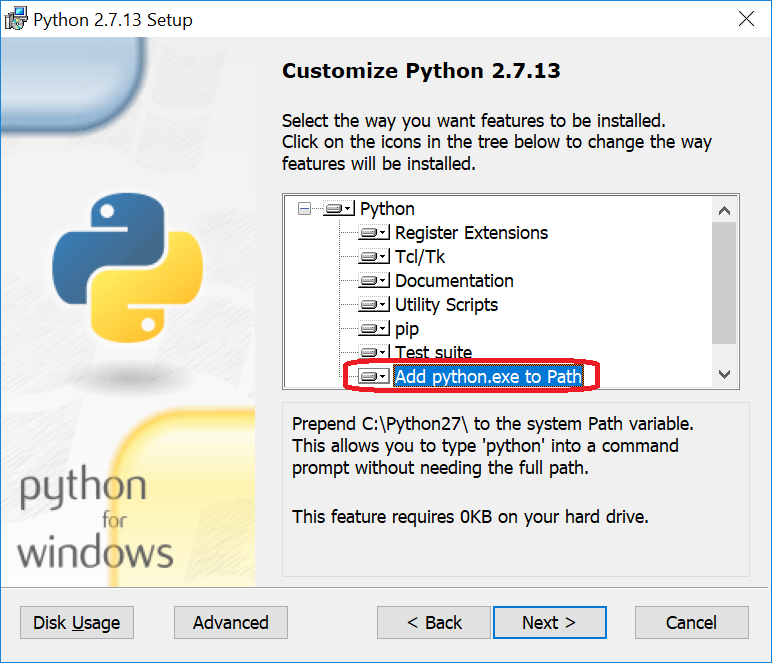
# The mbed target needs some tools to work properly:

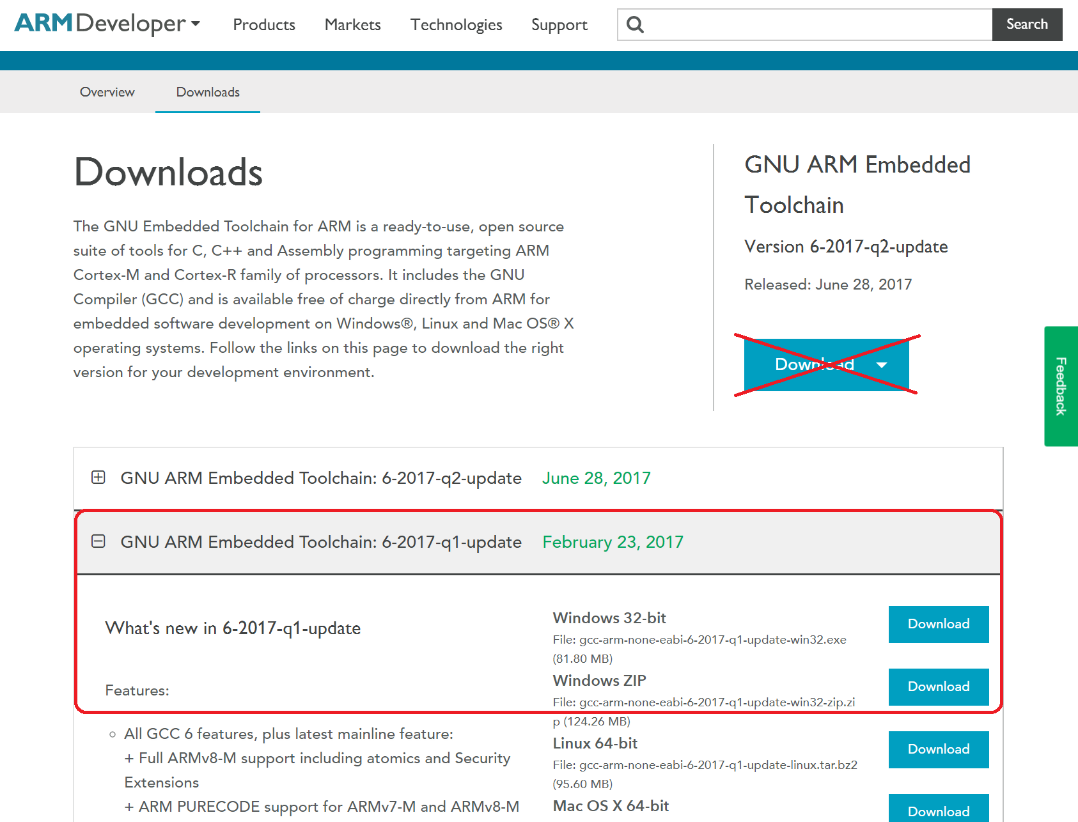
1. **Python** - mbed target uses mbed tools which are Python scripts, so you need Python to use it. Please install version 2.7.13 of Python: https://www.python.org/downloads/release/python-2713/

It is not compatible with Python 3!

During the installation it is necessary to choose the option **Add python.exe to Path**

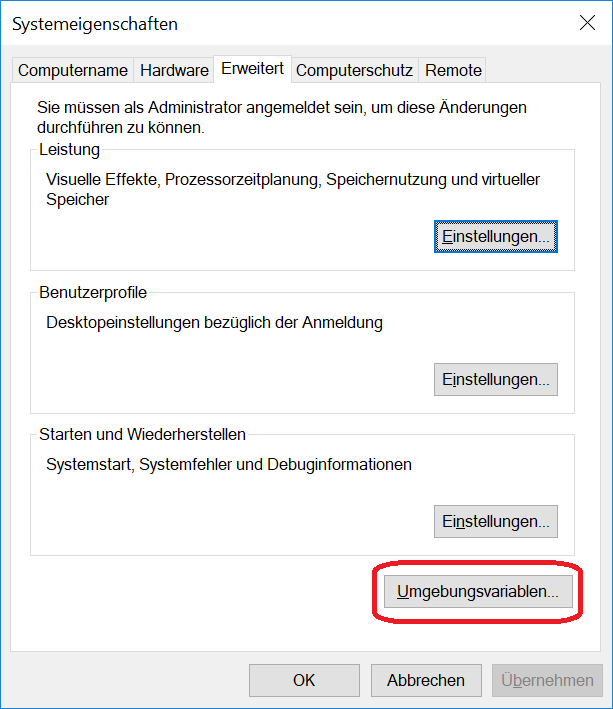


1. **GNU ARM Embedded Toolchain** – The code generation and building process incorporates the gcc\_arm toolchain. Because the currently included mbed os version 5.5 is tested with GCC 6 (6-2017-q1-update) this version should be preferred. You can find it here: https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads

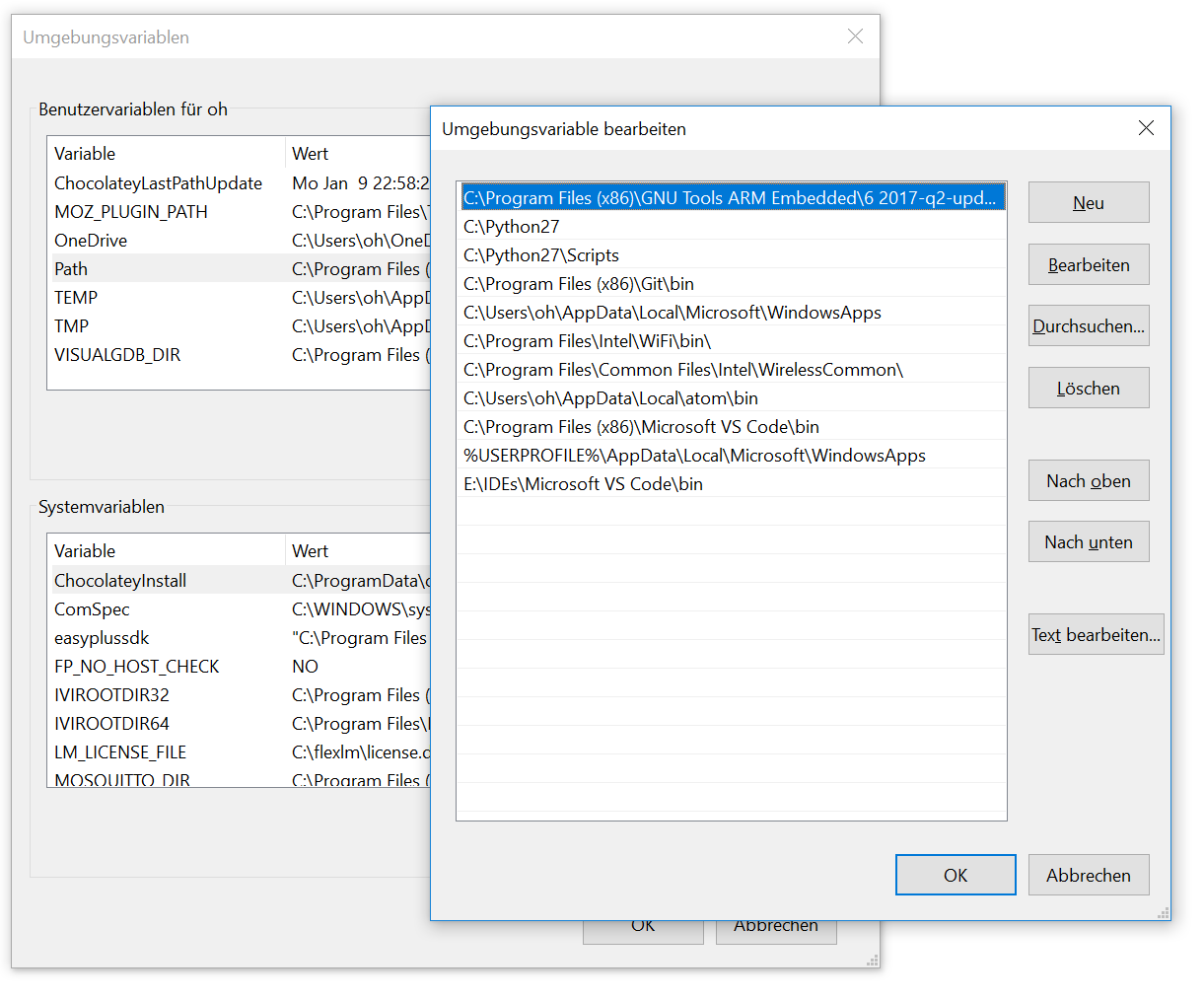


The gcc tools needs to be in the Windows Path:

Open the system properties dialog and press the button environment variables.



Check the user variable Path if the gcc path is contained. When not add it.



1. Further necessary tools are **mbed-cli** and **mbed-ls**. They can be installed manually with pip, the Python package management system:

C:\mbed\_target>pip install mbed-cli

and

C:\mbed\_target>pip install mbed-ls

But both tools can also be installed during the Matlab mbed\_target setup.

There exists an mbed CLI for Windows Installer under <https://docs.mbed.com/docs/mbed-os-handbook/en/latest/dev_tools/cli_install/>. This setup install all except for mbed-ls but additionally the not yet needed source code management system git and mercurial. This installer may easy the installation.

# Matlab Setup

The mbed\_target directory can be copied somewhere into the directory tree. But it is recommendable to use a path as close as possible to the root. The usage of the document director or the desktop may not work in some cases. Windows has a restriction in the maximum command line length and the mbed\_target build process could exceed this length in certain circumstances. As a second requirement the folder name mbed\_target must not be changed and has to be unique.

The root directory of mbed\_target directory contains an m-file: setup\_mbed\_target.m. This files has to be run once before using mbed\_target. It adds several folders to the Matlab path and checks/installs several tools in the following sequence:

1. Checking the folder blocks\mex if it contains compiled s-functions. If the mexw64 files are missing the setup tries to compile the c-sourcen. In this case an external c-compiler is necessary. Please see the documentation of mex.
2. Checking the existence of the gcc\_arm compiler. The windows commandline tool ‘where’ is used for this. This can be done manually with 'where arm-none-eabi-gcc'. The output should look like:

C:\Users\oh>where arm-none-eabi-gcc

E:\IDEs\GNUToolsARMEmbedded\6\_2017q1\bin\arm-none-eabi-gcc.exe

When the toolchain is not correctly installed, e.g. the entry in the path variable is missing, an error is generated. In this case the existence of the toolchain, the correct version and the correct Windows path has to be checked.

The GNU ARM Embedded Toolchain version: gcc version 6.3.1 20170215 (release) [ARM/embedded-6-branch revision 245512] (GNU Tools for ARM Embedded Processors 6-2017-q1-update) is recommended for the included version mbed os 5.5.

1. Checking the existence of Python. Please be aware that version 2.7 is necessary. The mbed build system is currently not compatible with Python 3! Because the Python package installer pip is used, a Python version newer than 2.7.9 is necessary.

The version 2.7.13 is recommended for the included version mbed os 5.5.

1. When the Python check was ok then the mbed requirements are installed. This can also be done manually with ‘pip install -r requirements.txt’ in the mbed\_target\targets\mbed-os folder.
2. When the Python check was ok then a list of mbed os 5 compatible targets is listed. This can also be done manually with ‘python tools\project.py -S targets’ in the mbed\_target\targets\mbed-os folder.
3. Checking the existence of mbedls. This is an additional mbed tool to check a connected target board. When the tool is not available the setup asked to install it. This can also be done manually with ‘pip install mbed-ls’.
4. The last check is the verification of the mbed\_target installation path. Only the name is checked and not the path length. Strange compiler or linker error later when generating a Simulink model can be an indication of a too long installation path.

The output in the Matlab command window should be checked after the execution of setup\_mbed\_target.m if an error happened or a version number does not meets the requirements.