

IoTSSC LAB #1

GETTING STARTED WITH EMBEDDED PROGRAMING*

INTRODUCTION TO MBED ONLINE COMPILER

*Manual based on updated ARM University Program 2014 material and mbed-cli documentation.

*Manual updated in 2019 for the FRDM-K64F boards

HARDWARE/ SOFTWARE SETUP

1. Update board firmware [optional]
 - Download the latest mbed interface firmware from <https://armmbed.github.io/DAPLink/?board=FRDM-K64F>
 - **Note that for the following steps the micro usb port that powers the board is next to the reset button.**
 - Press and hold down the reset button, while connecting the USB cable between the board and the computer. You should then see an LED blinking on the board.
 - The FRDM-K64F Development Kit should now enumerate as "BOOTLOADER" (you can open a file browser to see if such a drive has been recognized).
 - Copy the binary file that you downloaded previously to the BOOTLOADER drive.
 - Wait until the drive dismounts and the LED is flashing continuously on your board.
 - Re-plug the board (this time without pressing the reset button). The board should enumerate and mount as DAPLINK.

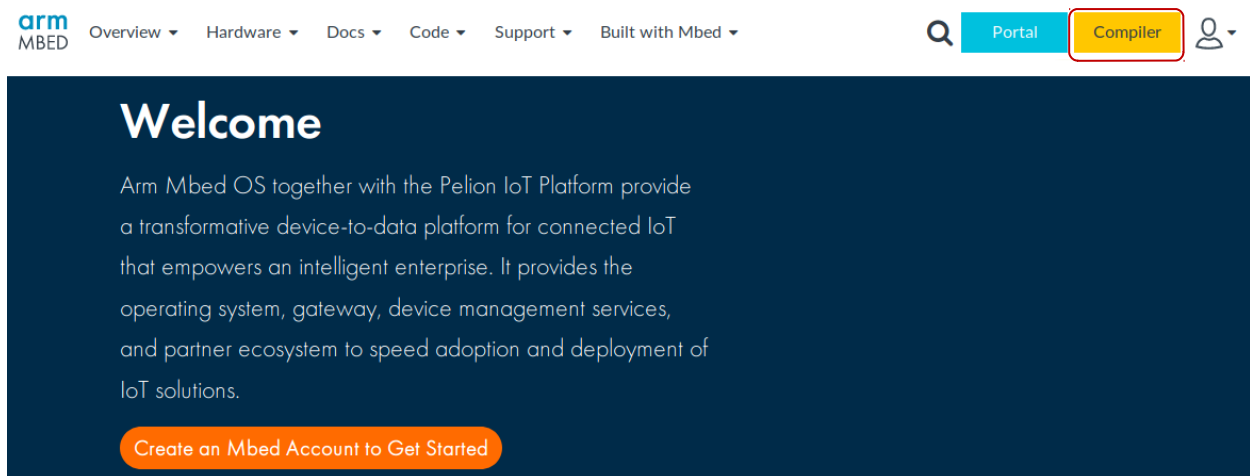
Note that for some lab exercises, you will have to download a program that you obtain by exporting a project from the mbed online compiler (see the "Exporting from mbed online compiler" section below).

EXPORTING FROM MBED ONLINE COMPILER

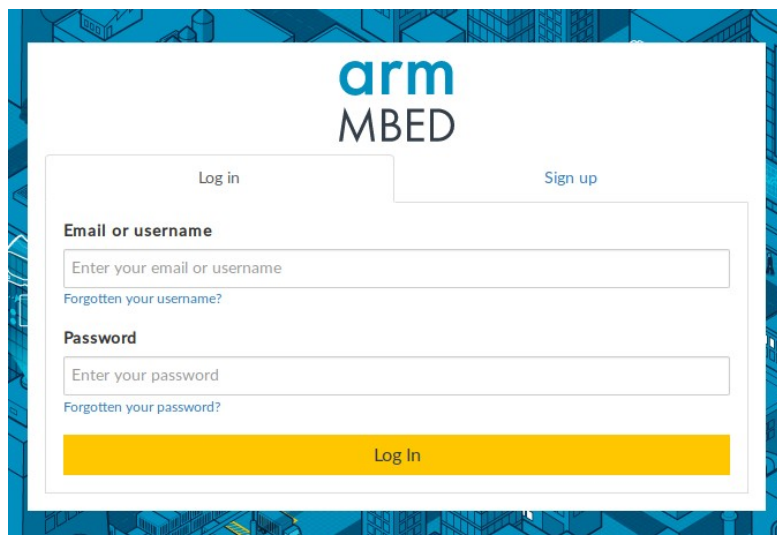
The mbed online Compiler provides a lightweight online C/C++ IDE that is pre-configured to let you quickly write programs, compile them, and download them to be executed on your mbed Microcontroller. The mbed online compiler is web based, hence you don't have to install or set up anything to get started with mbed.

CREATE YOUR FIRST MBED ONLINE PROJECT

1. Go to <https://mbed.org>, and click “compiler”

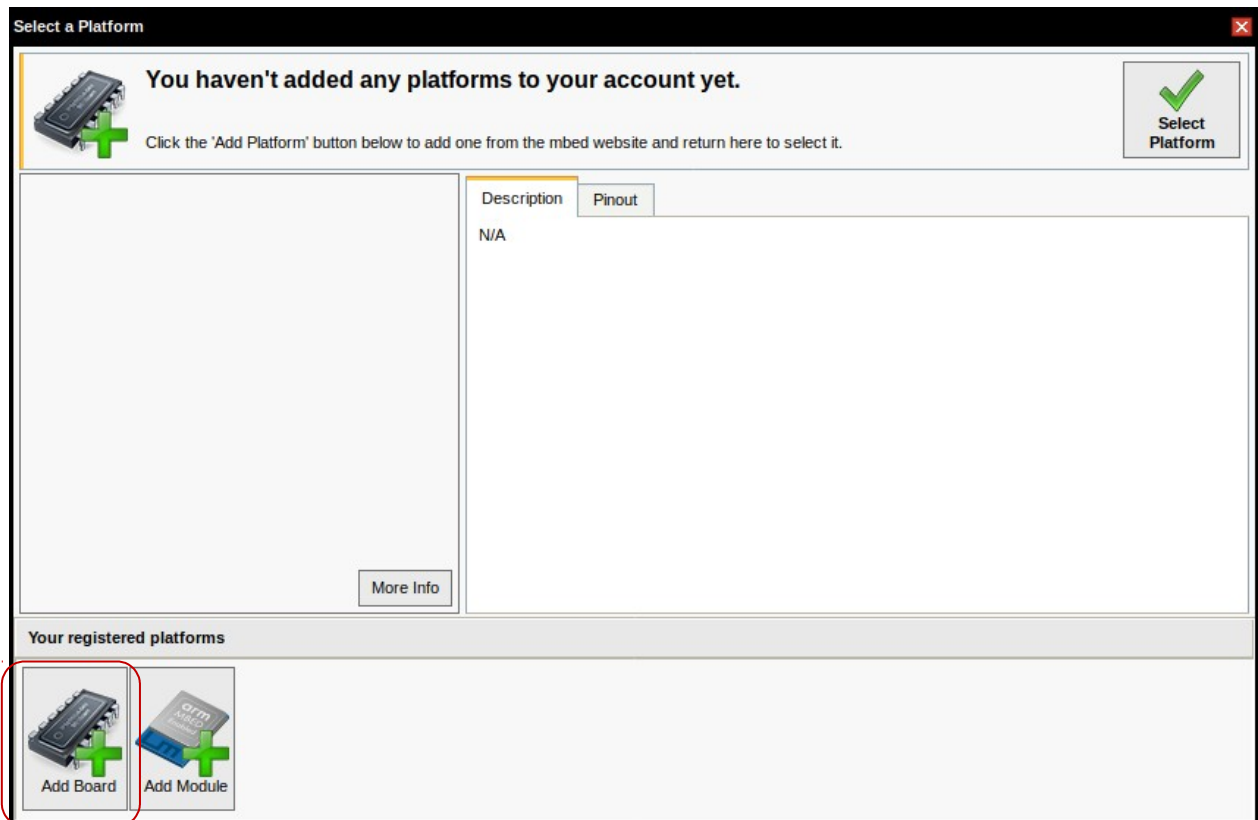
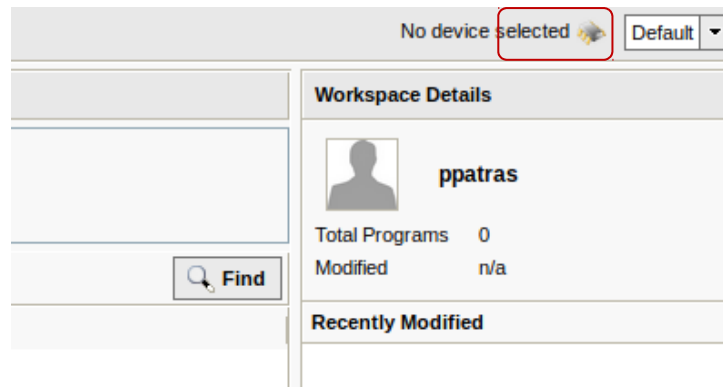


2. Register an account and then login



3. Open the Compiler. The main mbed IDE will be displayed. Here you can manage your projects, create new ones, import programs from the mbed.org community...

- On the top-right corner select the board you are currently working with. If it's the first time using this board press add a new platform.



Browse for your device among all the mbed enabled platforms. You can filter by manufacturer (**NXP Semiconductors**) to find the FRDM-K64F board faster.

Development boards

Build your Mbed projects with IoT development boards for Arm Cortex processors and microcontrollers. Mbed supports key MCU families including STM32, Kinetis, LPC, PSoC and nRF52, helping you to develop Internet of Things products quickly, securely and efficiently.

Filters

☐ Mbed Enabled

- Advanced (2)
- Baseline (18)
- Pellion Device Ready (2)

☐ Mbed OS support


- Mbed OS 2 (13)
- Mbed OS 5.4 (7)
- Mbed OS 5.5 (10)
- Mbed OS 5.6 (10)
- Mbed OS 5.7 (7)
- Mbed OS 5.8 (8)
- Mbed OS 5.9 (9)
- Mbed OS 5.10 (9)
- Mbed OS 5.11 (9)
- Mbed OS 5.12 (9)
- Mbed OS 5.13 (9)
- Mbed OS 5.14 (9)
- Mbed OS 5.15 (9)

☐ Communication

- CAN (4)
- Ethernet (4)
- USB Device (10)
- USB Host (7)


18 results

Sort by: Relevance




FRDM-K64F

- Cortex-M4, 120MHz
- 1024KB Flash, 256KB RAM
- Ethernet, USB Crystal-less, SD




FRDM-K66F


- Cortex-M4, 180MHz
- 2MB Flash, 256KB RAM
- Ethernet, microSD Card, Audio




FRDM-K82F

- Cortex-M4, 150MHz
- 256KB Flash, 256KB RAM
- USB, QuadSPI, Bootloader







Select the target board, then click 'Add to your Mbed Compiler'.

3, and K24 MCUs.

or
points.
K64F
1024KB
es
is fully
edom

Table of Contents

1. Overview

2. MCU Features

3. Board Features

4. Board Block Diagram

5. Board Pinout

6. PC Configuration

7. Debug Interface

8. Get Started with mbed

9. Flash a project binary

10. Open existing Project

11. Create new Project

12. Technical Doc

13. Software Materials

14. Supported NXP

To compile a program for this board using Mbed CLI, use `k64f` as the target name.

Board Partner

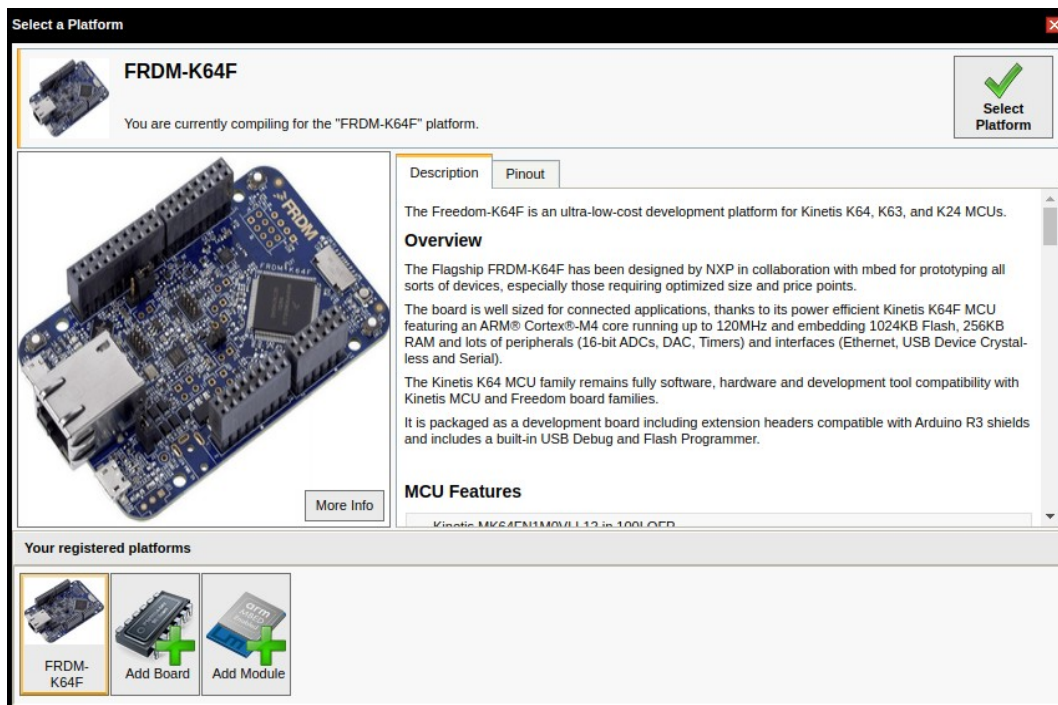
NXP

NXP is a leading semiconductor company founded by Philips more than 50 years ago.

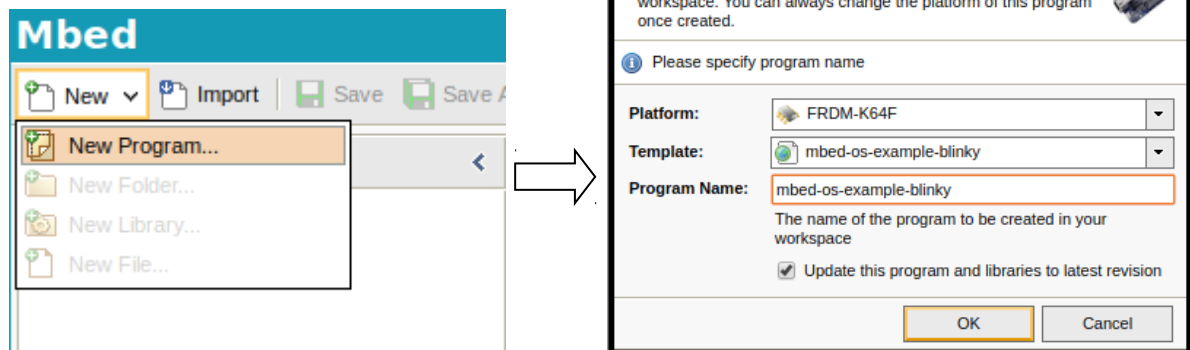
Mbed Enabled

- Advanced
- Baseline
- Pellion Device Ready

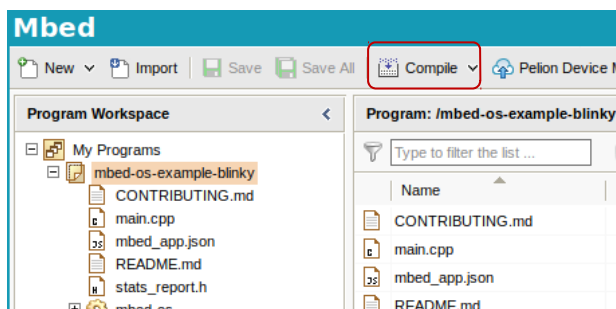
Go back to the Compiler interface. You should now be able to select this platform.



5. Create a helloworld project (blinky LED program).



6. Compile the program.



7. The program file will be generated and downloaded to your default download directory (set by your web browser).
8. Connect the board to your PC via an USB cable, DAPLINK will appear as a removable storage device.
9. Copy the downloaded program file to the DAPLINK root directory.
10. The board will automatically reset and run the latest copied program file.

SETTING UP A LOCAL DEVELOPMENT ENVIRONMENT

You may often wish to develop embedded code locally, to speed up deployment and avoid relying on Internet connectivity at all times. While some of the steps listed below are not necessary on DICE, since some software packages have been installed already, you will need to follow closely the steps below if you wish to set up a development environment on your own machine.

INSTALLING THE GNU EMBEDDED TOOLCHAIN FOR ARM (NOT NEEDED ON DICE)

The code you will develop for the embedded boards will be written in C, C++, or assembly. In order to be able to install on the boards an application that you develop on a PC, you will need to cross-compile the source tree for the target platform (for this lab the FRDM-K64F development kit). There exist multiple compiler options, but the recommended one is the GNU arm embedded toolchain. You can find the latest release and platform specific installation instructions at

<https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>

DICE machines already have the gcc-arm toolchain installed, hence you do not need to reproduce this step (which wouldn't be possible anyway, given that super user privileges would be required).

SETTING UP A PYTHON VIRTUAL ENVIRONMENT

This step might not be required on your personal computer. On the other hand, given that you only have limited privileges on DICE, you will have to set up a virtual environment, so as to be able to install the necessary packages that will allow you to develop and deploy embedded applications. In these labs, we will rely on the Conda environment management system. Using this can also prove useful on your on machine, if you are working on multiple projects that have different dependencies and require different libraries.

To set up Conda for your DICE user, follow steps 2 and 3 in the guide available at <https://github.com/CSTR-Edinburgh/mlpractical/blob/mlp2018-9/lab1/notes/environment-setup.md> making sure that you given an appropriate name to the environment used for this course, e.g.

```
conda create -n iotssc python=3
```

INSTALL THE ARM MBED COMMAND-LINE TOOL

While you should be able to do most of the work only using the gcc-arm compiler, you will find out that you often need to meet certain dependencies and reuse already existing code that is public. To this end, we will use the Arm mbed command-line tool, which is Python based.

With the Conda environment active, install and configure the mbed-cli as follows:

```
pip install mbed-cli
mbed config -G GCC_ARM_PATH "/opt/gcc-arm-none-eabi/bin"
```

If you have downloaded the GCC ARM toolchain on your own (non-DICE) Linux system. Copy the toolchain folder to the "/opt/" directory and run:

```
mbed config -G GCC_ARM_PATH "/opt/gcc-arm-none-eabi-9-2019-q4-major/bin"
(Replace "gcc-arm-none-eabi-9-2019-q4-major" with downloaded toolchain name)
```

CREATE YOUR FIRST MBED PROJECT USING MBED-CLI

We will create our first project once again based on the LED blinking example. For this, first import the example, then navigate into that project folder and compile the code with nRF51-DK as the target platform. Following this sequence of commands will also deploy the application on the board:

```
mbed import https://github.com/ARMmbed/mbed-os-example-blinky
cd mbed-os-example-blinky
mbed compile --target K64F --toolchain GCC_ARM --flash
```

SYSTEM DEBUGGING

To board will also expose a serial port when connected to a host PC, which can be used for debugging purposes. First, find out what is the identifier of the serial port exposed on the machine that you are using. You can do this with the following command

```
mbed detect
```

This should produce output that indicates on which serial port the board is using. You should see an identifier of the form /dev/ttyXYZ0. To connect to that port and inspect the embedded system reporting, on DICE you can use the following command

```
screen /dev/ttyXYZ0 9600
```

Remember to modify `ttyXYZ0` to the actual identifier on your system. Here 9600 is the port's Baud rate.

EXERCISE

Starting from the example above, write an embedded application that executes a continuous loop in which the LED on the board is blinked once every second, each time with one of the red, green, and blue colours.