***Internet of Things Course***

**LAB 1**

**Getting Started with Embedded Programming**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc44064362)

[1.1 Lab overview 1](#_Toc44064363)

[2 Requirements 1](#_Toc44064364)

[3 Updating board firmware (optional) 4](#_Toc44064365)

[4 Creating your first Mbed online project 4](#_Toc44064366)

[5 Mbed Studio Setup 9](#_Toc44064367)

[5.1 Mbed Studio 9](#_Toc44064368)

[5.1.1 Preparing a workspace 9](#_Toc44064369)

[5.1.2 Creating or importing programs 9](#_Toc44064370)

[5.1.3 Build and running a program 11](#_Toc44064371)

[6 Setting up a Local Development Environment 12](#_Toc44064372)

[6.1 Installing the GNU embedded toolchain for Arm 12](#_Toc44064373)

[6.2 Install the Arm Mbed Command-line Interface (CLI) 12](#_Toc44064374)

[7 Create Your First Mbed Program Using the CLI 12](#_Toc44064375)

[7.1 System debugging 12](#_Toc44064376)

[8 Exercise 13](#_Toc44064377)

# Introduction

## Lab overview

In this lab, we will introduce the Arm Mbed compiler and learn how to set up a local environment for developing embedded applications.

At the end of the lab, you should know how to create a new Mbed project, import code example, compile, and execute applications for embedded devices.

# Requirements

In this lab, we will be using the following hardware:

* DISCO-L475VG-IOT01A embedded board

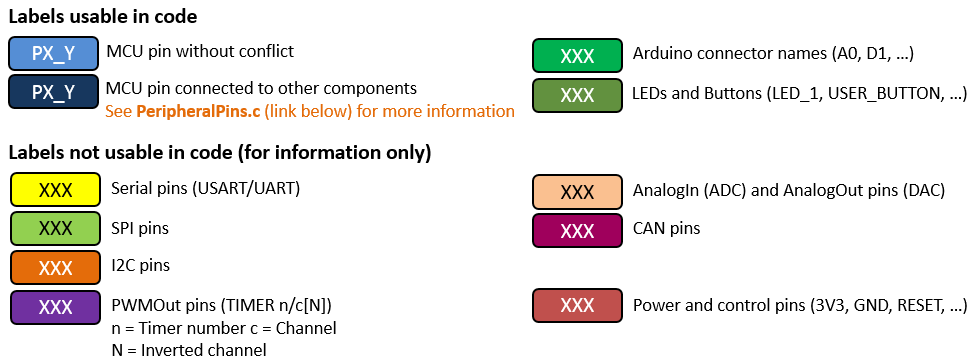


Figure : DISCO-L475VG-IOT01A board

**Board Features**

* 64-Mbit Quad-SPI (Macronix) Flash memory
* Bluetooth® V4.1 module (SPBTLE-RF)
* Sub-GHz (868 or 915MHz) low-power-programmable RF module (SPSGRF-868 or SPSGRF-915)
* Wi-Fi® module Inventek ISM43362-M3G-L44 (802.11 b/g/n compliant)
* Dynamic Near-field Communication (NFC) tag based on M24SR with its printed NFC antenna
* Two digital omnidirectional microphones (MP34DT01)
* Capacitive digital sensor for relative humidity and temperature (HTS221)
* High-performance 3-axis magnetometer (LIS3MDL)
* 3D accelerometer and 3D gyroscope (LSM6DSL)
* 260-1260hPa absolute digital output barometer (LPS22HB)
* Time-of-Flight and gesture-detection sensor (VL53L0X)
* Two push-buttons (user and reset)
* USB OTG FS with Micro-AB connector
* Expansion connectors:
  + Arduino™ Uno V3
  + PMOD
* Flexible power-supply options: ST LINK USB VBUS or external sources
* On-board ST-LINK/V2-1 debugger/programmer with USB re-enumeration capability: mass storage, virtual COM port, and debug port

**Board Pinout**



**Arduino Compatible Headers:**

A close up of a device

Description automatically generatedA picture containing implement, pencil

Description automatically generated

Figure : Arduino headers

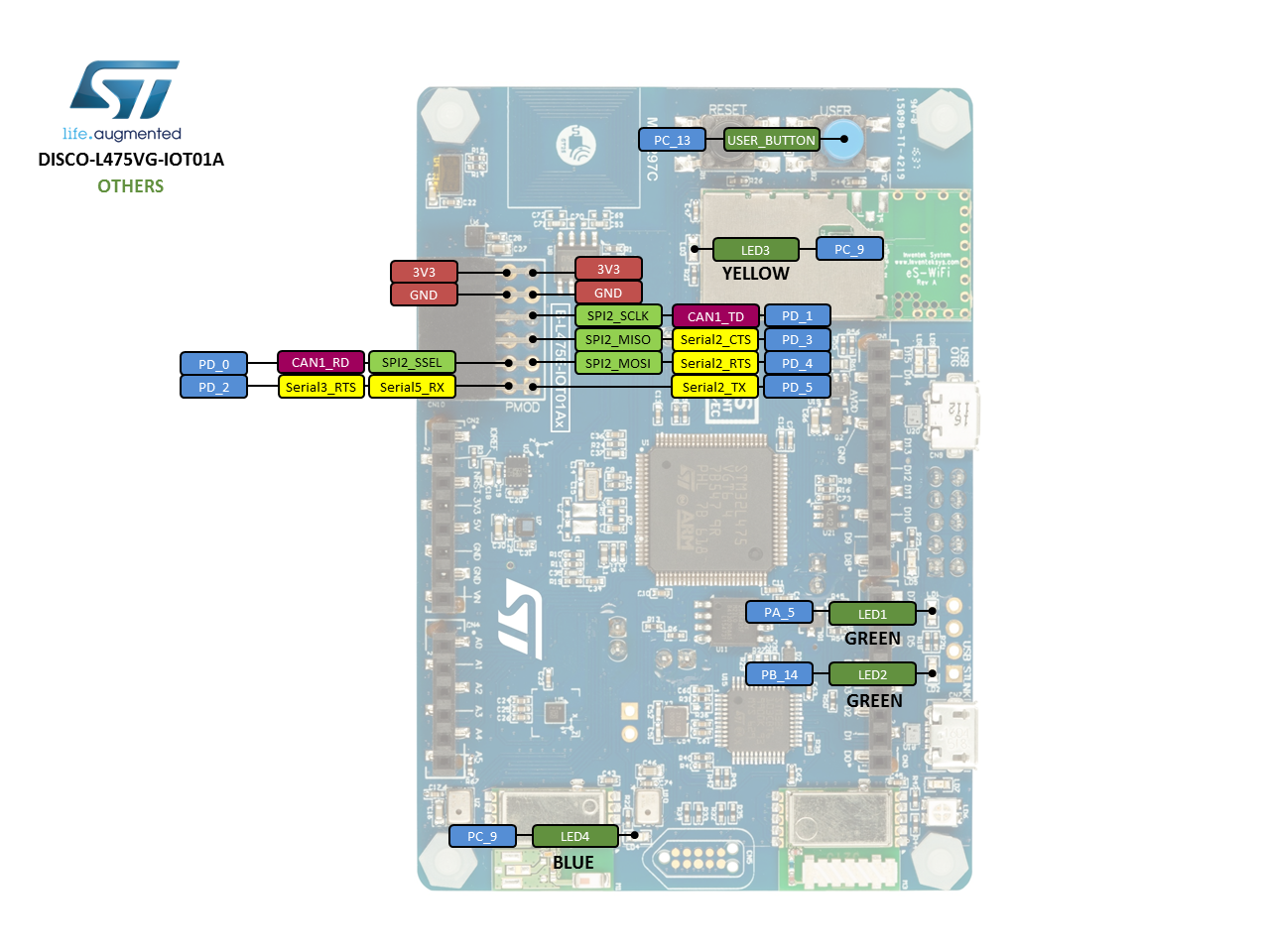


Figure : Board pinout

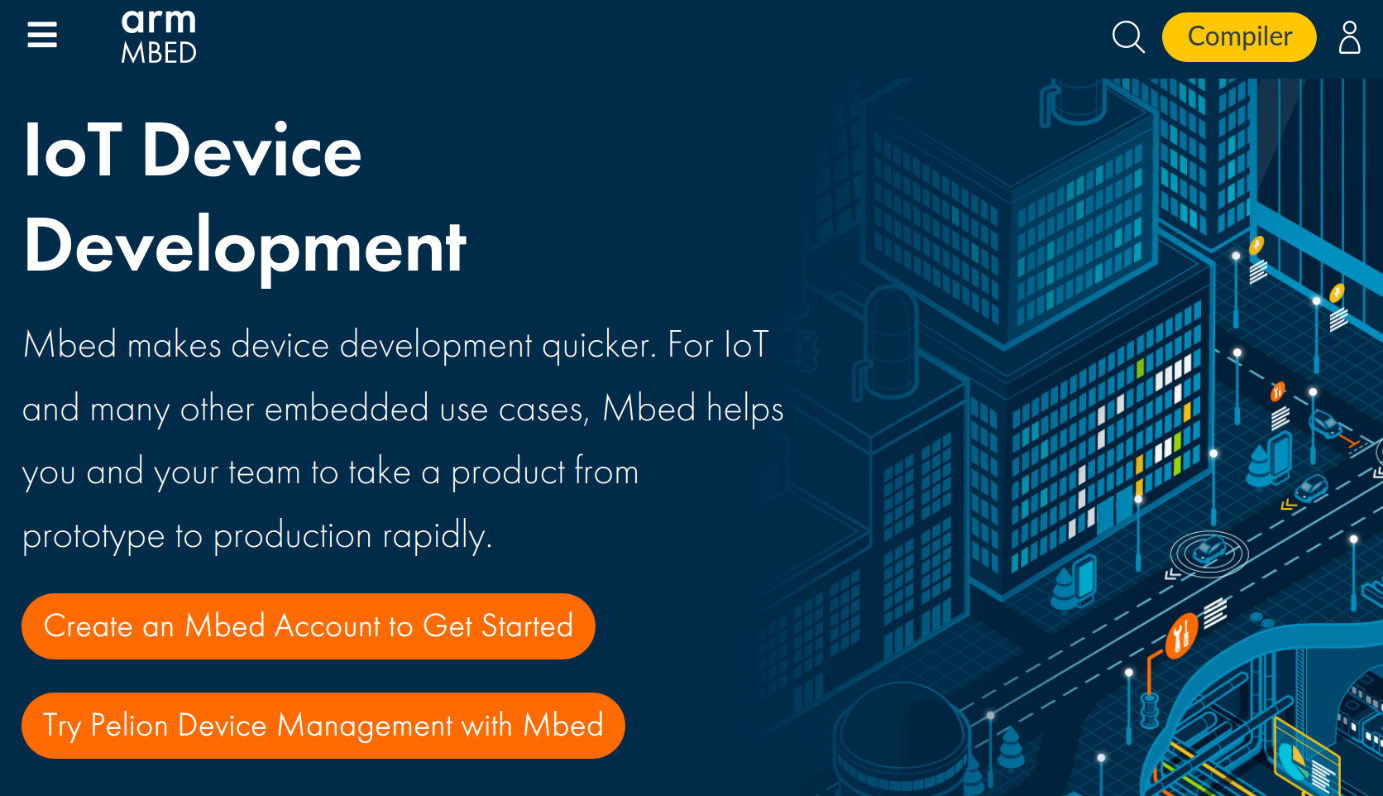
# Updating board firmware (optional)

* + Download the latest Mbed interface firmware from <https://armmbed.github.io/DAPLink/?board=MTB_STM32L475>
  + Press and hold down the reset button, while connecting the USB cable between the board and the computer.
  + It should now enumerate as “CRP DISABLD” (you can open a file browser to see if such a drive has been recognized).
  + Delete the file named firmware.bin, then drag and drop or copy the new bin file.
  + Wait for the file copy operation to complete.
  + Power-cycle the board. It should now enumerate and mount as DAPLINK or the name of the board.

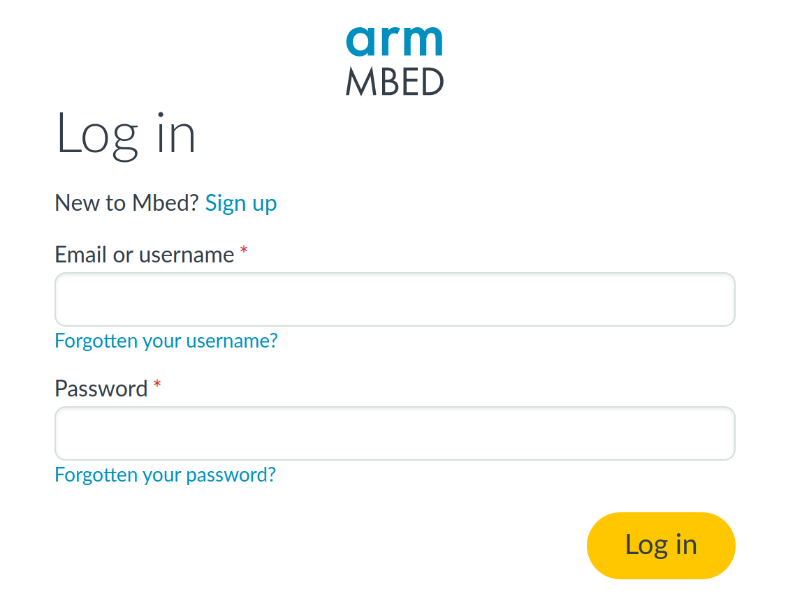
# Creating your first Mbed online project

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.

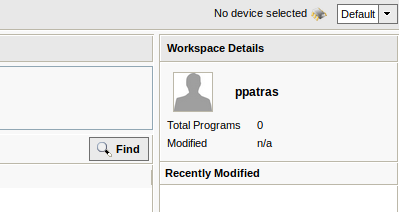
1. Go to https://www.mbed.com, and click “compiler”



1. Register an account and then login



1. Open the Compiler. The main Mbed IDE will be displayed. Here you can manage your projects, create new ones, import program from the mbed.org community…
2. Select the board on the top-right corner you are currently working with. If it’s the first time using this board press “Add a new platform.”



**user**



Browse for your device among all the Mbed enabled platforms. You can filter by manufacturer (**STMicroelectronics**) to find the DISCO-475VG-IOT01A board faster.

A screenshot of a cell phone

Description automatically generated

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

A screenshot of a cell phone

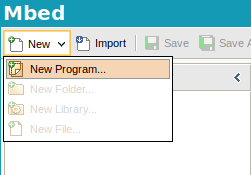
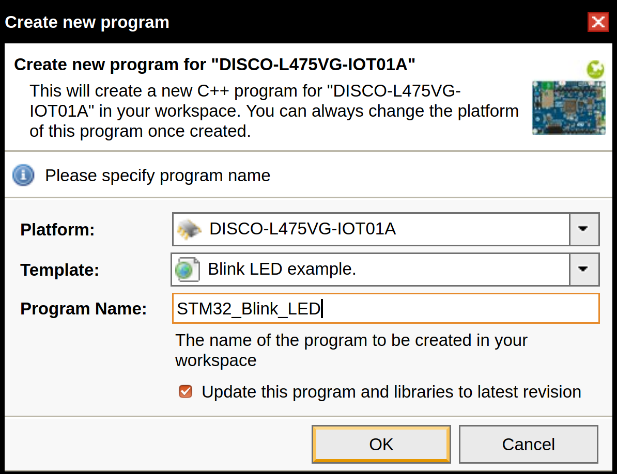
Description automatically generated

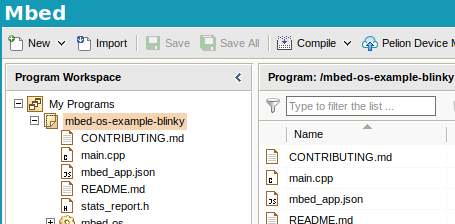
Go back to the Compiler interface. You should now be able to select this platform.

A screenshot of a cell phone

Description automatically generated

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



1. Compile the program file which will be generate a binary file that will be downloaded to your default download directory (set by your web browser).
2. Connect the board to your PC via an USB cable; the Mbed board will appear as a removable storage device.
3. Copy the downloaded program file to the Mbed board’s root directory.
4. Reset the Mbed board; the latest copied program file will be the default program to run.

# Mbed Studio Setup

## Mbed Studio

Mbed Studio is a free IDE that allows developers to create Mbed OS 5 applications and libraries. This will provide the functionality we require for completing the further labs.

Ensure that you have installed Mbed Studio from the following link: <https://os.mbed.com/studio/>.

You will also need to register an account and use it to sign in when launching the IDE.

### Preparing a workspace

Upon installation of Mbed Studio, a workspace named “Mbed Programs” is created for you in your home directory. A workspace is a location in the filesystem that contains your Mbed programs. This includes both imported and created projects.

You can change this workspace by clicking “File” and then selecting “Open Workspace”. Navigate to the folder you wish to make the new workspace and select it.



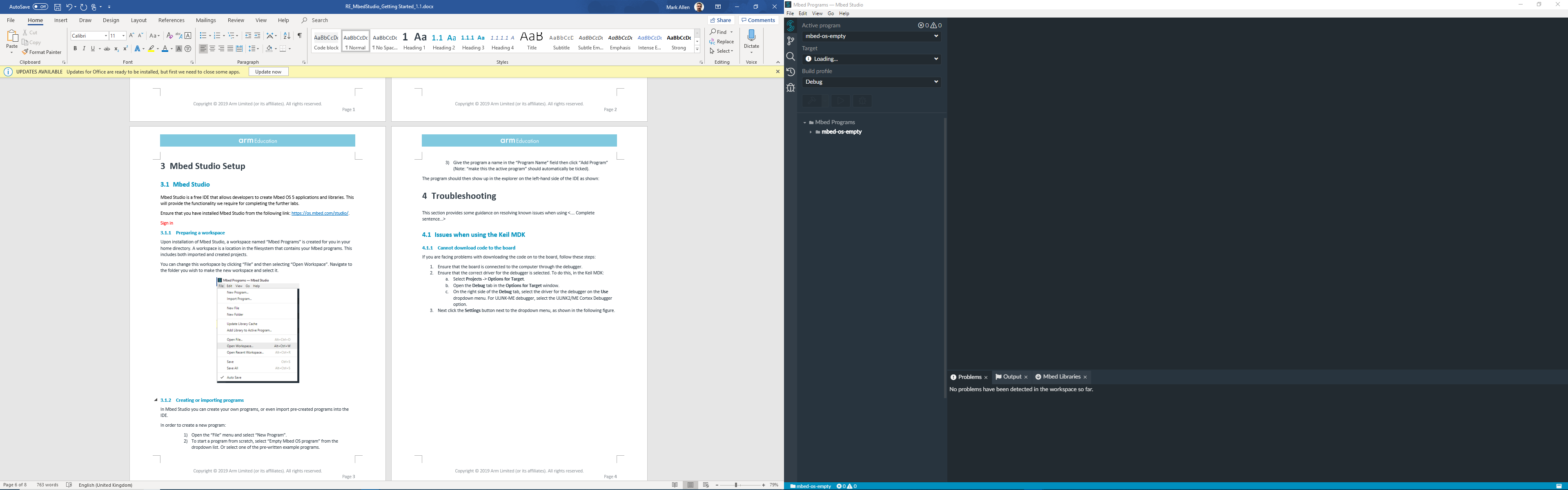
### Creating or importing programs

In Mbed Studio you can create your own programs, or import pre-created programs into the IDE.

In order to create a new program:

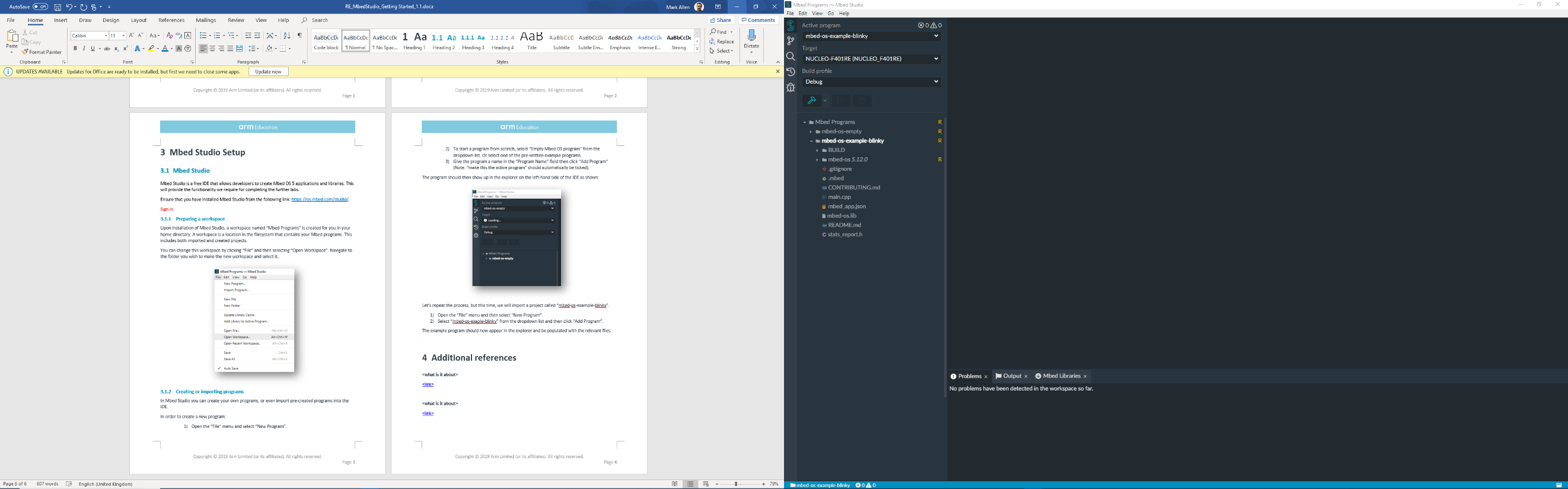
1. Open the “File” menu and select “New Program”.
2. To start a program from scratch, select “Empty Mbed OS program” from the dropdown list. Or select one of the pre-written example programs.
3. Give the program a name in the “Program Name” field then click “Add Program” (Note: “make this the active program” should automatically be ticked).

The program should then show up in the explorer on the left-hand side of the IDE as shown:



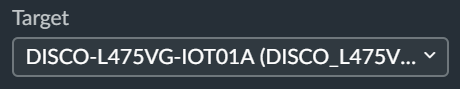
Let’s repeat the process, but this time, we will import a project called “mbed-os-example-blinky”.

1. Open the “File” menu and then select “New Program”.
2. Select “mbed-os-example-blinky” from the dropdown list and then click “Add Program”.

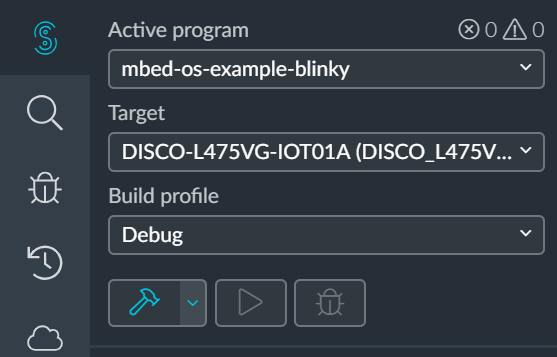
The example program should now appear in the explorer and be populated with the relevant files (this may require reopening the workspace to work):

### Build and running a program

Now you can plug in your Mbed enabled board and select it from the list of targets, in this case we will be using the Disco-L475VG-IOT01A:



Once the board is selected, ensure that “mbed-os-example-blinky” is the current active program and then select the “Build program” button:



As a result of the build process, Mbed will create a .bin file, which will contain result of compiling your program into machine code for whichever device you are using. Copying this file to your boards internal storage will have the effect of loading the program onto your board, at which point the program (for instance a blinking LED) should run!

Alternatively, pressing “Run program” should automatically build and load the program to your board. You may need to press the reset switch on the board in order for the new program to run.

# Setting up a Local Development Environment

You may often wish to develop embedded code locally to speed up deployment and avoid relying on the internet connectivity at all times. Follow the steps below if you wish to set up a development environment on your own machine.

## Installing the GNU embedded toolchain for Arm

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 DISCO-475VG-IOT01A development kit). There exist multiple compiler options, but the recommended one is the GNU Arm embedded toolchain (GCC). You can find the latest release and platform-specific installation instructions at <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>

## Install the Arm Mbed Command-line Interface (CLI)

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. On a Linux host type:

pip install mbed-cli

mbed config -G GCC\_ARM\_PATH "/usr/bin"

# Create Your First Mbed Program Using the 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 DISCO-L475VG-IOT01A 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 -m DISCO\_L475VG\_IOT01A -t GCC\_ARM -f

## System debugging

The 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 an output that indicates 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, 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 each LED, 1 and 2 on the board, blink once every second consecutively, then all of them flash simultaneously for a second, and the process repeats.