



EE 3070 Design Project

Technical Training 1 Laboratory Manual (Arduino IDE and Programming)

2021/22 Semester A

Purposes of this training

- To get familiar with Arduino board and the programming platform (Arduino IDE)
- To get familiar with Arduino basic programming

Schedules

1. Each individual practises the exercises and completes the tasks given in this manual.
Remember to give demo to the tutors as specified in the manual.
2. If you finish this manual fast, you can go ahead to read Technical Training II manual and start to work on it
3. Take some time to read the report template to understand what are required.

Note: Students are not expected to leave earlier from the lab session.

1. Arduino Boards

Arduino is an open-source platform used for building electronic systems. It consists of hardware (microcontroller) and software tool (Integrated Development Environment, IDE). The IDE facilitates the programming and also downloads program to the hardware board.

More details can be found in the Arduino website. (<http://arduino.cc/en/Guide/Introduction>)

There are different types of Arduino boards with different features available in the market. In this technical training, two types of Arduino boards may be used. They are Mega 2560 ver 3 and Micro.

1.1 Arduino Mega 2560

Fig. 1 shows a photo of Arduino Mega 2560. It provides many I/O pins and memory, so that more complex projects can be implemented. Pin layout of Mega 2560 ver 3 is given in Fig. 2.

Self-Reading Material: <https://store.arduino.cc/usa/mega-2560-r3> ([Overview and FAQ](#))



Fig. 1. Photo of Arduino Mega 2560

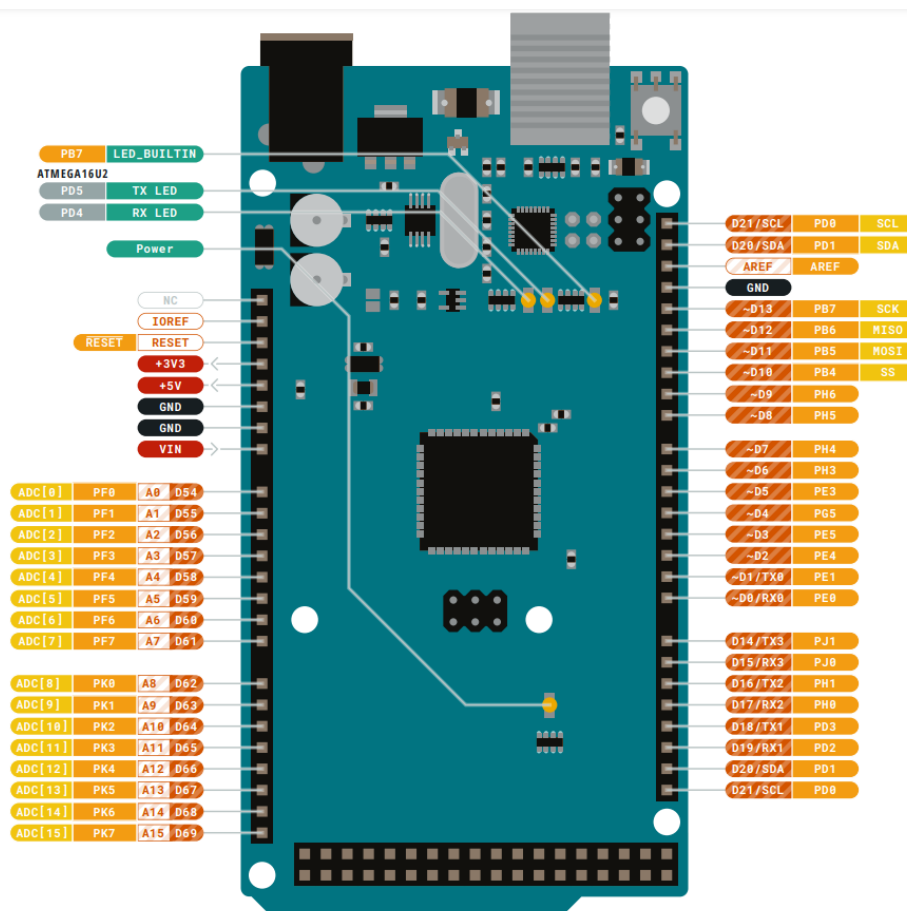


Fig. 2. Pinout of Arduino Mega 2560

1.2 Arduino Micro

Fig. 3 (a) and (b) show the photo of Arduino Micro and its pin layout, respectively.

Self-Reading Material: <https://store.arduino.cc/usa/arduino-micro> (Overview and FAQ)

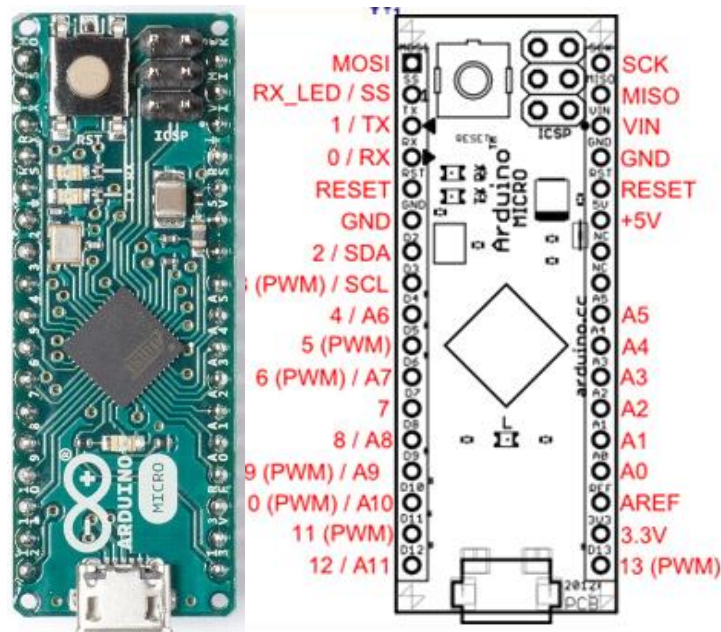


Fig. 3 (a) Photo (b) Pin layout of Arduino Micro

Arduino Programming

To program an Arduino board, you need to use the Arduino IDE. A web-version and a downloaded version are available. Here, only the downloaded version is focused. If you are going to use the web-version, you may visit the following URL and create your own account.

URL for web editor: <https://create.arduino.cc/editor>

1.3 Installation of Arduino IDE

The followings describe the procedures for installing the Arduino IDE and drivers to your PC/notebook.

- According to the OS of your PC/notebook (eg. Windows or Mac OS X), download the corresponding Arduino IDE from <https://www.arduino.cc/en/Main/Software>. You can click **Just Download**.

The screenshot shows the Arduino IDE 1.8.15 download page. The page has a teal header with navigation links: HARDWARE, SOFTWARE, CLOUD, DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. Below the header, the word "Downloads" is displayed. The main content area features the Arduino logo and the text "Arduino IDE 1.8.15". It describes the IDE as open-source software for writing and uploading code to Arduino boards. Below this, there is a "SOURCE CODE" section mentioning GitHub. To the right, a "DOWNLOAD OPTIONS" section lists various download links for different operating systems and architectures. Yellow arrows point from the "Windows" and "Mac OS X" labels to the corresponding download options. The "Windows" options include "Windows Win 7 and newer" and "Windows ZIP file". The "Mac OS X" option is "Mac OS X 10.10 or newer". Other options include "Windows app Win 8.1 or 10", "Linux 32 bits", "Linux 64 bits", "Linux ARM 32 bits", and "Linux ARM 64 bits". At the bottom of the download options, there are links for "Release Notes" and "Checksums (sha512)".

Operating System	Download Options
Windows	Windows Win 7 and newer Windows ZIP file Windows app Win 8.1 or 10
Mac OS X	Mac OS X 10.10 or newer
Linux	Linux 32 bits Linux 64 bits Linux ARM 32 bits Linux ARM 64 bits



Fig. 4. Screen showing the download of Arduino IDE.

- b) Run the downloaded executable file (eg. Arduino-1.8.x-windows.exe for Windows) to install the Arduino IDE, along with the driver if needed. (Note: The installation of IDE may need the Administrator password of the PC/Notebook. If you are using the PC in the Lab and need to install the Arduino IDE, please contact the in-charge technician/demonstrator.)

1.4 Brief Guides for IDE

Connect the Arduino board to the PC using a USB cable. Wait until the initialization of the board finished (Note: The orange IDE on the board no longer blinking). For the first time, it may take a moment.

- a) Start Arduino IDE, check the settings
- **Board:** Select Arduino Micro OR Arduino Mega, according to your Arduino board

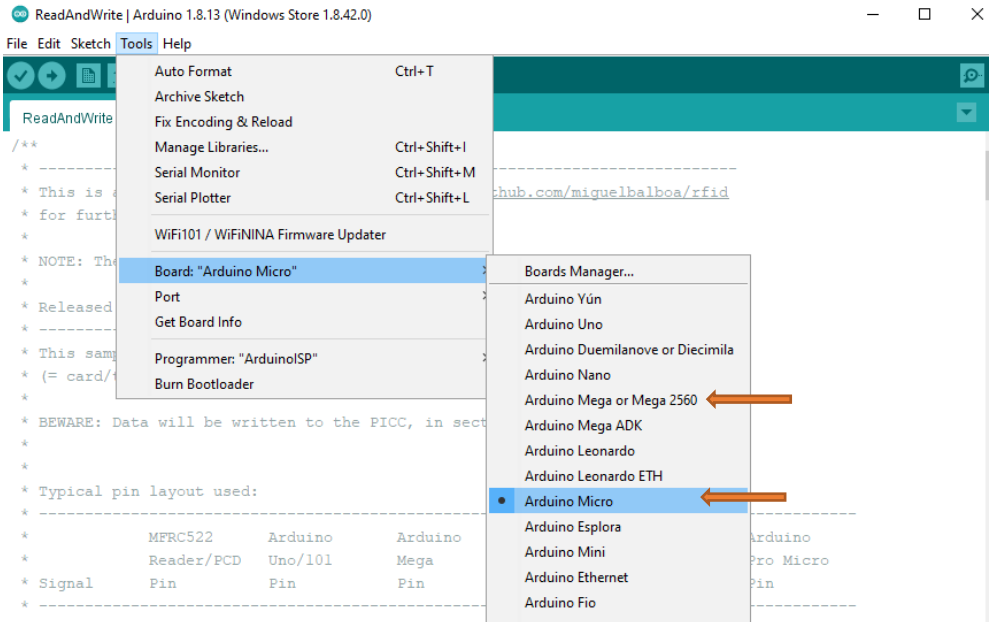


Fig. 5. Selection of Arduino boards

- **Port:** Select the proper communication port; Com XX (Arduino Micro) or (Arduino Mega) [Your Port number may be different from the example shown in Fig. 6]

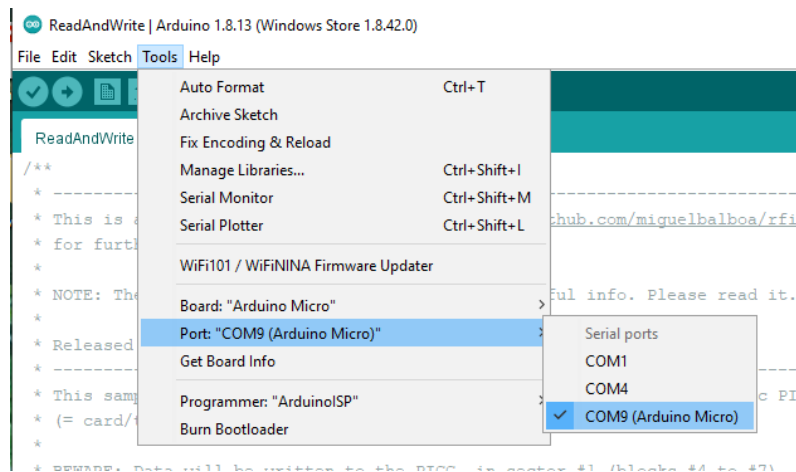


Fig. 6. Selection of communication Port

1.5 Managing Library

For some tasks, specific libraries may be needed. Procedures to add a library are as follows.

- a) Click Tools->Manage Libraries.

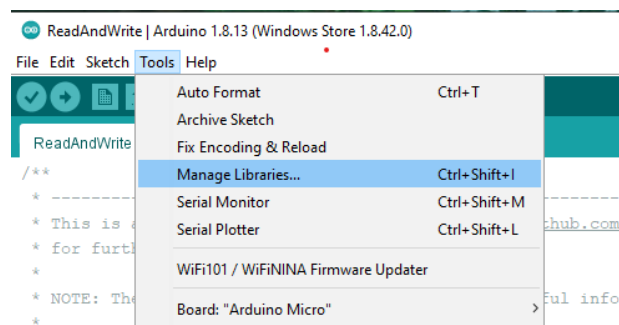


Fig. 7. Menu for Manage Libraries

- b) Search the library by typing its name and press Enter. The corresponding library will be shown. You can then install the version that you want.

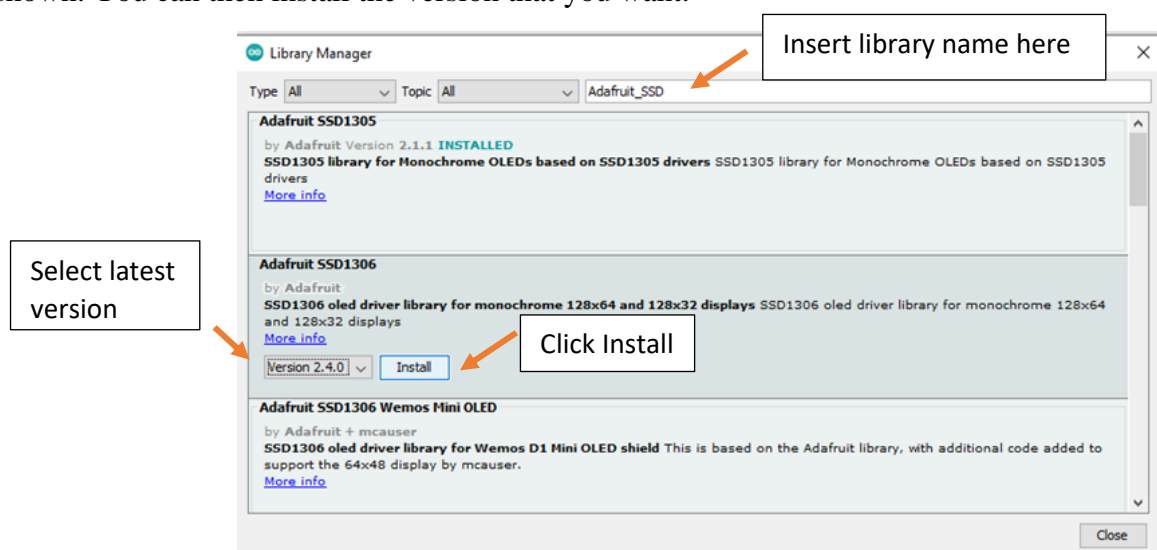


Fig. 8. Search and install Libraries

1.6 Examples in the Libraries

Usually, examples are provided by the libraries. They can be found by click File->Examples.

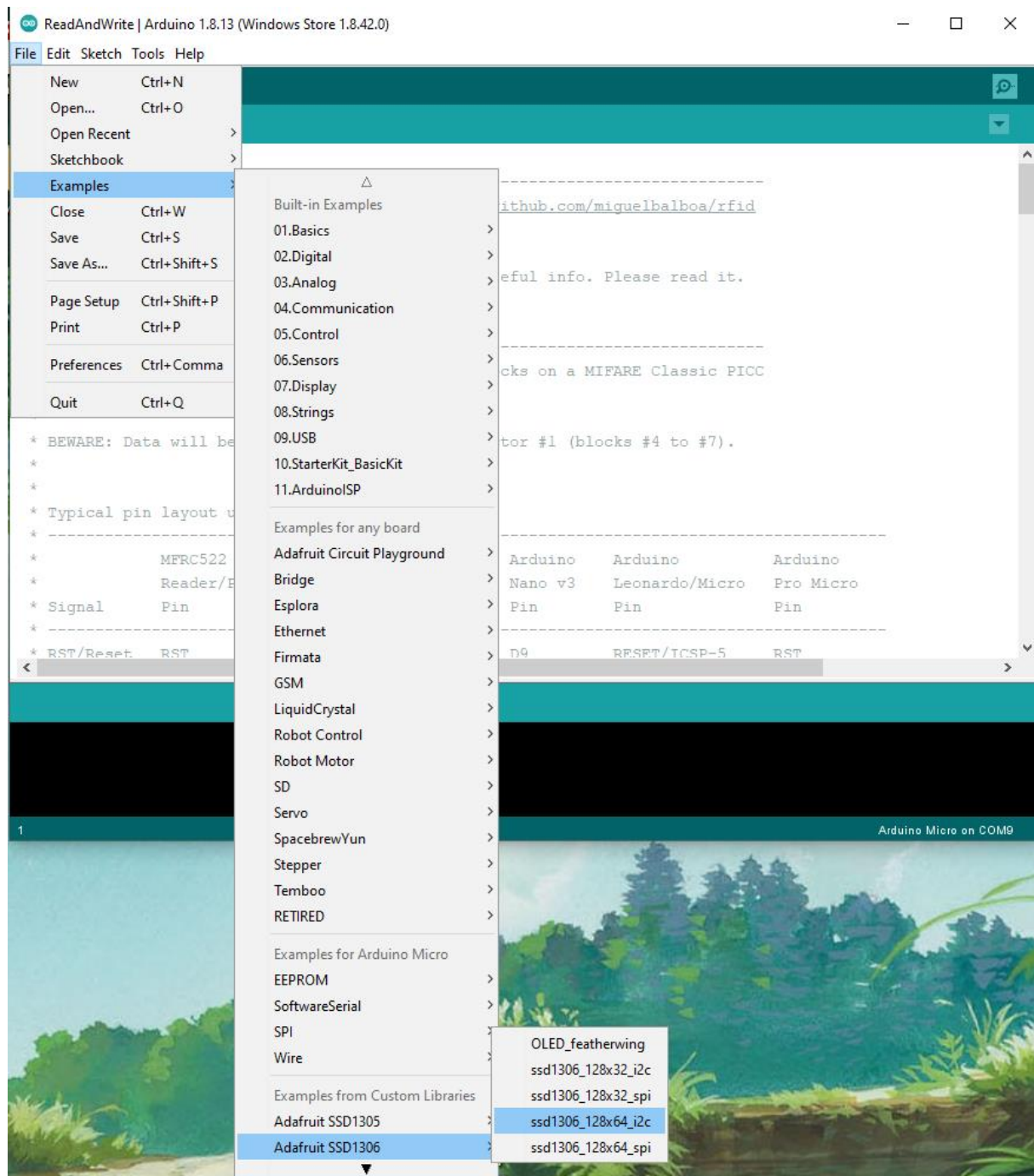


Fig. 9. Program examples in Libraries

2. Software Design

Arduino program (also called sketch) follows C++ syntax, but with some functions related to the hardware. As a controller program (called as “sketch”), it has a basic structure.

2.1 Basic Structure of a Sketch

The sketch (i.e. a program) in Arduino consists of two inevitable functions, called `setup()` and `loop()`. (See Fig. 10)

a) `setup()`

The `setup()` function is called when a sketch starts. You can use it to initialize variables, set pin modes, start using libraries, etc. The setup function will only run once, after each power-up or reset of the Arduino board.

b) `loop()`

After running the `setup()`, the `loop()` function will be run. It will loop consecutively.

c) Before `setup()`, we usually include libraries, define some global variables, create some other functions [Note: Those variables and functions can then be used in `setup()` and `loop()`].

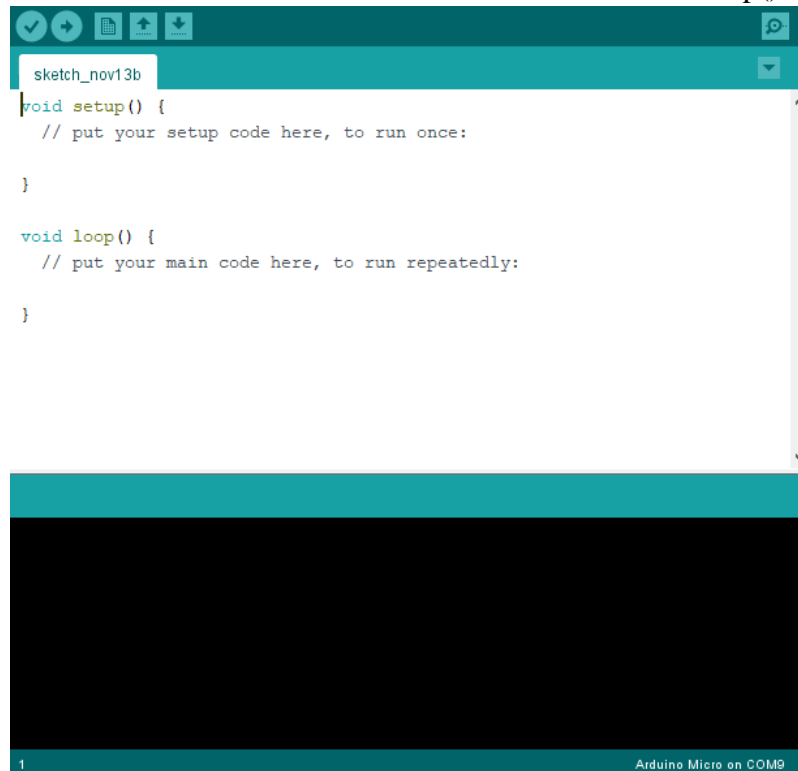




Fig. 10. Basic Arduino program structure

2.2 Program Compilation and Uploading

After writing your program, you can first compile it to see whether there is any syntax error. The compile button is . If your program is correct, you will see the memory usage. If your program is correct, you can upload it to the Arduino board. The uploading button is .

3. Serial Monitoring

We can establish Serial communication between a PC and an Arduino board, if they are connected via an USB cable. It is useful for debugging and also as an I/O interface of your program.

3.1 Functions/Commands

a) Set up

- `Serial.begin(val)` : val is the baud rate.
- `Serial.end()`: disable the serial communication

Note: The baud rate in the program and the serial monitor should be the same.

b) Sending data to PC

- `Serial.println` and `Serial.print`: print the texts (with or without line return)

c) Receiving data from PC

- `Serial.available()` : get the number of byte available for reading from the serial port
- `Serial.parseInt()` : get the first valid integer from the serial buffer
- `Serial.readString()`: read characters from serial buffer and return as a string.

3.2 Serial Monitor

To receive / send data at PC side, you'll need to open the **Serial Monitor**.

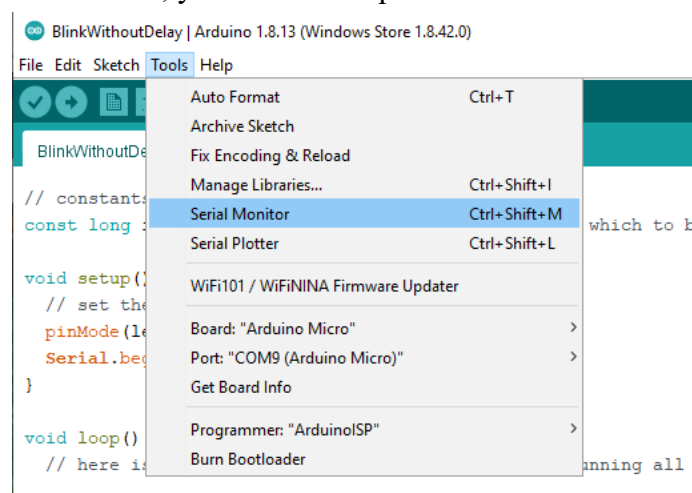


Fig. 11. Menu for Serial Monitor function

You need to make sure the baud rate setting at the terminal matching with your program.

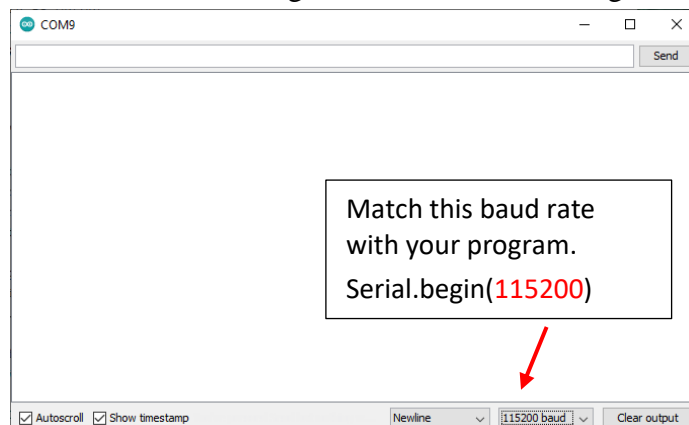


Fig. 12. Serial Monitor

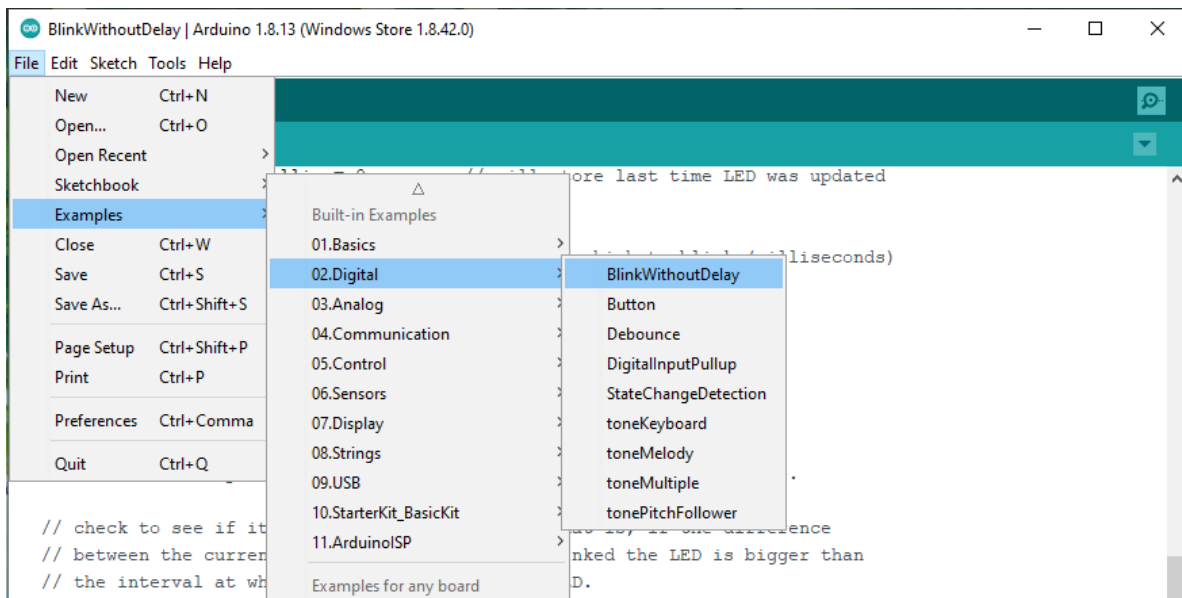
Self-Practicing Exercises



If you are using your own PC/notebook, you have to first install the Arduino IDE as described in Sec. 2.1.

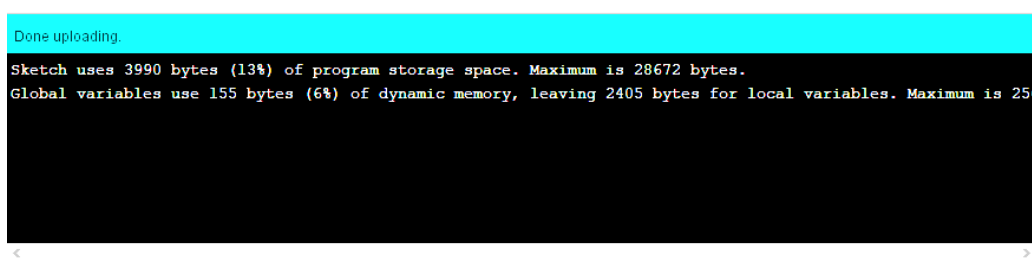
Practice 1:

Here, we'll go through an example available in Arduino IDE.

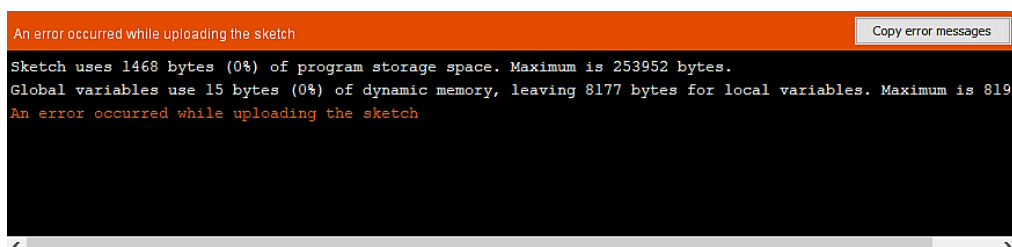
- Select the correct Board and Port for your Arduino board, as described in Sec. 2.2
- Pick the example **BlinkwithoutDelay** as shown below



- Press  to compile the program, notice different kinds of memories usage.
- Pressing  button (Note: The board should be connected with PC using USB cable). It will compile the code again and upload the program to the Arduino board. If no error is detected, the Status Bar would display 'Done Uploading'.

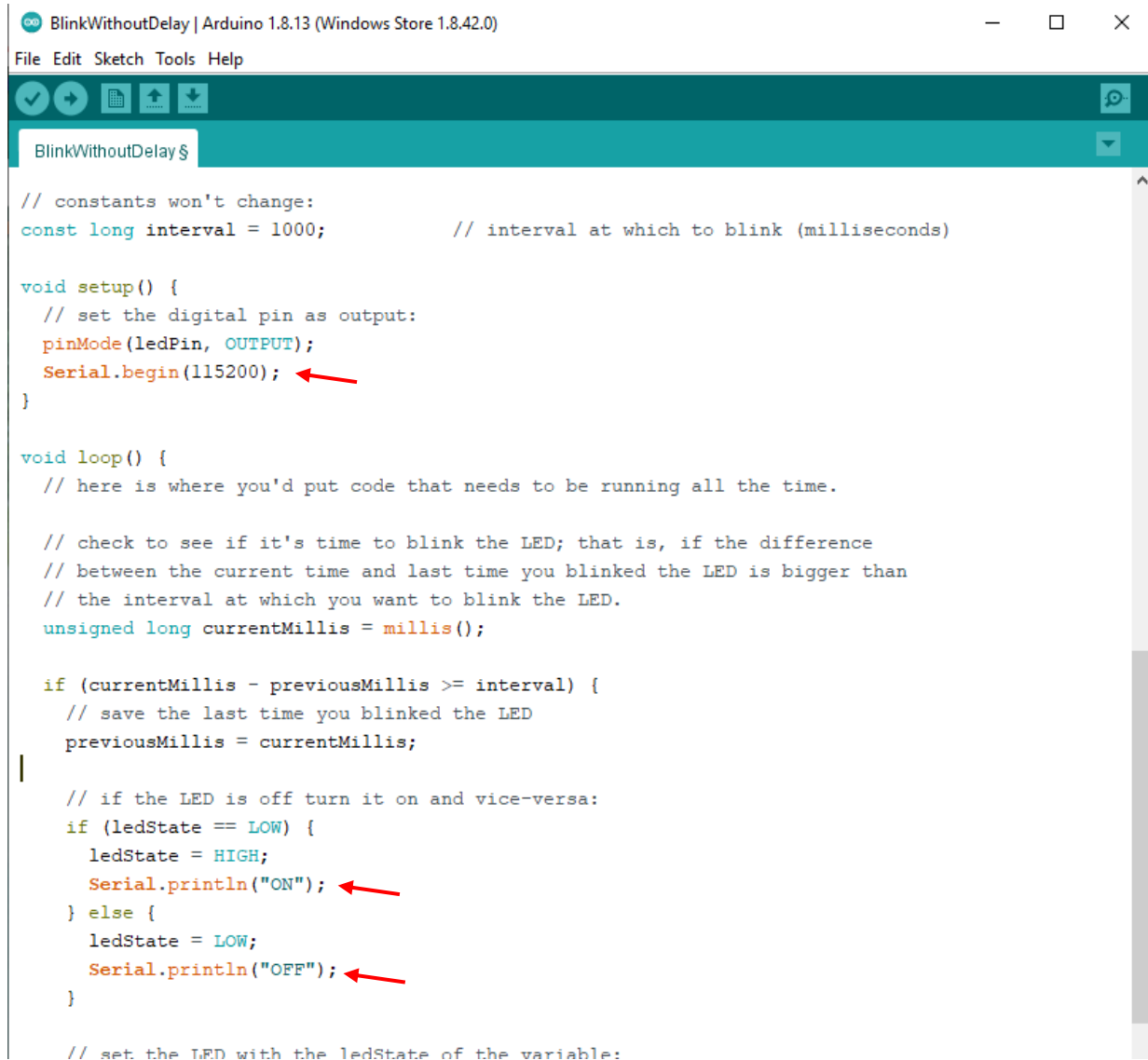


If error occurs, some error messages would be displayed. For example, below shows the case that Arduino IDE cannot find the board on the given port. Correction is then needed, say, go to Tool → Port and select the correct port for the IDE.



Practice 2:

Modify the BlinkwithoutDelay as follows and observe the output



```
// constants won't change:
const long interval = 1000;           // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  // here is where you'd put code that needs to be running all the time.

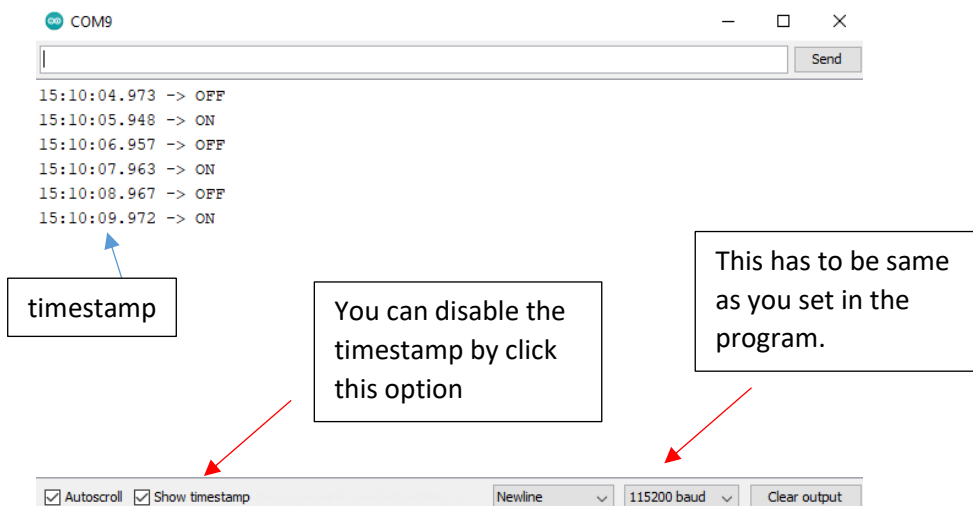
  // check to see if it's time to blink the LED; that is, if the difference
  // between the current time and last time you blinked the LED is bigger than
  // the interval at which you want to blink the LED.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
      Serial.println("ON");
    } else {
      ledState = LOW;
      Serial.println("OFF");
    }
  }

  // set the LED with the ledState of the variable:
}
```

The output should be as follows



```
COM9
15:10:04.973 -> OFF
15:10:05.948 -> ON
15:10:06.957 -> OFF
15:10:07.963 -> ON
15:10:08.967 -> OFF
15:10:09.972 -> ON
```

timestamp

You can disable the timestamp by click this option

This has to be same as you set in the program.

☒ Autoscroll ☒ Show timestamp Newline 115200 baud Clear output

After you finished Practice 1 and 2, you can start on the following Training Tasks.

Training Tasks

Task 1. Understanding the Arduino Board

(You can refer to Self-reading Materials in Sec. 1.1 and 1.2)

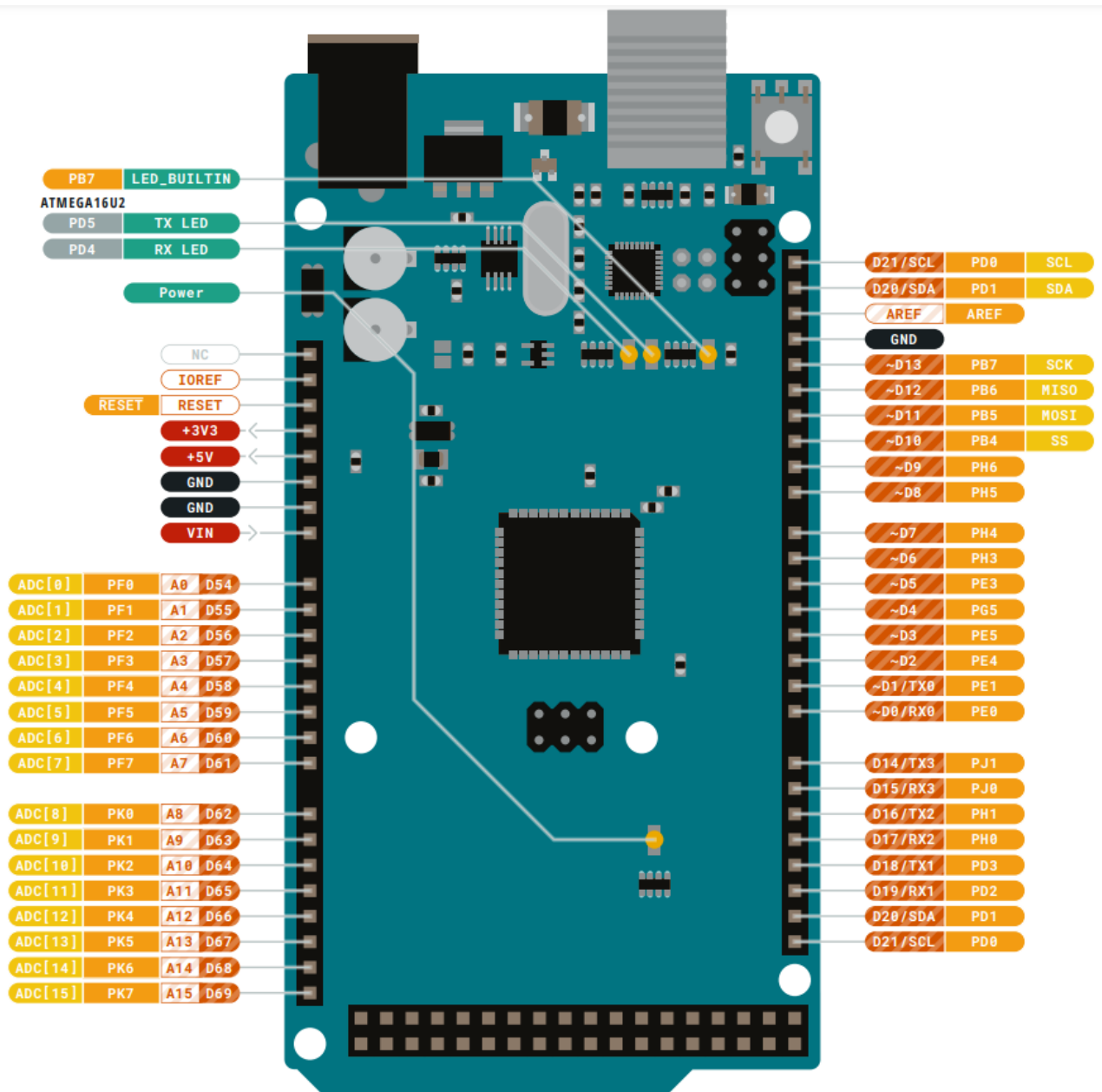
Task 1.1

*By checking the technical specification of **MEGA 2560 ver 3**, answer the following questions*

<i>Questions</i>	<i>Your answers</i>
1) <i>What is the operating voltage of Mega 2560 ver 3?</i>	
2) <i>In maximum, how many digital I/O pins you can assign in Mega 2560 ver 3?</i>	
3) <i>Which I/O pins can provide PWM output (give the pin numbers)?</i>	
4) <i>What is the smallest non-zero average voltage that PWM output can provide?</i>	
5) <i>How many pins can be used to read a sensor for which its output is ranged from 0V to 3V?</i>	
6) <i>Can we use an I/O pin to directly light up a LED if it needs 2V and 10mA (without using any driving circuit)?</i>	
7) <i>If you use your Arduino board to read an analog input, what is the possible largest digital value you obtained?</i>	
8) <i>How many serial communication pairs that Mega 2560 has?</i>	
9) <i>In maximum, how many bytes you can use for your program code?</i>	
10) <i>List all the pins that support external interrupts?</i>	

Task 1.2

Answer questions to get familiar with pin assignment for communications.



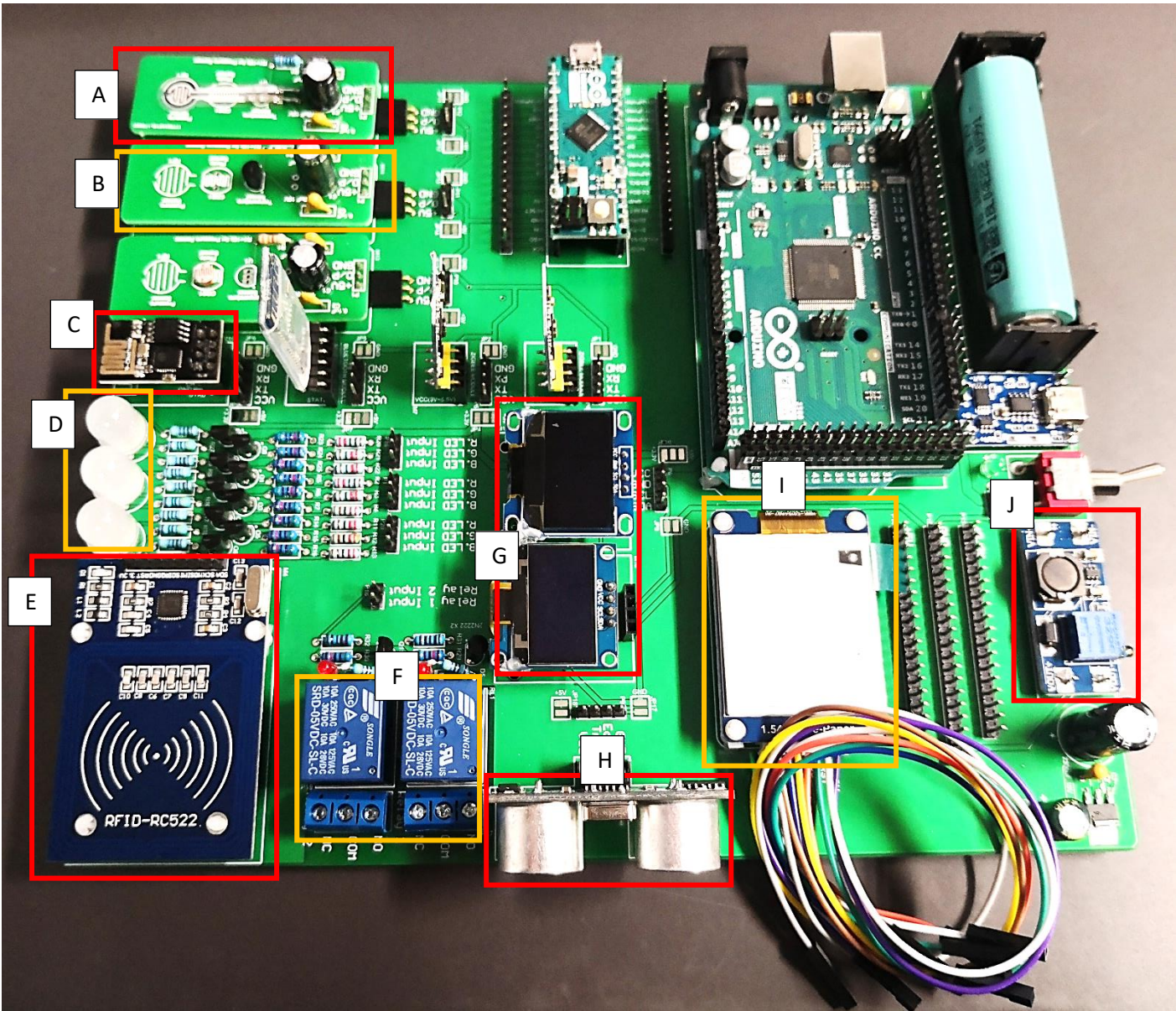
This project will make use of different types of serial communications. They are based on UART, I2C and SPI, respectively. [Please refer to Lecture notes]

- There are four default UART Serial communications. One of them is also used for the Serial monitor (So, you are not to use that serial, if you use Serial monitor). Circle the transmitter and receiver pins of that serial port, and mark as UART
- I2C communications use 2 pins, one is clock SCL and one is data SDA. Circle the default pins that assigned for these functions, and mark as I2C.
- SPI communications use 4 pins. They are clock (SCK), slave select (SS), master-input-slave-output (MISO) and master-output-slave-input (MOSI). Circle the default pins that assigned for these functions and mark as SPI.

Task 2 Get Familiar with the project platform

Task 2.1 Get Familiar with the modules

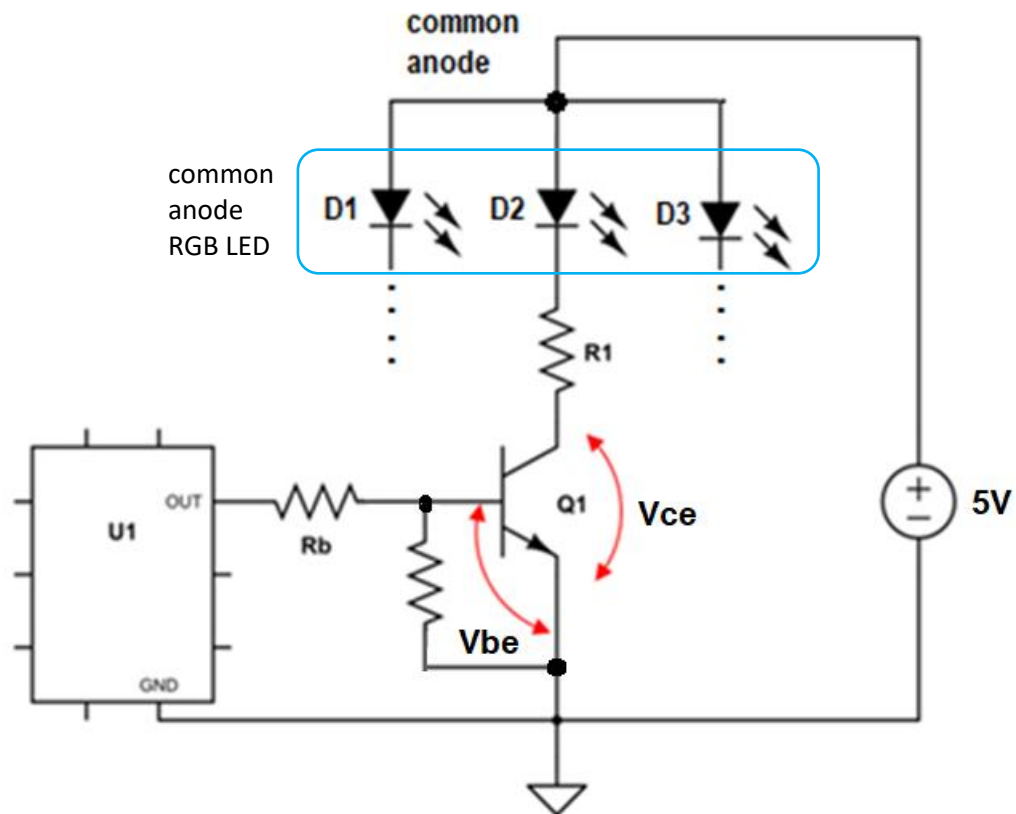
You'll be given a project platform. Followings are the photo that contain all modules. In your board, you may not have all the modules, but you should be able to identify the locations.



Modules	The name of the module
A	
B	
C	
D	
E	
F	
G	
H	
I	
J	

Task 2.2 Transistor Circuit to drive LEDs

The driving circuit for RGB LED is shown below. For clarity, only one circuit is given.



Explain how the driving circuit works to turn D2 on and off

Task 3. Programming Tasks

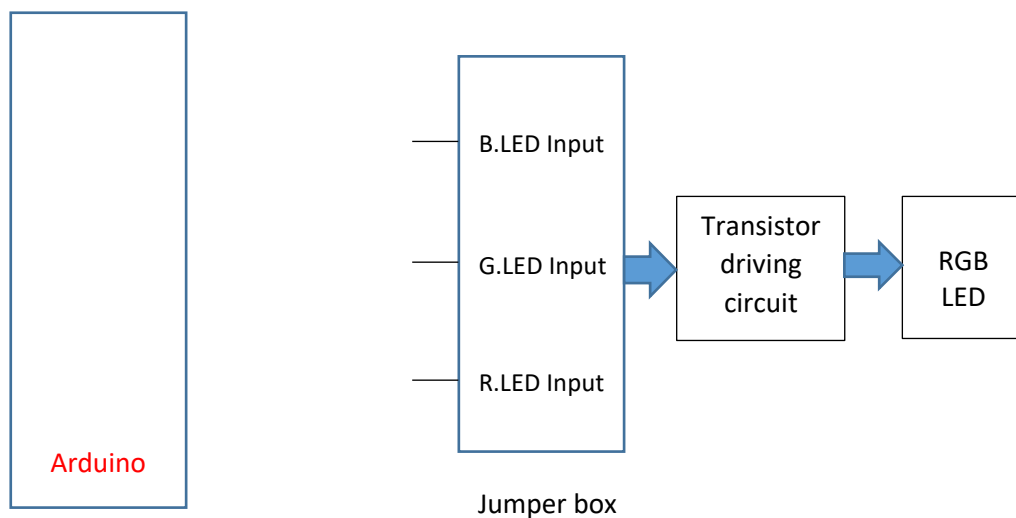
You are to complete all programming tasks given below.

Task 3A: Display color with RGB LED

Turn on the RGB LED to show three colors in sequence repetitively, each holds for 1 second
(Color 1 → Color 2 → Color 3 → Color 1 → ...)

- i) Draw the connection (in block diagram)
- ii) Draw the flow chart of your design
- iii) Do the actual programming

Provide your connection (in simple block diagram)



Note: The actual colors depend on your SID.

Last digit in your SID	Color 1	Color 2	Color 3
0	Blue	Green	Magenta
1	Red	Blue	Yellow
2	Yellow	Green	Blue
3	Magenta	Red	Green
4	Blue	Magenta	Red
5	Green	Yellow	Blue
6	Magenta	Blue	Red
7	Red	Yellow	Green
8	Yellow	Blue	Red
9	Green	Red	Yellow

Flow Chart (Specify how to control the I/O pins, instead of giving the colors)

Remarks: Save your program as SID_task3A.ino for later report usage

CHECKPOINT: SHOW THE FLOWCHART AND RESULTS TO THE TUTOR

Task 3B: Display colors based on command inputs

Control the color display of a RGB LED by entering commands in the Serial monitor. You have to describe your own mapping between the commands and the colors.

- i) Give a description on your function mapping
- ii) Do the actual programming

Remarks: Save your program as SID_task3B.ino for later report usage

CHECKPOINT: SHOW THE RESULT AND EXPLAIN TO THE TUTOR

Task 3C: Display different brightness based on command inputs

Control the color and brightness of a RGB LED by entering commands in the Serial monitor. For example, Rxx indicate RED color with brightness of xx% (You may define your own commands). But, in your design, you need to implement at least two colors and 3 levels of brightness

- i) Give a description on your function mapping
- ii) Do the actual programming

Hint: analogWrite() is to be used

Remarks: Save your program as SID_task3C.ino for later report usage

CHECKPOINT: SHOW THE RESULT AND EXPLAIN TO THE TUTOR