board	pyboard	PYBv1.1 with STM32F405RG	58	1.5.4
micropython	v1.16	frozen	rp2	GENERIC	11	1.8.0
micropython	v1.16	frozen	stm32	GENERIC	11	1.8.0
micropython	v1.16	frozen	stm32	PYBD_SF2	14	1.8.0
micropython	v1.17	documentation	-	micropython	47	1.9.12
micropython	v1.17	board	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
micropython	v1.17	frozen	esp32	GENERIC	21	1.9.12
micropython	v1.17	frozen	esp32	M5STACK_ATOM	22	1.9.12
micropython	v1.17	frozen	esp32	UM_FEATHERS2	23	1.9.12
micropython	v1.17	frozen	esp32	UM_TINYPICO	23	1.9.12
micropython	v1.17	frozen	esp32	UM_TINYS2	22	1.9.12
micropython	v1.17	board	esp8266	ESP module with ESP8266	73	1.5.4
micropython	v1.17	frozen	esp8266	GENERIC	29	1.9.12
micropython	v1.17	frozen	esp8266	GENERIC_512K	11	1.9.12
micropython	v1.17	frozen	mimxrt	GENERIC	10	1.9.12
micropython	v1.17	frozen	nrf	GENERIC	1	1.9.12
micropython	v1.17	board	pyboard	PYBv1.1 with STM32F405RG	59	1.5.4
micropython	v1.17	frozen	rp2	GENERIC	11	1.9.12
micropython	v1.17	board	rp2	Raspberry Pi Pico with RP2040	51	1.5.2
micropython	v1.17	frozen	stm32	GENERIC	11	1.9.12
micropython	v1.17	frozen	stm32	PYBD_SF2	14	1.9.12
micropython	v1.17	frozen	unix	GENERIC	9	1.9.12
micropython	v1.18	documentation	-	micropython	49	1.10.0
micropython	v1.18	board	esp32	ESP32 module (spiram) with ESP32	76	1.5.4
micropython	v1.18	frozen	esp32	GENERIC	21	1.10.0
micropython	v1.18	frozen	esp32	LOLIN_S2_MINI	22	1.10.0
micropython	v1.18	frozen	esp32	LOLIN_S2_PICO	24	1.10.0
micropython	v1.18	frozen	esp32	M5STACK_ATOM	22	1.10.0
micropython	v1.18	frozen	esp32	UM_FEATHERS2	23	1.10.0
micropython	v1.18	frozen	esp32	UM_FEATHERS2NEO	22	1.10.0
micropython	v1.18	frozen	esp32	UM_TINYPICO	23	1.10.0
micropython	v1.18	frozen	esp32	UM_TINYS2	22	1.10.0
micropython	v1.18	board	esp8266	ESP module with ESP8266	73	1.5.4
micropython	v1.18	frozen	esp8266	GENERIC	29	1.10.0

Table 1 – continued from previous page

family	version	type	port	board	count	stubber versio
micropython	v1.18	frozen	esp8266	GENERIC_512K	11	1.10.0
micropython	v1.18	frozen	mimxrt	GENERIC	11	1.10.0
micropython	v1.18	frozen	nrf	GENERIC	1	1.10.0
micropython	v1.18	frozen	rp2	ARDUINO_NANO_RP2040_CONNECT	23	1.10.0
micropython	v1.18	frozen	rp2	GENERIC	13	1.10.0
micropython	v1.18	board	rp2	Arduino Nano RP2040 Connect with RP2040	60	1.5.3
micropython	v1.18	frozen	samd	GENERIC	1	1.10.0
micropython	v1.18	frozen	stm32	GARATRONIC_PYBSTICK26_F411	8	1.10.0
micropython	v1.18	frozen	stm32	GENERIC	11	1.10.0
micropython	v1.18	frozen	stm32	PYBD_SF2	14	1.10.0
micropython	v1.18	board	stm32	PYBv1.1 with STM32F405RG	57	1.5.6
micropython	v1.18	frozen	teensy	GENERIC	2	1.10.0
micropython	v1.18	frozen	unix	GENERIC	9	1.10.0
micropython	v1.19	documentation	-	micropython	50	1.10.0
micropython	v1.19	frozen	esp32	GENERIC	21	1.9.12
micropython	v1.19	frozen	esp32	LILYGO_TTGO_LORA32	24	1.9.12
micropython	v1.19	frozen	esp32	LOLIN_C3_MINI	22	1.9.12
micropython	v1.19	frozen	esp32	LOLIN_S2_MINI	22	1.9.12
micropython	v1.19	frozen	esp32	LOLIN_S2_PICO	24	1.9.12
micropython	v1.19	frozen	esp32	M5STACK_ATOM	22	1.9.12
micropython	v1.19	frozen	esp32	UM_FEATHERS2	23	1.9.12
micropython	v1.19	frozen	esp32	UM_FEATHERS2NEO	22	1.9.12
micropython	v1.19	frozen	esp32	UM_FEATHERS3	22	1.9.12
micropython	v1.19	frozen	esp32	UM_PROS3	22	1.9.12
micropython	v1.19	frozen	esp32	UM_TINYPICO	23	1.9.12
micropython	v1.19	frozen	esp32	UM_TINYS2	22	1.9.12
micropython	v1.19	frozen	esp32	UM_TINYS3	22	1.9.12
micropython	v1.19	frozen	esp8266	GENERIC	29	1.9.12
micropython	v1.19	frozen	esp8266	GENERIC_512K	11	1.9.12
micropython	v1.19	frozen	mimxrt	GENERIC	11	1.9.12
micropython	v1.19	frozen	nrf	GENERIC	8	1.9.12
micropython	v1.19	frozen	renesas-ra	GENERIC	10	1.9.12
micropython	v1.19	frozen	renesas-ra	RA4M1_CLICKER	1	1.9.12
micropython	v1.19	frozen	renesas-ra	RA4M1_EK	1	1.9.12
micropython	v1.19	frozen	rp2	ARDUINO_NANO_RP2040_CONNECT	27	1.9.12
micropython	v1.19	frozen	rp2	GENERIC	14	1.9.12
micropython	v1.19	frozen	samd	GENERIC	1	1.9.12
micropython	v1.19	frozen	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
micropython	v1.19	frozen	stm32	GARATRONIC_PYBSTICK26_F411	8	1.9.12
micropython	v1.19	frozen	stm32	GENERIC	11	1.9.12
micropython	v1.19	frozen	stm32	LEGO_HUB_NO6	14	1.9.12
micropython	v1.19	frozen	stm32	PYBD_SF2	14	1.9.12
micropython	v1.19	frozen	teensy	GENERIC	2	1.9.12
micropython	v1.19	frozen	unix	GENERIC	9	1.9.12
micropython	v1.19	frozen	windows	GENERIC	7	1.9.12
micropython	v1.19.1	documentation	-	micropython	50	1.9.12
micropython	v1.19.1	board	esp32	ESP32 module (spiram) with ESP32	72	1.9.11
micropython	v1.19.1	frozen	esp32	GENERIC	21	1.9.12
micropython	v1.19.1	frozen	esp32	LILYGO_TTGO_LORA32	24	1.9.12

Table 1 – continued from previous page

family	version	type	port	board	count	stubber version
micropython	v1.19.1	frozen	esp32	LOLIN_C3_MINI	22	1.9.12
micropython	v1.19.1	frozen	esp32	LOLIN_S2_MINI	22	1.9.12
micropython	v1.19.1	frozen	esp32	LOLIN_S2_PICO	24	1.9.12
micropython	v1.19.1	frozen	esp32	M5STACK_ATOM	22	1.9.12
micropython	v1.19.1	frozen	esp32	UM_FEATHERS2	23	1.9.12
micropython	v1.19.1	frozen	esp32	UM_FEATHERS2NEO	22	1.9.12
micropython	v1.19.1	frozen	esp32	UM_FEATHERS3	22	1.9.12
micropython	v1.19.1	frozen	esp32	UM_PROS3	22	1.9.12
micropython	v1.19.1	frozen	esp32	UM_TINYPICO	23	1.9.12
micropython	v1.19.1	frozen	esp32	UM_TINYS2	22	1.9.12
micropython	v1.19.1	frozen	esp32	UM_TINYS3	22	1.9.12
micropython	v1.19.1	board	esp8266	ESP module (1M) with ESP8266	57	1.9.11
micropython	v1.19.1	frozen	esp8266	GENERIC	29	1.9.12
micropython	v1.19.1	frozen	esp8266	GENERIC_512K	11	1.9.12
micropython	v1.19.1	frozen	mimxrt	GENERIC	11	1.9.12
micropython	v1.19.1	frozen	nrf	GENERIC	8	1.9.12
micropython	v1.19.1	frozen	renesas-ra	GENERIC	10	1.9.12
micropython	v1.19.1	frozen	renesas-ra	RA4M1_CLICKER	1	1.9.12
micropython	v1.19.1	frozen	renesas-ra	RA4M1_EK	1	1.9.12
micropython	v1.19.1	frozen	rp2	ARDUINO_NANO_RP2040_CONNECT	27	1.9.12
micropython	v1.19.1	frozen	rp2	GENERIC	14	1.9.12
micropython	v1.19.1	board	rp2	Arduino Nano RP2040 Connect with RP2040	68	1.9.11
micropython	v1.19.1	board	rp2	Raspberry Pi Pico with RP2040	52	1.9.11
micropython	v1.19.1	frozen	samd	GENERIC	1	1.9.12
micropython	v1.19.1	frozen	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
micropython	v1.19.1	frozen	stm32	GARATRONIC_PYBSTICK26_F411	8	1.9.12
micropython	v1.19.1	frozen	stm32	GENERIC	11	1.9.12
micropython	v1.19.1	frozen	stm32	LEGO_HUB_NO6	14	1.9.12
micropython	v1.19.1	frozen	stm32	PYBD_SF2	14	1.9.12
micropython	v1.19.1	board	stm32	PYBv1.1 with STM32F405RG	55	1.9.11
micropython	v1.19.1	frozen	teensy	GENERIC	2	1.9.12
micropython	v1.19.1	frozen	unix	GENERIC	9	1.9.12
micropython	v1.19.1	frozen	windows	GENERIC	7	1.9.12
micropython	v1.9.3	frozen	esp8266	micropython	15	1.8.0
micropython	v1.9.3	frozen	stm32	micropython	3	1.8.0
micropython	v1.9.3	frozen	unix	micropython	2	1.8.0
micropython	v1.9.3-L	board	-	ESP module with ESP8266	57	1.1.2
micropython	v1.9.4	board	-	ev3	80	1.3.2
micropython	v1.9.4	frozen	esp32	micropython	16	1.8.0
micropython	v1.9.4	frozen	esp8266	micropython	15	1.8.0
micropython	v1.9.4	frozen	stm32	micropython	4	1.8.0
micropython	v1.9.4	frozen	unix	micropython	2	1.8.0
micropython	v1.9.4-L	board	-	ESP module with ESP8266	43	1.1.2

10.2 Loboris

family	version	type	port	board	count	stubber version	path
loboris	v3.2.24	board	esp32	ESP32 board with ESP32	68	1.0.0	stubs/loboris-v3_2_24-esp32
loboris	v3.2.24	frozen	esp32	loboris	17	1.5.5a2	stubs/loboris-v3_2_24-frozen
loboris	v3.2.9	board	esp32	ESP32 board with ESP32	68	1.1.2	stubs/loboris-v3_2_9-esp32

10.3 Lvgl

family	version	type	port	board	count	stubber version	path
lvgl	v8.1	library	esp32	ESP32 module (spiram) with ESP32	3	1.4.2	stubs/lvgl-v8_1_0_dev-esp32

10.4 M5stack

family	version	type	port	board	count	stubber version	path
m5stack	v1.11	board	-	ESP32 module with ESP32	129	1.3.1	stubs/m5stack_flowui-v1_4_0-beta

10.5 Pyb

family	version	type	port	board	count	stubber version	path
pyb	-	library	pyb	micropython-pyb by Daryl Schults	1	-	stubs/cpython_pyboard

10.6 Pycopy

family	version	type	port	board	count	stubber version	path
pycopy	-	frozen	-	included frozen modules	0	manual	stubs/pycopy-v0_0_0-frozen

10.7 Ulab

family	version	type	port	board	count	stubber version	path
ulab	-	library	-	generic	9	1.3.7	stubs/micropython-ulab

10.8 Metrics

Total stub modules: 5670

CHAPTER

ELEVEN

ALL STUBS BY TYPE

This page provides an overview of all collected and generated module stubs by type.

Type	family	version	port	board	#	stubber versio
board	loboris	v3.2.24	esp32	ESP32 board with ESP32	68	1.0.0
board	loboris	v3.2.9	esp32	ESP32 board with ESP32	68	1.1.2
board	m5stack	v1.11	-	ESP32 module with ESP32	129	1.3.1
board	micropython	v1.10	esp32	ESP32 module with ESP32	64	1.5.4
board	micropython	v1.10	pyboard	PYBv1.1 with STM32F405RG	50	1.5.4
board	micropython	v1.10-Latest	-	ESP module with ESP8266	67	1.1.0
board	micropython	v1.11	esp32	ESP32 module with ESP32	64	1.5.4
board	micropython	v1.11-Latest	-	ESP module with ESP8266	65	1.1.0
board	micropython	v1.12	esp32	ESP32 module (spiram) with ESP32	66	1.5.4
board	micropython	v1.12	pyboard	PYBv1.1 with STM32F405RG	51	1.5.4
board	micropython	v1.13	-	PYBv1.1 with STM32F405RG	47	1.3.4
board	micropython	v1.13	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
board	micropython	v1.13	esp32	ESP32 module (spiram) with ESP32	73	1.5.0
board	micropython	v1.14	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
board	micropython	v1.14	esp8266	ESP module with ESP8266	73	1.5.4
board	micropython	v1.15	esp32	ESP32 module (spiram) with ESP32	72	1.5.4
board	micropython	v1.15	esp8266	ESP module with ESP8266	73	1.5.4
board	micropython	v1.15	pyboard	PYBv1.1 with STM32F405RG	58	1.5.4
board	micropython	v1.16	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
board	micropython	v1.16	esp8266	ESP module with ESP8266	73	1.5.4
board	micropython	v1.16	pyboard	PYBv1.1 with STM32F405RG	58	1.5.4
board	micropython	v1.17	esp32	ESP32 module (spiram) with ESP32	75	1.5.4
board	micropython	v1.17	esp8266	ESP module with ESP8266	73	1.5.4
board	micropython	v1.17	pyboard	PYBv1.1 with STM32F405RG	59	1.5.4
board	micropython	v1.17	rp2	Raspberry Pi Pico with RP2040	51	1.5.2
board	micropython	v1.18	esp32	ESP32 module (spiram) with ESP32	76	1.5.4
board	micropython	v1.18	esp8266	ESP module with ESP8266	73	1.5.4
board	micropython	v1.18	rp2	Arduino Nano RP2040 Connect with RP2040	60	1.5.3
board	micropython	v1.18	stm32	PYBv1.1 with STM32F405RG	57	1.5.6
board	micropython	v1.19.1	esp32	ESP32 module (spiram) with ESP32	72	1.9.11
board	micropython	v1.19.1	esp8266	ESP module (1M) with ESP8266	57	1.9.11
board	micropython	v1.19.1	rp2	Arduino Nano RP2040 Connect with RP2040	68	1.9.11
board	micropython	v1.19.1	rp2	Raspberry Pi Pico with RP2040	52	1.9.11
board	micropython	v1.19.1	stm32	PYBv1.1 with STM32F405RG	55	1.9.11
board	micropython	v1.9.3-Latest	-	ESP module with ESP8266	57	1.1.2

Table 1 – continued from previous page

Type	family	version	port	board	#	stubber versio
board	micropython	v1.9.4	-	ev3	80	1.3.2
board	micropython	v1.9.4-Latest	-	ESP module with ESP8266	43	1.1.2
documentation	micropython	latest	-	micropython	50	1.8.0
documentation	micropython	v1.17	-	micropython	47	1.9.12
documentation	micropython	v1.18	-	micropython	49	1.10.0
documentation	micropython	v1.19	-	micropython	50	1.10.0
documentation	micropython	v1.19.1	-	micropython	50	1.9.12
frozen	loboris	v3.2.24	esp32	loboris	17	1.5.5a2
frozen	micropython	latest	esp32	GENERIC	24	1.9.12
frozen	micropython	latest	esp32	LILYGO_TTGO_LORA32	27	1.9.12
frozen	micropython	latest	esp32	LOLIN_C3_MINI	25	1.9.12
frozen	micropython	latest	esp32	LOLIN_S2_MINI	25	1.9.12
frozen	micropython	latest	esp32	LOLIN_S2_PICO	27	1.9.12
frozen	micropython	latest	esp32	M5STACK_ATOM	25	1.9.12
frozen	micropython	latest	esp32	UM_FEATHERS2	26	1.9.12
frozen	micropython	latest	esp32	UM_FEATHERS2NEO	25	1.9.12
frozen	micropython	latest	esp32	UM_FEATHERS3	25	1.9.12
frozen	micropython	latest	esp32	UM_PROS3	25	1.9.12
frozen	micropython	latest	esp32	UM_TINYPICO	26	1.9.12
frozen	micropython	latest	esp32	UM_TINYS2	25	1.9.12
frozen	micropython	latest	esp32	UM_TINYS3	25	1.9.12
frozen	micropython	latest	esp8266	GENERIC	27	1.9.12
frozen	micropython	latest	esp8266	GENERIC_512K	10	1.9.12
frozen	micropython	latest	mimxrt	GENERIC	11	1.9.12
frozen	micropython	latest	nrf	arduino_nano_33_ble_sense	11	1.9.12
frozen	micropython	latest	nrf	GENERIC	8	1.9.12
frozen	micropython	latest	renesas-ra	GENERIC	10	1.9.12
frozen	micropython	latest	renesas-ra	RA4M1_CLICKER	1	1.9.12
frozen	micropython	latest	renesas-ra	RA4M1_EK	1	1.9.12
frozen	micropython	latest	rp2	ARDUINO_NANO_RP2040_CONNECT	30	1.9.12
frozen	micropython	latest	rp2	GENERIC	14	1.9.12
frozen	micropython	latest	rp2	PICO_W	17	1.9.12
frozen	micropython	latest	rp2	WEACTSTUDIO	15	1.9.12
frozen	micropython	latest	samd	GENERIC	11	1.9.12
frozen	micropython	latest	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
frozen	micropython	latest	stm32	GARATRONIC_PYBSTICK26_F411	1	1.9.12
frozen	micropython	latest	stm32	GENERIC	9	1.9.12
frozen	micropython	latest	stm32	LEGO_HUB_NO6	12	1.9.12
frozen	micropython	latest	stm32	LEGO_HUB_NO7	12	1.9.12
frozen	micropython	latest	stm32	PYBD_SF2	12	1.9.12
frozen	micropython	latest	stm32	PYBV10	10	1.9.12
frozen	micropython	latest	teensy	GENERIC	2	1.9.12
frozen	micropython	latest	unix	GENERIC	11	1.9.12
frozen	micropython	latest	windows	GENERIC	7	1.9.12
frozen	micropython	v1.10	esp32	micropython	16	1.8.0
frozen	micropython	v1.10	esp8266	micropython	15	1.8.0
frozen	micropython	v1.10	stm32	micropython	4	1.8.0
frozen	micropython	v1.10	unix	micropython	2	1.8.0
frozen	micropython	v1.11	esp32	micropython	16	1.8.0

Table 1 – continued from previous page

Type	family	version	port	board	#	stubber versio
frozen	micropython	v1.11	esp8266	micropython	15	1.8.0
frozen	micropython	v1.11	stm32	micropython	4	1.8.0
frozen	micropython	v1.11	unix	micropython	2	1.8.0
frozen	micropython	v1.12	common	GENERIC	2	1.7.4
frozen	micropython	v1.12	esp32	GENERIC	14	1.7.4
frozen	micropython	v1.12	esp32	RELEASE	18	1.7.4
frozen	micropython	v1.12	esp32	TINYPICO	16	1.7.4
frozen	micropython	v1.12	esp8266	GENERIC	15	1.7.4
frozen	micropython	v1.12	esp8266	RELEASE	29	1.7.4
frozen	micropython	v1.12	stm32	GENERIC	4	1.7.4
frozen	micropython	v1.12	stm32	PYBD_SF2	7	1.7.4
frozen	micropython	v1.13	esp32	GENERIC	21	1.8.0
frozen	micropython	v1.13	esp32	RELEASE	25	1.8.0
frozen	micropython	v1.13	esp32	TINYPICO	23	1.8.0
frozen	micropython	v1.13	esp8266	GENERIC	22	1.8.0
frozen	micropython	v1.13	esp8266	RELEASE	27	1.8.0
frozen	micropython	v1.13	stm32	GENERIC	11	1.8.0
frozen	micropython	v1.13	stm32	PYBD_SF2	14	1.8.0
frozen	micropython	v1.14	esp32	GENERIC	21	1.8.0
frozen	micropython	v1.14	esp32	RELEASE	25	1.8.0
frozen	micropython	v1.14	esp32	TINYPICO	23	1.8.0
frozen	micropython	v1.14	esp8266	GENERIC	28	1.8.0
frozen	micropython	v1.14	stm32	GENERIC	11	1.8.0
frozen	micropython	v1.14	stm32	PYBD_SF2	14	1.8.0
frozen	micropython	v1.15	esp32	GENERIC	21	1.8.0
frozen	micropython	v1.15	esp32	RELEASE	25	1.8.0
frozen	micropython	v1.15	esp32	TINYPICO	23	1.8.0
frozen	micropython	v1.15	esp8266	GENERIC	28	1.8.0
frozen	micropython	v1.15	stm32	GENERIC	11	1.8.0
frozen	micropython	v1.15	stm32	PYBD_SF2	14	1.8.0
frozen	micropython	v1.16	esp32	GENERIC	21	1.8.0
frozen	micropython	v1.16	esp32	M5STACK_ATOM	22	1.8.0
frozen	micropython	v1.16	esp32	RELEASE	25	1.8.0
frozen	micropython	v1.16	esp32	UM_FEATHERS2	23	1.8.0
frozen	micropython	v1.16	esp32	UM_TINYPICO	23	1.8.0
frozen	micropython	v1.16	esp32	UM_TINYS2	22	1.8.0
frozen	micropython	v1.16	esp8266	GENERIC	28	1.8.0
frozen	micropython	v1.16	esp8266	GENERIC_512K	11	1.8.0
frozen	micropython	v1.16	mimxrt	GENERIC	10	1.8.0
frozen	micropython	v1.16	rp2	GENERIC	11	1.8.0
frozen	micropython	v1.16	stm32	GENERIC	11	1.8.0
frozen	micropython	v1.16	stm32	PYBD_SF2	14	1.8.0
frozen	micropython	v1.17	esp32	GENERIC	21	1.9.12
frozen	micropython	v1.17	esp32	M5STACK_ATOM	22	1.9.12
frozen	micropython	v1.17	esp32	UM_FEATHERS2	23	1.9.12
frozen	micropython	v1.17	esp32	UM_TINYPICO	23	1.9.12
frozen	micropython	v1.17	esp32	UM_TINYS2	22	1.9.12
frozen	micropython	v1.17	esp8266	GENERIC	29	1.9.12
frozen	micropython	v1.17	esp8266	GENERIC_512K	11	1.9.12

Table 1 – continued from previous page

Type	family	version	port	board	#	stubber versio
frozen	micropython	v1.17	mimxrt	GENERIC	10	1.9.12
frozen	micropython	v1.17	nrf	GENERIC	1	1.9.12
frozen	micropython	v1.17	rp2	GENERIC	11	1.9.12
frozen	micropython	v1.17	stm32	GENERIC	11	1.9.12
frozen	micropython	v1.17	stm32	PYBD_SF2	14	1.9.12
frozen	micropython	v1.17	unix	GENERIC	9	1.9.12
frozen	micropython	v1.18	esp32	GENERIC	21	1.10.0
frozen	micropython	v1.18	esp32	LOLIN_S2_MINI	22	1.10.0
frozen	micropython	v1.18	esp32	LOLIN_S2_PICO	24	1.10.0
frozen	micropython	v1.18	esp32	M5STACK_ATOM	22	1.10.0
frozen	micropython	v1.18	esp32	UM_FEATHERS2	23	1.10.0
frozen	micropython	v1.18	esp32	UM_FEATHERS2NEO	22	1.10.0
frozen	micropython	v1.18	esp32	UM_TINYPICO	23	1.10.0
frozen	micropython	v1.18	esp32	UM_TINYS2	22	1.10.0
frozen	micropython	v1.18	esp8266	GENERIC	29	1.10.0
frozen	micropython	v1.18	esp8266	GENERIC_512K	11	1.10.0
frozen	micropython	v1.18	mimxrt	GENERIC	11	1.10.0
frozen	micropython	v1.18	nrf	GENERIC	1	1.10.0
frozen	micropython	v1.18	rp2	ARDUINO_NANO_RP2040_CONNECT	23	1.10.0
frozen	micropython	v1.18	rp2	GENERIC	13	1.10.0
frozen	micropython	v1.18	samd	GENERIC	1	1.10.0
frozen	micropython	v1.18	stm32	GARATRONIC_PYBSTICK26_F411	8	1.10.0
frozen	micropython	v1.18	stm32	GENERIC	11	1.10.0
frozen	micropython	v1.18	stm32	PYBD_SF2	14	1.10.0
frozen	micropython	v1.18	teensy	GENERIC	2	1.10.0
frozen	micropython	v1.18	unix	GENERIC	9	1.10.0
frozen	micropython	v1.19	esp32	GENERIC	21	1.9.12
frozen	micropython	v1.19	esp32	LILYGO_TTGO_LORA32	24	1.9.12
frozen	micropython	v1.19	esp32	LOLIN_C3_MINI	22	1.9.12
frozen	micropython	v1.19	esp32	LOLIN_S2_MINI	22	1.9.12
frozen	micropython	v1.19	esp32	LOLIN_S2_PICO	24	1.9.12
frozen	micropython	v1.19	esp32	M5STACK_ATOM	22	1.9.12
frozen	micropython	v1.19	esp32	UM_FEATHERS2	23	1.9.12
frozen	micropython	v1.19	esp32	UM_FEATHERS2NEO	22	1.9.12
frozen	micropython	v1.19	esp32	UM_FEATHERS3	22	1.9.12
frozen	micropython	v1.19	esp32	UM_PROS3	22	1.9.12
frozen	micropython	v1.19	esp32	UM_TINYPICO	23	1.9.12
frozen	micropython	v1.19	esp32	UM_TINYS2	22	1.9.12
frozen	micropython	v1.19	esp32	UM_TINYS3	22	1.9.12
frozen	micropython	v1.19	esp8266	GENERIC	29	1.9.12
frozen	micropython	v1.19	esp8266	GENERIC_512K	11	1.9.12
frozen	micropython	v1.19	mimxrt	GENERIC	11	1.9.12
frozen	micropython	v1.19	nrf	GENERIC	8	1.9.12
frozen	micropython	v1.19	renesas-ra	GENERIC	10	1.9.12
frozen	micropython	v1.19	renesas-ra	RA4M1_CLICKER	1	1.9.12
frozen	micropython	v1.19	renesas-ra	RA4M1_EK	1	1.9.12
frozen	micropython	v1.19	rp2	ARDUINO_NANO_RP2040_CONNECT	27	1.9.12
frozen	micropython	v1.19	rp2	GENERIC	14	1.9.12
frozen	micropython	v1.19	samd	GENERIC	1	1.9.12

Table 1 – continued from previous page

Type	family	version	port	board	#	stubber versio
frozen	micropython	v1.19	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
frozen	micropython	v1.19	stm32	GARATRONIC_PYBSTICK26_F411	8	1.9.12
frozen	micropython	v1.19	stm32	GENERIC	11	1.9.12
frozen	micropython	v1.19	stm32	LEGO_HUB_NO6	14	1.9.12
frozen	micropython	v1.19	stm32	PYBD_SF2	14	1.9.12
frozen	micropython	v1.19	teensy	GENERIC	2	1.9.12
frozen	micropython	v1.19	unix	GENERIC	9	1.9.12
frozen	micropython	v1.19	windows	GENERIC	7	1.9.12
frozen	micropython	v1.19.1	esp32	GENERIC	21	1.9.12
frozen	micropython	v1.19.1	esp32	LILYGO_TTGO_LORA32	24	1.9.12
frozen	micropython	v1.19.1	esp32	LOLIN_C3_MINI	22	1.9.12
frozen	micropython	v1.19.1	esp32	LOLIN_S2_MINI	22	1.9.12
frozen	micropython	v1.19.1	esp32	LOLIN_S2_PICO	24	1.9.12
frozen	micropython	v1.19.1	esp32	M5STACK_ATOM	22	1.9.12
frozen	micropython	v1.19.1	esp32	UM_FEATHERS2	23	1.9.12
frozen	micropython	v1.19.1	esp32	UM_FEATHERS2NEO	22	1.9.12
frozen	micropython	v1.19.1	esp32	UM_FEATHERS3	22	1.9.12
frozen	micropython	v1.19.1	esp32	UM_PROS3	22	1.9.12
frozen	micropython	v1.19.1	esp32	UM_TINYPICO	23	1.9.12
frozen	micropython	v1.19.1	esp32	UM_TINYS2	22	1.9.12
frozen	micropython	v1.19.1	esp32	UM_TINYS3	22	1.9.12
frozen	micropython	v1.19.1	esp8266	GENERIC	29	1.9.12
frozen	micropython	v1.19.1	esp8266	GENERIC_512K	11	1.9.12
frozen	micropython	v1.19.1	mimxrt	GENERIC	11	1.9.12
frozen	micropython	v1.19.1	nrf	GENERIC	8	1.9.12
frozen	micropython	v1.19.1	renesas-ra	GENERIC	10	1.9.12
frozen	micropython	v1.19.1	renesas-ra	RA4M1_CLICKER	1	1.9.12
frozen	micropython	v1.19.1	renesas-ra	RA4M1_EK	1	1.9.12
frozen	micropython	v1.19.1	rp2	ARDUINO_NANO_RP2040_CONNECT	27	1.9.12
frozen	micropython	v1.19.1	rp2	GENERIC	14	1.9.12
frozen	micropython	v1.19.1	samd	GENERIC	1	1.9.12
frozen	micropython	v1.19.1	stm32	ARDUINO_PORTENTA_H7	23	1.9.12
frozen	micropython	v1.19.1	stm32	GARATRONIC_PYBSTICK26_F411	8	1.9.12
frozen	micropython	v1.19.1	stm32	GENERIC	11	1.9.12
frozen	micropython	v1.19.1	stm32	LEGO_HUB_NO6	14	1.9.12
frozen	micropython	v1.19.1	stm32	PYBD_SF2	14	1.9.12
frozen	micropython	v1.19.1	teensy	GENERIC	2	1.9.12
frozen	micropython	v1.19.1	unix	GENERIC	9	1.9.12
frozen	micropython	v1.19.1	windows	GENERIC	7	1.9.12
frozen	micropython	v1.9.3	esp8266	micropython	15	1.8.0
frozen	micropython	v1.9.3	stm32	micropython	3	1.8.0
frozen	micropython	v1.9.3	unix	micropython	2	1.8.0
frozen	micropython	v1.9.4	esp32	micropython	16	1.8.0
frozen	micropython	v1.9.4	esp8266	micropython	15	1.8.0
frozen	micropython	v1.9.4	stm32	micropython	4	1.8.0
frozen	micropython	v1.9.4	unix	micropython	2	1.8.0
frozen	pycopy	-	-	included frozen modules	0	manual
library	lvgl	v8.1	esp32	ESP32 module (spiram) with ESP32	3	1.4.2
library	pyb	-	pyb	micropython-pyb by Daryl Schults	1	-

Table 1 – continued from previous page

Type	family	version	port	board	#	stubber versio
library	ulab	-	-	generic	9	1.3.7

Total modules 5670

CHAPTER

TWELVE

INDICES AND REFERENCE

- genindex
- search