***Embedded Systems Essentials with Arm:  
Get Practical with Hardware***

**LAB 0**

**Introducing the STM Nucleo-64 Development Board and the Mbed Development Environment**

Contents

[1 Introduction 1](#_Toc67645674)

[2 Resources 1](#_Toc67645675)

[3 Introducing the Nucleo F401RE Board 2](#_Toc67645676)

[4 Simple Programs with Mbed Studio and the Online Compiler 4](#_Toc67645677)

[4.1 The Online Compiler 4](#_Toc67645678)

[4.2 Mbed Studio 6](#_Toc67645679)

[4.3 Taking Things Further, in Either Environment 8](#_Toc67645680)

[5 Linking the Nucleo to a Breadboard Circuit 8](#_Toc67645681)

[6 Troubleshooting 10](#_Toc67645682)

[7 For Background Reading 11](#_Toc67645683)

[8 Conclusion 11](#_Toc67645684)

[9 References 11](#_Toc67645685)

# Introduction

The aim of this laboratory (lab) is to introduce the STM Nucleo-64 development board, and to review the Mbed development environment. The laboratory has these objectives:

* to introduce the STM Nucleo-64 Development Board;
* to review use of Mbed Studio, and the online compiler;
* to download simple programs to the standalone Nucleo board, and demonstrate successful working;
* to build a simple circuit on a breadboard, and link to the Nucleo board, again with successful program download;
* to review simple debugging techniques.

Note that this lab series anticipates that you have some knowledge of C/C++ programming, you then get the opportunity to practice and develop these skills.

# Resources

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

**Mbed Studio**, an Integrated Development Environment (IDE) designed to streamline development and prototyping using Mbed enabled microcontrollers and development boards

**OR**

**Mbed online compiler**, a simple on-line IDE. Note that Arm may not support the online compiler indefinitely. However it remains a very useful and easy-to-use tool as long as it is there.

Hardware components you will need are shown in Table 1.

|  |  |
| --- | --- |
| **Item** | **Qty.** |
| STM32F401 Nucleo-64 Development Board | 1 |
| Bread Board | 1 |
| Jumper Wires (kit) | 1 |
| LED with internal current-limiting resistor | 1 |

**Table 1: List of Required Parts**

# Introducing the Nucleo F401RE Board

The Nucleo F401RE development board provides a powerful, low-cost and versatile platform for the prototyping of mbed-enabled devices. It’s one of a family of similar boards, each one with slightly different features. Reference 1 provides a User Manual for the full family. This is a useful resource, but gives more information then we would immediately need for these labs.

The Nucleo F401RE is pictured in Figure 1, with main features identified in Figure 2. An immediate point of interest is that the printed circuit board (PCB) is perforated, with interface circuitry on an upper sub-board, and the main circuit on a lower, larger one. Communication with the board is made through the USB mini B connector at its top; this is used to download programs, and as one means of providing power.

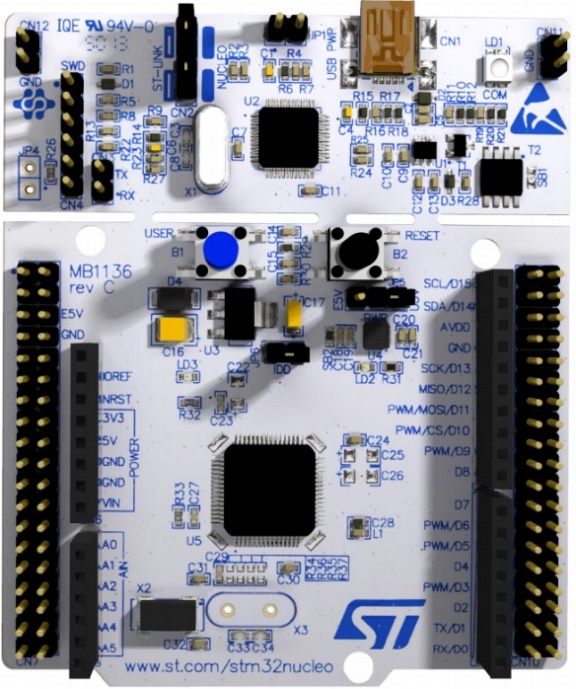


Figure 1: The NUCLEO F401RE board

At the heart of the Nucleo board is the powerful STM 32-bit microcontroller. This has an ARM Cortex-M4 core. The full power and versatility of the microcontroller can be glimpsed by viewing its data sheet (Reference 2); this is a major document and is absolutely not essential for this lab. Note the bicoloured (red/green) LED (LD1) on the top right of the Nucleo board, the power LED (LD3), the black Reset Button (B2), the green LED (LD2) below this, and the blue User Button (B1). LD2 is the only LED on the board which can be controlled by the user. There are a number of jumper links on the board. For our purposes these should not be changed or moved.

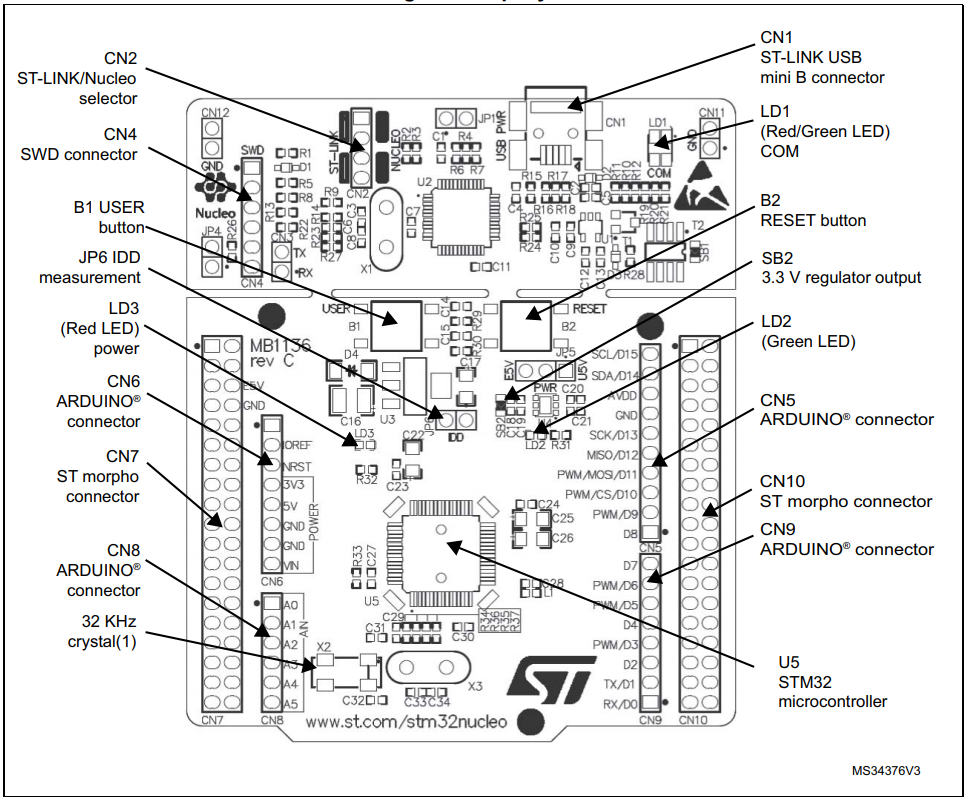


Figure 2: The Nucleo F401RE board, main features

The board has two main sets of connectors. One set (CN5, CN6, CN8, CN9) allows direct connection to Arduino expansion boards, while the other (CN7 and CN10) provides more complete interface capability. We will use the Arduino connectors, as detailed in Figure 3.

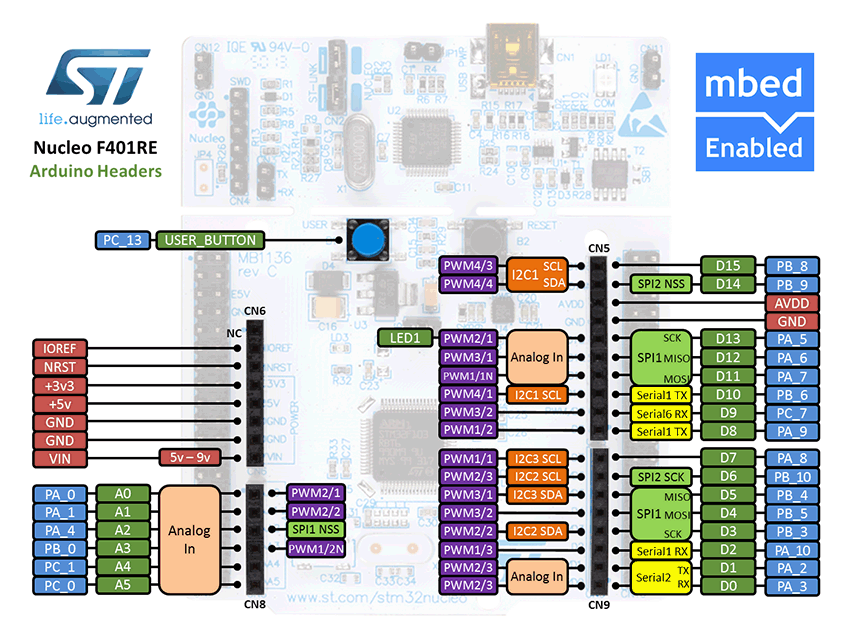


Figure 3: The Nucleo F401RE Arduino-Compatible Pin Descriptions

# Simple Programs with Mbed Studio and the Online Compiler

Two options for developing programs are outlined here. They are not the only ones, but do provide different opportunities. Mbed Studio contains the most features. It must be downloaded to your computer, which allows you to work offline, and at your own pace. The online compiler is an alternative but lacks some of the sophistication that Mbed Studio gives. Be warned that Arm are expected to stop supporting the online compiler at some time in the future. **We recommend using Mbed Studio for consistency throughout the labs, but will give an example of how to use the online compiler. The Accompanying lab videos will only use Mbed Studio.**

Read and apply either Section 4.1 or 4.2 below, depending on which development platform you plan to use. Note that the procedures and screen captures as shown were made using a PC running Windows 10. Some minor differences will occur with different computers or operating systems. Needless to say, Arm are continuously updating their software products, so changes due to this will also occur.

## 4.1 The Online Compiler

The compiler can be accessed at [https://ide.mbed.com/compiler](about:blank) . If you’re planning to use it, create an account as instructed. When you have registered and are viewing the compiler, select **New Program** (top left), and you will be offered the choices seen in Figure 4.

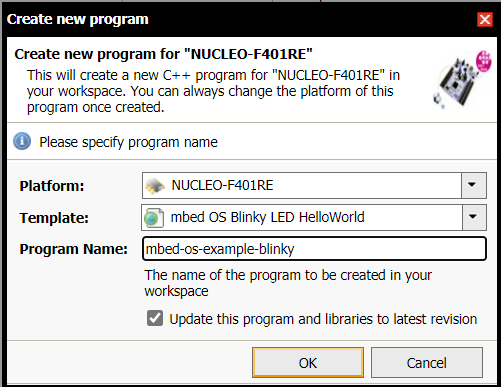
.

Figure 4: Selection Dialog Box for **new program**

You will need to select the correct platform for this development. The compiler remembers previous boards you have worked with. To select a new one, go to [Development boards | Mbed](https://os.mbed.com/platforms/) [Development boards | Mbed](about:blank), and find your chosen target board. In our case, you can go direct to [NUCLEO-F401RE | Mbed](about:blank) . Then click on the **Add to your Mbed Compiler** button shown, and the compiler will add the chosen board to its drop-down **Platform** list.



Choose the program template shown in Figure 4, and select a program name of your choice. Click OK, and you should then see the screen of Figure 5, which shows library resources linked to your new program.

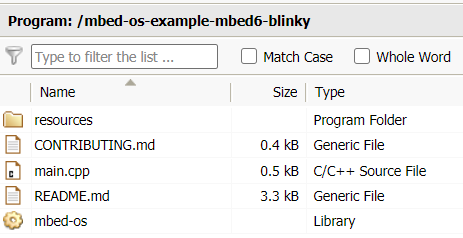


Figure 5: Library Resources Linked to New Program

Click on **main.cpp;** a program listing similar to Figure 6 should appear. Note that the templates are sometimes adjusted, so don’t worry if yours is a little different.

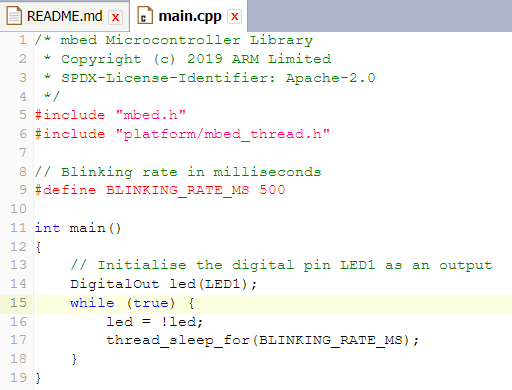


Figure 6: The mbed Blinky Template

Click **Compile** on the toolbar at the top of the screen. The compile process will start, with status displayed in the box shown in Figure 7.

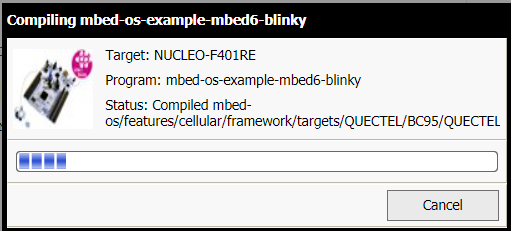


Figure 7: Compile status update

Compile output also appears in the sub-screen of Figure 8 (bottom right of the main screen). This may contain information or warnings which at present have little meaning to you. Don’t worry about these, but wait for the **Success!** statementwhich should finally arrive.

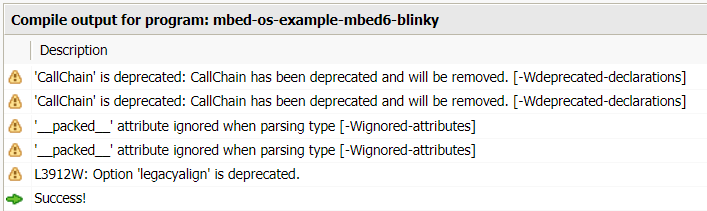


Figure 8: Compile status update

The final output of a successful compile is a binary file which can be downloaded (sometimes called “flashed” - because program memory use Flash memory technology) to program memory. This will appear as in Figure 9a). Plug your Nucleo board into the USB port of your computer, LD1 and LD3 should both light, indicating power is present. It should then be recognised and appear in File Explorer, as shown, labelled NODE\_F401RE, in Figure 9b).

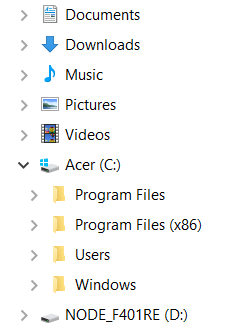
 

Figure 9 a) the Output Binary File, at Foot of Compiler Screen b) Nucleo Board Appearing in File Explorer

Drag the binary file of Figure 9a) to the Nucleo symbol in File Explorer. The bi-colour LED (LD1) on the board should flicker between red and green as download occurs. The program should then immediately start running, with LD2 blinking at the rate specified in the program.

## 4.2 Mbed Studio

Mbed Studio is a free Integrated Development Environment (IDE) for Mbed OS (Operating System) application and library development, including all the tools you need in a single package. You can thus create, compile and debug your Mbed programs on the desktop. Download Studio from [Mbed Studio | Mbed](about:blank) , and install.

To develop a program, you need to indicate the target hardware, and select a program template to work with. If you just plug in your Nucleo device to the computer, it should detect this and automatically select it as the target. Select **New Program**, and in the screen of Figure 10 choose **mbed-os-example-blinky** as template. This causes a folder of slightly over 1 Gbyte size (a significant memory demand) to be created on your hard drive, which will contain all necessary software features.

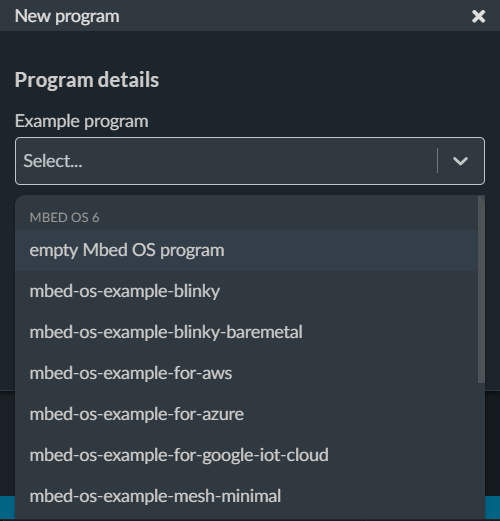


Figure 10: Selecting a Template in Mbed Studio

The screen of Figure 11 should then appear, giving a sub-screen on the right showing the program, as well as the project summary information and tools on the left. This can be toggled on and off by clicking on the Studio symbol,  , top left. Note the three important buttons for Build, Run and Debug, appearing in this sub-screen:  . The program displayed should be the same as, or very similar to that shown in Figure 6.

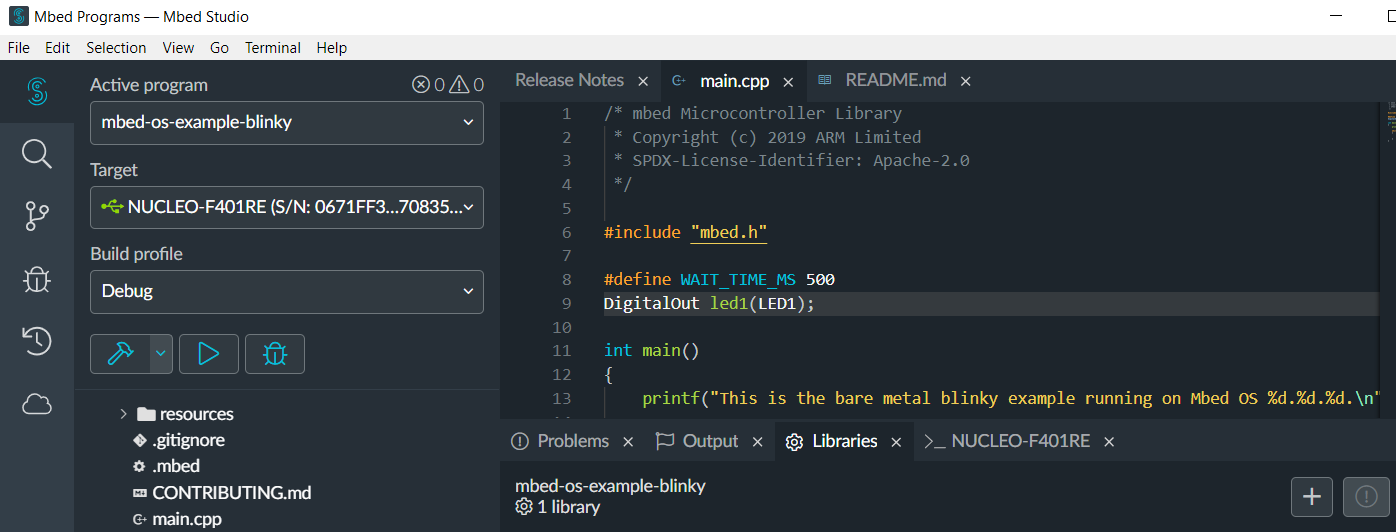


Figure 11: Mbed Studio Screen with blinky template selected

Click the Run symbol  to cause the program to compile and download. Even for a tiny program such as this, compilation in Mbed Studio -at least for the first iteration of any program - is an extended process. A rolling screen, bottom right, indicates progress. Finally it ends up with some useful data about the program build, as seen in Figure 12.

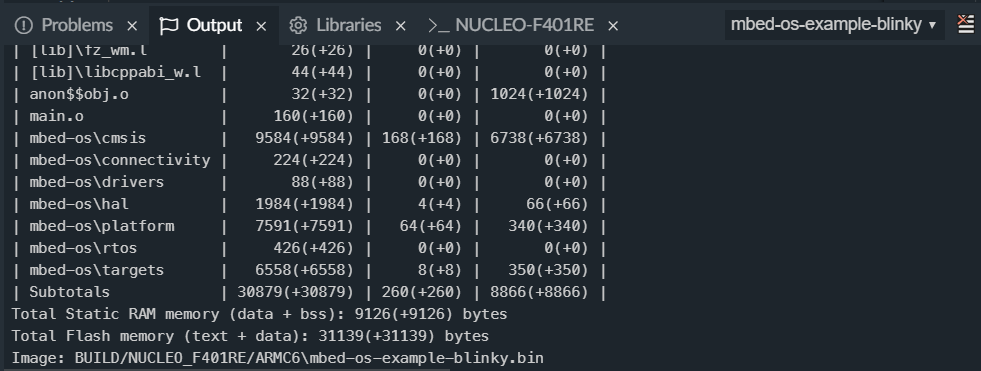


Figure 12: Mbed Studio Screen, Build Output

At the end of the Build process, if it has been successful, the binary program code is automatically downloaded to the Nucleo board. The bi-coloured LED should flash between its two colours during this – look out for it! The program immediately starts running; the green LED should flash at the rate prescribed in the program.

## 4.3 Taking Things Further, in Either Environment

Make simple changes to the program to gain experience of going through the development cycle of program change, compilation, download and test, and hence to persuade yourself that you’re in charge.

Try some or all of the following:

* Changing the blink rate;
* Emitting alternate long and short blinks;
* Repeatedly sending the Morse code signal for “SOS”, i.e. short short short long long long short short short, with a longer break (e.g. two seconds) between each transmission;
* Making use of the User Button as a program input to control the action of the LED. You can invoke the button with the line below, where **my\_name** is a variable name you choose.

DigitalIn my\_name(USER\_BUTTON);

# Linking the Nucleo to a Breadboard Circuit

Of course we’ve kept life rather simple by just running a program on the Nucleo board. In real life we need to connect to external circuits. We now take a simple solderless “breadboard”, and connect it to the Nucleo.

The parts for this first build are pictured in Figure 13. The breadboard allows wires and component pins to be inserted, making electrical connection with others within the same row of pins. Connections run lengthwise along each edge; in this version with blue and red lines running alongside – these are generally used for power supply and earth. Connections run crossways in the main body of the board, here giving five possible pin locations per connected line. Jumper wires can be purchased as kits, or can simply be made from single-strand solid-core insulated wire. It is very useful to have different colours and lengths.

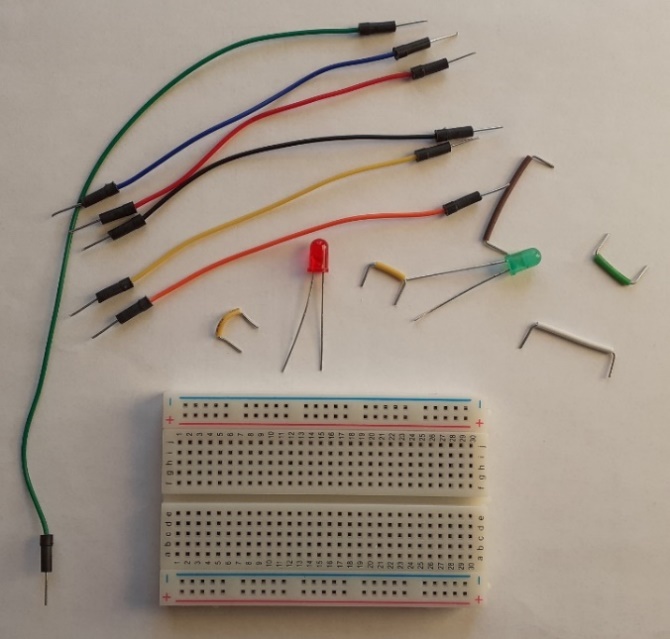


Figure 13: Parts for a Simple Breadboard Build

To make connection to the Nucleo, we now need to begin to understand in detail how the connectors work. Information was first given in Figure 3; Figure 14 simplifies this, and shows only the connections that we expect to use. Note that these connections are also marked on the Nucleo board itself. Input/output labels used on these connections are the same as can be used in a program.

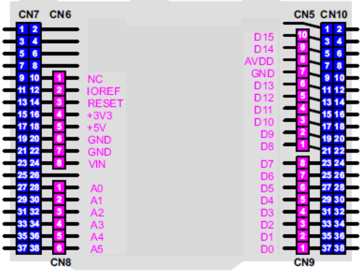


Figure 14: Connections on Nucleo Board, Arduino Connectors Only

A very simple circuit build is shown in Figure 15. Ground is linked from Pin 7 of Connector CN5, going to one of the lengthway breadboard connections. Then D10, chosen for LED output, is connected to the anode (i.e. the longer lead) of a red LED. The cathode of the LED is connected to ground.

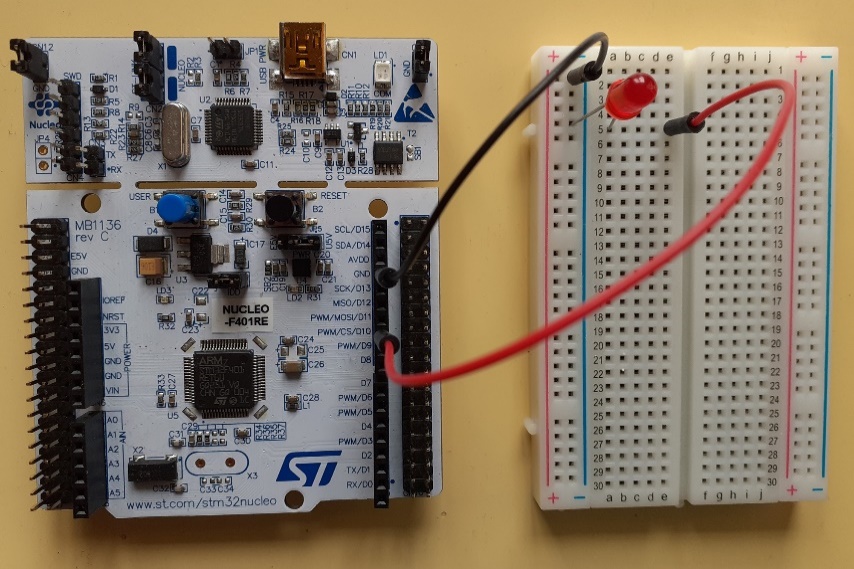


Figure 15: Mbed Studio Screen, Build Output

Enter the code of Program Example 1 into either the on-line compiler or Mbed Studio, compile and download. You should then see the red LED blinking at half the rate of the on-board one.

/\* Simple example of breadboard build for Lab 0

Off-board LED blinks at half the rate of on-board.

\*/

#include "mbed.h"

DigitalOut led1(LED1); // Initialise the digital pin LED1 as an output

DigitalOut led2(D10);

// Blinking rate in milliseconds

#define BLINKING\_RATE\_MS 500

int main(){

while (true) {

led1 = !led1;

thread\_sleep\_for(BLINKING\_RATE\_MS);

led1 = !led1;

led2 = !led2;

thread\_sleep\_for(BLINKING\_RATE\_MS);

}

}

Program Example 1: Blinking Off-board LED at half the Rate of On-Board

# Troubleshooting

Although the simple programs and circuits used in this lab are likely to work first time, it is worth getting into the mindset of systematic troubleshooting, for when things don’t go right. Everyone building and commissioning microcontroller-based devices has the experience at some time or other of switching on a circuit, and finding that nothing seems to work! The combination of untried software running in untried hardware creates particular demands on your troubleshooting skills. First check these points:

* Are all circuit connections correctly made? Check that power and ground are correctly connected, and that LEDs are the right way round.
* Is the board and circuit powered correctly? If you have either, use a digital voltmeter or oscilloscope to check power distribution.
* Has the program compiled and downloaded properly? In the Nucleo, check for that little flickering bi-colour LED at the moment of download.
* Is your program logical, and does it really do what you’re hoping? Remember that a successful compile only tells you that the program is grammatically correct, not that it will lead to correct functioning.

If you have checked all the above without success, return to a “known good” hardware and/or software version, and make incremental changes from there. “Known good” includes the “Blinky” program template in the on-line compiler or Mbed Studio, downloaded to the Nucleo board.

# For Background Reading

There isn’t at the time of writing a stand-alone manual for the Mbed API or operating system, although the guidance material on the web effectively constitutes an on-line manual. References 3-7 identify five of the main landmarks of this Mbed OS guidance; taken together they give deep insights into the Mbed programming environment. Get used to accessing and finding your way around these, as you get deeper into the Mbed world. References 8 and 9 are very good summaries of C and C++, providing useful “rule books” for each language. Most programs appearing in this lab series use only C, so it’s certainly worth having a copy of Reference 8. However the APIs themselves are of course written in C++, and as one moves into the C++ environment a copy of Reference 9 can also be useful.

# Conclusion

You should now have the core skills needed to create a simple program, and successfully download it to the Nucleo board. We build on these skills in the labs that follow.

# References

1. UM1724 User Manual. STM32 Nucleo-64 boards (MB1136). [https://www.st.com/resource/en/user\_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf](about:blank)
2. STM32F401xB STM32F401xC Data Sheet [Arm® Cortex®-M4 32b MCU+FPU, 105 DMIPS, 256KB Flash/64KB RAM, 11 TIMs, 1 ADC, 11 comm. interfaces (st.com)](about:blank)

*The five web pages below are significant landmarks of the online Mbed manual*

1. Introduction to ARM Mbed OS6

[Introduction - Introduction to Mbed OS 6 | Mbed OS 6 Documentation](https://os.mbed.com/docs/mbed-os/v6.6/introduction/index.html)

1. Full Mbed API listing

[Full API list - API references and tutorials | Mbed OS 6 Documentation](https://os.mbed.com/docs/mbed-os/v6.6/apis/index.html)

1. Mbed Tutorials and Examples

[Tutorials and official examples - Tutorials and examples | Mbed OS 6 Documentation](https://os.mbed.com/docs/mbed-os/v6.6/tutorials/index.html)

1. Mbed Components

[Components | Mbed](https://os.mbed.com/components/)

1. Mbed Forums

[Arm Mbed OS support forum - Get support for Arm Mbed OS from our community and support team](https://forums.mbed.com/)

*These books are excellent reference points while programming in C and/or C++*

1. Peter Prinz and Ulla Kirch-Prinz. (2002). *C Pocket Reference*. O’Reilly. ISBN 0-596-00436-2.

1. Kyle Loudon. (2003). *C++ Pocket Reference*. O’Reilly. ISBN 978-0-596-00496-5.