ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and Electrical Engineering

Machine Learning on Microcontrollers

227-0155-00G

Exercise 1

Installation Guide for ML on MCU Labs

ETH Center for Project-Based Learning

Monday 16th February, 2026

1 Introduction

Welcome to the laboratory of Machine Learning on Microcontrollers.

In this exercise, you will learn to install Docker and to create a Docker environment that we will use in our future exercises.

2 Notation

Student Task: Parts of the exercise that require you to complete a task will be explained in a shaded box like this.

Note: You find notes and remarks in boxes like this one.

3 What is Docker?

Docker is a platform for building and running applications in lightweight, isolated *containers*. A container bundles everything an application needs (OS libraries, Python, packages, and startup commands) so that the same code runs consistently on Linux, macOS, and Windows.

Why we use it here:

- **Reproducibility**: everyone runs the exact same environment.
- **Isolation**: no conflicts with system Python or other projects.
- **Portability**: one setup works across different machines and OSes.

4 What is Jupyter Notebook (Lab)?

Jupyter Notebook (and Jupyter Lab) is an interactive, web-based environment for writing and running code, visualizing results, and documenting experiments in a single place. Each notebook consists of cells that can contain code, text, formulas, and plots.

Why we use it here:

- **Exploration**: iterate quickly on experiments and visualize results inline.
- **Documentation**: combine code, figures, and explanations for reproducible reports.
- **Convenience**: runs in your browser; the server runs inside our Docker container.

5 Preparation

To ensure a smooth and identical setup across all machines, this course uses a Dockerized environment that already includes Python, system libraries (e.g., `ffmpeg`, `libsndfile`), and all Python packages from `requirements.txt`. At a high level, you will:

1. Install Docker.
2. Clone the course repository.
3. Build the Docker image once with `docker compose build`.
4. Run Jupyter with `docker compose up` and you will be ready for Python exercises.

6 Docker-based Environment

6.1 Installation

Docker can be installed on all major platforms.[1] Please select your corresponding OS and CPU family and download&install Docker desktop.

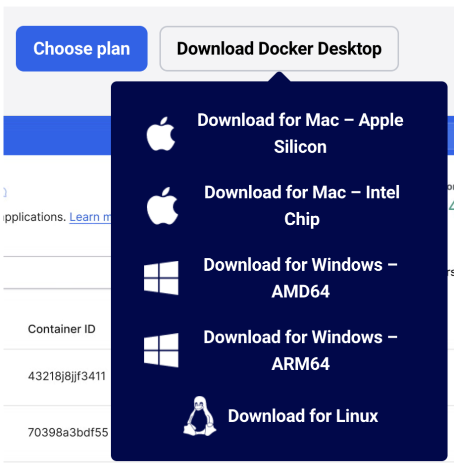

Choose plan
Download Docker Desktop
Download for Mac – Apple Silicon
Download for Mac – Intel Chip
Download for Windows – AMD64
Download for Windows – ARM64
Download for Linux
Container ID
43218j8jjf3411
70398a3bdf55

Figure 1: Downloading Docker Desktop

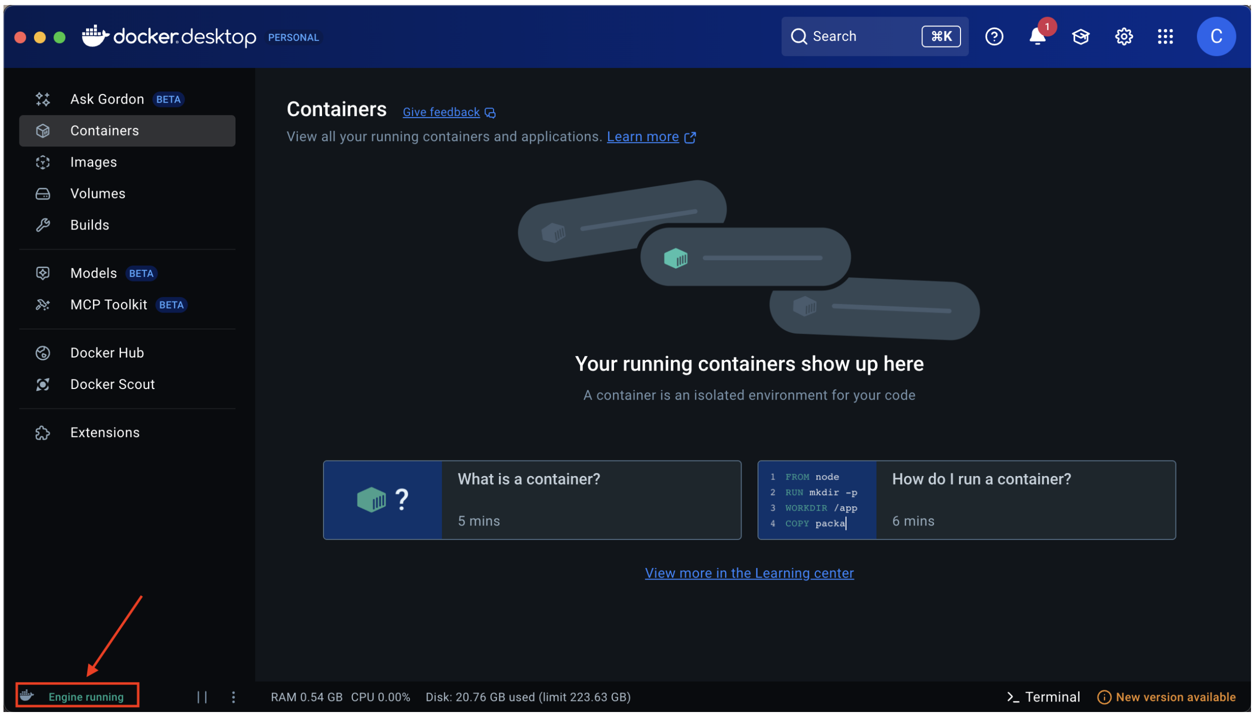
Note:

For **Windows and macOS**, downloading Docker Desktop and installing it via the graphical installer is sufficient. For **Linux**, you will be redirected to the *Supported platforms* page. Select your distribution (e.g., **Ubuntu**), then under *Install Docker Desktop* click *Download the latest DEB package*. After the download completes, install it from the terminal:

```
sudo apt-get update
sudo apt-get install ./docker-desktop-amd64.deb
```

[1] See: `https://www.docker.com/products/docker-desktop/`

After installation, launch Docker Desktop and ensure the engine is running.



View all your running containers and applications.

Your running containers show up here

A container is an isolated environment for your code

```
FROM node
RUN mkdir -p
WORKDIR /app
COPY packa
```

RAM 0.54 GB CPU 0.00% Disk: 20.76 GB used (limit 223.63 GB)

Figure 2: Docker Desktop Engine Running

Verify the installation in a terminal:

- `$ docker --version`
- `$ docker compose version`

6.2 Repository Layout and Files

Start from a fresh clone of the original repository:

- `$ git clone https://github.com/ETH-PBL/MLonMCU.git`

Your folder should look like:

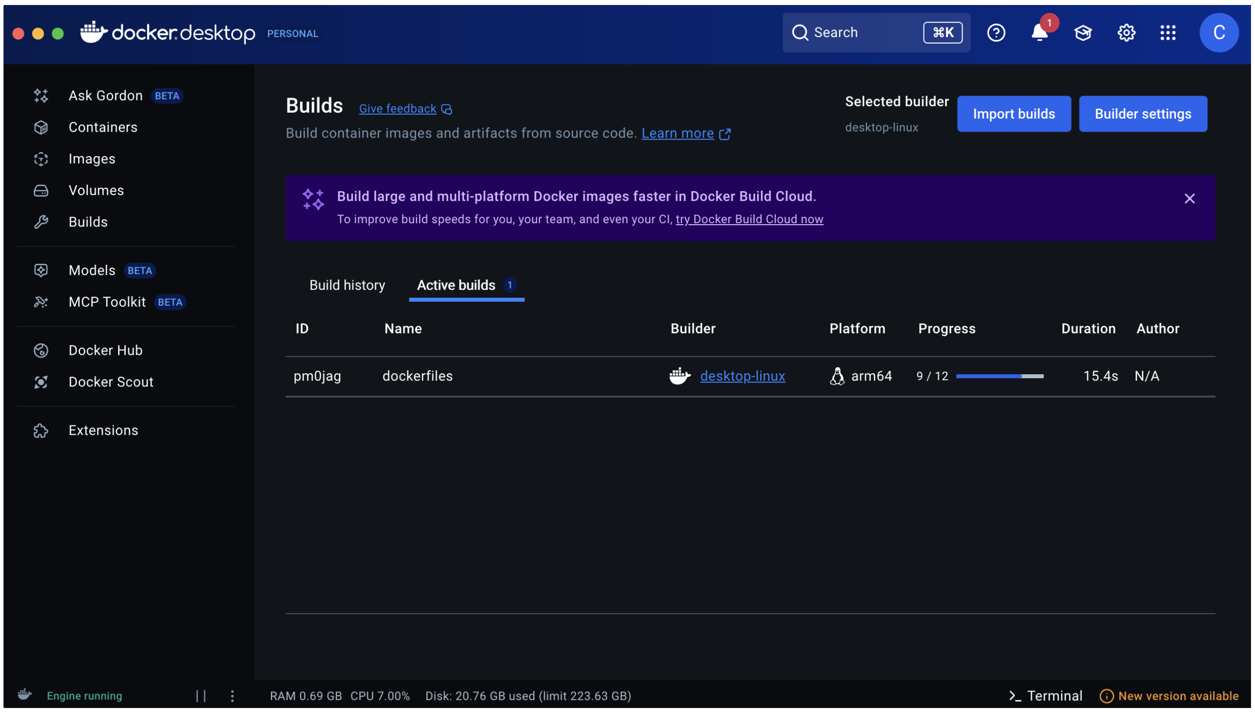
```
MLonMCU/
├─ex1
├─ex2
├─...
└─dockerfiles/
   ├─Dockerfile
   ├─README.md      ← Please follow the setup instructions in parallel with the README
   ├─docker-compose.yml
   ├─requirements.txt
   └─start.sh
```

6.3 Build the Image

Change to the `dockerfiles` directory and build:

- `$ cd MLonMCU/dockerfiles`
- `$ docker compose build`

The first build may take some time. You can check the build status following **Builds** ⟶ **Active Builds**.



Build container images and artifacts from source code. Learn more

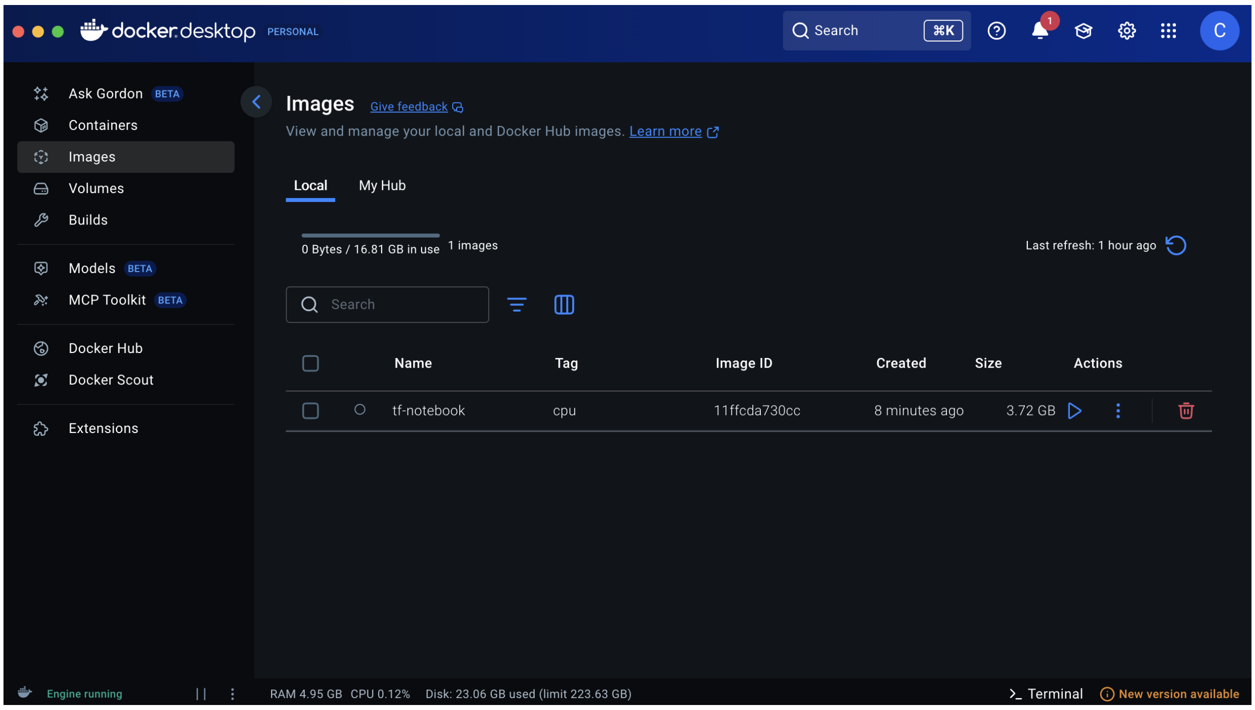
Build large and multi-platform Docker images faster in Docker Build Cloud.
To improve build speeds for you, your team, and even your CI, try Docker Build Cloud now

ID	Name	Builder	Platform	Progress	Duration	Author
pm0jag	dockerfiles	desktop-linux	arm64	9 / 12	15.4s	N/A

RAM 0.69 GB CPU 7.00% Disk: 20.76 GB used (limit 223.63 GB)

Figure 3: Docker Desktop Active Builds

When it finishes, you can see the built image from **Images** $\longrightarrow$ **Local**.



Name	Tag	Image ID	Created	Size	Actions
tf-notebook	cpu	11ffcda730cc	8 minutes ago	3.72 GB	

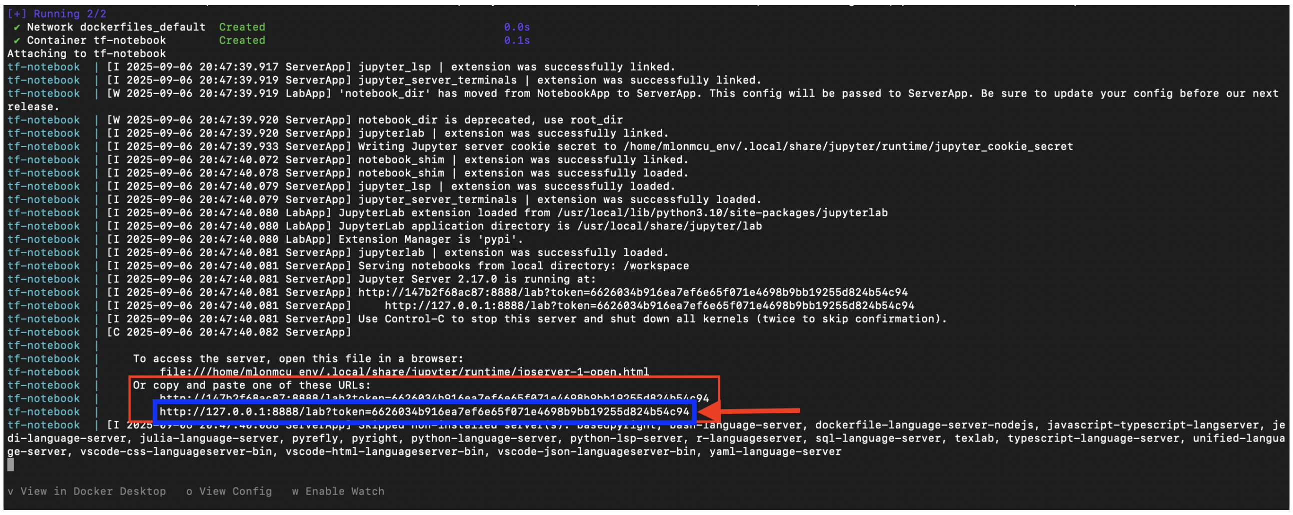
Figure 4: Docker Desktop Built Image

6.4 Run Jupyter

Start the container and launch Jupyter:

- `$ docker compose up`

When it starts, the terminal prints a URL like:



```
[+] Running 2/2
 ✔ Network dockerfiles_default  Created                                   0.0s
 ✔ Container tf-notebook        Created                                   0.1s
Attaching to tf-notebook
tf-notebook  | [I 2025-09-06 20:47:39.917 ServerApp] jupyter_lsp | extension was successfully linked.
tf-notebook  | [I 2025-09-06 20:47:39.919 ServerApp] jupyter_server_terminals | extension was successfully linked.
tf-notebook  | [W 2025-09-06 20:47:39.919 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next
release.
tf-notebook  | [W 2025-09-06 20:47:39.920 ServerApp] notebook_dir is deprecated, use root_dir
tf-notebook  | [I 2025-09-06 20:47:39.920 ServerApp] jupyterlab | extension was successfully linked.
tf-notebook  | [I 2025-09-06 20:47:39.933 ServerApp] Writing Jupyter server cookie secret to /home/mlonmcu_env/.local/share/jupyter/runtime/jupyter_cookie_secret
tf-notebook  | [I 2025-09-06 20:47:40.072 ServerApp] notebook_shim | extension was successfully linked.
tf-notebook  | [I 2025-09-06 20:47:40.078 ServerApp] notebook_shim | extension was successfully loaded.
tf-notebook  | [I 2025-09-06 20:47:40.079 ServerApp] jupyter_lsp | extension was successfully loaded.
tf-notebook  | [I 2025-09-06 20:47:40.079 ServerApp] jupyter_server_terminals | extension was successfully loaded.
tf-notebook  | [I 2025-09-06 20:47:40.080 LabApp] JupyterLab extension loaded from /usr/local/lib/python3.10/site-packages/jupyterlab
tf-notebook  | [I 2025-09-06 20:47:40.080 LabApp] JupyterLab application directory is /usr/local/share/jupyter/lab
tf-notebook  | [I 2025-09-06 20:47:40.080 LabApp] Extension Manager is 'pypi'.
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp] jupyterlab | extension was successfully loaded.
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp] Serving notebooks from local directory: /workspace
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp] Jupyter Server 2.17.0 is running at:
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp] http://147b2f68ac87:8888/lab?token=6626034b916ea7ef6e65f071e4698b9bb19255d824b54c94
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp]     http://127.0.0.1:8888/lab?token=6626034b916ea7ef6e65f071e4698b9bb19255d824b54c94
tf-notebook  | [I 2025-09-06 20:47:40.081 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
tf-notebook  | [C 2025-09-06 20:47:40.082 ServerApp]
tf-notebook  |
tf-notebook  |     To access the server, open this file in a browser:
tf-notebook  |         file:///home/mlonmcu_env/.local/share/jupyter/runtime/jpserver-1-open.html
tf-notebook  |     Or copy and paste one of these URLs:
tf-notebook  |         http://147b2f68ac87:8888/lab?token=6626034b916ea7ef6e65f071e4698b9bb19255d824b54c94
tf-notebook  |         http://127.0.0.1:8888/lab?token=6626034b916ea7ef6e65f071e4698b9bb19255d824b54c94
tf-notebook  | [I 2025-09-06 20:47:40.088 ServerApp] Skipped non-installed server(s): basedpyright, bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, je
di-language-server, julia-language-server, pyrefly, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-langua
ge-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server

v View in Docker Desktop   o View Config   w Enable Watch
```

Figure 5: Jupyter Notebook URL

Copy the link that looks like:

```
http://127.0.0.1:8888/lab?token=XXXXXXXXXXXXXXXXXXXXXXXX
```

and paste it into your web browser (Google Chrome is recommended). Keep this terminal open for the entire Jupyter session

Student Task 1 (Docker Setup): Using the instructions above, prepare your Docker environment and verify you can run the exercises:

1. Install Docker Desktop for your OS.
2. Clone the repository: `git clone https://github.com/ETH-PBL/MLonMCU.git`
3. Build and run the docker.

Now you can close the docker by typing `docker compose down` to your terminal. You should see the following output verifying that the running environment is closed:

```
(base) your@username dockerfiles % docker compose down
[+] Running 2/2
Container tf-notebook        Removed
Network dockerfiles_default  Removed
```

7 Running the Docker-Based Environment Test

This section explains how to run `test_env.ipynb` inside the Docker container.

7.1 Start the Container

1. Open a terminal and navigate to the course Docker folder:

   `MLonMCU/dockerfiles`

2. Launch the container with Docker Compose:

   `docker compose up`

3. Wait until a tokenized JupyterLab URL appears in the logs (e.g., `http://127.0.0.1:8888..`). Open this link in your browser.
4. In JupyterLab's file browser, open the freshly prepared workspace and navigate to **ex1/task/test_env.ipynb**
5. Run the notebook (run the cell with Shift + Enter or use `Run All` button) and verify the output.

7.2 What You Should See

- Tables for **Package Versions** and **Import-Only Checks** with `OK` statuses and matching versions.
- Some **Functional Smoke-Tests** table with all tests showing `PASS`.
- A **Final Status**: `ALL TESTS PASSED`.
- Two figures created in the same folder: `test_plot.png` and `model_visualization.png`.

Student Task 2 (Environment Test):

1. Do the printed versions match those in `MLonMCU/dockerfiles/requirements.txt`?
2. What kind of model visualization do you see, and why? ______________________
3. What do you see in the test plot? ______________________

7.3 Shut Down the Container

When you are done, stop and remove the services:

```
docker compose down
```

That is all for the Python-related setup.

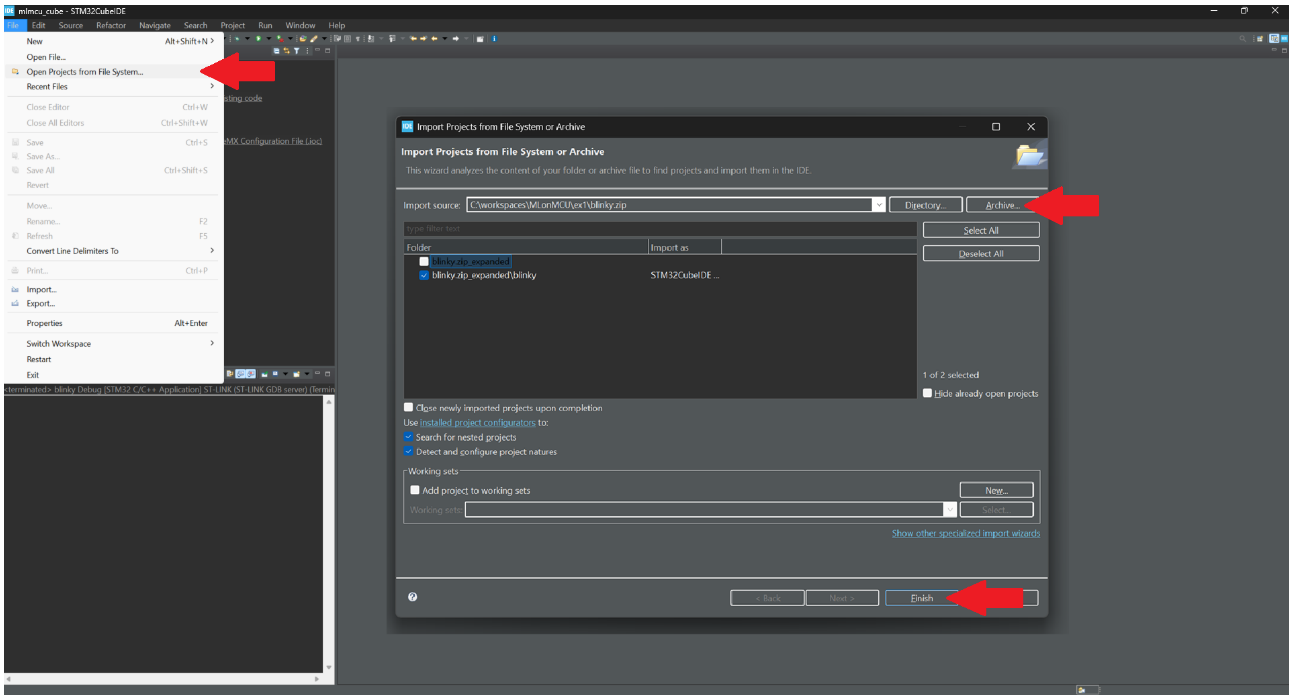
8 STM32 CUBE IDE

In the following weeks you will learn how to configure and program the STM Microcontroller (MCU) that you will use throughout the first half of this course. The development board for this course is the B-U585I-IOT02A featuring an STM32U585AI low-power MCU with a maximum clock frequency of 160 MHz, 786 KB of RAM and 2 MB of flash memory. The processor core is based on the ARM Cortex M33, featuring advanced Digital Signal Processor (DSP) operation capabilities and a floating-point unit.

Today, we will simply test the integrity of our MCU and verify the correctness of the STM32 CUBE IDE installation.

Student Task 3 (Blinking LED):

1. Import the blinky project in the STM32 CUBE IDE as shown on the screenshot figure 6. Do not unzip the project before importing it as archive.
2. Compile the code using the BUILD button and flash it to the MCU using the RUN button.
3. What is the behavior of the MCU?



Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\workspaces\MLonMCU\ex1\blinky.zip

Folder	Import as
☐ blinky.zip_expanded	
☑ blinky.zip_expanded\blinky	STM32CubeIDE ...

1 of 2 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

Working sets

☐ Add project to working sets

Working sets:

Show other specialized import wizards

<terminated> blinky Debug [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Termin

Figure 6: Importing Projects

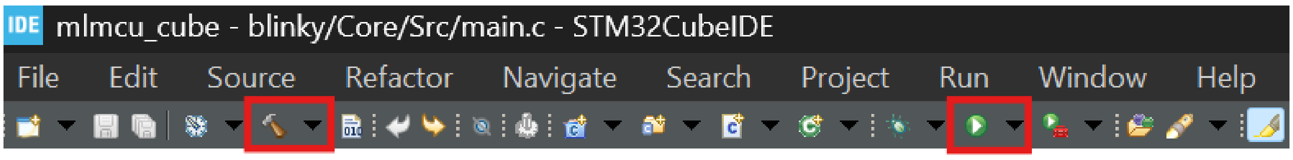


Figure 7: Flashing a Project

$\mathcal{E}$ **Congratulations! You have reached the end of the exercise.
If you are unsure of your results, discuss with an assistant.** $\mathcal{E}$