# 임베디드컴퓨팅
## Embedded Computing (0009488)
# Start Arduino

2022년 2학기

정보기술대학 정보통신공학과
김 영 필
ypkim@inu.ac.kr

인천대학교
INCHEON NATIONAL UNIVERSITY

## Contents

- First Arduino programming
- Basic C language syntax review

## Preparation for your first sketch

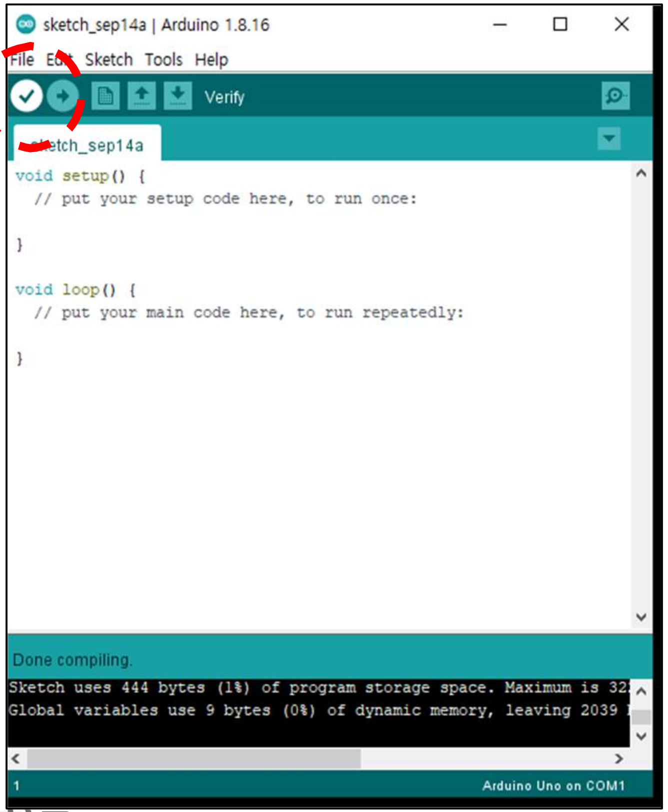
- **Check your device connection**
  - USB-to-Serial device driver is working?
    - Device manager shows it?
- **Connect your device with a cable**
  - A-B type USB cable
- **Check your IDE setting**
  - Go to 'Tools' menu
  - Select Board
  - Select Port
  - Try 'Get board info.'
    - Working?

# Try this code!

- Type all lines of code

```
void setup() {
  pinMode(13 OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

- Verify and Compile it!
  - Sketch->Verify/Compile or,
  - Ctrl+R



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

}

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

}
```

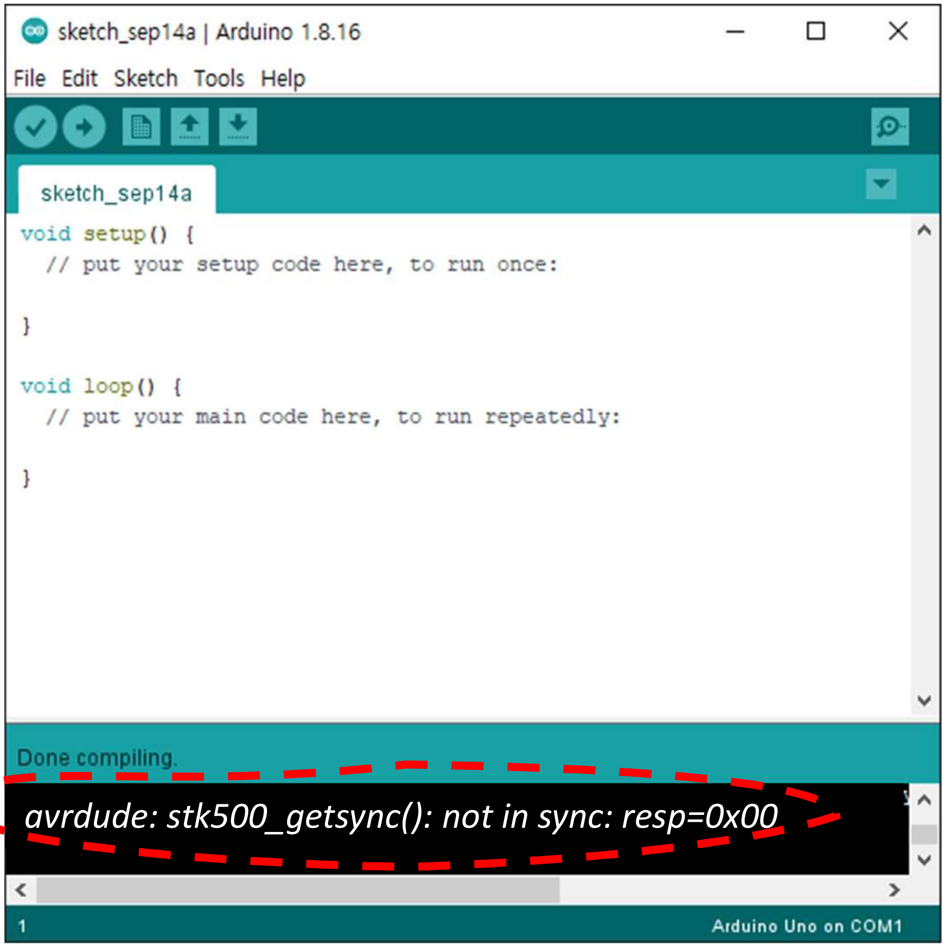
Done compiling.

```
Sketch uses 444 bytes (1%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039
```

# Caution: IDE uses D0, D1 pins

- If you see error messages like this..
  - *avrdude: stk500_getsync(): not in sync: resp=0x00*
- Check extended boards (shield) which try to use those pins
  - Remove them, and try it again.
- IDE also uses pins to transfer program images; avoid a collision!

- Check below black text box

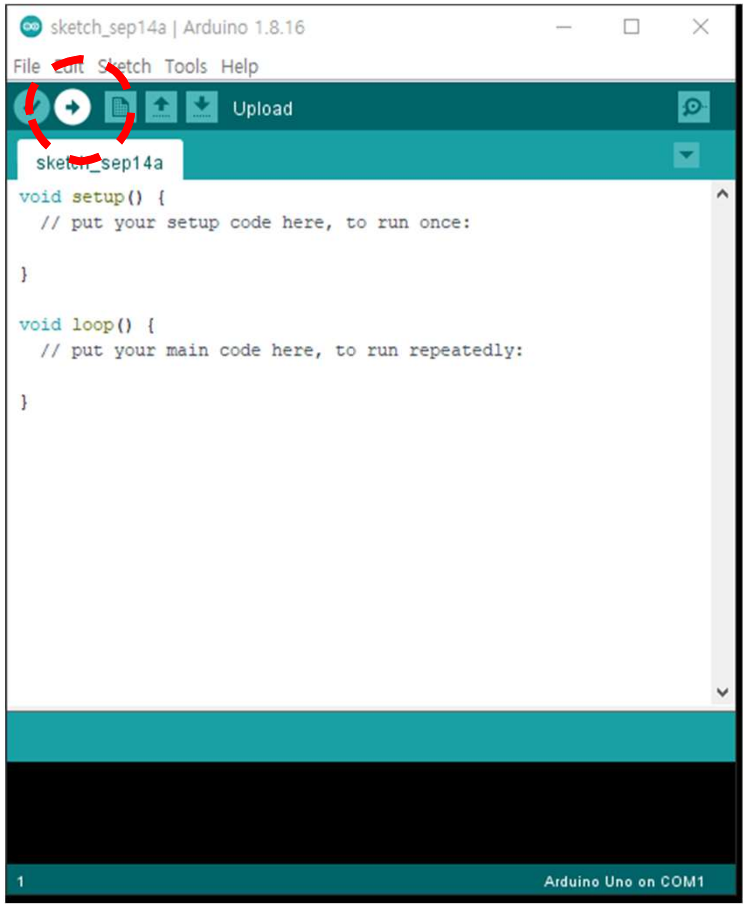


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

```
avrdude: stk500_getsync(): not in sync: resp=0x00
```

# Let’s Upload our output!

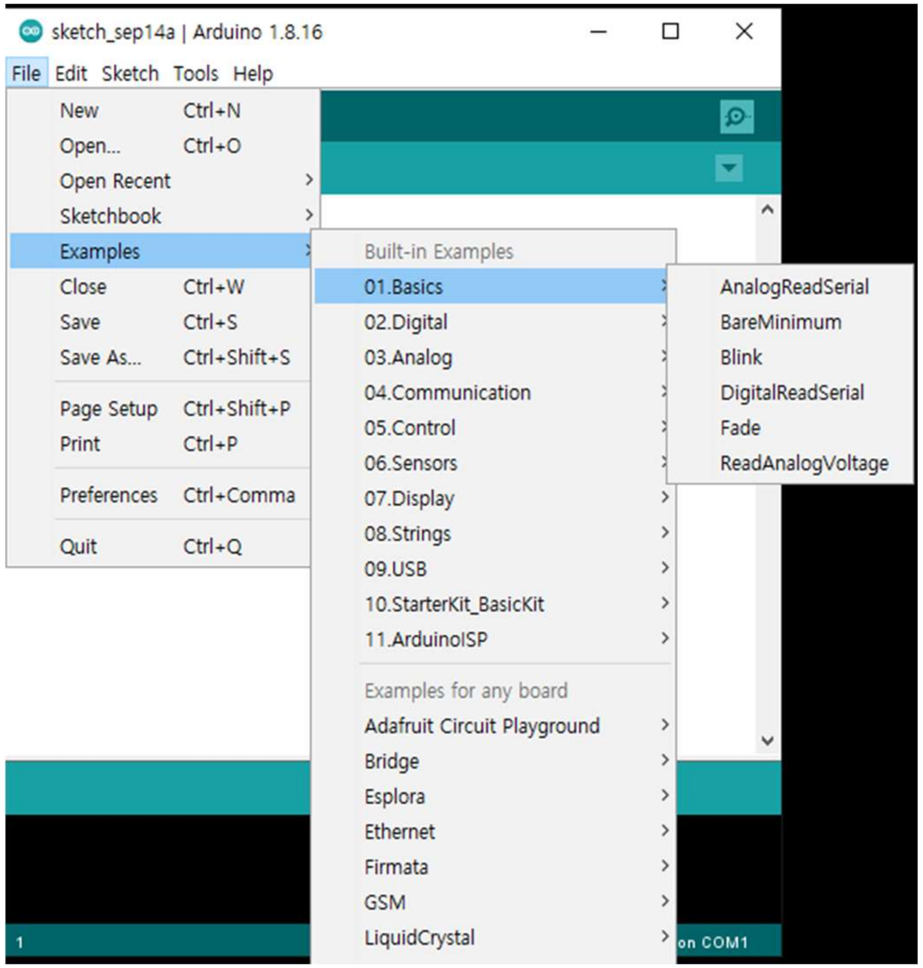
- Sketch->Upload or, Ctrl+U



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

}

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

}
```

- Watch your device!!!
  - See blinking three LEDs?
    - Sketch code transfer is going on.
  - Now, Which number of LED is blinking?
    - Then, it is working or, something is wrong…;

# Try Example code!



- Open sketch code
- Verify / Compile it
- Resolve errors
- Upload it and Enjoy

# Lets modify sketch code!

- Put detailed comments
  - As much as you can

    understand the code part
- Try to change variables
- Try to put code for printing debug messages
- Try to insert sub-loop control or conditional branch

- Not remember C syntax??

# Basic structure of sketch code

- Comments
  - To explain
    - for you and your colleagues
  - To plan
    - for fast prototying and lazy implementation
  - To show
    - Creator, Persission, Rights, History etc.
- void setup()
  - Put your code
    - Runs once at startup.
- void loop()
  - Put your code
    - Runs repeatedly

- Language reference
  - Functions, Variables, Structures
  - https://www.arduino.cc/reference/en/
  - We will explore the aboves on demand

## Blink's built-in functions

- **pinMode( )**
  - Syntax
    - pinMode(**pin**, **mode**)
  - Parameters
    - **pin**: the Arduino pin number to set the mode of.
    - **mode**: INPUT, OUTPUT, or INPUT_PULLUP.
    - See the Digital Pins page
      - https://www.arduino.cc/en/Tutorial/Foundations/DigitalPins
  - Return
    - Nothing

- Digital I/O
  - digitalRead()
  - digitalWrite()
  - **pinMode()**

## Digital Pins

- INPUT
  - Default Pin state
  - For nothing connected, get random pin states or noises
- INPUT_PULLUP
  - When no input, set input pin as known state
- OUTPUT
  - Set the pin state as a low impedance state
  - Can provide a substantial amount of current to other circuits

# Blink’s built-in functions

- **digitalWrite( )**
  - Syntax
    - digitalWrite(pin, value)
  - Parameters
    - **pin**: the Arduino pin number to set the mode of.
    - **value**: HIGH or LOW.
      - 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
  - Return
    - Nothing

- Digital I/O
  - digitalRead()
  - **digitalWrite()**
  - pinMode()

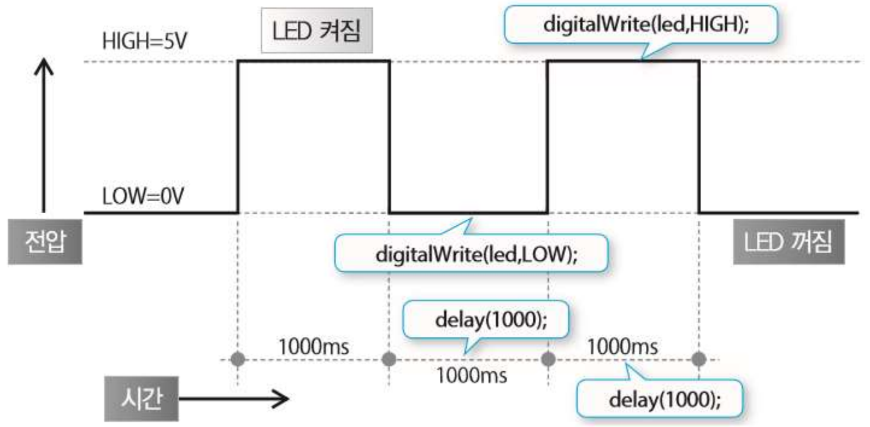
# Blink's built-in functions

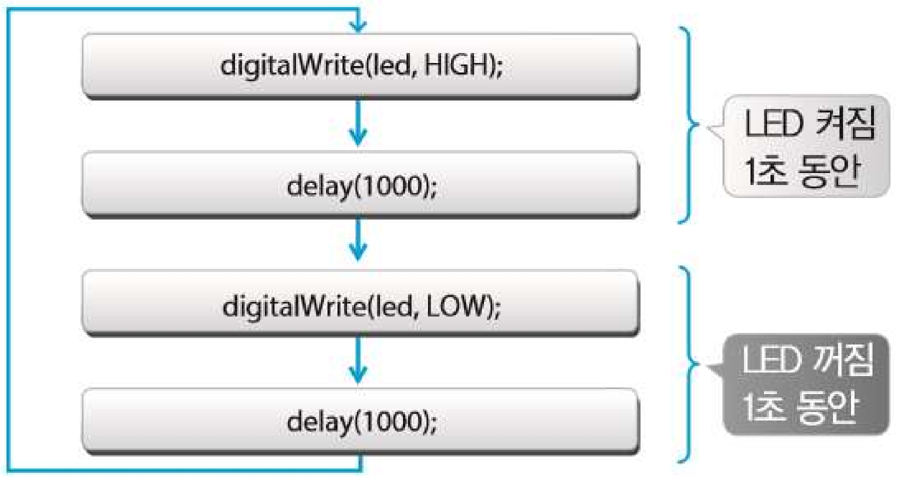
- **delay( )**
  - Pauses the program for the amount of time (in milliseconds) specified as parameter
  - Syntax
    - delay(ms)
  - Parameters
    - **ms**: the number of milliseconds to pause. (unsigned long)
  - Return
    - Nothing

- Time
  - **delay()**
  - delayMicroseconds()
  - micros()
  - millis()

# Blink code explanation

- Blinking mechanism
- Flow chart





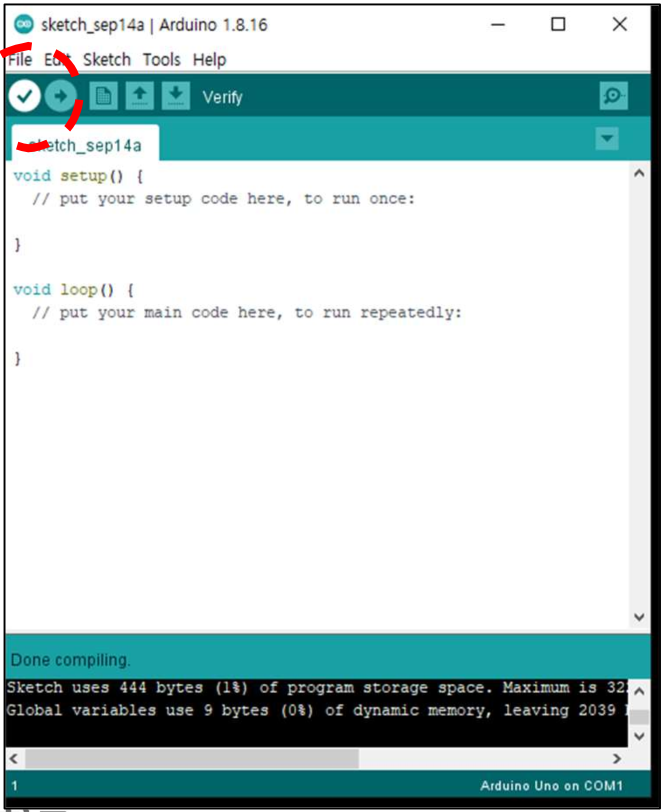
**Source: 길벗, "모두의 아두이노"**

# Try this code!

- Type all lines of code

```
void setup()
  Serial.begin(9600);
}

void loop() {
  Serial.print("*** Arduino test ****");
  Serial.println("+++ Uno R3 test +++");
  delay(300);
}
```

- Verify and Compile it!
  - Sketch->Verify/Compile or,
  - Ctrl+R



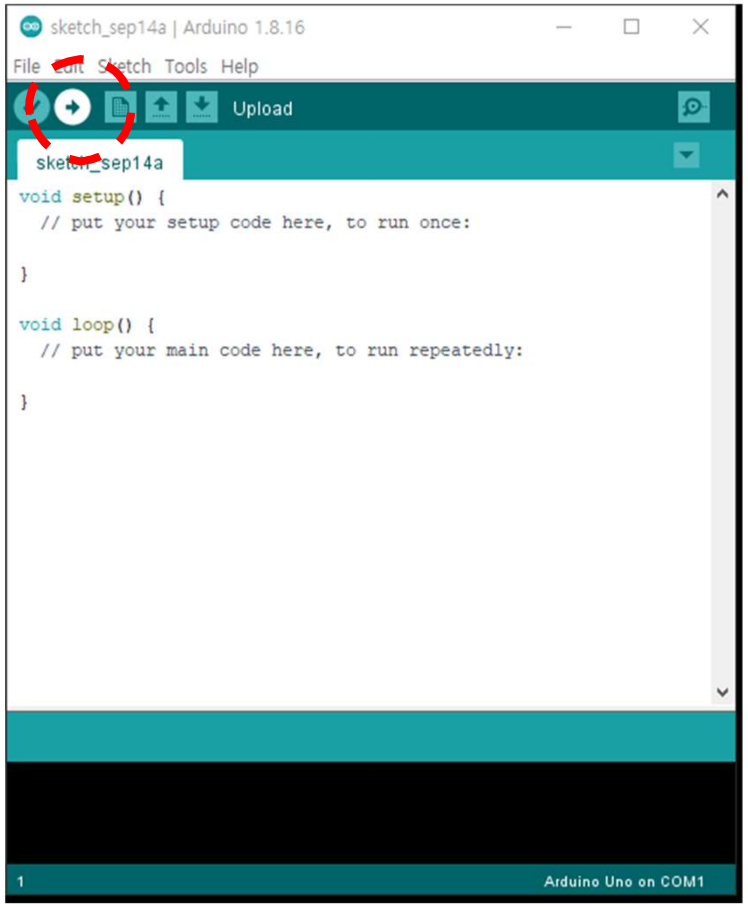
```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Done compiling.

```
Sketch uses 444 bytes (1%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039
```

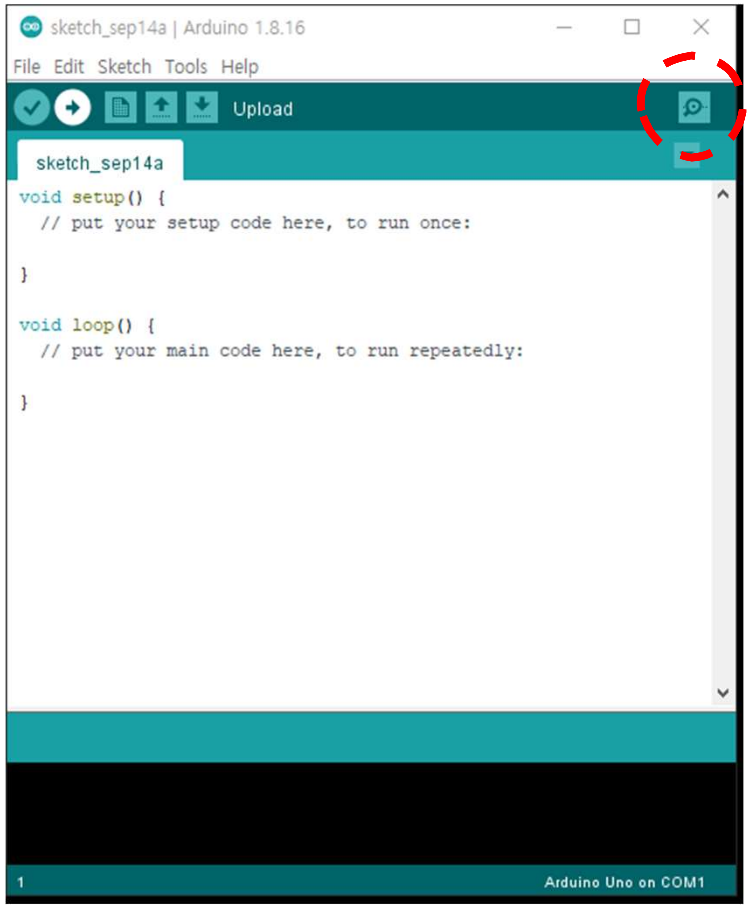
# Let’s Upload our output!

- Sketch->Upload or, Ctrl+U



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

}

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

}
```

- Tools->Serial monitor or, Ctrl+Shift+M



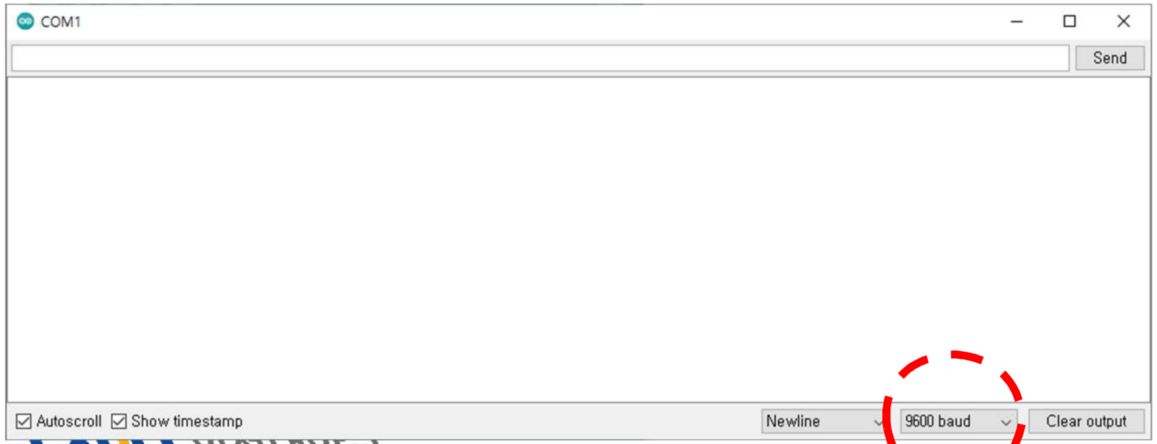
```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

## Serial monitor

- A tool built in to the Arduino IDE allowing sending and receiving serial data to and from a connected board.
- Keep the same  in serial monitor and your sketch code

-  (in Arduino)
  - Shorthand of ***"bits per second"***, signifying the speed at which two devices are communicating

```
void setup()
  Serial.begin(9600);
}
```


COM1

Send

Autoscroll ☑ Show timestamp ☑

Newline

9600 baud

Clear output

# Serial's built-in functions

- **Serial.begin( )**
  - Sets the data rate in bits per second (baud) for serial data transmission.
  - Syntax
    - Serial.begin(speed)
    - Serial.begin(speed, config)
  - Parameters
    - **Serial:** serial port object.
    - **speed:** in bits per second (baud) (data type: long)
    - **config:** sets data, parity, and stop bits. (Default: SERIAL_8N1)
  - Return
    - Nothing

- Communication
  - Serial
    - …
    - **begin()**
    - …
    - print()
    - println()
    - …
    - write()
  - Stream

# Serial's built-in functions

- **print( )**
  - Prints data to the serial port as human-readable text.
  - Syntax
    - Serial.print(val)
    - Serial.print(val, format)
  - Parameters
    - **val**: the value to print. (any data type.)
    - **format**: DEC, HEX, BIN, …
  - Return
    - the number of bytes written (size_t)

- Communication
  - Serial
    - …
    - begin()
    - …
    - **print()**
    - println()
    - …
    - write()
  - Stream

# Serial's built-in functions

- **println( )**
  - Prints data to the serial port as human-readable ASCII text followed by a character (ASCII 13, or '₩r') and a character (ASCII 10, or '₩n').
  - Others are same to print()

- Communication
  - Serial
    - …
    - begin()
    - …
    - print()
    - **println()**
    - …
    - write()
  - Stream

# Serial's built-in functions

- **write( )**
  - Writes data to the serial port.
  - Syntax
    - Serial.write(val)
    - Serial.write(str)
    - Serial.write(buf, len)
  - Parameters
    - **val**: the value to print. (any data type.)
    - **str:** a string to send (bytes)
    - **buf:** an array to send (bytes)
    - **len:** the number of bytes to be sent from array
  - Return
    - the number of bytes written (size_t)

- Communication
  - Serial
    - …
    - begin()
    - …
    - print()
    - println()
    - …
    - **write()**
  - Stream

# How to utilize serial comm.?

- Used to examine the Arduino status from the computer or to send values to the Arduino.
  - Check the values of device
  - Observe the variables during program running
- When debugging
  - Check the status of the Arduino to find out what's wrong with the program.
- Can send the value entered with the keyboard, keypad, or mouse to the Arduino, and print the entered value on the computer.
- When monitoring sensor values
  - Serial monitor is also used by connecting the sensor