NANYANG POLYTECHNIC
EG431D Data Acquisition

Lesson 1

Course:	Diploma in Engineering (Electronic and Digital Engineering)
Module:	EG431D Data Acquisition
Title:	MCU and Arduino Development Platform

Objective:

This lesson introduces learners to the fundamentals of using Microcontroller Unit (MCU) as a Data Processing System. Learners will also become proficient with the Arduino Integrated Development Environment (IDE) for developing applications.

Learning Objectives:

- Describe the process of embedded system development using microcontrollers, including hardware and software integration, according to industry standards.
- Describe the significance of the Arduino platform in the evolution of prototyping and development for IoT and embedded systems.
- Explain how the Arduino development environment enables cross-compatibility between different hardware architectures and simplifies the creation of scalable embedded systems applications.
- Demonstrate the steps to install necessary libraries for application support.
- Demonstrate how to compile and link a sketch (application program), and how to upload and execute a sketch on an Arduino Board.

1. Introduction

The Microcontroller Unit (MCU) is a highly efficient, compact data processing system that plays a central role in managing and coordinating tasks within embedded systems. As the primary processor, the MCU combines a CPU, memory, and input/output (I/O) peripherals on a single chip, enabling it to handle real-time data processing while consuming minimal power. In sensor-based applications, the MCU collects raw data from various sensors, processes it using predefined algorithms, and takes immediate actions based on the analysis. Its architecture is optimized for executing specific, repetitive tasks quickly, making it ideal for real-time functions like controlling motors, adjusting lighting, and analyzing signals for defect detection. With on-chip memory for efficient data storage and support for communication protocols (e.g., UART, SPI, I2C), the MCU can seamlessly transmit processed data to other system components or external devices, positioning it as a vital controller for embedded and IoT applications.

Historically, application programs for MCUs were written in assembly language due to the need for low-level hardware control. However, the structured and user-friendly nature of the C programming language has made it a preferred choice for developing applications on microcontrollers. The Arduino Integrated Development Environment (IDE) further simplifies the process, allowing developers to write, compile, and upload code entirely in C. The open-source Arduino IDE provides a comprehensive, accessible tool for microcontroller programming, offering a one-stop solution for application development. Its support for C allows developers to create efficient and maintainable code, enhancing the overall functionality and performance of MCU-based applications.

The Arduino IDE features a user-friendly interface with a text editor for writing C source code, a message area, a serial console, a toolbar for common functions, and various menus for easy navigation. This environment provides basic error checking for C syntax, compiles the code, and establishes a communication link with the Arduino Board's onboard bootloader for uploading compiled machine code to the program memory. Running on a host PC, the IDE connects to the Arduino hardware via a USB cable, facilitating both program uploads and real-time communication. This streamlined development workflow enables rapid prototyping and testing, making it ideal for developers and hobbyists alike working on MCU-based projects.

In this lesson, we will become familiar with the Arduino Integrated Development Environment (IDE).

2. Setting Up the Working Directory

Before we start creating application software, let’s set up a working directory as your personal workspace for the semester.

(A) On Windows desktop, create a new folder and call it **EG431D**. To do it, you right click on the mouse to open a pop-up menu, choose menu item **New**, select menu sub-item **Folder**, and you should see a new folder created on the desktop. Rename it as **EG431D**.

(B) Please use the mouse and open the EG431D folder. Create a new folder within the EG431D folder and call it **libraries**. This newly created folder stores all custom libraries for Arduino.

(C) This EG431D folder will be known as the **working directory** or **user directory**. Please backup this directory to your portable hard disk drives or USB flash drives at the end of each lab session.

Note: You are only required to setup this directory once for the entire semester, and it is advisable for you to use the same PC throughout this semester.

3. Starting the Arduino IDE

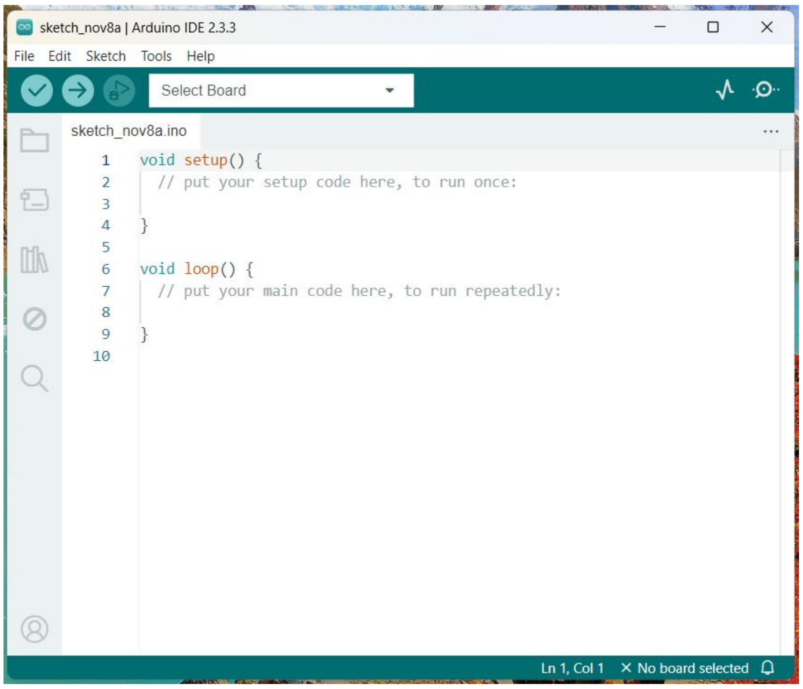
Next, we are ready to explore the Arduino Integrated Development Environment (IDE). An Arduino IDE comprises of:

1) Text Editor,
2) C Compiler and a Linker,
3) Serial Terminal (for debugging)
4) Serial Plotter
5) Application Loader,

On the pc desktop, you should see the following application icon for Arduino IDE (Figure 3.1):

Figure 3.1 – Arduino IDE Program Icon

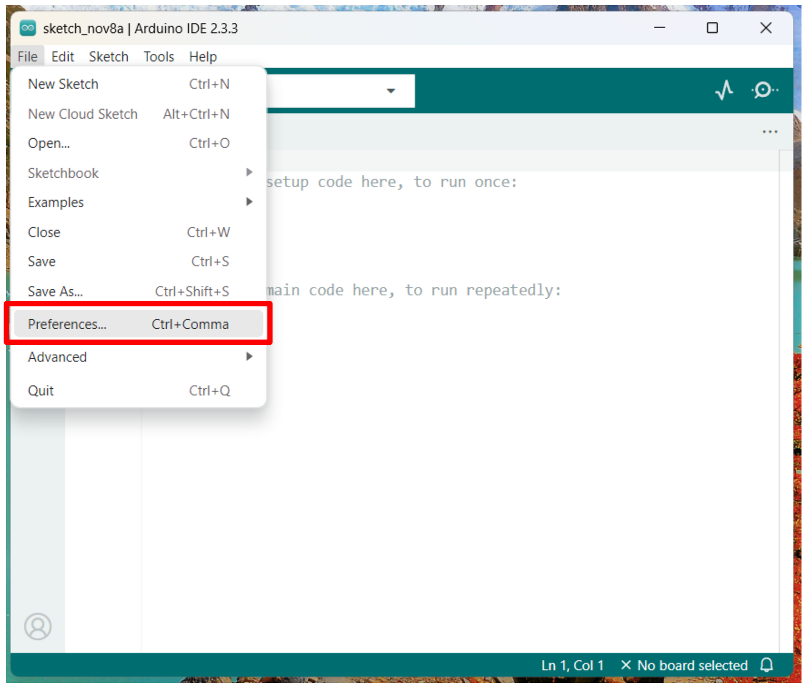
Double click on the icon and you should see the Arduino IDE application running (Figure 3.2) with a blank sketch.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 3.2 – Blank Sketch

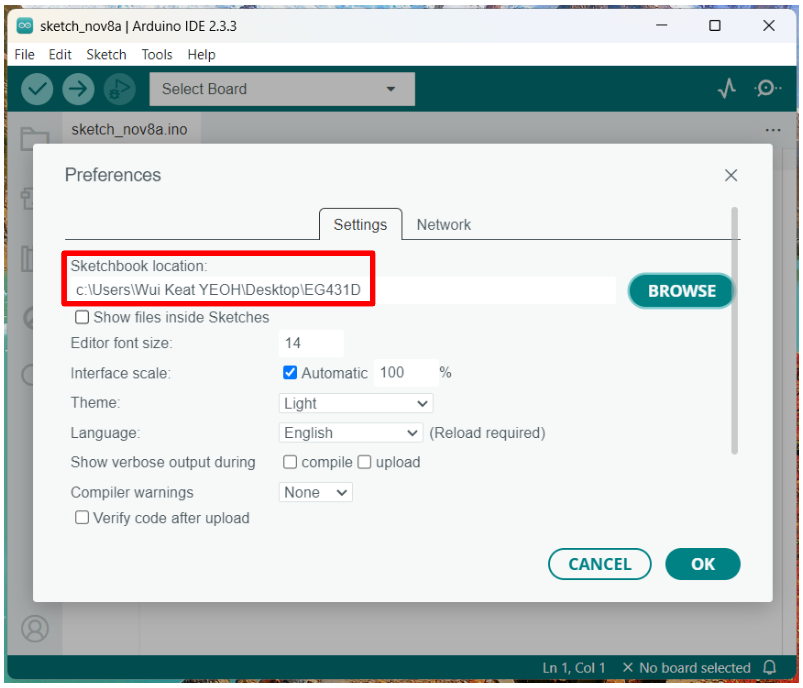
Click on the menu item **File** and select item **Preference** as illustrated in Figure 3.3.



```
setup code here, to run once:

main code here, to run repeatedly:
```

Figure 3.3 – File -> Preference

After clicking **Preference** you should see a pop up item page as illustrated in Figure 3.4. Click **Browse** to select the EG431D folder that you have just created. Once you have done, click on **OK** button to save the change.



Sketchbook location:
c:\Users\Wui Keat YEOH\Desktop\EG431D

☐ Show files inside Sketches

Editor font size:	14
Interface scale:	☑ Automatic 100 %
Theme:	Light
Language:	English (Reload required)
Show verbose output during	☐ compile ☐ upload
Compiler warnings	None

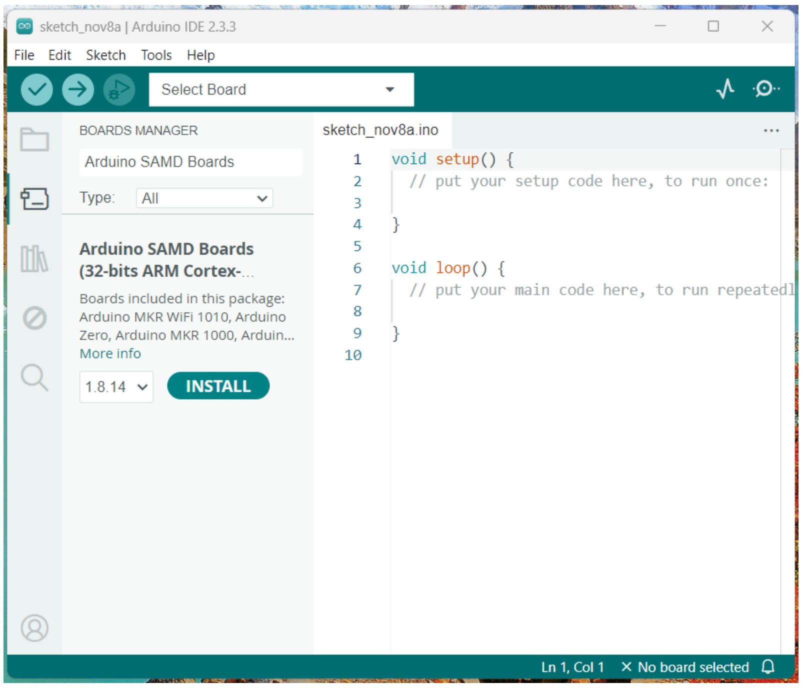
☐ Verify code after upload

Figure 3.4 – Preference Pop Up

4. Installing Board Support and Creating a new application program (sketch) in Arduino

4.1 Installing Board Support

We must specify the type of Arduino Board to be used for. We are using the 32-bit Arduino MKR WiFi 1010 board for this semester. By referring to Figure 4.1, click on the menu item **Tools**, select **Board** -> **Board Manager**. In the Search Entry Box, type **Arduino SAMD Boards** as illustrated in Figure 4.1. Press the **INSTALL** button to install the required Board Support files and wait for the installation to complete (Answer **Yes** to all questions asked and **Install** the required driver when prompted).



Arduino SAMD Boards (32-bits ARM Cortex-...

Boards included in this package: Arduino MKR WiFi 1010, Arduino Zero, Arduino MKR 1000, Arduin...
More info

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedl

}
```

Figure 4.1 – Board Manager

4.2 Creating a new application program (sketch) in Arduino

In the Arduino IDE, application program is called a sketch. It is the unit of code that is uploaded to and run on the Arduino Board. When the Arduino IDE is launched, it will open the last opened sketch. In this situation, click on the menu item **File** and choose **New**, and a new blank sketch will appear. This blank sketch is ready for you to enter C codes for a new application program.

We must also specify the type of Arduino Board to be used for the new application. We are using the 32-bit **Arduino MKR WiFi 1010** board for this semester. By referring to Figure 4.2, click on the menu item **Tools** and select **Board** -> **Arduino SAMD Boards** Scroll down the list of Arduino Boards and select **Arduino MKR WiFi 1010**.

Next is to set up a valid serial port for the Arduino IDE (PC) to communicate with the Arduino MKR 1010 board. The Arduino Board is required to be connected to the development PC via a micro-USB cable. Plug one end of the micro-USB cable to the micro-USB port on the Arduino Board, and the other end of the USB cable to the PC with Arduino IDE.

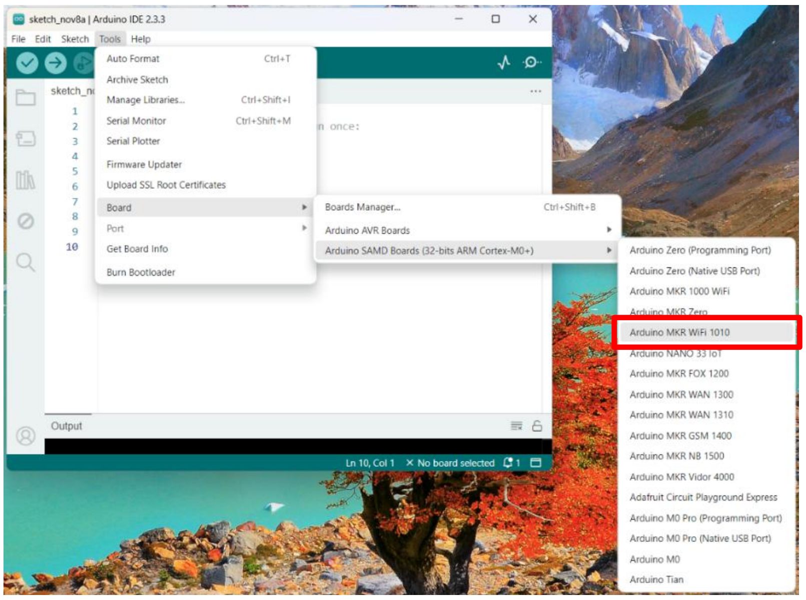

sketch_nov8a | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board
Port
Get Board Info
Burn Bootloader
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
Arduino SAMD Boards (32-bits ARM Cortex-M0+)
Arduino Zero (Programming Port)
Arduino Zero (Native USB Port)
Arduino MKR 1000 WiFi
Arduino MKR Zero
Arduino MKR WiFi 1010
Arduino NANO 33 IoT
Arduino MKR FOX 1200
Arduino MKR WAN 1300
Arduino MKR WAN 1310
Arduino MKR GSM 1400
Arduino MKR NB 1500
Arduino MKR Vidor 4000
Adafruit Circuit Playground Express
Arduino M0 Pro (Programming Port)
Arduino M0 Pro (Native USB Port)
Arduino M0
Arduino Tian
Output
Ln 10, Col 1 × No board selected 1

Figure 4.2 – Board -> Arduino SAMD Boards

After establishing a proper USB (serial) connection, click on the menu item **"Tools"**, select **"Port"** and search for a **COM** port that includes a label **Arduino MKR WiFi 1010** as illustrated in Figure 4.3. Click to select the required COM port. When this step is completed, please select **Arduino MKR WiFi 1010** as the connected Arduino Board from the drop-down menu as illustrated in Figure 4.4.

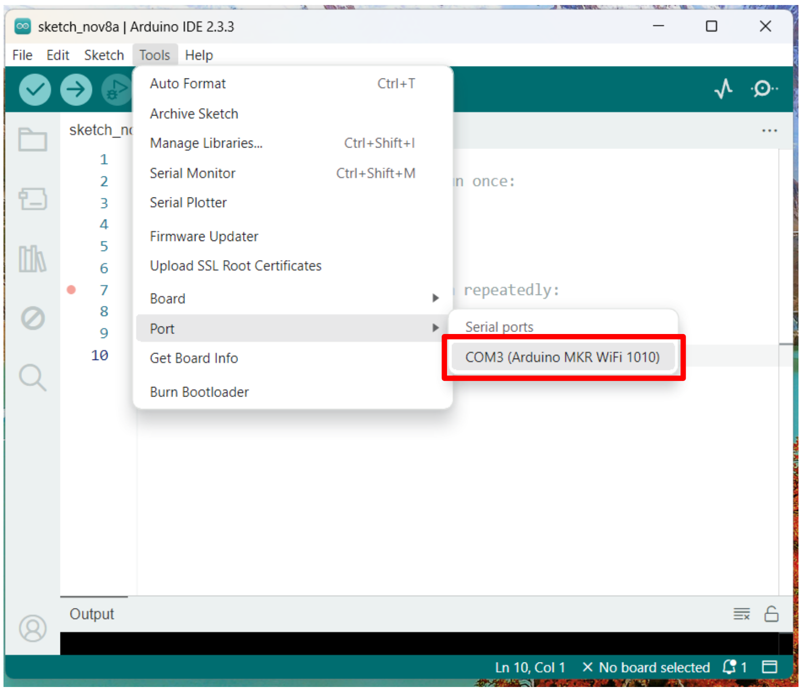
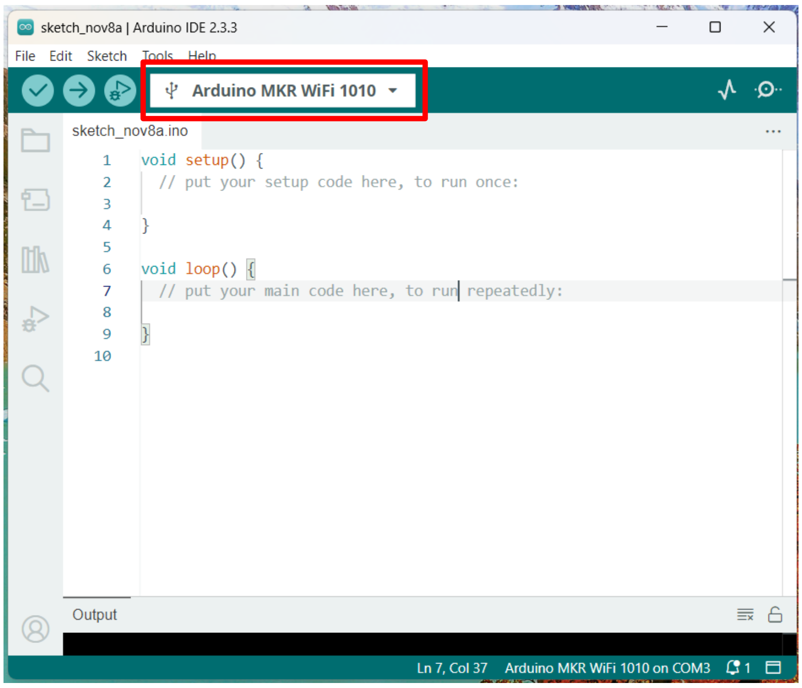

sketch_nov8a | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board
Port
Get Board Info
Burn Bootloader
Serial ports
COM3 (Arduino MKR WiFi 1010)
Output
Ln 10, Col 1 × No board selected 1

Figure 4.3 – Tools -> Port


sketch_nov8a | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino MKR WiFi 1010
sketch_nov8a.ino
Output
Ln 7, Col 37 Arduino MKR WiFi 1010 on COM3 1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}

```

Figure 4.4 – Select the Connected Arduino Board

In-order to verify that you have selected the correct connected Arduino Board and COM port, you can check on the Arduino IDE's board status message at the IDE's bottom right. Figure 4.4 shows a board status message **Arduino MKR WiFi 1010 on COM3** at the bottom right of the IDE. This message indicates that the Arduino Board is interfaced to a PC on COM3.

Up to this stage, you have successfully created a new blank Arduino sketch, installed the Board Support files, select a proper Arduino Board, and configure the required COM port for PC (Arduino IDE) to Arduino Board communication.

5. Install application support libraries for Arduino

Application development supports are in the form of custom or contributed code libraries. In Arduino IDE, libraries contain encapsulated functions for various tasks, algorithms and external hardware supports. These libraries can be installed from online repositories by using the built-in library manager. Click on the menu item **Sketch** and select **Include Library**. From the list, click on the first item **"Manage Libraries"** as illustrated in Figure 5.1.

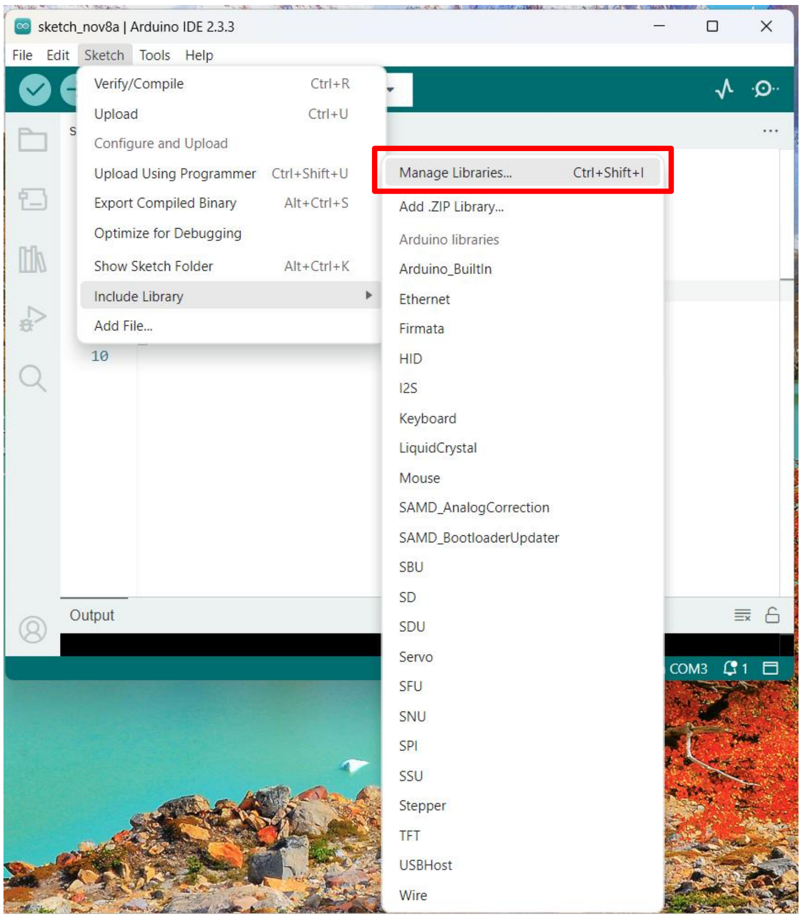


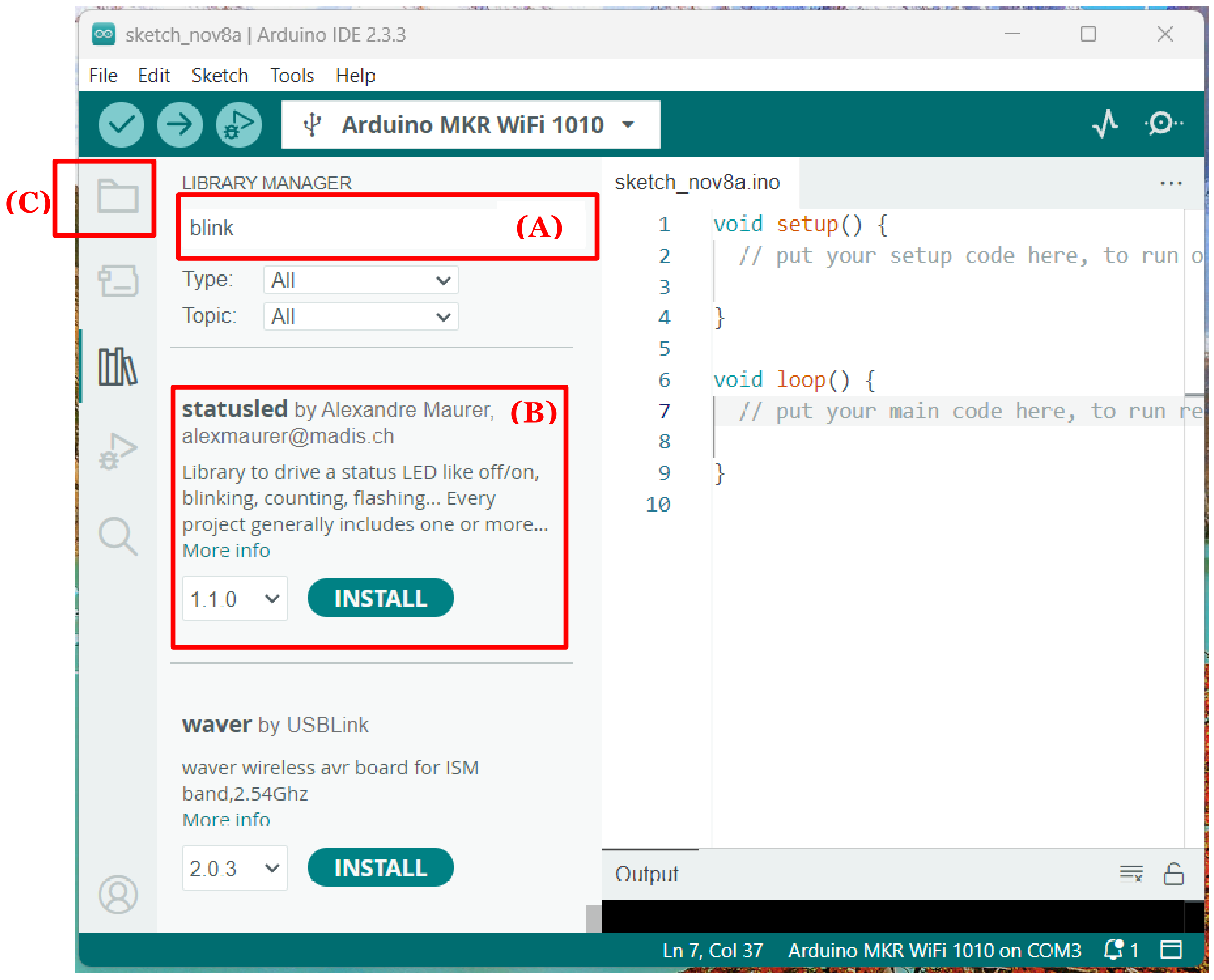
Figure 5.1 – Manage Libraries

A Library Manager window popup will appear. The Library Manager contains a search function for you to search and filter through the published libraries based on keywords. This will assist to narrow down the available libraries based on the entered keywords.

The Arduino IDE Library Manager manages open-source libraries for all types of Arduino Boards. The manager pulls in the required library source (online) and installs it locally in the **libraries** folder of your EG431D working directory. Prior to the start of application development, it is advisable to browse through the available libraries published by the Library Manager. You may not require developing custom function to perform a specific task if an equivalent library to support the task is available. This can effectively save some time and coding effort.

Let's try one example to install a **StatusLED** library for Arduino Board. Use the Library Manager and enter the keyword **blink** into the search area as shown in Figure 5.2 (A). You will see a list of libraries with the keyword. Browse through the list and click on **INSTALL** button to install the **statusled** library as shown in Figure 5.2 (B).

Once the library has been successfully installed, you will see the word **installed** beside the library name. Up to this stage, you have invoked the Arduino IDE Library Manager to install a library that provides Status LED functionality for the Arduino Board. Click on the **Folder Icon** as shown in Figure 5.2 (C) to exit the Library Manager.


sketch_nov8a | Arduino IDE 2.3.3
File Edit Sketch Tools Help
Arduino MKR WiFi 1010
(C)
(A)
(B)
LIBRARY MANAGER
sketch_nov8a.ino
Output
Ln 7, Col 37 Arduino MKR WiFi 1010 on COM3

LIBRARY MANAGER

blink

Type: All

Topic: All

statusled by Alexandre Maurer, alexmaurer@madis.ch

Library to drive a status LED like off/on, blinking, counting, flashing... Every project generally includes one or more...
More info

1.1.0 INSTALL

waver by USBLink

waver wireless avr board for ISM band,2.54Ghz
More info

2.0.3 INSTALL

```
void setup() {
  // put your setup code here, to run o

}

void loop() {
  // put your main code here, to run re

}
```

Figure 5.2 – Installing a StatusLED Library

6. Compile, Link and Execute a sketch in Arduino

Let's open a simple example to be targeted to the Arduino Board. By referring to Figure 6.1, click on the menu item **File** and select **Examples**. Navigate the list of examples to **01. Basics** and click on **Blink**.

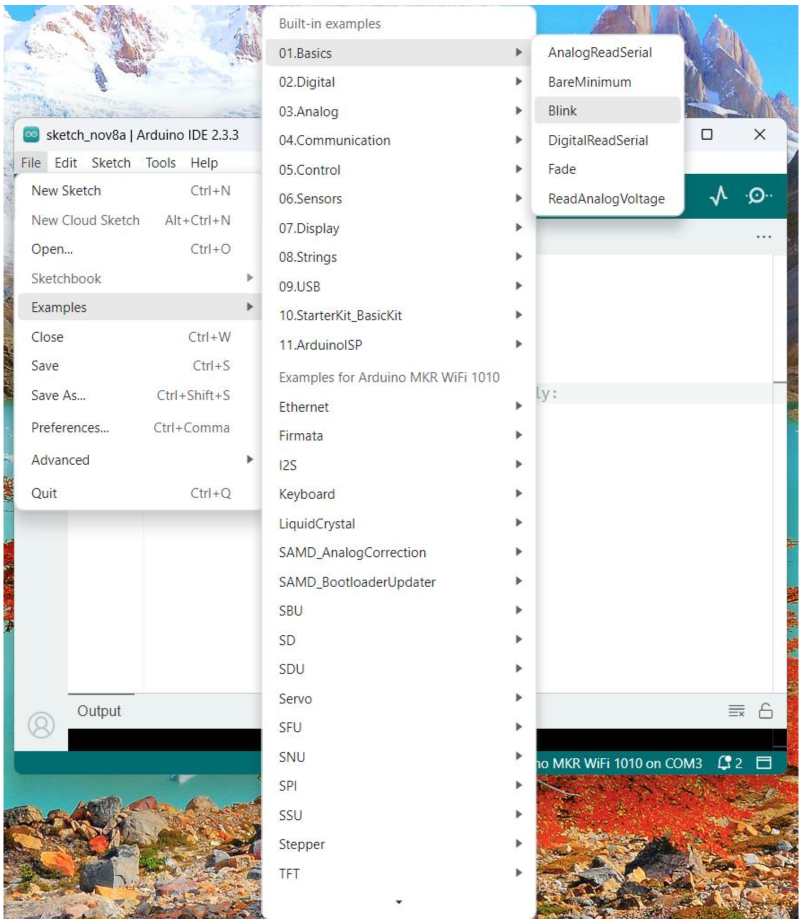
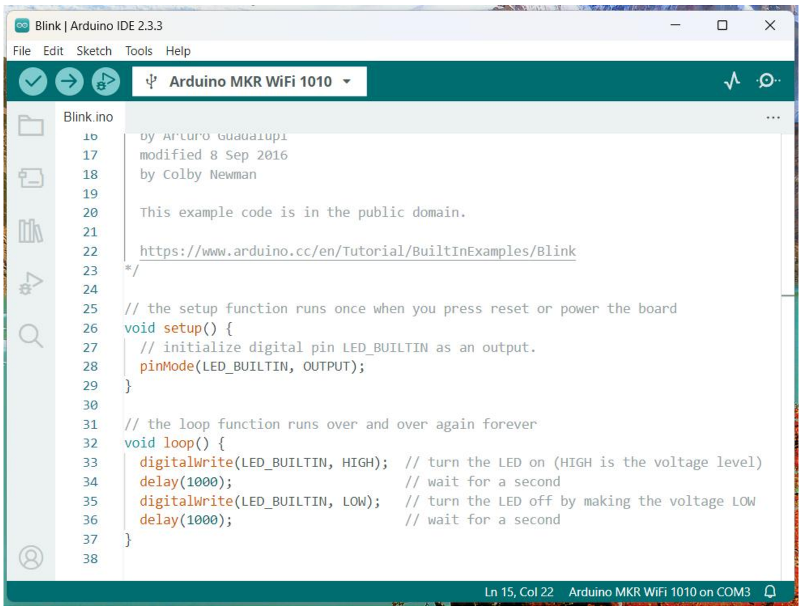


Figure 6.1 – Blink Example

The example sketch illustrated in Figure 6.2 will cause the LED on the Arduino Board to blink continuously at a fixed interval. Take note that this is a sketch from a provided example. You are required to save a copy of the example to your EG431D working directory that you have created earlier.

Click on the menu item **File** and select **Save As** to save a copy of the example code. A popup window will appear, navigates to the EG431D folder that you have created earlier, and provides a new file name to the Blink example as **Blink**. Click on **Save** to complete the procedure.



```
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

Ln 15, Col 22   Arduino MKR WiFi 1010 on COM3

Figure 6.2 – Newly Created Blink.ino

You are free to modify the saved copy of the **Blink** sketch. We now proceed to compile and link the sketch. Press the button on the Arduino IDE to compile and link the sketch. Take note of the compiler and linker messages shown at the bottom section of the IDE. You are required to clear any identified coding errors (if any).

Once the **Blink** sketch is compiled and linked without errors, you can upload the sketch to an Arduino Board. First, we need to connect the board to a PC via the provided USB cable. Once the board and the PC are physically connected, please verify that you have selected the correct Arduino Board and COM port.

You will proceed to upload the sketch to the Arduino Board by pressing button on the Arduino IDE. You will observe a series of activities at the bottom section of the IDE during the sketch uploading process. Once the upload has been completed, you will see the notice of completed uploading. The Arduino Board will now automatically execute the uploaded sketch.

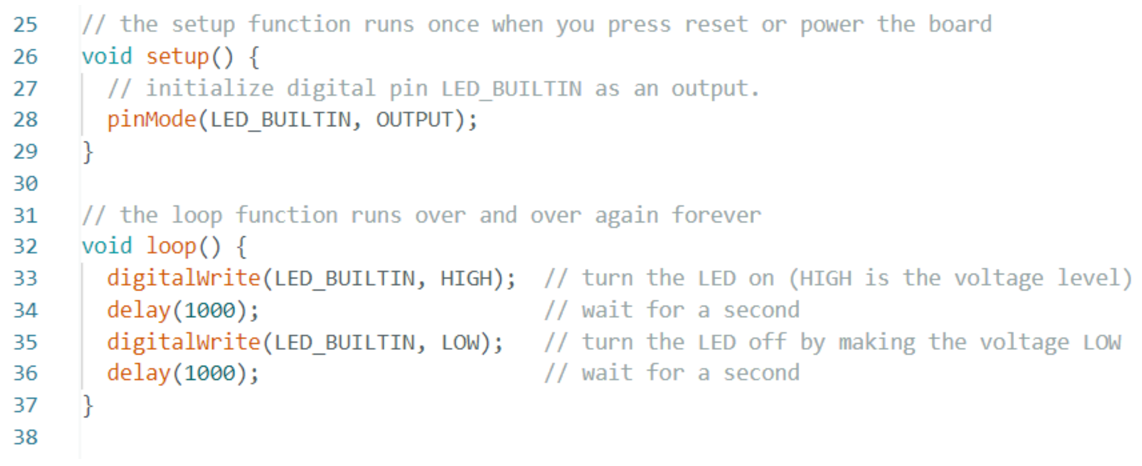
Take note that the button also performs the compile and link functions prior to uploading the sketch to the Arduino Board.

Observes the Arduino Board carefully to locate the blinking LED.

In this final stage, you have successfully compiled and linked a simple sketch (or application program) and upload the sketch to the Arduino Board.

7. Program analysis

The simple sketch blinks the on-board LED at a fixed rate. The code for blinking the LED is shown in Figure 7.1.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}

```

Figure 7.1 – Simple Sketch (Blink.ino)

Review Questions:

1. What is the time for one complete LED blink?
2. Analyze the code in Figure 8.1, can you identify the function that set the blink time?
3. If the time required for one complete LED blink is 1 second, how should the code in Figure 7.1 be modified to accommodate this change?

- The End -