# Lab 1 – IoT and Embedded Systems

## Prerequisities

For this lab you will need:

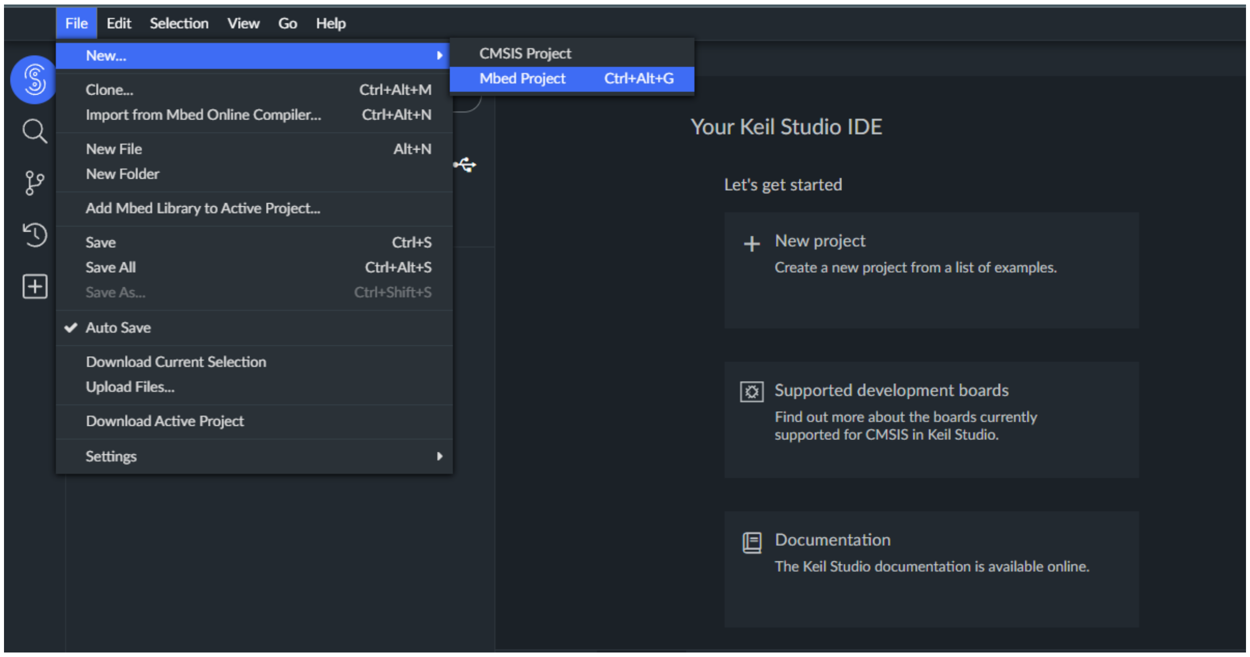
- STM32L4 Discovery kit IoT node (B-L475E-IOT01A)
- A USB Cable
- ARM Keil Studio Cloud (Requires registration)
- MATLAB (version 2021b)

## Part 1 : Introduction to Keil Studio

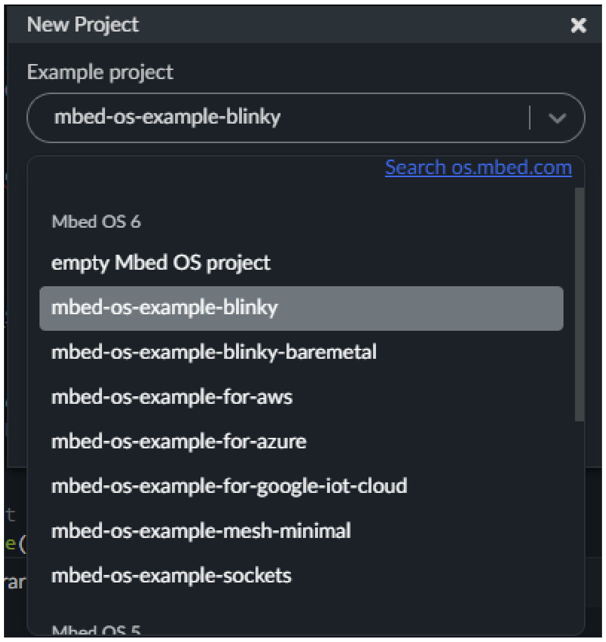
Arm Keil Studio is a free to use, browser-based IDE for the evaluation and development of embedded, IoT, and Machine Learning software for Cortex-M devices. To work with development boards over USB, you must use Keil Studio in a desktop browser that supports the WebUSB standard: Google Chrome or Microsoft Edge (Chromium).

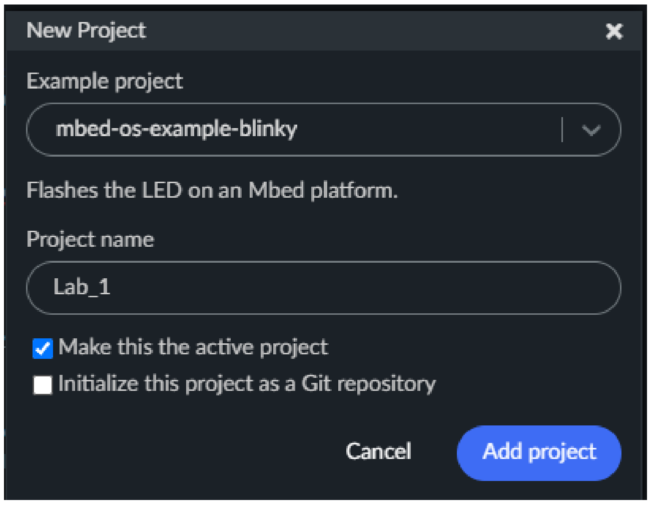
After registration, you can access the IDE. To verify the connection, we will setup an example from the environments predefined code templates.

First, create a new project. Click File > New... > Mbed Project or Click on New Project on Let's get started section:

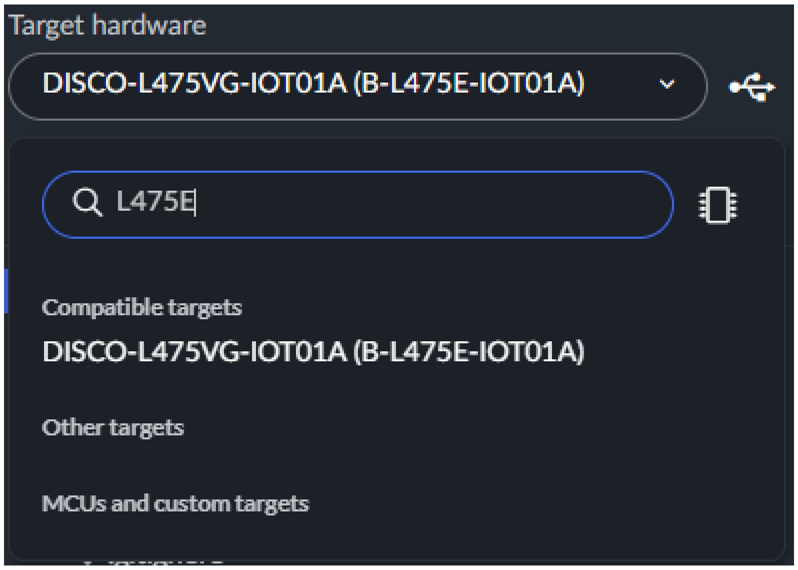

File
Edit
Selection
View
Go
Help
New...
CMSIS Project
Mbed Project Ctrl+Alt+G
Clone... Ctrl+Alt+M
Import from Mbed Online Compiler... Ctrl+Alt+N
New File Alt+N
New Folder
Add Mbed Library to Active Project...
Save Ctrl+S
Save All Ctrl+Alt+S
Save As... Ctrl+Shift+S
Auto Save
Download Current Selection
Upload Files...
Download Active Project
Settings
Your Keil Studio IDE
Let's get started
New project
Create a new project from a list of examples.
Supported development boards
Find out more about the boards currently supported for CMSIS in Keil Studio.
Documentation
The Keil Studio documentation is available online.

Choose mbed2-example-blinky. The name of the project is automatically set. Change it to "Lab 1". Make sure you have chosen the "Make this the active project" option.

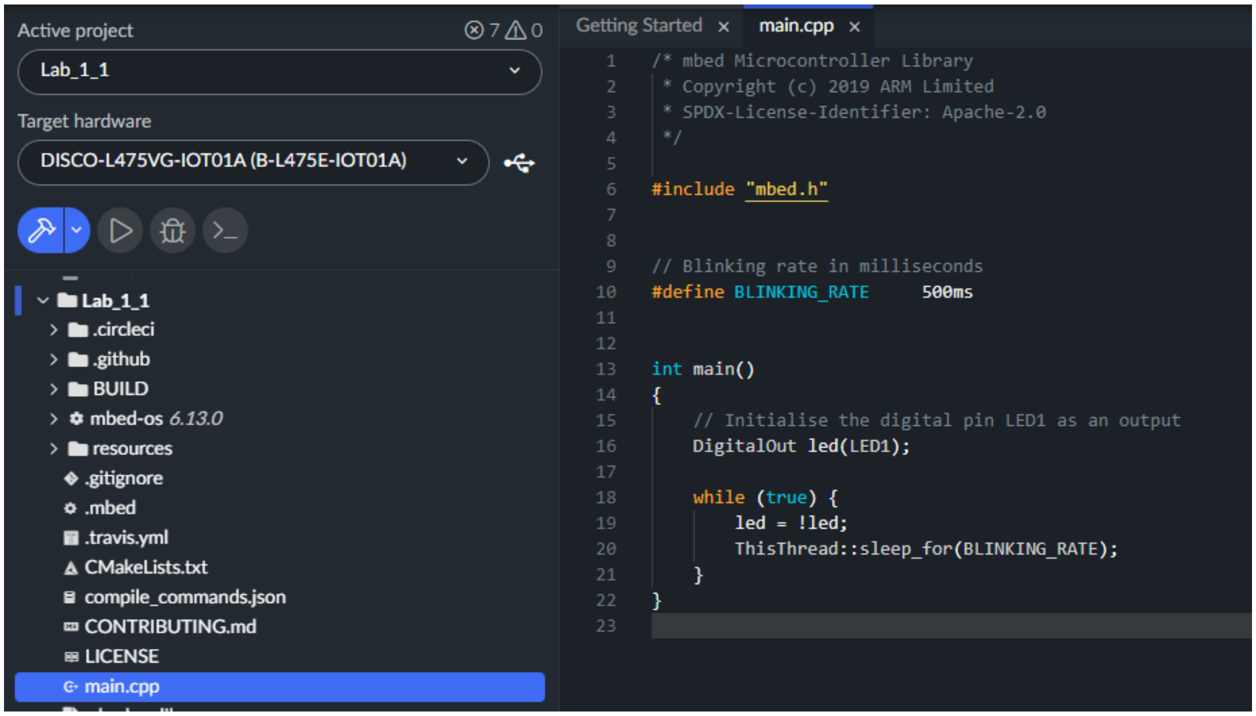



New Project
Example project
mbed-os-example-blinky
Flashes the LED on an Mbed platform.
Project name
Lab_1
Make this the active project
Initialize this project as a Git repository
Cancel
Add project

Then connect STM32L4 Discovery kit IoT node to your workstation (should be discoverable if you choose STM32 STLink). Several LEDs should light up on your board. Back on the IDE, go to Target Hardware and choose your device:



After that, simply run the project. After the build, the code is downloaded from your browser. You can review the code by opening main.cpp:



```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"


// Blinking rate in milliseconds
#define BLINKING_RATE     500ms


int main()
{
    // Initialise the digital pin LED1 as an output
    DigitalOut led(LED1);

    while (true) {
        led = !led;
        ThisThread::sleep_for(BLINKING_RATE);
    }
}

```

## Part 2 : Create a Serial connection

The objective of this exercise is to simulate the exchange of data among an edge device – your node- and a hub – your computer. The scenario involves the following:

- The hub sends some data to the node
- A led blinks to indicate data are read from the serial
- The node responds with the data that received

To establish communication, you first need to create a serial connection to both ends. First go to Keil and replace main.cpp code with the following:

```
#include "mbed.h"


// Maximum number of element the application buffer can contain

#define MAXIMUM_BUFFER_SIZE                                  32


// Create a DigitalOutput object to toggle an LED whenever data is received.

static DigitalOut led(LED1);


// Create a BufferedSerial object with a default baud rate.

static BufferedSerial serial_port(USBTX, USBRX);


int main(void)

{

   // Set serial port properties

   serial_port.set_baud(9600);

   serial_port.set_format(8, BufferedSerial::None, 1);


   // Application buffer to receive the data

   char buf[MAXIMUM_BUFFER_SIZE] = {0};

   // Continuously

   ????{

         // Read serial condition

          ????


          // Toggle the LED.

          ????


          // Echo the input back to the serial

          ????

      }
```

Replace the question marks with the required code. Next, open Matlab environment and create a new script named “Lab1_Serial_Comm”. Paste the following code in it:

```
%% Script to read constantly the Serial synchronously
clear all
close all

%Clear communication channel
delete(instrfind)

% Init Serial with the same settings as in node’s code
COM_PORT_NUMBER = "????"; % run the command ‘seriallist’ to retrieve this
baud_rate = ????;
data_bits = ????;
myComPort = serial(COM_PORT_NUMBER,'BaudRate',baud_rate,'DataBits',data_bits);

% Init variables
datain = '';
dataout = 'Hello World';

% Open communication channel
fopen(myComPort);

% Continuously
???
  % Send one or more characters to the STM32 device
  ????

  % Pause for a while to slow down the exchange
  ????

  % Read the available UART data as ‘uint8’ and display the result
  ????
   disp(char(datain'));
end

% Close communication channel
fclose(myComPort);
```

Replace the question marks with the required code. For the COM_PORT_NUMBER, run the command ‘seriallist’ in MATLAB command prompt to derive the active ports in your computer.