Arduino Leonard

Learning with HBE ARDUINO KIT

Hanback Co., Ltd.

Table of Contents

Chapter 1

HBE ARDUINO KIT

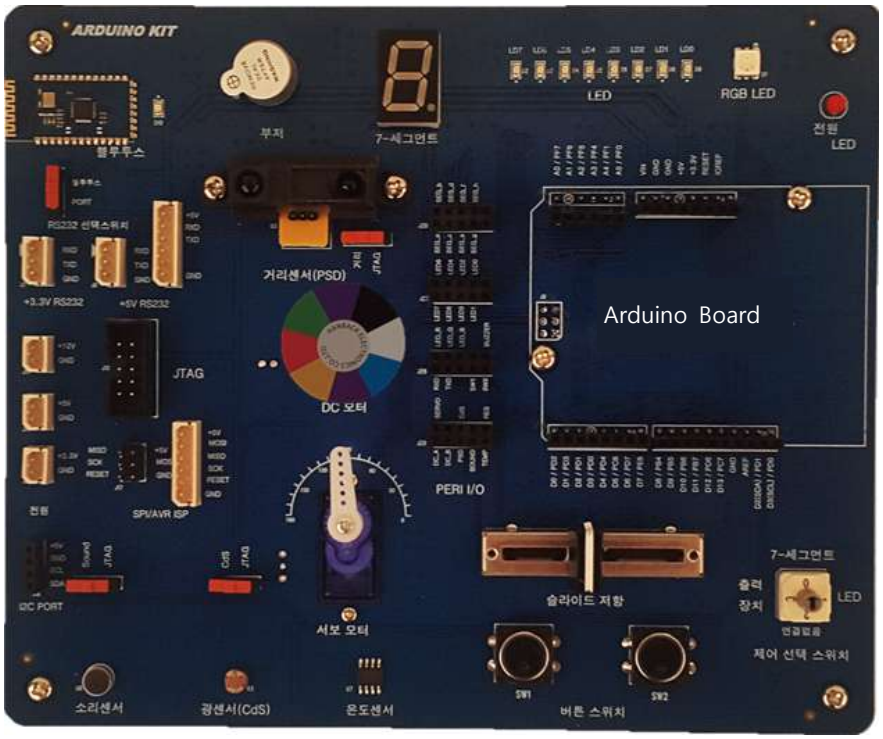
Learning Objectives

1. Understand ARDUINO KIT board.
2. Understand Development Environment.
3. Understand How to Use ARDUINO program.

1. HBE ARDUINO KIT

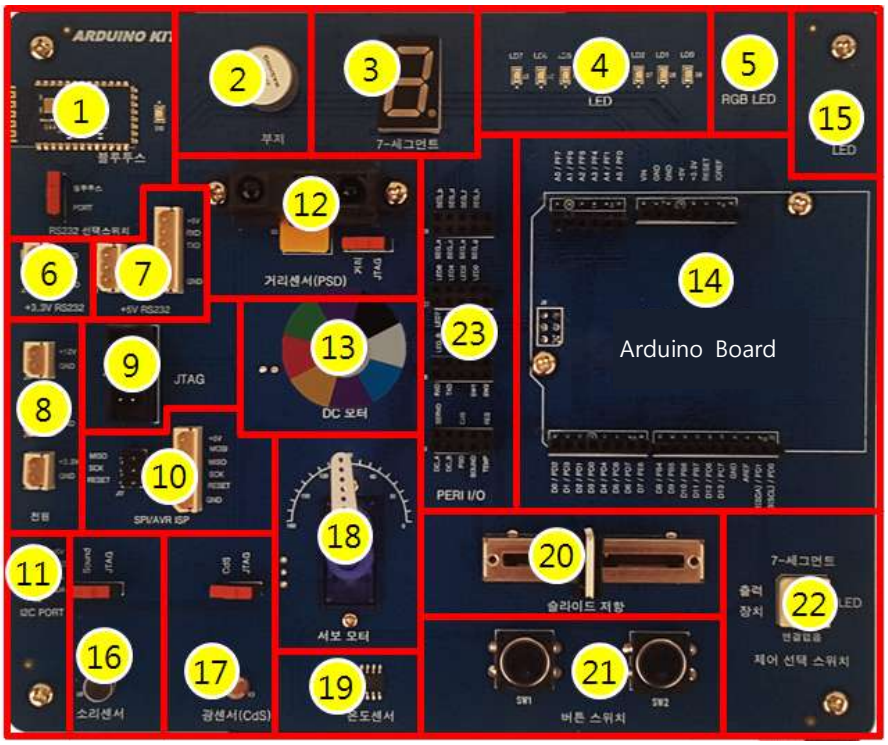
1.1. Overview of HBE ARDUINO KIT Board

HBE ARDUINO KIT board is shown in [Figure 1-1] below. This chapter explains easily using picture and figure so that even beginners who do not have much basic knowledge can follow equipment setting → test → result → verification by themselves.



[Figure 1-1] HBE ARDUINO KIT Board

HBE ARDUINO KIT board consists of Arduino Leonardo and each function module, and the signal lines between each module are connected in the PCB. [Figure 1-2] summarizes the module configuration of the board.



[Figure 1-2] HBE ARDUINO KIT Board

1.1.1. Features and Configuration

HBE ARDUINO KIT board has the following specification

① Bluetooth Module

② Buzzer : Sound Generation

③ 7-Segment (FND) : Displays numbers from 0 to 9 and letters from A to F

④ LED : 8 Green LED

⑤ RGB LED : Red, green, and blue LED consists of one LED

⑥ 3.3V RS232 Port : UART communication port with 3.3V level

⑦ 5V RS232 Port : UART communication port with 5V level

⑧ Power : 12V, 5V, 3.3Vpower supply

⑨ JTAG Port : Program JTAG Port

⑩ ISP Port : Program ISP Port

⑪ I2C Port : I2C(TWI) Communication Port

⑫ PSD Sensor : Distance Measuring Sensor

⑬ DC Motor

⑭ Arduino Board : Arduino Leonardo mounted on the floor of Board

⑮ Power LED : Light on/off when power is applied

⑯ Sound Sensor : Sensor that detects Sound

⑰ Optical Sensor : Sensor that detects Lignt

⑱ Servo Motor

⑲ Temperature Sensor : Sensor that measures temperature

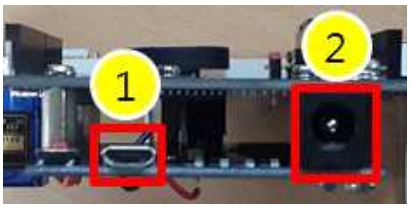
⑳ Variable Resistance: Sensor that changes voltage

㉑ Button Switch

㉒ Control selection switch: switch to select 7-segment, LED, output de vice or disconnect all pins (no connection)

㉓ Peripheral device connecting port: Port connecting sensors and control ling device mounted on the board

The following picture is the Arduino Leonardo side view from the left s ide of the Arduino Kit.



[Figure 1-3] Side of Arduino Kit Board

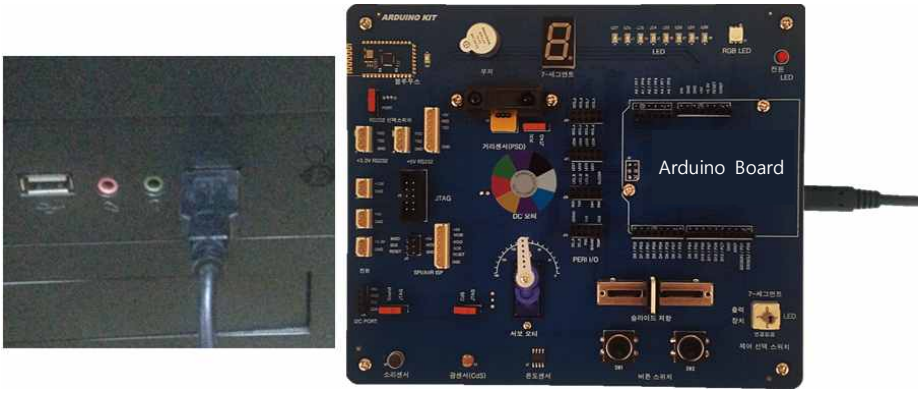
① Program Port : Program upload to Arduino IDE and Serial Communicatio n Port

② Power input port: DC jack that can apply 12V power

In the figure above, ① and ② supply power to Arduino Kit board, and when connecting two at the same time, DC jack power is supplied.

1.1.2. How to Use HBE-ARDUINO KIT Board

Connect Arduino Leonardo and PC with Micro USB cable as shown belo w.



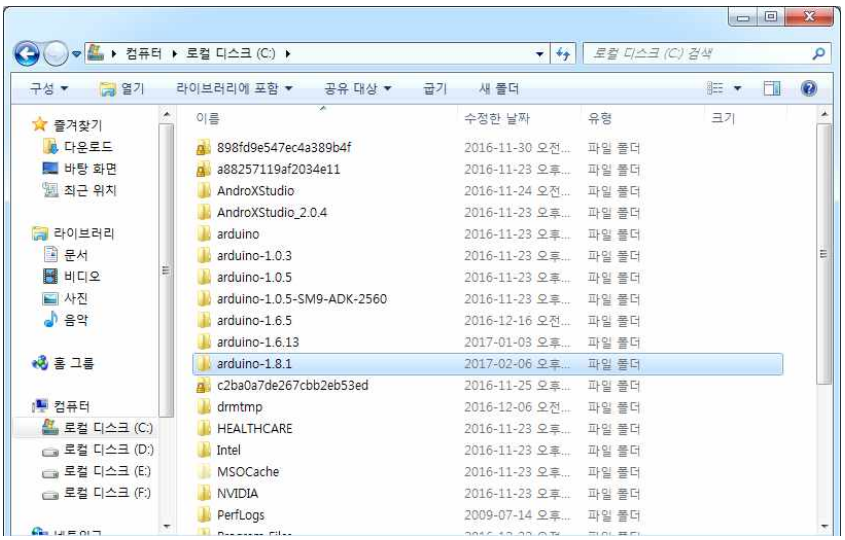
[Figure 1-4] Micro USB Cable Connection

When the USB cable is connected, power is supplied and the power LED lights up. In this state, you can upload the program to Arduino Leonardo of the HBE ARDUINO KIT board and check the operation with other modules. See below for more details.

1.2. Development Environment and USB Driver Installation

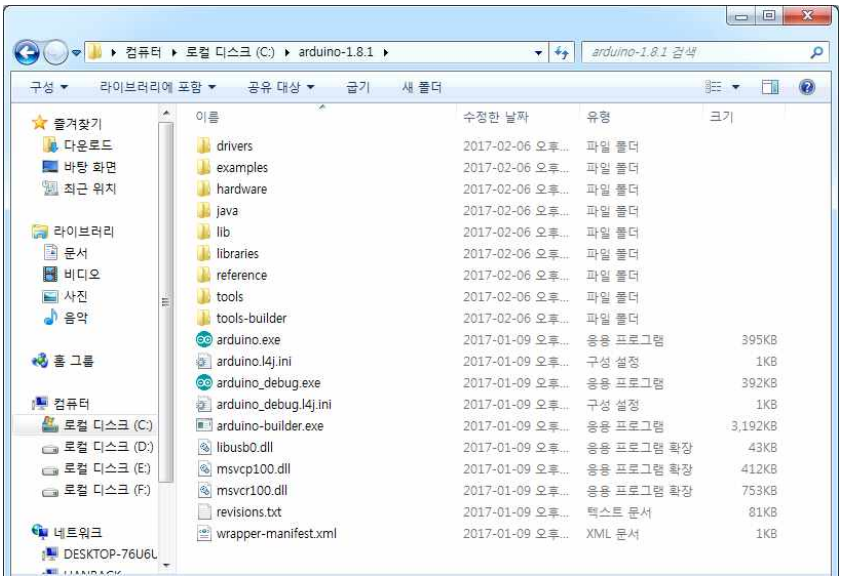
1.2.1. Install Development Environment

① To install the development environment, first unzip Arduino-1.8.1.zip in the “Software\Arduino” folder of the software provided and copy it to C:\.



[Figure 1-5] Copy Arduino-1.8.1

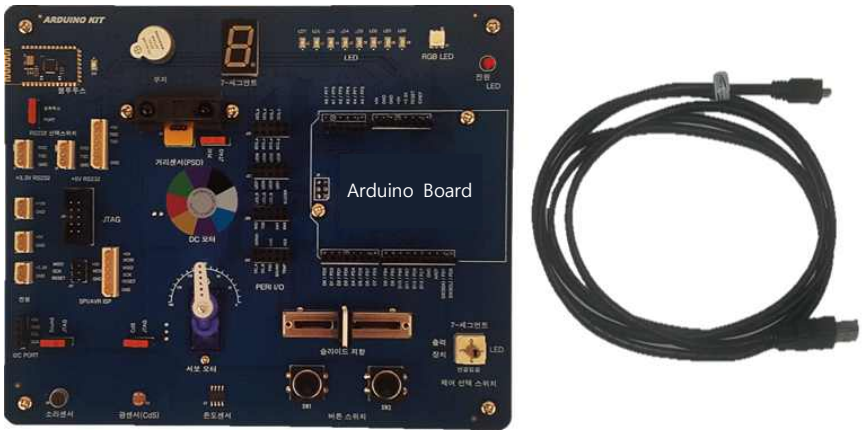
② As shown in Figure [1-5], when copying is completed, installation of the development environment is completed.



[Figure 1-6] Complete Arduino-1.8.1 Installation

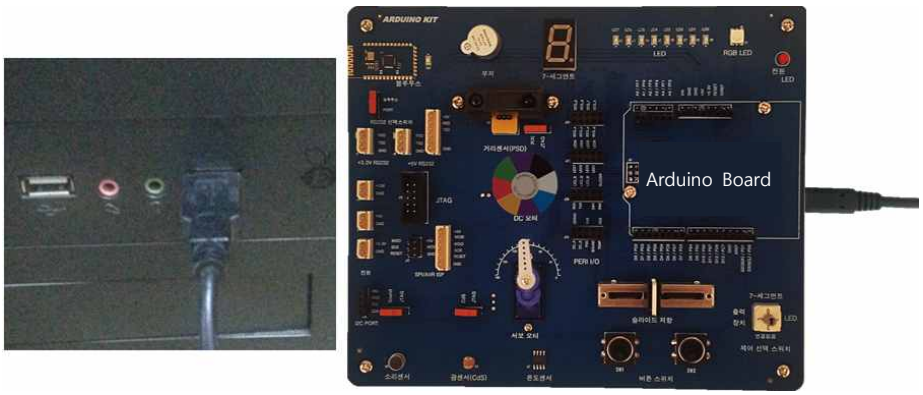
1.2.2. USB Driver Installation

① Prepare the ARDUINO Kit board and USB A(M)-Micro 5pin cable as shown in the following figure.



[Figure 1-7] Arduino Kit Board (left) &
USB A(M)-Micor 5pin Cable (right)

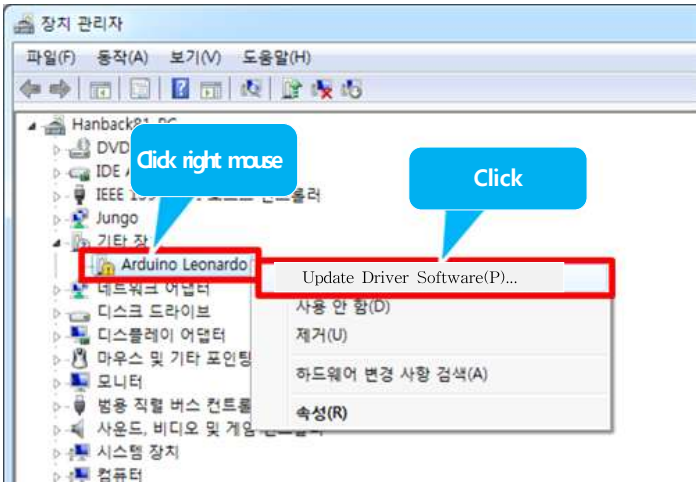
② As shown in the following figure, connect the programming port of the Leonardo board from the ARDUINO Kit board and PC with a USB A(M)-Micro 5pin cable.



[Figure 1-8] Connect the PC's USB Port (left) and Arduino Kit (right) with a cable

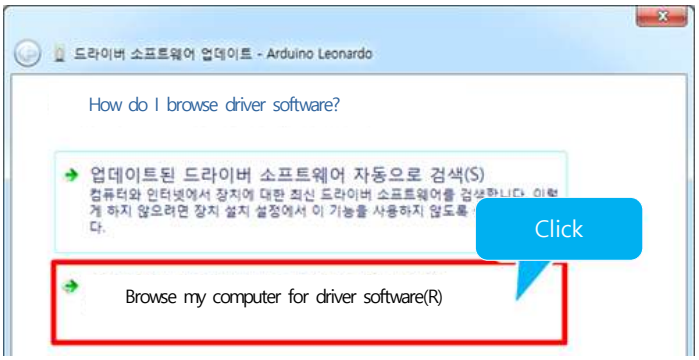
③ Control Panel - System and Security - System - Device Manager. A

s shown in the picture, “Arduino Leonardo” is displayed on the other de vice. Click the right mouse button on “Arduino Leonardo” and execute “Update Driver Software”.



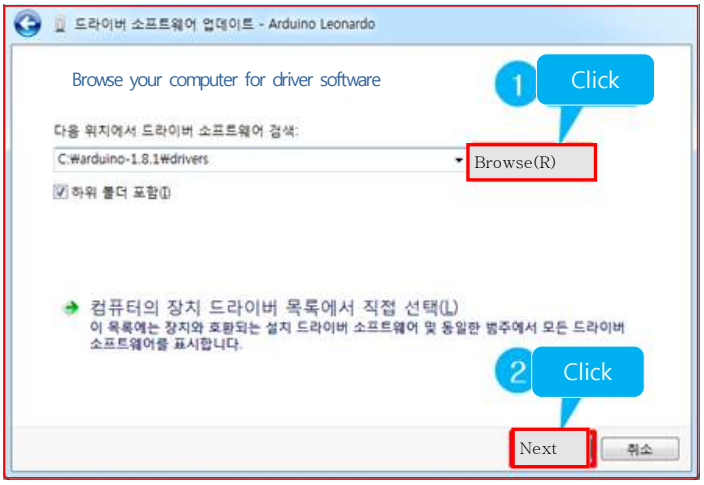
[Figure 1-9] Install Driver 1

④ Select Browse my computer for driver software as shown in the foll owing picture.



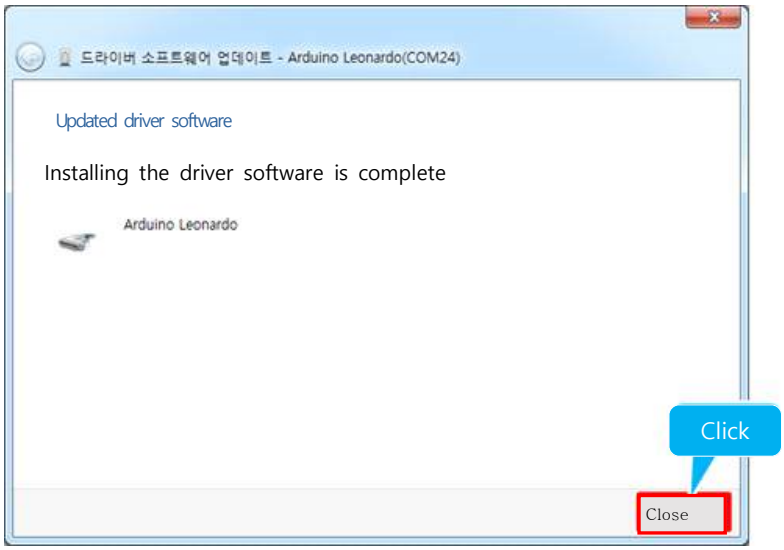
[Figure 1-10] Install Driver 2

⑤ Click Browse and select C:\Arduino-1.8.1\driver folder and click Next.



[Figure 1-11] Install Driver 3

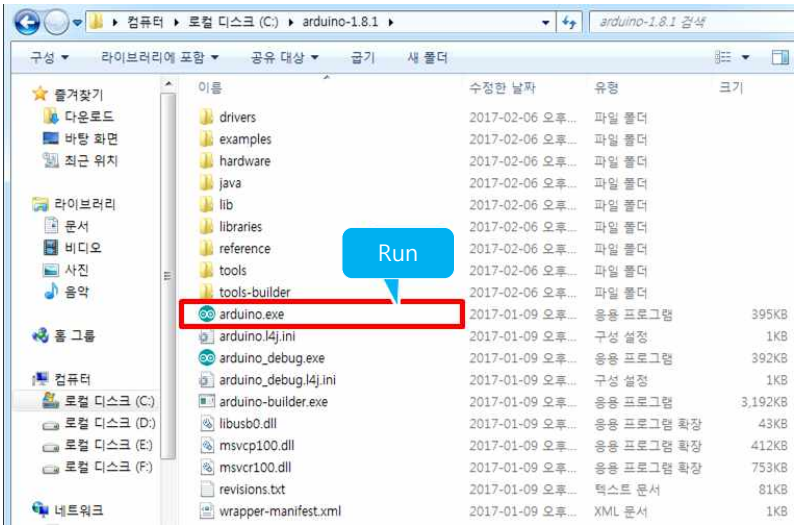
⑥ When the driver installation is complete, click Close.



[Figure 1-12] Complete Driver Installation

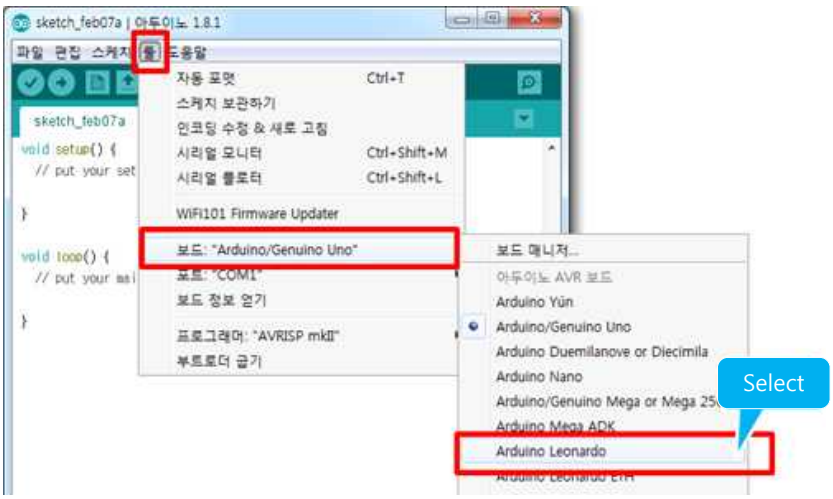
1.3. Arduino Setting

① Run arduino.exe as shown in the Figure.



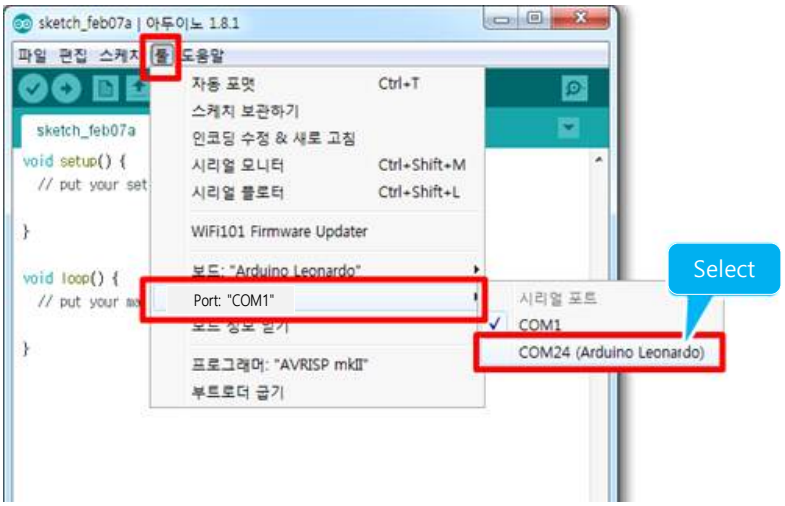
[Figure 1-13] Run Arduino

② Set the board to Arduino Leonardo in the Tools menu of the IDE as shown in the picture.



[Figure 1-14] Board Setting

③ Set the port to COM24 (Arduino Leonardo) in the tool menu of the IDE as shown in the picture.



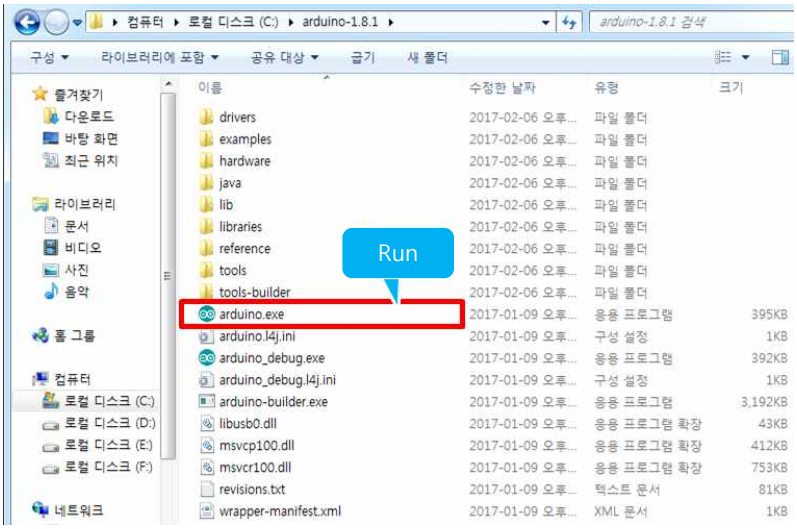
[Figure 1-15] Port Setting

④ If the lower part shown in Arduino IDE is the same as the setting value, the board and port setting may be omitted.

[Figure 1-16] Check Board and Port Setting Value

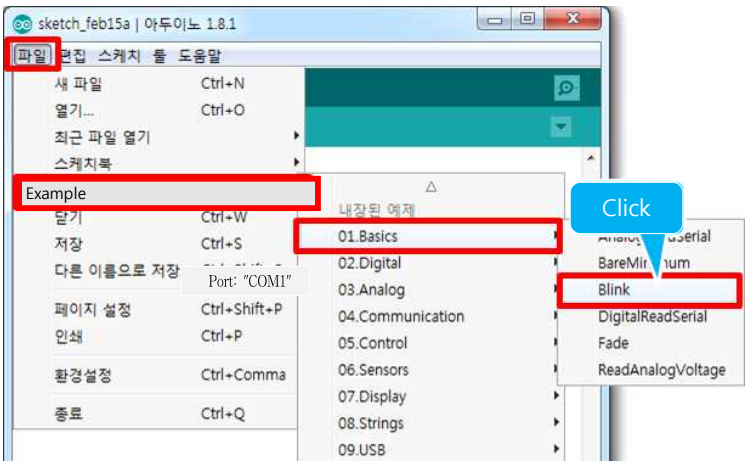
1.4. Program Upload

① Run arduino.exe as shown in the picture.



[Figure 1-17] Run Arduino

② Click Blink Example among Example - 01.Basics in IDE File menu as shown below.

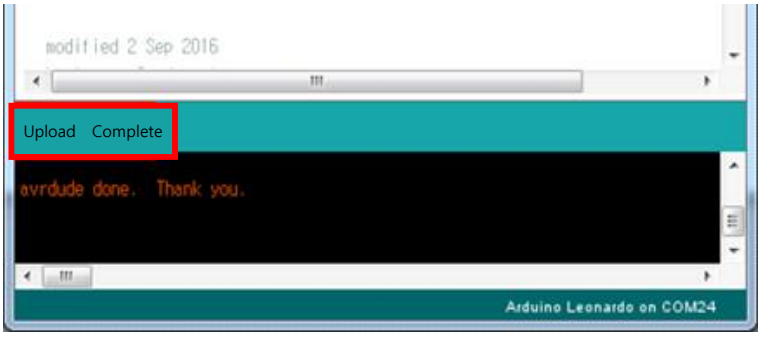


[Figure 1-18] Open Blink Example

③ Execute upload as shown below. Upload is executed after compilation is complete.

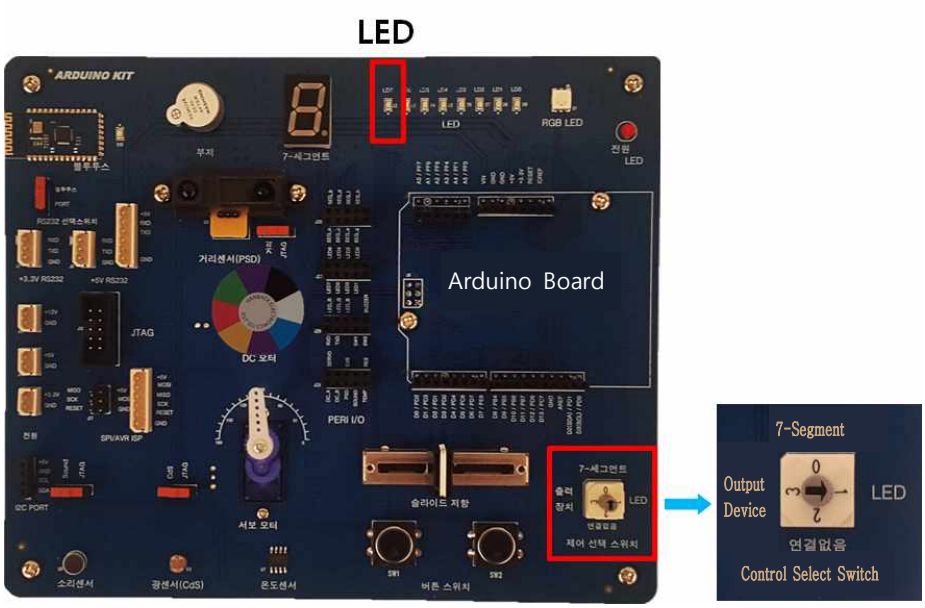
[Figure 1-19] Execute Upload

④ Upload is complete when it is displayed as shown in the picture.



[Figure 1-20] Upload is Complete

⑤ If LED as shown in the following figure turns ON/OFF every 1 second, the program upload is completed normally.



[Figure 1-21] Select Control Switch and Check Program Operation

This program upload process should be familiar because it is required while doing the program practice.

Chapter 2

C Language

Learning Objectives

1. Understand C Language.

2. C Language

A computer is a machine that processes data according to instruction, and a program sets how to process this data. The language that the computer understands is a binary number consisting of 0 and 1, and internal switches are changed to ON/OFF by 1 and 0 and get operated. Due to this feature, it is not possible to do inaccurate work, so detailed and specific instruction should be given. Language of the binary number is called machine language and is divided into Assembly Language and High-Level Language depending on the type.

2.1. Concept of Program

2.1.1. What is the Program Language?

2.1.1.1. Machine Language

It is a language consists of 0 and 1 and the basic of programming

language, which is the most suitable language that can be directly decoded by a computer. However, because it is difficult to understand and to use without sufficient knowledge, many programming languages have been developed in terms of versatility and time, and writing programs in machine language has almost disappeared

2.1.1.2. Assembly Language

Also called symbolic language, it is a language that can only be implemented when translated into machine language by an assembler. It consists of a label, an operation, and an operand. The label refers to the instructions of the program, the operation is a symbol of special instruction for performing such as move, add, and subtraction, and the operand represents a register or storage area where data will be processed and saved. Compared to machine language, assembly language is easy to write programs because it uses symbolic codes, and it is easy to understand because it is easy to modify, delete, and add contents. However, program written in assembly language for a specific type can only be processed on that machine but not on other type of machine, and are more difficult to read, write, and manage than programs written in high-level language.

2.1.1.3. High-Level Language

Assembly language was more convenient than machine language, but it was inconvenient to express many instructions in symbols, so languag e that could work at a higher level was required, and later a high-level language was developed for this purpose. This allowed programmer to write program independently, no longer dependent on the architecture or processor of a particular computer. In order for a program written in a n advanced language to be performed on a computer, it is called a comp iler that converts sentences in the advanced language into machine lang

uage that the computer can understand.

2.1.2. What is C Language?

C language is a programming language suitable for system technology such as operating system or language processing system, and not only basic program structure but also detailed structure such as bit manipulation are possible to describe. Most Unix operating system for mini-computer is described in this language. Recently, it has been widely used as a language to commonize software for microcomputer.

C language has several advantages. C language is concise compared to other languages, which is a key feature of C language. In addition, C language is an efficient language, and program written in C language is small in size, runs fast, and uses memory effectively. This is a great advantage when writing commercial program with assembly language level efficiency. C language is also capable of low/high level programming, and most of the electronic device hardware control is developed in C language. In the past, the assembly language was used widely, but assembly language had to change the program according to the CPU used, whereas C language was easier to maintain when the embedded program was written. C Language can delicately control various hardware equipment by manipulating pointer and bit. In addition, since C language supports programming in module unit and enables divisional compilation, high-level programming is possible and various techniques of software engineering such as top-down design, structured programming, and modular design can be applied. In addition, portability is good, which means that a program written once can be easily ported to hardware with a different CPU. However, C language is difficult for beginner to learn because it is not a language for education but a language used in the industrial field.

Algorithm is essential to use C language, which refers to the

step-by-step procedure that a computer performs a certain task. Expressing an algorithm formally is an important factor in writing a program, and depending on whether the algorithm is good or bad, there can be a big difference in time or functionality even in the process of obtaining the same result. Algorithm should have the following features:

① Input: There may be data provided from outside.

② Output: at least one outcome

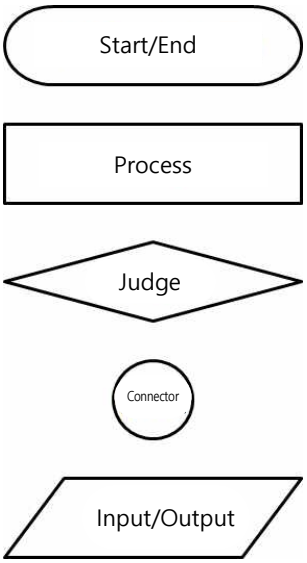
③ Clarity: Each command should be clear.

④ Finite: When the algorithm is ordered, it ends after a limited number of steps processing.

⑤ Effectiveness: All commands must be clear and practicable.

2.1.3. What is Algorithm?

Algorithm can be expressed differently, but it is usually expressed in a flow chart. It expresses the logic sequence or operation sequence in a program using geometric symbol. Rectangle is used for processing, diamond is used for judgment, and parallelogram is used for input/output processing.



[Figure 2-1] Flowchart Sign

However, if the algorithm is complicated, flowchart becomes complicated to describe, and once the algorithm is written, each step is written using a programming language.

This writing is called coding, and implementing a commonly developed algorithm can be done in any programming language. Description of the desired work using a programming language is called a source code, and the source code is usually entered using an editor and saved as a text file. Windows can also be used as a simple editor, such as Notepad, or you can use an editor built into Visual C++, an integrated development environment.

2.1.4. Build and Compile

After creating the source file, the next step is to build the source,

which is to Compile. The compiler analyzes the source file and converts it into machine language so that it can be executed on a specific computer. Integrated development environment such as Visual C++ includes a compiler. During compilation, it analyzes the sentence in the source file to check if they are written in accordance with the syntax, and if an error is found, it notifies the user of the error and terminates compilation. If there is no error, the compiler converts each statement into machine language. This machine language file is called an object file. However, even if the conversion is made, it may not be executed as intended due to an algorithmic problem, and this error is called a logical error. In this case, the source program must be modified and recompiled. These modification is called debugging, and the tool used for this is called a debugger. Typical program receives data from the outside, processes the data, and outputs the result.

2.2. Components of C Program

2.2.1. Program

When a program starts in Arduino, the void setup() function is executed once. And the program inside the loop() function repeats infinitely. This is to prevent all microcomputers from stopping running when the program is terminated.

In the first example, after performing an addition operation with two integers, the operation result is consecutively displayed on the serial monitor. In this program, we use the concept of variables to store data in memory, perform operations, and see how to output the value of a variable using the print() function. Write an example as follows.

Example 2-1) : "Serial Monitor" Program

◆ Serial.begin() : This function specifies the serial communication speed.

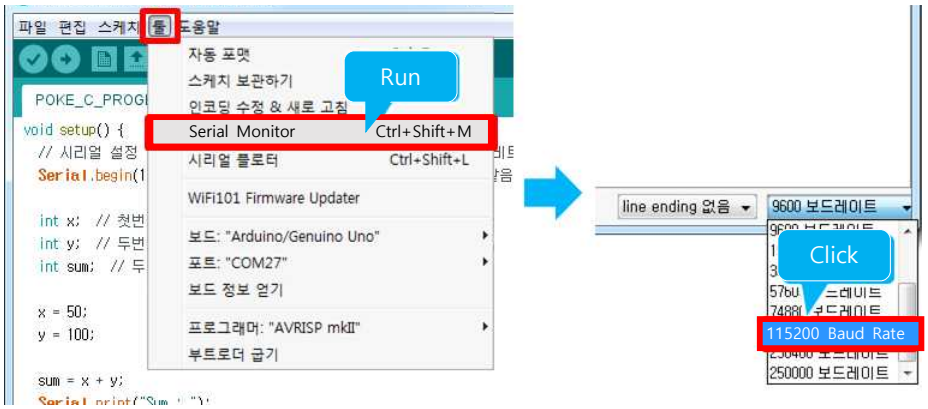
◆ Serial.print(), Serial.println() : Both functions do the same role. If you put in a variable and a string other than an array, the value is output. However, the println() function appends a newline character after outputing.

ARDUINO_KIT_PROGRAM1.ino

```
void setup() {
  // Serial Setting : Rate 115200, Data 8 bits, No Parity, 1 Stop bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  // Wait for the serial port to be connected
  while(!Serial); // If serial communication is connected, Serial =1.

  int x;  // Declare a variable to store the first integer
```

```
  int y;  // Declare a variable to store the second integer
  int sum;  // Declare a variable to store the sum of two integers

// Assignment Operation
   x = 50;
   y = 100;
  // Arithmetic operation (addition)
   sum = x + y;

   // Outputs the operation value in serial.
  Serial.print("Sum : ");
  Serial.println(sum);
}

void loop() { }
```

After the upload is completed, run the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. Set the speed to 115200 after execution.



[Figure 2-2] Execute Serial Monitor and Set Baud Rate

The result is output as shown in the following figure. Sum= 150 is dis played on the serial monitor.

[Figure 2-3] Output Result Value

2.2.2. Comments

Comment is a text written between /* and */ that explains what the program does, and does not affect the execution result of the program. Comments are not essential, but are intended to improve readability or to avoid confusion during program maintenance. Another way to comment is to put // before the text to be commented. This is common in C++ and Java. The following is how to add comment. First, the contents between "/*" and "*/" become comment.

```
/* one line comment */
/* comment
   over
   two lines */
```

And // the following line is annotated.

```
// This line is fully comment
int x; // from here to the end of the line is comment
```

Indentation is important when writing code, because indentation adds space at the same level to improve readability of the source code. If you don't indent, it runs the same, but it's a difficult program to read. Also, it is recommended to write only one sentence per line, and it is good to

put appropriately blank line between sentences to make it easier to understand. Relatively simple programs can be read without any problem without indentation, but as it grows longer it becomes very difficult to read without indentation.

2.2.3. Preprocessor

Preprocessor is a directive, and all preprocessor directives start with # sign, usually starts with the first column. There must be no space between the # symbol and include, and it is a sentence meaning to include stdio, a header file (h), in the source code. Here, a header file is a text file containing a part of code and has a “.h” extension.

2.2.4. Function

Function is a parenthetical set of processing steps that perform a specific function, such as taking input and producing output. The tasks performed by a function are listed in braces ({}) and these detailed steps are called sentences, entering a sentence outside the function other than a few cases is an error and always ends with a semicolon (;) at the end of the sentence. In addition, a function is a basic unit that consists C program and can be directly created or imported from an external source. Function directly written by the user is called a user-defined function, and a function imported and used from outside is called a library function.

In order to execute the statement inside the function, the function must be called. At this time, calling means executing the function using the function name. To call a function, just write down the name of the function, and when the function is called, the sentences in the function are executed sequentially from top to bottom. One program can have multiple functions, of which the first priority is the main() function in a

general compiler and the void setup() function in an Arduino compiler. The void loop() function runs indefinitely after the setup() function runs once.

2.2.5. Variable

Variable is a memory space used for temporarily storing data used by a program. Depending on what data you store, you can also decide whether to store integer, real number, or character. To use a variable in C language, first declare a variable, which is to inform the compiler in advance to use a variable.

The way to declare a variable is to write down the data type and the name of the variable. The data type specifies the type of data to be stored by a variable, and there are integer type, real number type, and character type.

2.2.5.1. Integer Type: short, int, long, Real Number Type :float, double

Integer type is the most basic data type and can store integers. In C language, it is divided into short, int, and long according to the number of bits that store integer. Integer in mathematics has a range up to infinity, but in computer, the number of assigned bits is limited, so only a limited range can be expressed. In the method of expressing integer, there are modifiers called unsigned and signed. Unsigned represents only non-negative value, and Signed represents that negative numbers are also possible. However, since the number of allocated bits is limited, there are some things to be careful about. When performing arithmetic operations such as addition using integer variable, overflow occurs if the result exceeds the range indicated by the integer type, and the program starts from the beginning again if it is outside the limit.

Overflow is a situation in which the number assigned to a variable is

too large to store. Float variables cannot store numbers greater than about 1*1038. Assigning a value larger than this will cause an overflow, and a warning will be issued in compiling.

Contrary to overflow, there is underflow, which is a situation where floating point number is too small to represent. Since the float type has a maximum of 6 significant digits, the smallest value in the exponent is 8-38. Since the exponent already has the smallest value, the compiler tries to adjust the mantissa to match it, but since the significant digit is 6, so it causes to make the value of 0.

[Table 2-1] Integer Type, Real Type Declaring Commands

Type	Volume	Contents	Remarks
boolean	1byte	true, false(bool type)	true(=1) or false(=0)
char	1byte	character (ASCII code)	-128~+128
byte(unsigned char)	1byte	byte	0~255
int	2bytes	Integer (same as short	-32,768~+32,767
unsigned int	2bytes	integer without sign	0~65,536
long	4bytes	long integer	-2147,483,648 ~ +2147,483,647
unsigned long	4bytes	long integer without sign	0~4,294,967,295
float	4bytes	real number	-3,4028235E+38 ~+3,4028235E+38
double	4bytes	real number(same as float)	-

2.2.5.2. character type

It means a single letter, number, symbol, etc. in Korean or English. Number is xpressed in letter to make it easier for people to see a computer that expresses everything in number. To exchange character

data, a common standard was required, and C language uses one of them, a standard called ASCII (American Standard Code for Information Interchange). This standard has 33 non-printable control character codes and 95 printable character codes. ASCII codes represent characters using numbers from 0 to 127. So, in fact, only 7 bits are required to represent only ASCII code. However, since the char type is 8 bits, half of the space remains even after expressing ASCII code, and the remaining 1 bit is occupied by extended ASCII code. Since character is expressed as integer, data types that can store integers can also be stored as character and variables of char type can be declared as char test;. If the character A is stored in the variable test, 65, the ASCII code value representing A, is set to test = 65; In order to avoid the inconvenience of writing ASCII code values, C uses single quotation marks (''), and when expressed as 'A', it means the letter A.

So test = 'A'; becomes the same code as the above code.

2.2.6. Identifier and Reserved Word

Identifier must be created according to the following rules, consisting of English letter, number, and an underscore _, and no space in the middle of the identifier. The first letter of an identifier must be an alphabetic character or an underscore _. Identifiers cannot start with a number. And uppercase and lowercase letters are distinguished.

Identifier same as keyword of C language is not allowed. Keyword has its own meaning in C language and are also called reserved word. Keyword is prohibited from being redefined or used by user, and there are keywords like this.

auto	break	case	char
double	else	enum	extern
int	long	register	return
struct	switch	typedef	union
const	continue	default	do
float	for	goto	if
short	signed	sizeof	static
unsigned	void	volatile	while

2.3. Equation and Operator

A computer is basically a calculating machine. Therefore, let's take a look at the operators provided in the program.

Types of Operator	Operator	Meaning
assignment	=	Assign right into left
arithmetic	+ - * / %	Four arithmetic operations and the remainder operations
sign	+ -	Display positive and negative number
increment,	++ --	Increasing and decreasing operation
relation	< > == != <= >=	Compare Right and Left
logic	&& || !	Logical AND, OR
condition	?	Choose according to conditions
comma	,	Run operand sequentially
bitwise operator	& | ^ ~ << >>	Bitwise AND, OR, XOR, Shift, Invert
sizeof operator	sizeof	Returns the size of a data type or variable in bytes
type conversion	(type)	Convert the data type of a variable or constant
pointer operator	* & []	Calculate the address, extract the content of where the pointer points
structure operator	. ->	Referencing member of structure

2.3.1. Arithmetic Operator

It is an operator that performs basic arithmetic operations such as addition, subtraction, multiplication, division, and the remainder operation.

Operator	Sign	Example	Result Value
addition,	+	9 + 5	14
subtraction,	-	9 - 5	4
division	/	9 * 5	45
multiplication	*	9 / 5	1
remainder	%	9 % 5	4

Example 2-2) : "Addition Operation" program

Let's implement a simple addition program.

◆ Serial.available() : function checks whether serial data has been entered

◆ Serial.parseInt() : function inputs an integer number as a serial

◆ Serial.print() : function that outputs any variable except an array

◆ Serial.println() : Add a newline character after outputing

ARDUINO_KIT_PROGRAM2.ino

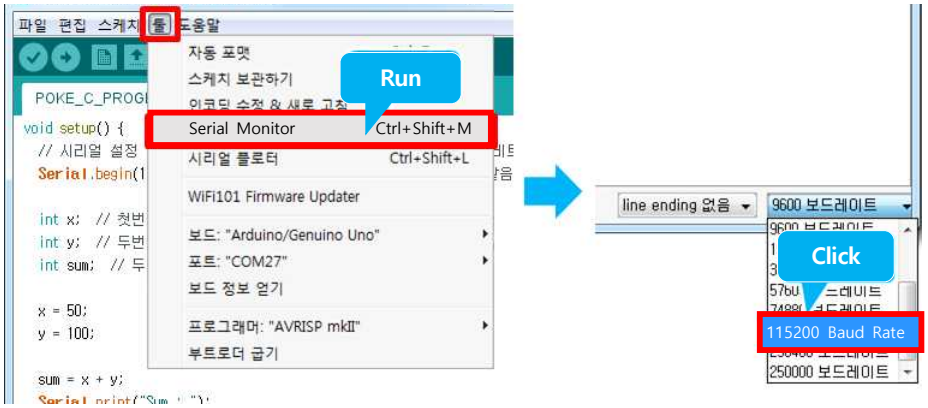
```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, No parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  //Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int x;  // Declares a variable to store the first integer
```

```
  int y;  // Declares a variable to store the second integer
  int result;  // Declares a variable to store the sum of two integers

  // Outputs the contents of “Please enter the first integer” serially
  Serial.print("Please enter the first integer : ");
  // Wait until there is an input value
  while(Serial.available() == 0);

  // Stores the first entered integer value
  x = Serial.parseInt();
  // Outputs the input integer value
  Serial.println(x);

  // Outputs the contents of “Please enter the second integer” serially
  Serial.print("Please enter the second integer : ");
  // Wait until there is an input value
  while(Serial.available() == 0);

  // Stores the second entered integer value
  y = Serial.parseInt();
  // Outputs the input integer value
  Serial.println(y);

  // Assigns the addition operation of the input two integers into the
result
  result = x + y;
  // Outputs the result of the addition operation
  Serial.print(x);
  Serial.print(" + ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(result);

  // Assigns the subtraction, operation of the input two integers into the
result
  result = x - y;
```

```
  // Outputs the result of the subtraction operation
  Serial.print(x);
  Serial.print(" - ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(result);

  // Assigns the multiplication operation of the input two integers into the result
  result = x * y;
  // Outputs the result of the multiplication operation
  Serial.print(x);
  Serial.print(" * ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(result);

  // Assigns the division operation of the input two integers into the result
  result = x / y;
  // Outputs the result of the division operation
  Serial.print(x);
  Serial.print(" / ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(result);

  // Assigns the rmainder operation of the input two integers into the result.
  result = x % y;
  // Outputs the result of the remainder operation
  Serial.print(x);
  Serial.print(" % ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(result);
```

```
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After execution, set the baud rate to 115200.

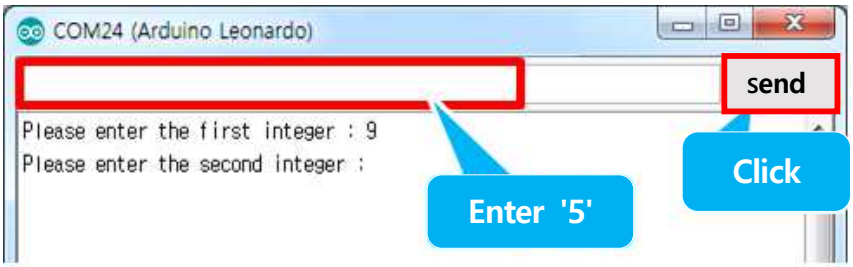


[Figure 2-4] Execute Serial Monitor and Set Baud Rate

When the output is shown in the following figure, enter '9' and click 'S end'.

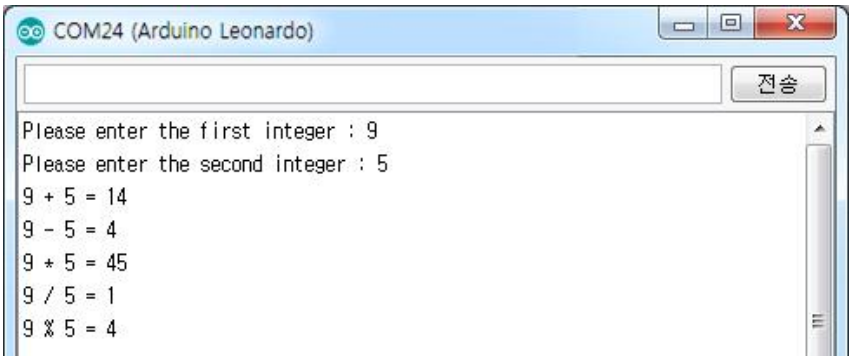
[Figure 2-5] Enter the First Integer

When the output is shown in the following figure, enter '5' and click 'S end'.



[Figure 2-6] Enter the Second Integer

The result value is output as shown in the following figure.



[Figure 2-7] Output Result Value

Example 2-3) : "Deposit Calculation " Program

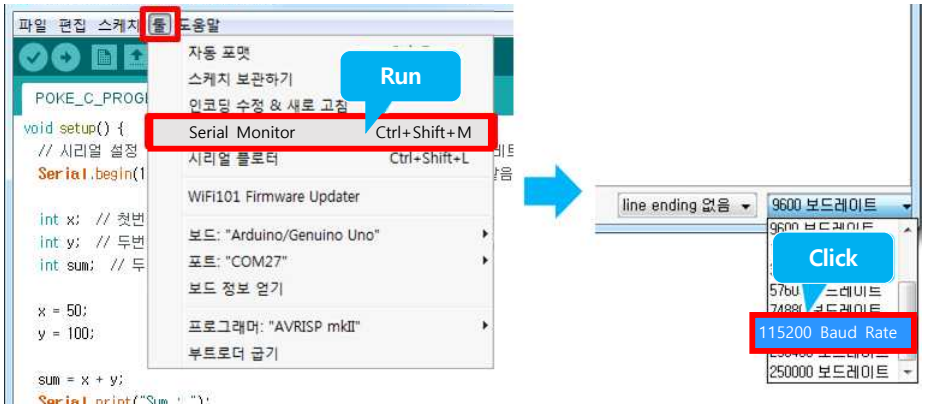
Let's write a program to calculate how much money you can save if you save 30% of your salary for 5 years. First, after receiving the amount of monthly salary, multiply it by 0.3, then multiply the result by 12 to calculate the amount you can save in one year. And multiply this by 5 again to calculate the amount you can save for 5 years.

ARDUINO_KIT_PROGRAM3.ino

```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8 bit, No parity, Stop 1 bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait until the serial port is connected
  while (!Serial); // If serial communication is connected, Serial =1

  int salary  ; // salary
  int deposit ; // deposit
  int deposit_sum ;

  Serial.print("Input salary ==> ");
  while( Serial.available()==0);
```

```
  // Save and output monthly salary input value
  salary = Serial.parseInt();
  Serial.println(salary);

  deposit = salary * 0.3;
  deposit_sum = deposit * 12 * 5;

  Serial.print("deposot_sum = ");
  Serial.println(deposit_sum);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-8] Execute Serial Monitor and Set Baud Rate

When the output is as shown below, enter '100' and click 'Send'.

[Figure 2-9] Enter Salary

The resulting value is output as shown in the following figure.

[Figure 2-10] Output Result Value

Example 2-4) : "Get the circumference of a circle" program

The radius of the circle is input from the user, the circumference of the circle is obtained, and then output to the serial monitor.

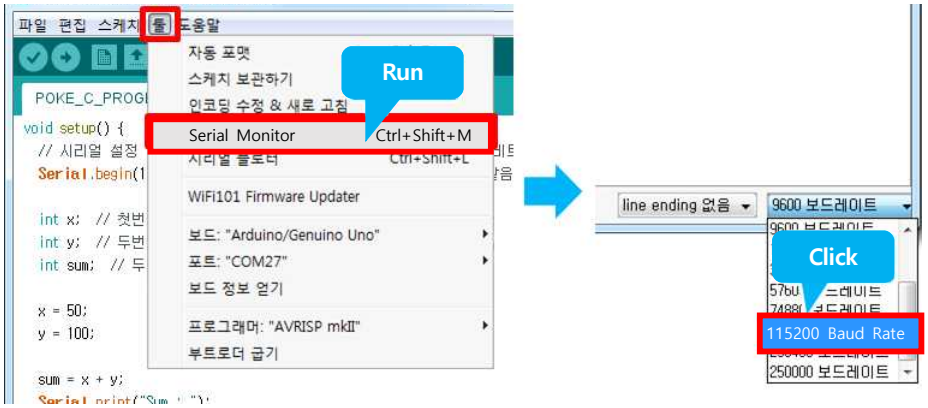
◆ Serial.parseFloat() : This is a function that inputs a real number as a serial number.

ARDUINO_KIT_PROGRAM4.ino

```
void setup() {
  // Serial Setting: Baud Rate 115200, Data 8 bit, No parity, Stop 1 bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait until the serial port is connected
  while (!Serial); // If serial communication is connected, Serial =1.

  float radius  ; // radius
  float circumference ; // circumference

  Serial.print("Input radius ==> ");
  while(Serial.available()==0) ;

  // Save radius and output radius
  radius = Serial.parseFloat();
  Serial.println(radius);

  //Calculate the circumference
```

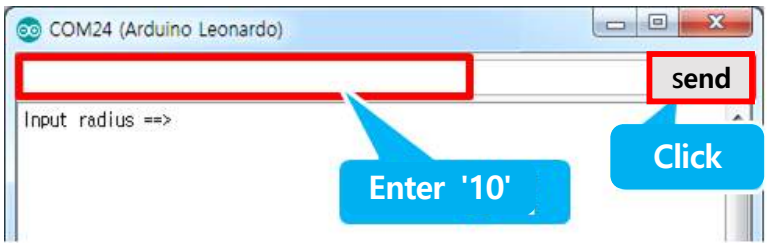
```
  circumference = 2.0 * radius * 3.14;

  Serial.print("circumference = ");
  Serial.println(circumference);

}

void loop() {}
```

After uploading is completed, execute the serial monitor in the tool as shown below to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-11] Execute Serial Monitor and Set Baud Rate

If the output is as shown in the following figure, enter the radius '10' a nd click 'Send'.



[Figure 2-12] Enter Radius

The resulting value is output as shown in the following figure.

[Figure 2-13] Output Result Value

2.3.1.1. Sign Operator

Sign operator is an operator that indicates or changes the sign of a variable or constant, changing a value to a negative number by adding - sign in front of it. This sign operator is unary operator because only one operand is required, but + and - are also called binary operator that represent addition and subtraction.

2.3.1.2. Increment Operator and Decrement Operator

Increment operator is an operator that increments or decrements the value of a variable by 1 using ++ sign or -- sign and is a unary

operator and takes only one operand. ++x operator increments the value of variable x by 1, and the -- x operator decrements the value of variable x by 1. The increment operator can come before or after the operand, and if it is used only for the purpose of increasing the value of variable x, x++ and ++x are the same, but if the purpose is to use the operation value after applying the increment operator, it must be distinguished. ++x increases the value of x first and then uses it in the formula, but x== increases the value of x after using the previous value of x in the formula. If the value of x is currently 1, then y becomes 2 at the value of y = ++x; but if y = x++; then the value of y becomes 1 because the value of x is first stored in y.

Increment/Decrement Operator	Differences
++x	increased value of x
x++	not increased, original value of x
--x	decreased value of x
x--	not decreased, original value of x

Even if there are parentheses in an equation that includes the increment operator x++, the increase in the value of x occurs after the calculation of the equation is completely finished.

Even if there is (1 + x++) + 1// parentheses, the increase in the value of x is performed last. In addition, the increment/decrement operator can only be used for variable, so it does not apply to constant or formula.

Let's take an example of the difference between ++x and x++.

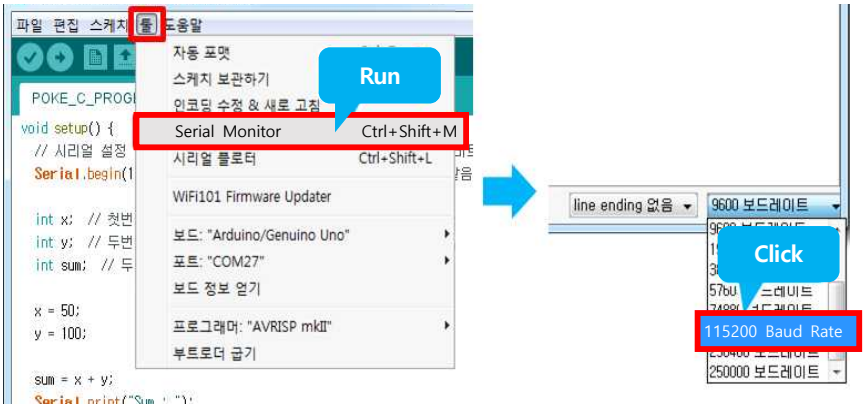
Example 2-5) : "Check increment/decrement operators" program

The user inputs the radius of the circle, calculates the circumference of the circle, and outputs it to the serial monitor.

ARDUINO_KIT_PROGRAM5.ino

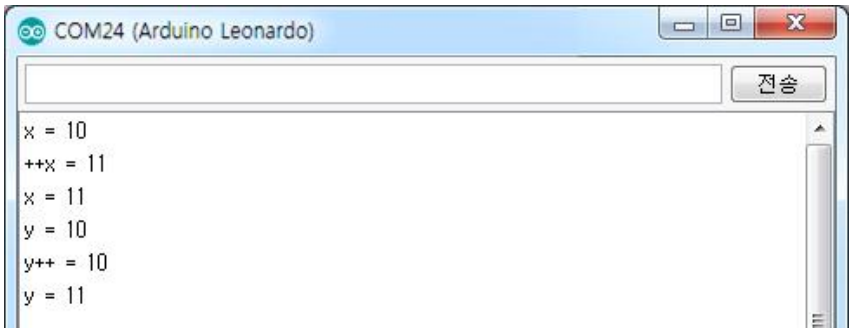
```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8 bit, No Parity, Stop 1 bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

 //Wait for the serial port to connect.
  while (!Serial); // If serial communication is connected, Serial =1.

  int x ;
  int y ;

  x = 10 ;
  // Output the initial value of x
  Serial.print("x = ");
  Serial.println(x);
  // Output the value of++x
  Serial.print("++x = ");
  Serial.println(++x);
  // Output the current value of x
  Serial.print("x = ");
  Serial.println(x);

  y = 10 ;
  // Output the initial value of y
  Serial.print("y = ");
  Serial.println(y);
  // Output the value of y++
  Serial.print("y++ = ");
  Serial.println(y++);
  // Output the current value of y
  Serial.print("y = ");
  Serial.println(y);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-14] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown in the following figure.



[Figure 2-15] Output Result Value

2.3.2. Assignment Operator

Assignment Operator is an operator that calculates and stores the value of a formula in a variable. The left side of the equal sign must be a va riable, and the right side of the equal sign can be any expression.

One of the parts that the beginner confuses is the equal sign (=). In mathematics, it is usually used to mean “equal”, but in C language, it m eans to store, so when using it, constants such as x = 100 are possible in variables, but Variables do not apply to constants, such as 100 = x.

2.3.3. Compound Assignment Operator

It is an operator that combines substitution and arithmetic operators, suc h as the compound substitution operator +=. The meaning of x + = y is equal to x = x + y.

Compound Assignment Operator	Meaning
x += y	x = x + y
x -= y	x = x - y

x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x &= y	x = x & y
x |= y	x = x | y
x ^= y	x = x ^ y
x >>= y	x = x >> y
x <<= y	x = x << y

2.3.4. Relational Operator

Relational operator is used to compare two operands and is often used t o determine the size.

Operator	Meaning
x == y	Are x and y the same?
x != y	Are x and y different?
x > y	Is x greater than y?
x < y	Is x lesser than y?
x >= y	Is x greater than or equal to y?
x <= y	Is x lesser than or equal to y?

The relationship formula produces a value of true or false, and in C language, true and false are represented as 1 or 0, so this formula is created as a value. At this time, the most common mistake is x = y, which is often represented by the same values of x and y, but this is different because it is a assignment operator, not a relation operator. Since this applies y to the variable of x, the value of the formula is

just x, and == must be used in order to use the relational operator.

2.3.5. Logical Operator

Logical operator combines several conditions to determine whether they are true or false. In C, operators that can bind conditions in various ways are prepared.

Operation	Meaning
x&&y	AND operation, true if both x and y are true, false otherwise
x||y	OR operation, true if either x or y is true, false if both are false
!x	NOT operation, false if x is true, true if x is false

It is the same as the operation of logic circuits learned in basic electroni cs.

x	y	x&&y	x||y
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

AND operator, &&, returns true only when both operands are true, and the OR operator returns true when only one operand is true.

If x is 10 or more and less than 20, the following equation is established.

(x >= 10) && (x < 20)

NOT operator makes the result of the operation false if the operand is true, makes the result of the operation true when the operand is false.

Giving some examples, in the case of !0, it is not 0, but since computer can only output 1 and 0, the resulting value is output as 1. In the case of !(x+y), if the value of the summed formula is 0, it is expressed as 1, and if it is 1, it is expressed as 0. Since a number other than 1, such as !123, is treated as a number other than 123 and made to 0, be careful whether it is positive or negative. In NOT operator, in the case of a basic logic circuit, if NOT is added to NOT, the true value comes in as a negation of negation, but in C language, if it is 0, it is false, and if it is not, it is considered to be true.

Example 2-6) : "Check Logical Operators" Program

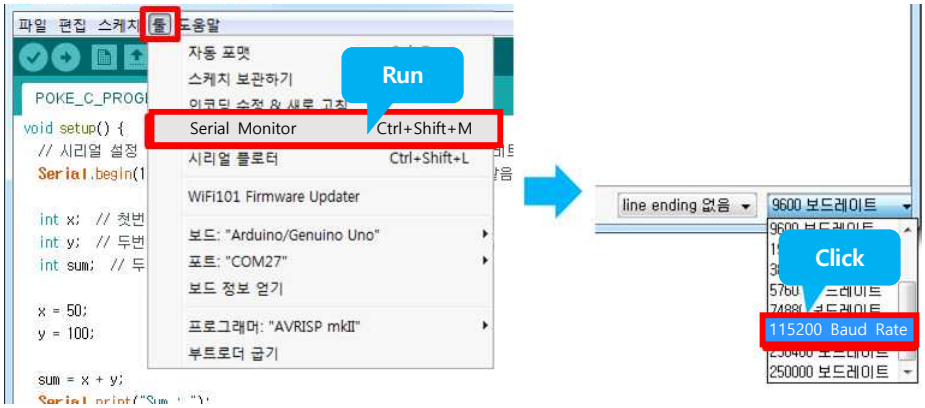
Check the function of the logical operator through the program.

ARDUINO_KIT_PROGRAM6.ino

```
void setup() {
  // Serial Setting : Baud rate 115200, Data 8 bits, No Parity, Stop 1 bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Waits for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int x , y;  // Declare a variable to store the integer

  // Outputs the contents of "Enter the first integer" serially
  Serial.print("Please enter the first integer : ");
  // Waits until there is an input value
  while(Serial.available() == 0);

  // Saves the first entered integer value
  x = Serial.parseInt();
  // Outputs the input integer value
  Serial.println(x);

  // Outputs the contents of "Enter the second integer" serially
```

```
  Serial.print("Please enter the second integer : ");
  // Wait until there is an input value.
  while(Serial.available() == 0);

  // Saves the second entered integer value
  y = Serial.parseInt();
  // Outputs the input integer value.
  Serial.println(y);

  // Outputs the result of AND operation of two input integers
  Serial.print(x);
  Serial.print(" && ");
  Serial.print(y);
  Serial.print(" : ");
  Serial.println(x && y);

  // Outputs the result of OR operation of two input integers
  Serial.print(x);
  Serial.print(" || ");
  Serial.print(y);
  Serial.print(" : ");
  Serial.println(x || y);

  // Outputs the result of NOR operation of the input x integer
  Serial.print("!");
  Serial.print(x);
  Serial.print(" : ");
  Serial.println(!x);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.

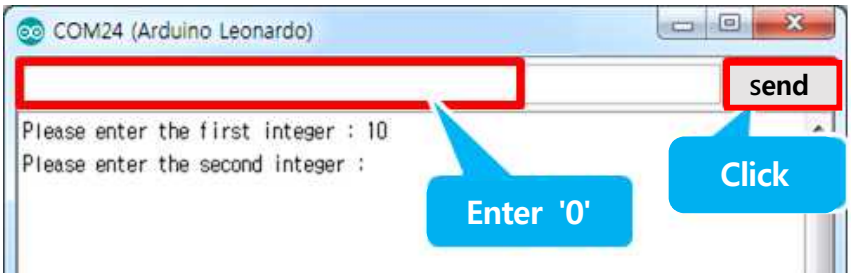


[Figure 2-16] Execute Serial Monitor and Set Baud Rate

When output is shown in the following figure, type '10' and click Send.

[Figure 2-17] Enter the First Integer

When output is shown in the following figure, type '0' and click 'Send'.



[Figure 2-18] Enter the Second Integer

When the result value is output shown in the following figure.

[Figure 2-19] Output Result Value

2.3.6. Conditional Operator

Conditional operator is the only ternary operator with three operands. You can also put statement like print() in conditional operator. For example, you can know when you write a statement that distinguishes age by inputting it.

(age > 20) ? print("adult\n") : print("teenagers\n");

Example 2-7) : "Check conditional operator" program

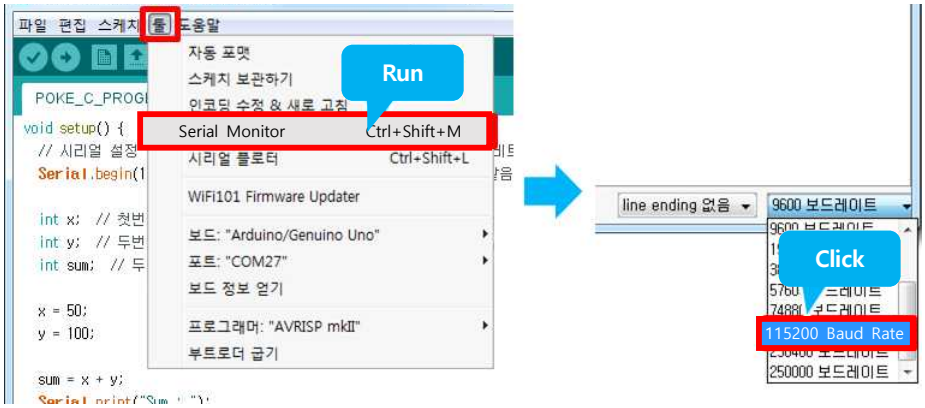
Let's look at the following example to find large and small number.

ARDUINO_KIT_PROGRAM7.ino

```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8 bits, No Parity, Stop 1 Bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Waits for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1.

  int x , y;  // Declares a variable to save the integer

  // Outputs the contents of "Enter the first integer" serially
  Serial.print("Please enter the first integer : ");
  // Waits for the input value
  while(Serial.available() == 0);

  // Saves the first input integer value
  x = Serial.parseInt();
  // Outputs the input integer value
  Serial.println(x);

  // Outputs the contents of "Enter the second integer" serially.
  Serial.print("Please enter the second integer : ");
  // Wait for the input value.
  while(Serial.available() == 0);

  // Saves the second input integer value
  y = Serial.parseInt();
  // Outputs the input integer value
  Serial.println(y);

  // Outputs a large number of two input integers as the result value.
```

```
  Serial.print("Max = ");
  Serial.println((x > y)? x : y);

  // Outputs a small number of two input integers as the result value
  Serial.print("Min = ");
  Serial.println((x < y)? x : y);
}

void loop() { }
```

After the upload is completed, run the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. Set the baud rate to 115200 after execution.



[Figure 2-20] Run Serial Monitor and Set Baud Rate

If the output is as shown in the following figure, enter '9' and click 'Send'.

[Figure 2-21] Enter the First Integer

When the output is as shown below, enter '5' and click 'Send'.

[Figure 2-22] Enter the Second Integer

The result value is output as shown below.

[Figure 2-23] Output Result Value

2.3.7. Comma Operator

Comma used as comma in normal sentence is also used as operator, but formula can be connected with comma. Each formula is executed sequentially from left to right. The result of the comma operator is saved as the value of the rightmost formula.

```
x = (1+2, 3-2);
```

After calculating 1+2 first, 3 is stored in x, but since 3-2 is calculated afterwards, 1 is eventually saved. Comma operator is used to put multiple sentences in a limited space.

2.3.8. Bitwise Operator

In a computer, all data is eventually expressed in bit, and (bit) is the smallest unit for storing information in a computer, and has only value of 0 and 1, and a frequently used Int-type Variable contains as many as 32 bits.

Operator	Meaning	Description
&	bit AND	1 if the corresponding bits of both operands are 1, otherwise 0
|	bit OR	1 if only one of the corresponding bits of two operands is 1, otherwise 0
^	bit XOR	0 if the corresponding bits of two operands have the same value, otherwise 1
<<	move to left	Move all bits to the left as many as specified number
>>	move to right	Move all bits to the right as many as specified number
~	bit NOT	0 makes 1 and 1 makes 0

Bitwise operator can only be applied to integer-type operand.

Bitwise AND operator performs an AND operation on a bit-by-bit basis. It is 1 only if both operands are 1.

0 & 0 = 0
1 & 0 = 0
0 & 1 = 0
1 & 1 = 1

Bitwise OR operator performs an OR operation on a bit-by-bit basis. If any of the operands are 1, it becomes 1.

0 | 0 = 0
1 | 0 = 1
0 | 1 = 1
1 | 1 = 1

Bitwise XOR operator performs an XOR operation on a bit-by-bit basis. 0 if the two operands are equal, 1 otherwise.

0 ^ 0 = 0
1 ^ 0 = 1
0 ^ 1 = 1
1 ^ 1 = 0

Bitwise NOT operation returns 1 if the operand is 0 and 0 if the operan d is 1.

~0 = 1
~1 = 0

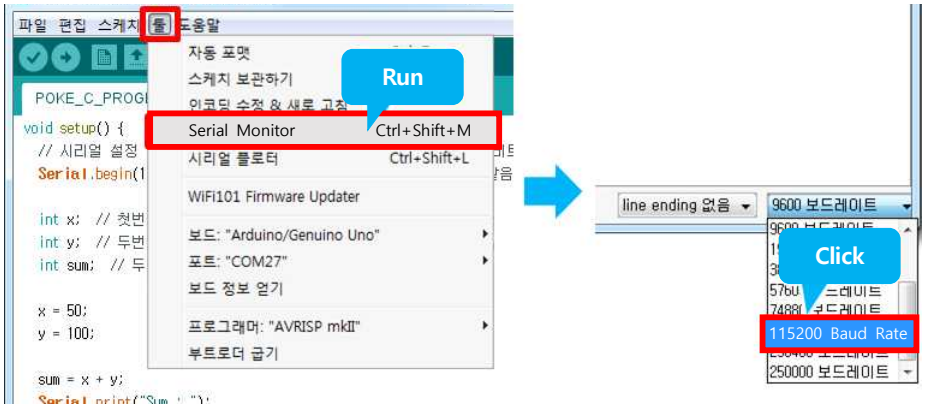
Example 2-8) : “Check Bitwise Operator” Program

Let's take a look at the following bitwise operator.

ARDUINO_KIT_PROGRAM8.ino

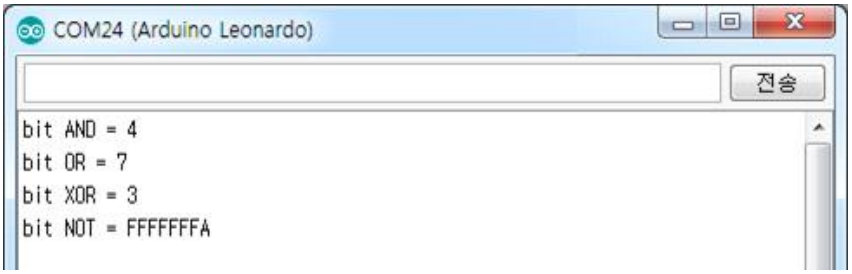
```
void setup() {
  // Serial Setting : Baud rate 115200, Data 8 bits, No Parity, Stop 1 bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Waits for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

 // A number with 0x before it means hexadecimal
  int x = 0x05, y = 0x06;

  // Outputs the value of AND operation
  Serial.print("bit AND = ");
  Serial.println(x & y, HEX);

  // Outputs the value of OR  operation
  Serial.print("bit OR = ");
  Serial.println(x | y, HEX);

  // Outputs the value of XOR operation
  Serial.print("bit XOR = ");

  Serial.println(x ^ y, HEX);

  // Outputs the value of NOT operation
  Serial.print("bit NOT = ");
  Serial.println(~x, HEX);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After executing, set the baud rate to 115200.



[Figure 2-24] Run Serial Monitor and Set Baud Rate

The resulting value is output as shown in the following figure.



.. [Figure 2-25] Output the Resulting Value

2.3.8.1. Bit Shift Operator (<<,>>)

Bit shift operator shifts all bits by a specified number. Since there are two directions to move, left and right, two operators << and >> are required.

Operator	Sign	Description
Left Bit Shift	<<	x<<y x bit shifts to the left by y places
Right Bit Shift	>>	x>>y x bit shifts to the right by y places

Left bitwise operator << shifts the entire bit pattern to the left by a specified number. Each time the bit is shifted to the left, the value is doubled.

2.3.9. Precedence and Combination Rule of Operator

2.3.9.1. Precedence of Operator

If there are two or more operators in one formula, there is a precedence and it works according to that priority. Put it in brackets to make the c alculation first.

Precedence	Operator	Combination Rule
1	() [] -> , ++(rear) --(rear)	->(left to right)
2	sizeof & (address) ++(front) --(front) ~ ! * (back reference) + (sign) -(sign), Type change	<-(right to left)
3	*(multiplication) / %	->(left to right)
4	+(addition) -(subtraction)	->(left to right)
5	<< >>	->(left to right)
6	< <= >= >	->(left to right)
7	== !=	->(left to right)
8	&(bit operation)	->(left to right)
9	^	->(left to right)
10	|	->(left to right)
11	&&	->(left to right)
12	||	->(left to right)
13	?(ternary)	<-(right to left)
14	= += *= /= %= &= ^= |= <<= >>=	<-(right to left)
15	,(comma)	->(left to right)

The order of precedence by the function of the operators is as follows.

comma < assignment < logic < relation < arithmetic < unary

The parenthesis operator () has the highest precedence, so if the programmer has the first part to calculate, enclose it in parentheses.

2.3.9.2. Combining Rules for Operators

As explained above, there are precedence, but there is also a combination rule in calculations. If x*y*z is calculated, it is calculated sequentially, but if x=y=z, it is calculated in reverse order.

2.4. Conditional Statement

2.4.1. Control Statement

In the program, each statement is executed sequentially, starting with the first part of the main() function and one statement at a time. However, since this alone cannot solve various problems, statements that affect the order by the situation are adjusted using control statement.

2.4.1.1. Conditional Statement

Depending on the conditions, it is required to choose one of two or more execution paths.

Statement that make decisions based on condition is called conditional statement, and if statement and switch statement belong to conditional statement.

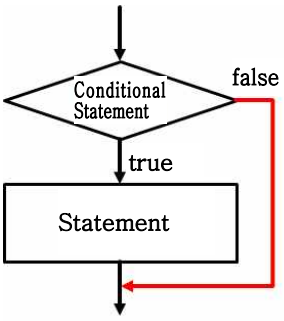
2.4.1.2. Loop Statement

Sometimes when we solve a problem, it can only be solved by repeating

the process. For example, a case in which a statement is repeated a certain number of times or a case in which it is repeated consecutively is called a loop statement. For statement and While statement belong to loop statement.

2.4.2. if Statement

In an if statement, condition is expressed as formula and the formula is called conditional expression, and if statement calculates a given conditional expression and executes the statement if the result value is true, and if it is false, the statement is not executed.



[Figure 2-26] Flowchart of
If Statement

The following example is a program that determines whether an input i nteger is even and outputs the integer value.

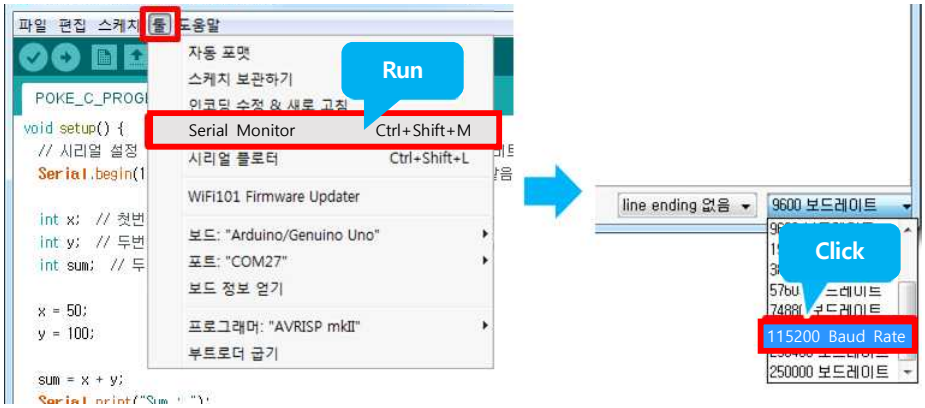
Example 2-9) : "Even Number-Determining" program

The following example is a program that determines whether an input integer is even and outputs the integer value.

ARDUINO_KIT_PROGRAM9.ino

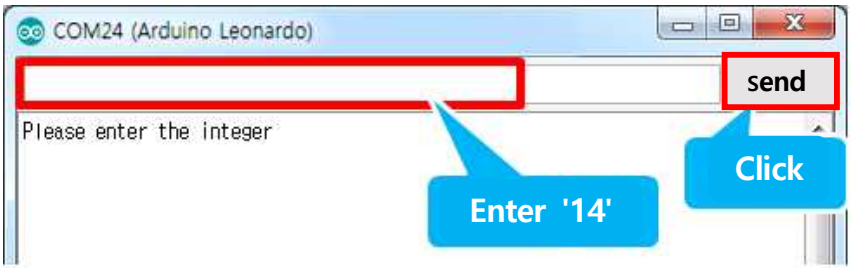
```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, No parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int number;  // Declare a variable to save the integer

  // Outputs the contents of “Please enter an integer” serially
  Serial.println("Please enter the integer");
  // Wait until the input value.
  while(Serial.available() == 0);

  // Save the entered integer value
  number = Serial.parseInt();

  // Determine whether the input integer is odd or even
  if((number % 2) == 0)
    // Outputs "This is an even number" serially
    Serial.println("It is an even number.");

   Serial.print("Input value : ");
   Serial.println(number);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown below to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-27] Execute Serial Monitor and Set Baud Rate

When the output is as shown below enter '14' and click 'Send'.



[Figure 2-28] Enter Integer

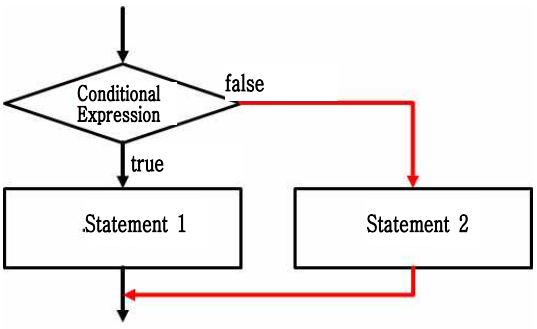
The resulting value is output as shown in the following figure.

[Figure 2-29] Output Resulting Value

Since the input integer 14 is an even number, “It is an even number.” is output. If it is an odd number, “It is an even number.” is not output.

2.4.3. if-else Statement

In if statement, only if the condition is true, it is processed, but if the condition is false, use 'else' after 'if' so as to process with conditions. For example, if the condition is true, statement 1 is executed, and if the condition is false, statement 2 is executed.

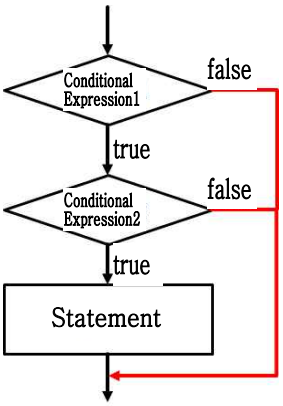


[Figure 2-30] Flowchart of if-else Statement

```
// Determine whether the input integer is even number.
if((number % 2) == 0)
  Serial.println("It is an even number");
else
  Serial.println("It is an odd number");
```

2.4.4. Multiple if Statement

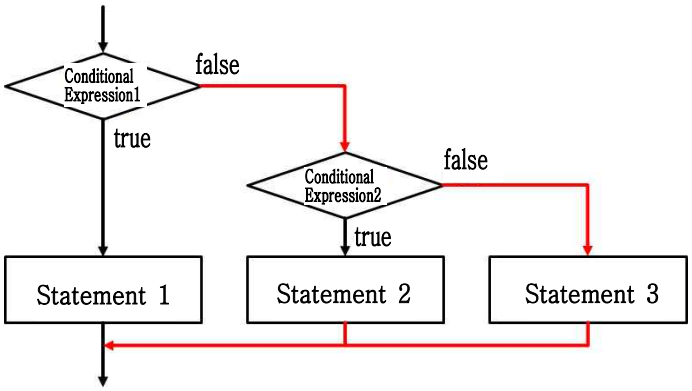
Since an if statement is also a single statement, other if statements can be included in an if statement.



[Figure 2-31]
Flowchat of Multiple
if Statement

```
if(score >= 90) {
  if(score >= 95);
    Serial.println("It's an A+ grade.");
  else
    Serial.println("It's an A0 grade.");
 }
```

If the score is 90 or higher, the multiple if statement is executed. In the multiple if-statement, if the score is 95 or higher, "It's an A+ grade"is output, otherwise "It's an A0 grade" is output. If you want to indicate that a consecutive if statement is divided into multiple statements depending on the condition, you can use 'else if' after the 'if statement' sequentially.



[Figure 2-32] Flowchart of consecutive if statement

```
if(Score >= 90)
  Serial.println("It's an A grade");
else if(Score  >= 80)
  Serial.println("It's a B grade");
else if(Score  >= 70)
  Serial.println("It's a C grade");
else if(Score  >= 60)
  Serial.println("It's a D grade");
else
  Serial.println("It's a F grade");
```

It is possible to check using consecutive if statements to determine a grade based on the score.

2.4.5. Switch Statement

Switch statement is similar to if statement, but switch statement determines the next statement to be executed according to the value of the control expression. When comparing with each case and if there is a

matching value, execute in sequence and switch statement is terminated after reaching the break statement.

```
switch(control expression)
{
 case c1 :
  statement1;
  break;
 case c2 :
  statement2;
  break;
 case c3 :
  statement3;
  break;
  ……
 default :
  statement;
  break;
}
```

If the value calculated by the user's control expression does not match the value of the case clause, the default clause is executed. Note that if there is no break statement, execute the statements in the selected case clause and then execute the next case clause consecutively. If there is no break statement, output the statements of case1 and then output the statements of case2. Let's take an example.

```
switch(control expression)
{
 case c1 :
 case c2 :
  statement2;
  break;
 case c3 :
  statement3;
```

```
    break;
    ……
  default :
    statement;
    break;
}
```

When the value of the control expression becomes c1, statement 2 is executed. Also, even if the control expression value becomes c2, statement 2 is also executed. If there is no break statement between case and case, it will be executed until the break statement reaches or the switch statement ends.

Example 2-10) : "Calculate the number of days in the month" program

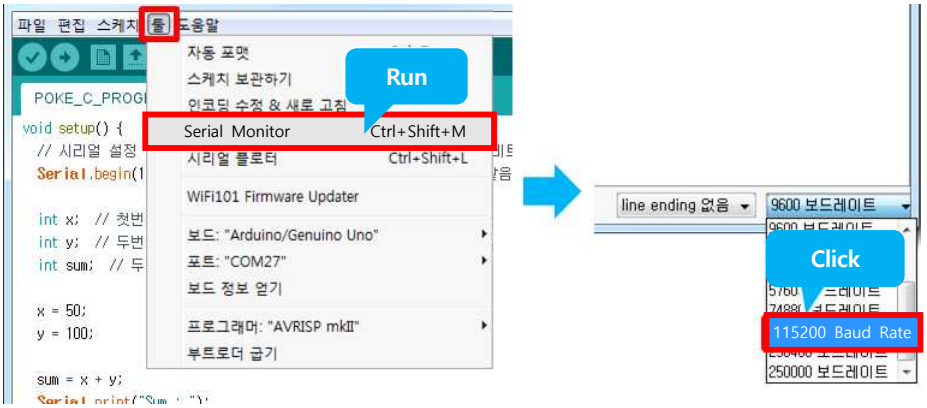
The following example is a program that inputs a month and outputs the number of days in that month.

ARDUINO_KIT_PROGRAM10.ino

```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int  month;  // Declare a variable to save the month
  int days;

  // Outputs the contents of "Please enter an integer" serially
  Serial.println("Please enter the month ");
  // Wait for the input value
  while(Serial.available() == 0);
```

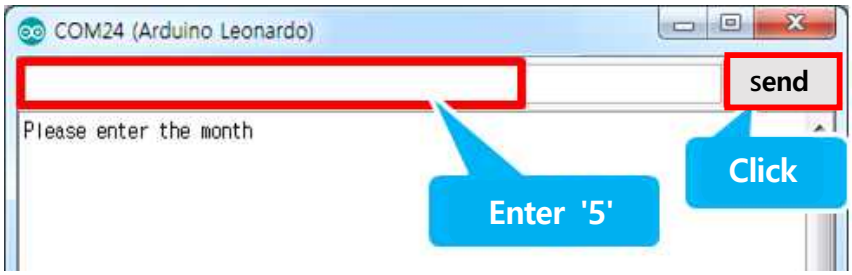
```
  // Save the entered month value
  month = Serial.parseInt();

 switch(month)
 {
  case 1: case 3 : case 5 : case 7 : case 8 :
  case 10 : case 12 :
                        days = 31 ;
  break;

  case 2: days = 28;
  break;

  default : days=30;
  break ;
 }

 Serial.print("The number of days in the  ");
 Serial.print(month);
 Serial.print("(month) is ");
 Serial.print(days);
 Serial.print("days.");
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown below to check the operation with the serial monitor. After running, set the baud rate to 115200.



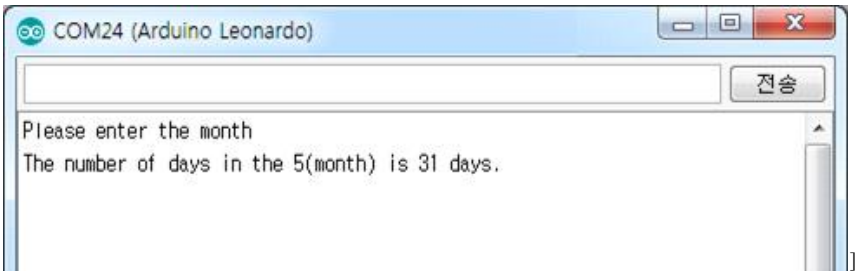
[Figure 2-33] Execute Serial Monitor and Set Baud Rate

When the output is as shown below, enter '5' and click 'Send'.



[Figure 2-34] Enter Month

The resulting value is output as shown in the following figure.



]

[Figure 2-35] Output Result Value

If you enter May, the number of days is 31 days is output.

2.5. Loop Statement

When solving problems, there are problems that can only be solved by repeating the process. Statement that repeats a process for a specified number of times is called loop statement. for statement and while statement belong to loop statement.

Sequential structure: Structure in which the given statements are executed sequentially. It is the most basic control structure.

Selection structure: This is a structure that selects and executes one of several execution paths by checking certain conditions as described in the previous chapter. It is a useful control structure that allows the computer to make logical decision.

RepeatStructure: Control structure that repeatedly executes the same statements until a certain condition is satisfied.

2.5.1. while Statement

While statement is a statement structure that repeatedly executes statements while a given condition is satisfied. Statement is repeatedly executed while the value of the conditional expression is true, and the repetition is stopped when the value of the conditional expression becomes false. If the value of the conditional expression is false from the beginning, the statement is never executed. Inside the while statement, the condition for continuing the repetition and the contents to be repeated must be included. Let's take an example.

```
while(i <  3) // condition
{
  // repeated statement
  size = i;
```

```
  i++;
}
```

To make this true, the value of i must be less than 3, and every time i is repeated from 0, the value is increased by 1 by i++ statement. When this increased value reaches 3, it becomes false, so the iteration ends.

Example 2-11) : "Unit conversion" program

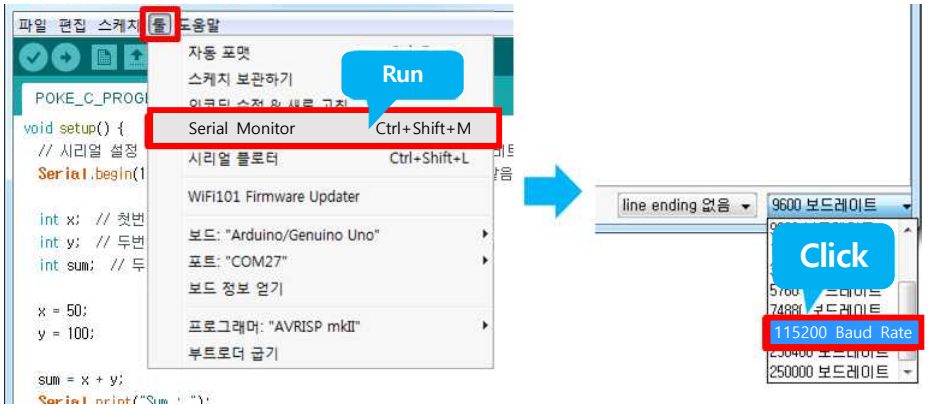
Program to convert meter to centimeter using while statement

ARDUINO_KIT_PROGRAM11.ino

```
void setup() {
 // Serial Setting: Baud Rate 115200, Data 8bit, No Parity, Stop 1bit
 Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

 //Wait for the serial port to connected
 while (!Serial); // If serial communication is connected, Serial =1.

 int centimeter;
 int meter = 0;

 while(meter < 5)
 {
   // Convert meter to centimeter and output.
   centimeter = meter * 100;
   Serial.print(meter);
   Serial.print("[m] is ");
   Serial.print(centimeter);
   Serial.println("[cm]");
   meter++;
 }
```

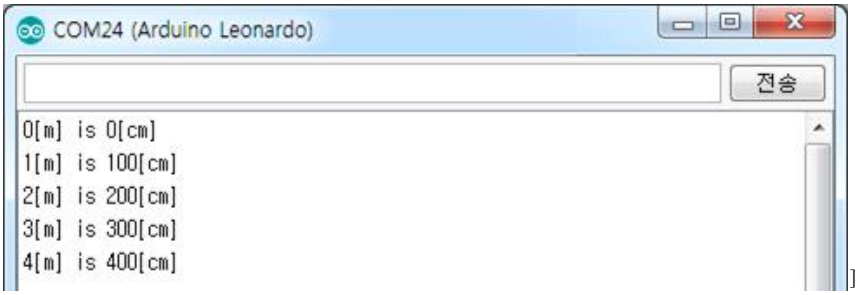
```
}

void loop() {}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-36] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown in the following figure.



[Figure 2-37] Output Result Value

2.5.2. do, while Statement

do, while statement is similar to the while statement, but the iteration c ondition is checked at the end of the loop, not at the beginning. do, whil e statement is executed at least once. The following is an example.

```
int i = 5;
do
{
    statement1;
    statement2;
} while(i < 4)
```

If i is 5 and the condition of the while statement is when i is less than 4, the condition becomes false from the start, but since it is executed at least once, statements 1 and 2 are executed once.

2.5.3. for Statement

for statement has a repeating structure that repeats a statement for a specified number of times and is also called 'for loop'. for statement consists of three parts: initial expression, conditional expression, and incremental expression, each separated by a semicolon (;).

```
for(initial expression; conditional expression; increment/decrement expression)
{
    iterated statement;
}
```

The following is an example.

```
for(i = 0; i <10; i++)
{
    Serial.println(i);
}
```

2.5.3.1. Initial Expression

Initial expression is executed only once before starting the iterative loop and is used to initialize the value of a variable. Set the initial value of the i variable to 0, such as i = 0.

2.5.3.2. Conditional Expression

Since the conditional expression determines whether the iteration continues or stops, it must be a relational or logical expression whose value can be evaluated as either true or false. Since this is calculated before iteration, if the value of the conditional expression is false, iteration does not occur, and if the value of the conditional expression is true, iteration occurs.

2.5.3.3. Increment/Decrement Expression

When one loop is executed, increment/decrement expression is executed, and the variable controlling the loop can be increased or decreased. When the increment/decrement expression is completed, the conditional expression is executed. For example, let's consider the example of outputting the name 20 times. Without loop statement, you have to write Serial.println() 20 times. However, with for statement, you write the code as follows.

```
for(i = 0: i <20; i++)
{
   Serial.println("Name");
}
```

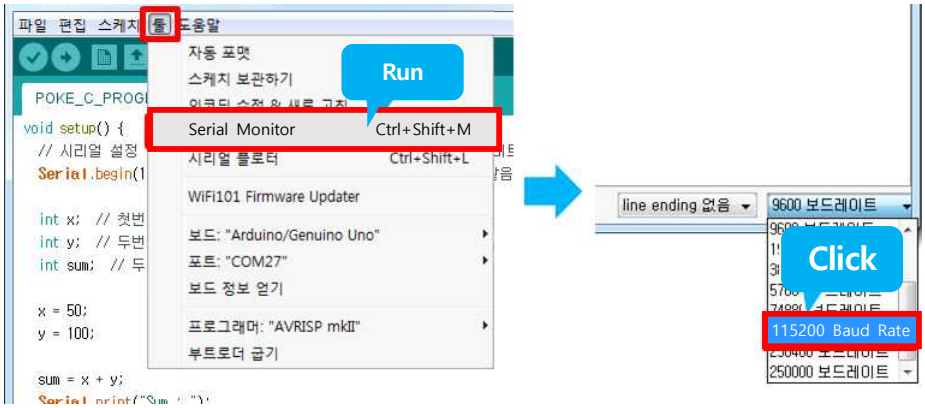
Example 2-12) : “Sum of integers from 1 to 100” program

Let's look at a program that calculates the sum of integers from 1 to 100.

ARDUINO_KIT_PROGRAM12.ino

```
void setup() {
  // Serial Setting : Baud Rate 115200, Data 8 bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1.

  int sum , i;

  sum = 0;
  for(i=1; i <= 100; i++)
  {
     sum += i;
  }
  Serial.print("Sum of 1 to 100 = ");
  Serial.println(sum);
}

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-38] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown in the following figure.

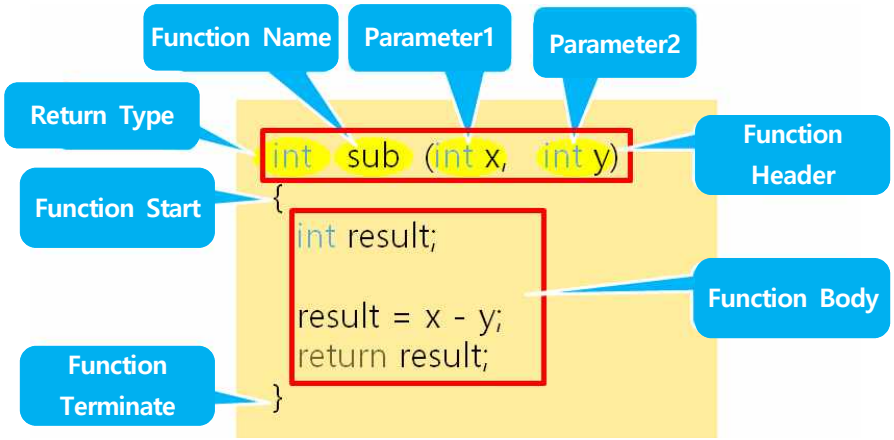
]

[Figure 2-39] Output Result Value

2.6. Function

2.6.1. Definition

Function is a basic component of a program, and a program is made up of several functions. Function can be divided into a function header and a function body. Return type, function name, and parameter list are called function header. Usually, the body is enclosed in brace, and there is the statement for the task in the body. Function name can just follow the rules for identifier but it is better fo imply the features of function.



[Figure 2-40] Structure of Function

2.6.1.1. Return Type

Return type is the data type returns to where have called after the function terminates processing, and data type includes char, int, long, double, etc. If it does not return a value, it is displayed as void. If the return type is not specified, an int type is assumed. Syntax error occurs when returning a value from a function specified as void.

```
// Function with no return value
void digitalWrite(unsigned char pin, unsigned char value);

// returns int type value
int digitalRead(unsigned char pin);
```

2.6.1.2. Parameter

When calling a function from outside, you can send the data required for the operation. The parameter is the variable that receives the data. Parameters can be many and can be separated by commas(,). For each parameter, data type and name is specified. If there is no parameter, write void in the variable location or just leave it blank.

```
// Function to control digital pin
void digitalWrite(unsigned char pin, unsigned char value);

// Function to read digital pin
int digitalRead(unsigned char pin);
```

The number of parameter and the number of argument must be same and types must be also same. The variable is defined as an int type, and an error occurs if float type data is passed. To call a function, write the name of the function and list the data the function needs in parentheses (). The statements in the function are not executed before the function is called, but when the function is called, the statements in the function are executed sequentially. After the execution is finished, it returns to its original place, and the result value at this time can be passed to the calling place. Once the function is called, code that was executing stops and moves to the function and executes. When it is finished, it resumes the previous operation.

Example 2-13) : “Input subfunction” program

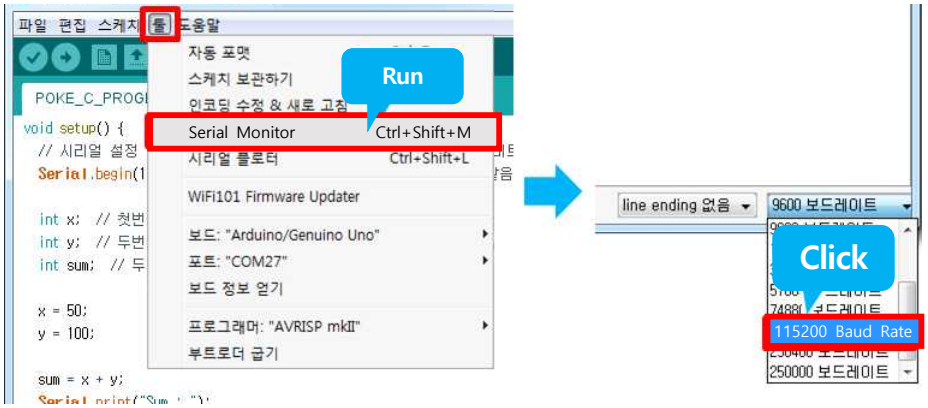
Let's look at an example of adding two integers. Example was written by changing the part that reads two integers into a function.

ARDUINO_KIT_PROGRAM13.ino

```
// Declare the function to be used at the beginning
int get_int(void) ;
void setup() {
  // Serial Setting : Baud rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

//Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int x;  // Declare a variable to save the first integer
  int y;  // Declare a variable to save the second integer

  x = get_int();
  y = get_int();


  Serial.print(x);
  Serial.print(" + ");
  Serial.print(y);
  Serial.print(" = ");
  Serial.println(x + y);
}

void loop() { }

int get_int(void)
{
  int n;

  // Outputs the contents of “Please enter an integer” serially
  Serial.print("Please enter the integer : ");

  // Wait for input value
  while(Serial.available() == 0);
```

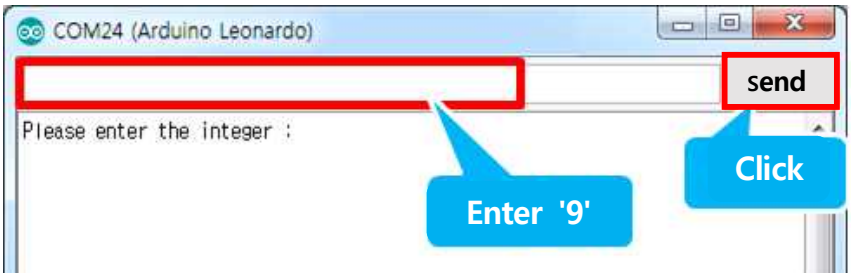
```
  // Save the entered integer value.
  n = Serial.parseInt();

  // Output the entered integer value.
  Serial.println(n);
  return n;
}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-41] Execute Serial Monitor and Set Baud Rate

If the output is as shown below, enter '9' and click 'Send'.

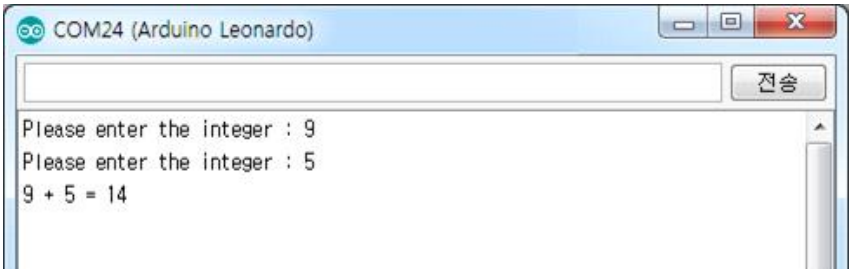


[Figure 2-42] Enter Integer 1

When the output is as shown below, enter '5' and click Send.

[Figure 2-43] Enter Integer 2

The resulting value is output as shown below.



[Figure 2-44] Output Result Value

Example 2-14) : "Function to get the square of an integer " Program

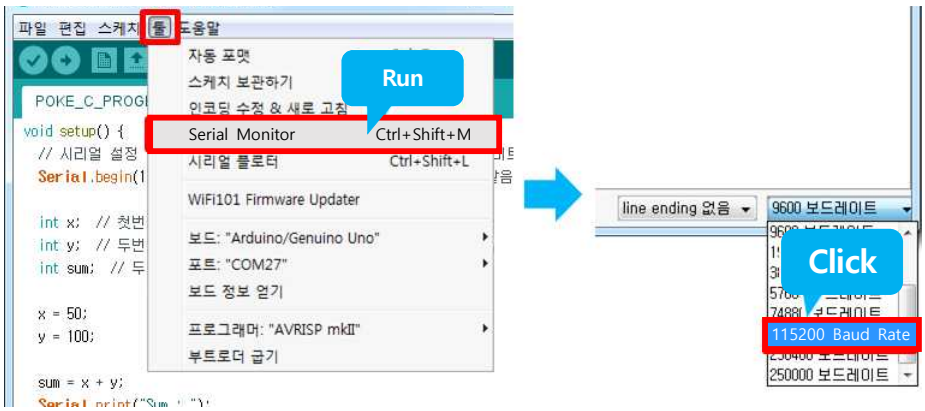
We will write a function program that calculates and returns the square of an integer with the following example.

ARDUINO_KIT_PROGRAM14.ino

```
int get_int(void);
int square(int n) ;

void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)


//Wait until the serial port is connected
  while (!Serial); // If serial communication is connected, Serial =1

  int num;  // Declare a variable to save the input integer
  int result;

  num = get_int();
  result = square(num);


  Serial.print("result =  ");
  Serial.println(result);
}

void loop() { }

int get_int(void)
{
  int n;

  // Outputs the contents of "Please enter an integer" serially
```

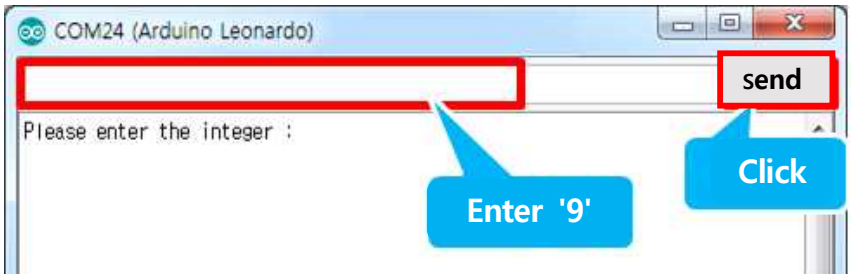
```
  Serial.print("Please enter the integer : ");

  // Wait for input value
  while(Serial.available() == 0);

  // Save the entered integer value
  n = Serial.parseInt();

  // Output the entered integer value
  Serial.println(n);
  return n;
}

int square(int n)
{
  return (n * n) ;
}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



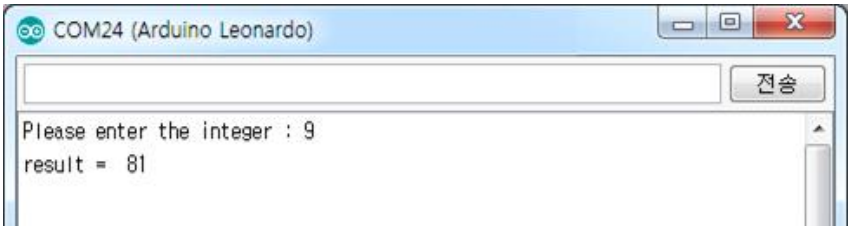
[Figure 2-45] Execute Serial Monitor and Set Baud Rate

If the output is as shown below, enter '9' and click 'Send'.



[Figure 2-46] Enter Integer

Resulting value is output as shown in the following figure.



[Figure 2-47] Output Result Value

2.6.2. Library Function

Functions are divided into user-defined function and library function. One of the reasons why the C language is widely used is that it supports library well. To use a library function, function prototype should be included first. In general, prototypes of library functions are not included individually, but header files that contain the prototypes can be included.

2.7. Function and Variable

2.7.1. Variable

Variables have several basic properties, such as scope, lifetime, and linkage.

2.7.2. Scope

Scope of variables is determined by the location where it is declared. Variables can be divided into global variable and local variable.

2.7.3. Local Variable

It is a variable defined in a function or block. Block refers to curly brace. Local variable must be defined at the beginning of the block and cannot be defined in the middle. It is not possible to declare a local variable after a variable declaration ends and a normal statement begins. Local variable is valid only within the region, so there is no error even if there is a variable of the same name within another region. Unlike global variable, local variable is created in a memory space called stack and initialized at the same time. Local variable is not automatically initialized to 0 by the compiler, and is set to meaningless value unless an initial value is set. Header of the function is also a kind of local

variable, and is initialized with the argument value when called.

Example 2-15) : "Local variable example" program

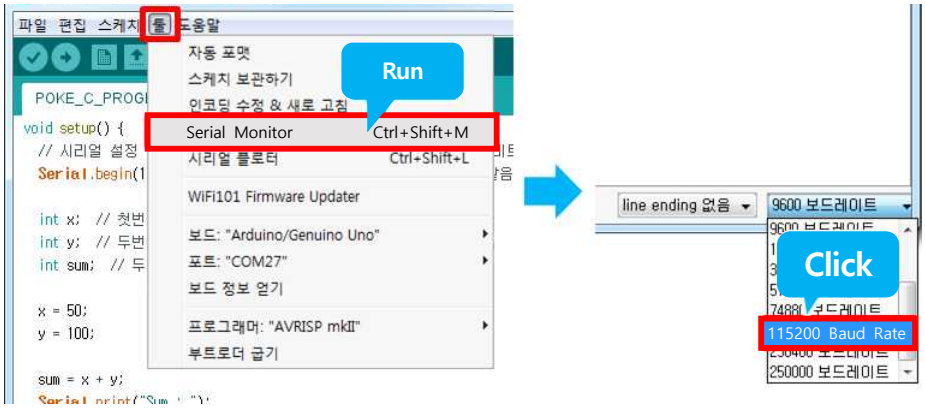
Let's write a program example for local variable.

ARDUINO_KIT_PROGRAM15.ino

```
void setup() {
  // Serial Setting : Baud Eate 115200, Data 8 bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while(!Serial); // If serial communication is connected, Serial =1

  int i;
   for(i = 0; i < 10 ; i++)
  {
    int num = 1;  // declare local variable

    Serial.print("num =  ");
    Serial.println(num);

    num = num + 1 ;
  }
 }

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial

monitor. After running, set the baud rate to 115200.



[Figure 2-48] Set Serial Monitor and Set Baud Rate

[Figure 2-49] Output Result Value

2.7.4. Global Variable

Global variable is declared outside a function and can be used anywhere in a file. If the programmer does not initialize it, it is initialized to 0 by the compiler. Global variable is created at the time of program start and remain in memory until the program is terminated. It is possible to make an access not only at the same time as the program but also from the entire area until it is terminated. It seems to be easy to use, but rather, the more complex the program is, the more it is difficult to know where the variable is used. If a local variable with the same name as the global variable is declared, the local variable takes precedence over the global variable.

Example 2-16) : "Global Variable Example" Program

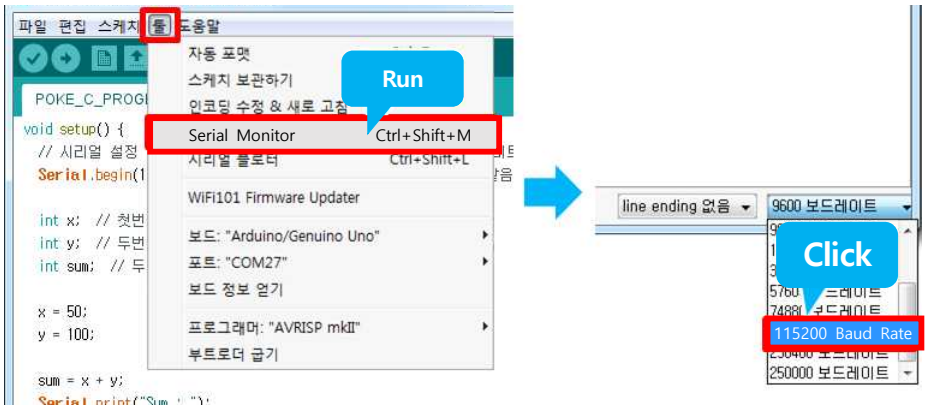
Let's write a program example for global variable.

ARDUINO_KIT_PROGRAM16.ino

```
int G = 1234; // Global variable setting

void sub1(void);
void sub2(void);

void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait until the serial port is connected
  while (!Serial); // If serial communication is connected, Serial =1

  sub1();
  sub2();
```

```
}

void loop() { }

void sub1(void)
{
    Serial.print("sub1 =  ");
    Serial.println(G);
}

void sub2(void)
{
    Serial.print("sub2 =  ");
    Serial.println(G);
}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-50] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown below.

[Figure 2-51] Output Result Value

2.7.5. Lifetime

Lifetime means the time that a variable lives, and if variables are classified according to this, they are divided into static allocation and automatic allocation.

2.7.5.1.1. Static Allocation

Storage space is continuously allocated to the variable while the program is running.

2.7.5.1.2. Automatic Allocation

When the block starts, storage space is allocated to the variable, and when the block ends, the storage space is returned.

Global variable is statically allocated, but local variable is automatically allocated by default. However, it can be changed to static allocation by attaching a storage type specifier.

2.8. Array

Array is one of the basic structures for storing large amount of data

and is a data storage area where many data of the same type are stored. Data in the array is accessed to each other, and data access is possible by attaching number to the array name. The following example is when declaring an array that can store x integer data.

```
int s[x];
```

Data stored in an array is called Array Element, and array element has a number called index or subscript attached to it, which is stored in a consecutive memory space. Arrays are basically listed in the order of data type, array name, and number of elements, and are expressed as follows.

```
int array[5];
```

The above statement is an sequence named array that can store 5 int type integers. Since the size is 5, each element is starting 0 and ending 4. Therefore, if array[5] = 10; is set, this is an incorrect definition because it is in the range of 5, not the range of 0 to 4. In the case of array declaration, it can be declared together with general variable, or several arrays can be declared together in one line, and the size of array cannot be designated as a variable, negative number, or real number.

2.8.1.1.1. Array Initialization

To initialize an array, separate the values with comma, surround them in curly braces {}, and assign them when declaring the array. If you declare it as follows, all elements are initialized to 0.

```
int array[5] = {0};
```

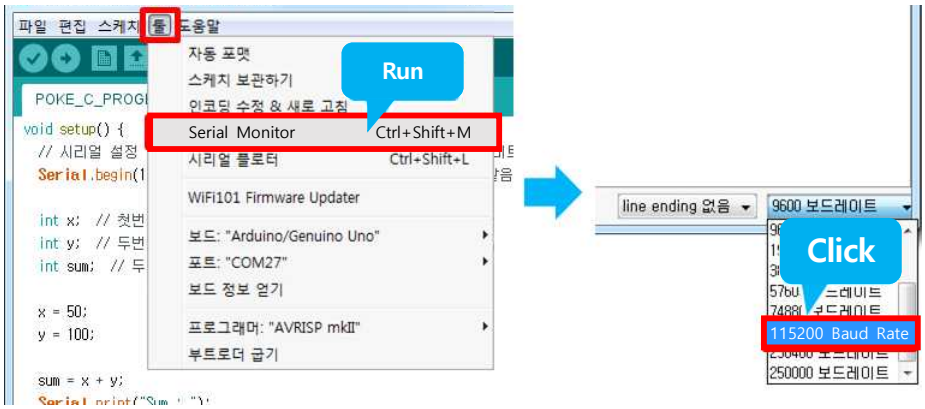
Example 2-17) : “Array Example (1)” Program

Let's write a program that allocates a value to an array element.

ARDUINO_KIT_PROGRAM17.ino

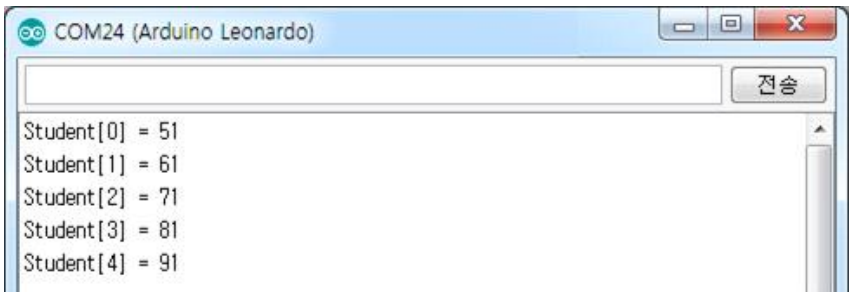
```
void setup() {
  // Serial Setting : Baud rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while (!Serial); // If serial communication is connected, Serial =1

  int i;
  int Student[5] ;

  Student[0] = 51;
  Student[1] = 61;
  Student[2] = 71;
  Student[3] = 81;
  Student[4] = 91;

  for( i= 0 ; i < 5 ; i++ )
  {
   Serial.print("Student[");
   Serial.print(i);
   Serial.print("] = ");
   Serial.println(Student[i]);
  }
 }

void loop() { }
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-52] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown below.



[Figure 2-53] Output Result Value

Example 2-18) : "Array Example(2)" Program

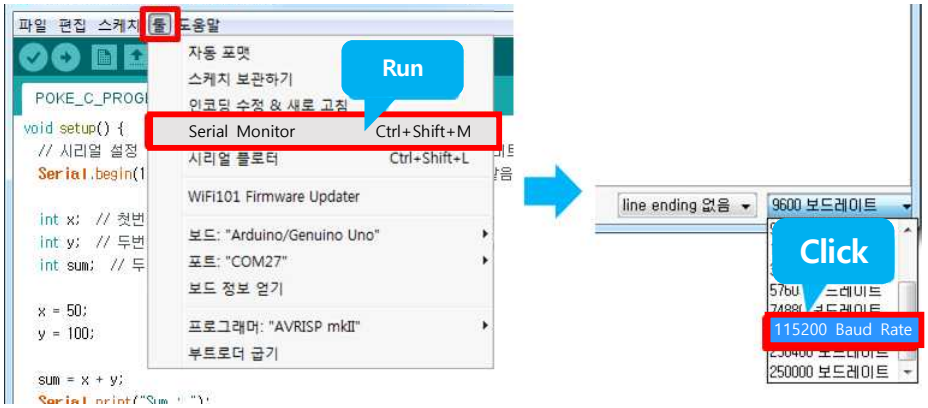
Let's write a program that calculates the average and sum using the values of array element.

ARDUINO_KIT_PROGRAM18.ino

```
#define SIZE  5 // Define the size of an array

void setup() {
```

```
  // Serial Setting : Baud Rate 115200, Data 8bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)

  //Wait for the serial port to be connected
  while(!Serial); // If serial communication is connected, Serial =1

  int i;

  // initialize the array
  int Student[SIZE] ={ 51, 61, 71, 81, 91 };
  int sum = 0;
  int Ave = 0;


  for( i= 0 ; i < SIZE ; i++ )
        sum = sum + Student[i];

  Ave = sum / SIZE;

 Serial.print("sum = ");
 Serial.println(sum);
 Serial.print("Ave = ");
 Serial.println(Ave);

 }

void loop() {
}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 2-54] Execute Serial Monitor and Set Baud Rate

The resulting value is output as shown.

[Figure 2-55] Output Result Value

Chapter 3

LED Control

Learning Objectives

1. Understand LED control method.
2. Understand how to control the brightness of RGB LED.

3. LED Control

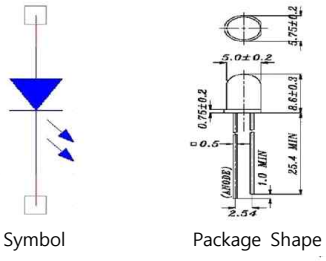
In this chapter, we will try the basic practice of turning on LED using HBE ARDUINO KIT board. We will send a signal to LED in order to make LED blink using Arduino Leonardo on the HBE ARDUINO KIT board. When the program starts, LED turns on and off every 0.5 seconds.

3.1. LED Structure and Driving Method

LED is an abbreviation for Light-emitting diode. This refers to a semiconductor device that emits light, and the color of the emitted light may be red, green, or yellow depending on the amount and type of crystal doping. Compared to general incandescent lamp, power consumption is only 1/5, response time is 1 million times faster, and duration is improved semi-permanently, so it is used for precision semiconductor equipment inspection device, electronic display panel such

as automobile dashboard, electronic display board, industrial equipment indicator, and various traffic safety signal.

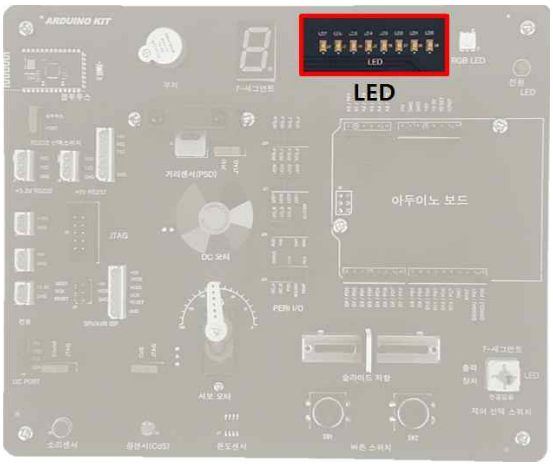
LED is a diode that makes a Pn junction, and that emits light by electron and hole recombining as current flows in the forward direction. The symbol of LED is as follows, and the electrical features are generally determined at a voltage drop of 1.5[V] to 2.5[V] at a current of 10 to 20[mA]. The shape of the LED is that the long leg is the Anode and the short leg is the Cathode.



[Figure 3-1] Symbol and Shape of LED

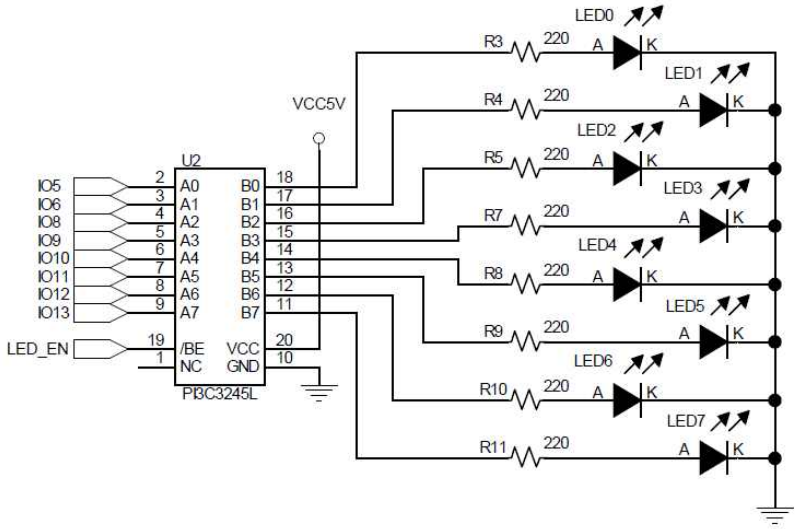
3.2. Location and Circuit of LED

The following shows the location of LED.



[Figure 3-2] Location of LED

The following shows the circuit diagram of LED module.



[Figure 3-3] Circuit of LED Module

In the circuit diagram of LED module, LED0 to LED7 are connected to the buffer chip PI3C3245L and connected to pins 5, 6, 8, 9, 10, 11, 12,

and 13 of the Arduino Leonardo. The buffer chip is controlled by the LED_EN signal and is connected when LED_EN signal is LOW and disconnected when LED_EN signal is HIGH.

3.3. LED Control Program Experiment

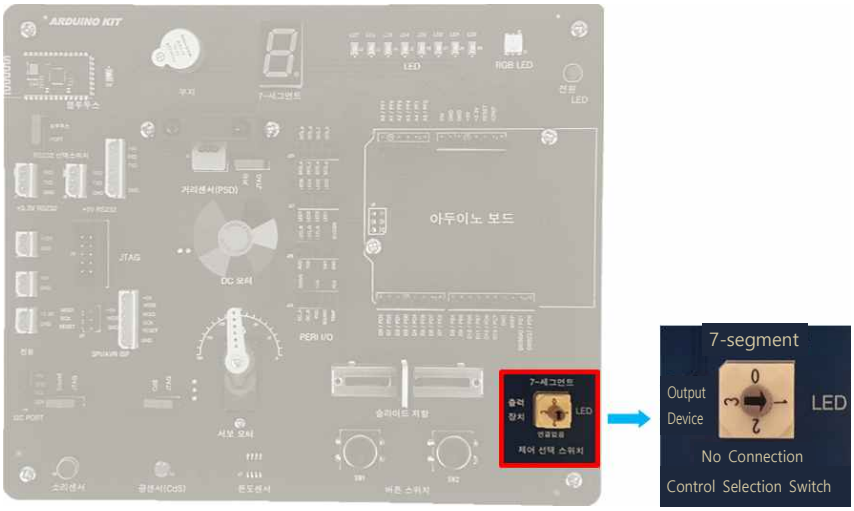
◉ LED on - Let's control LED

◆ Set the control selection switch of Arduino Kit board to LED and write a program that turns the LED ON (0.5 seconds)-OFF (0.5 seconds) at a cycle of 1 second.

● The following table shows the pin map where Arduino Leonardo and LED on ARDUINO KIT board are connected.

Arduino Pin Number	LED	Remarks
5	LED0	Control selection switch must be set to LED
6	LED1	
8	LED2	
9	LED3	
10	LED4	
11	LED5	
12	LED6	
13	LED7	

● For LED control, set the control selection switch on the ARDUINO KIT board to LED.



[Figure 3-4] Set Control Selection Switch to LED

ARDUINO_KIT_LED1.ino

```
// Save pin number of LED0 ~ 7
int pin_LED0 = 5;
int pin_LED1 = 6;
int pin_LED2 = 8;
int pin_LED3 = 9;
int pin_LED4 = 10;
int pin_LED5 = 11;
int pin_LED6 = 12;
int pin_LED7 = 13;

void setup() {
  // Set LED0 ~ 7 pins as output pins
  pinMode(pin_LED0, OUTPUT);
  pinMode(pin_LED1, OUTPUT);
  pinMode(pin_LED2, OUTPUT);
```

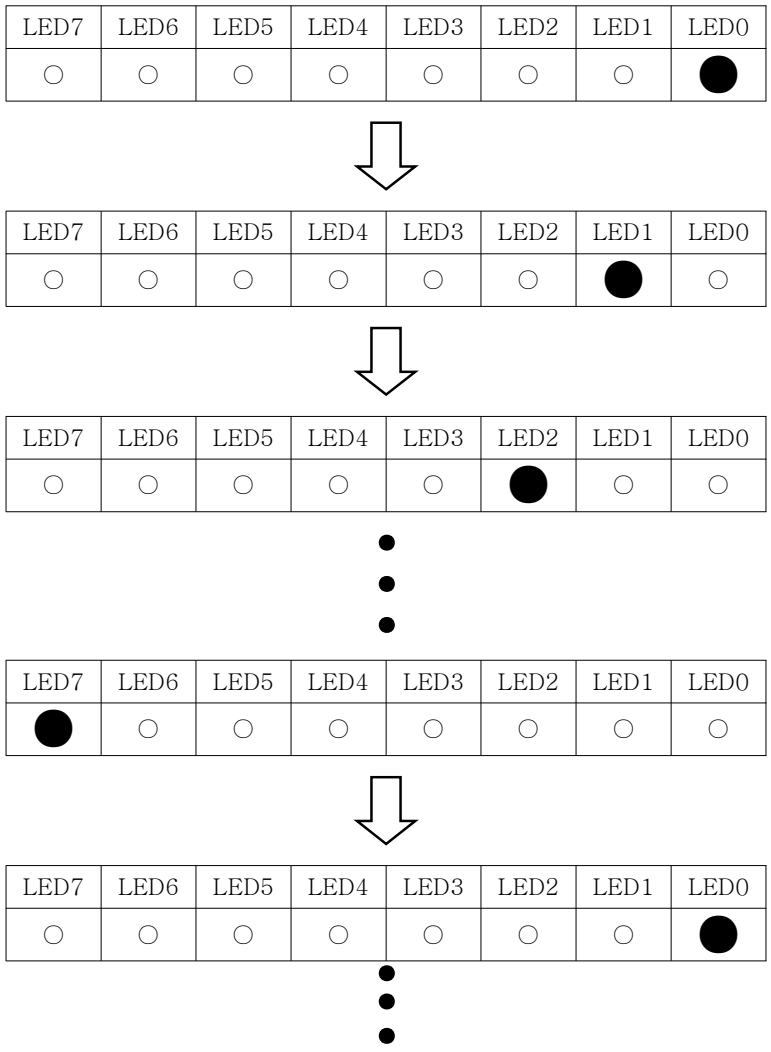
```
  pinMode(pin_LED3, OUTPUT);
  pinMode(pin_LED4, OUTPUT);
  pinMode(pin_LED5, OUTPUT);
  pinMode(pin_LED6, OUTPUT);
  pinMode(pin_LED7, OUTPUT);
}

void loop() {
  // Ouput HIGH(LED ON) to LED0 ~ 7pin
  digitalWrite(pin_LED0, 1);
  digitalWrite(pin_LED1, 1);
  digitalWrite(pin_LED2, 1);
  digitalWrite(pin_LED3, 1);
  digitalWrite(pin_LED4, 1);
  digitalWrite(pin_LED5, 1);
  digitalWrite(pin_LED6, 1);
  digitalWrite(pin_LED7, 1);

  delay(500);  // 0.5second Delay

  // Output LOW(LED OFF) to LED0 ~ 7pin
  digitalWrite(pin_LED0, 0);
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 0);
  digitalWrite(pin_LED6, 0);
  digitalWrite(pin_LED7, 0);
  delay(500);
}
```

EXPERIMENT 3-2

◉ Turn LED on

◆ Let's write a program that turns on one LED every 0.5 seconds.

● Control LED0 to LED7 to turn on one by one in the following order.



ARDUINO_KIT_LED2.ino

```
// Save LED0 ~ 7 pin number
int pin_LED0 = 5;
int pin_LED1 = 6;
int pin_LED2 = 8;
int pin_LED3 = 9;
int pin_LED4 = 10;
int pin_LED5 = 11;
int pin_LED6 = 12;
int pin_LED7 = 13;

void setup() {
  //Set LED0 ~ 7 pin as output pin
  pinMode(pin_LED0, OUTPUT);
  pinMode(pin_LED1, OUTPUT);
  pinMode(pin_LED2, OUTPUT);
  pinMode(pin_LED3, OUTPUT);
  pinMode(pin_LED4, OUTPUT);
  pinMode(pin_LED5, OUTPUT);
  pinMode(pin_LED6, OUTPUT);
  pinMode(pin_LED7, OUTPUT);
}

void loop() {
  // Only LED0 pin outputs HIGH(LED0 ON)
  digitalWrite(pin_LED0, 1);
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 0);
  digitalWrite(pin_LED6, 0);
  digitalWrite(pin_LED7, 0);
  delay(500);

  // Only LED1 pin outputs HIGHLED1 ON)
```

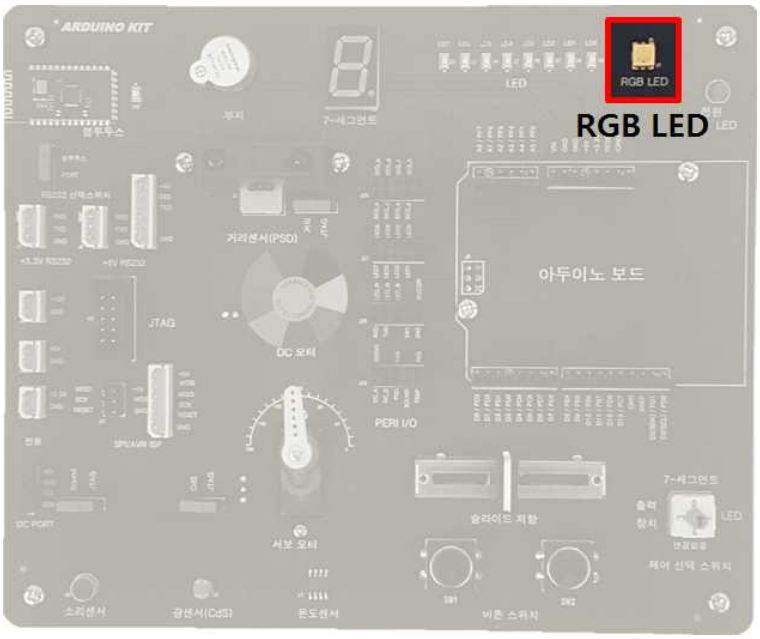
```
digitalWrite(pin_LED0, 0);
digitalWrite(pin_LED1, 1);
digitalWrite(pin_LED2, 0);
digitalWrite(pin_LED3, 0);
digitalWrite(pin_LED4, 0);
digitalWrite(pin_LED5, 0);
digitalWrite(pin_LED6, 0);
digitalWrite(pin_LED7, 0);
delay(500);

// Only LED2 pin outputs HIGH(LED2 ON)
digitalWrite(pin_LED0, 0);
digitalWrite(pin_LED1, 0);
digitalWrite(pin_LED2, 1);
digitalWrite(pin_LED3, 0);
digitalWrite(pin_LED4, 0);
digitalWrite(pin_LED5, 0);
digitalWrite(pin_LED6, 0);
digitalWrite(pin_LED7, 0);
delay(500);

// Only LED3 pin outputs HIGH(LED3 ON)
digitalWrite(pin_LED0, 0);
digitalWrite(pin_LED1, 0);
digitalWrite(pin_LED2, 0);
digitalWrite(pin_LED3, 1);
digitalWrite(pin_LED4, 0);
digitalWrite(pin_LED5, 0);
digitalWrite(pin_LED6, 0);
digitalWrite(pin_LED7, 0);
delay(500);

// Only LED4 pin outputs HIGH(LED4 ON)
digitalWrite(pin_LED0, 0);
digitalWrite(pin_LED1, 0);
digitalWrite(pin_LED2, 0);
```

```
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 1);
  digitalWrite(pin_LED5, 0);
  digitalWrite(pin_LED6, 0);
  digitalWrite(pin_LED7, 0);
  delay(500);

  // Only LED5 pin outputs HIGH(LED5 ON)
  digitalWrite(pin_LED0, 0);
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 1);
  digitalWrite(pin_LED6, 0);
  digitalWrite(pin_LED7, 0);
  delay(500);

  // Only LED6 pin outputs HIGH(LED6 ON)
  digitalWrite(pin_LED0, 0);
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 0);
  digitalWrite(pin_LED6, 1);
  digitalWrite(pin_LED7, 0);
  delay(500);

  // Only LED7 pin outputs HIGH(LED7 ON)
  digitalWrite(pin_LED0, 0);
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 0);
```

```
    digitalWrite(pin_LED6, 0);
    digitalWrite(pin_LED7, 1);
    delay(500);
}
```

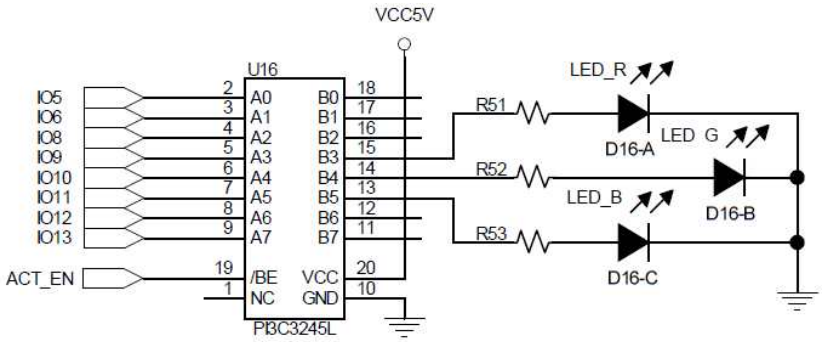
3.4. Location and Circuit of RGB LED

The following shows the location of RGB LED.



[Figure 3-5] RGB LED Location

The following shows the circuit diagram of RGB LED module.



[Figure 3-6] Circuit of RGB LED Module

Like the circuit diagram of LED module, in RGB LED module circuit diagram, LED_R, LED_G, LED_B are connected to the buffer chip PI3C3245L and connected to pins 9, 10, and 11 of the Arduino Leonardo. The buffer chip is controlled by ACT_EN signal and is connected when ACT_EN signal is LOW and disconnected when ACT_EN is HIGH.

3.5. RGB LED Control Program Experiment

◉ LED Brightness Control - Let's control the brightness of LED with PWM signal.

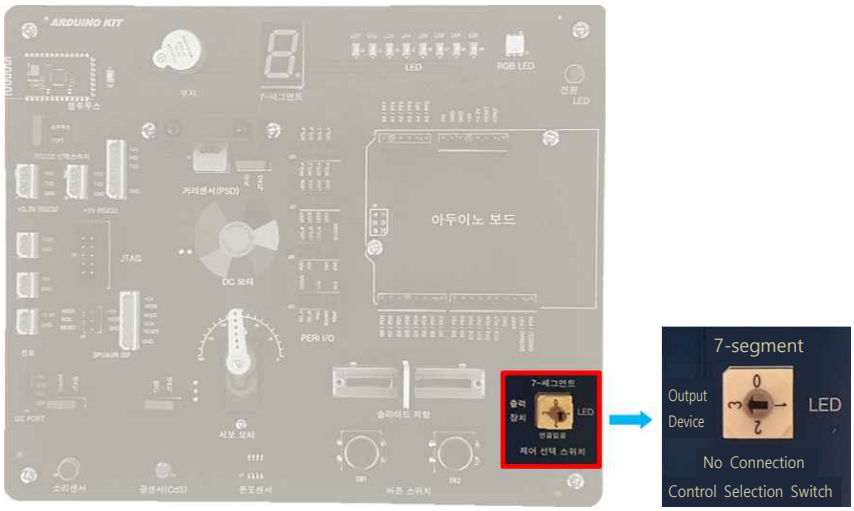
EXPERIMENT 3-3

◆ **Set the control selection switch of ARDUINO KIT board to the output device and write a program that infinitely repeats the operation of turning LED on and getting brighter while RGB LED is off.**

● The following table shows the pin map where Arduino Leonardo and RGB LED on the ARDUINO KIT board are connected.

Arduino Pin Number	LED	Remarks
9	LED_R	Control selection switch should be set to output device
10	LED_G	
11	LED_B	

For RGB LED control, set the control selection switch on the ARDUINO KIT board to the output device.



[Figure 3-7] Set Control Selection Switch to Output Device

ARDUINO_KIT_LED3.ino

```
// Save pin number of LED_R, LED_G, LED_B
int pin_LED_R = 9;
int pin_LED_G = 10;
int pin_LED_B = 11;

void setup() {
  // Set LED_R pin as output pin
  pinMode(pin_LED_R, OUTPUT);
  // Set LED_G pin as output pin
  pinMode(pin_LED_G, OUTPUT);
  // Set LED_B pin as output pin
  pinMode(pin_LED_B, OUTPUT);
}

void loop() {
  int brightness;
  // Red LED gets bright.
```

```
  for(brightness=0; brightness<256; brightness++)
  {
    analogWrite(pin_LED_R, brightness);
    delay(20);
  }
  // Red LED off.
  digitalWrite(pin_LED_R, 0);
  // Green LED gets bright.
  for(brightness=0; brightness<256; brightness++)
  {
    analogWrite(pin_LED_G,brightness);
    delay(20);
  }
  // Green LED off
  digitalWrite(pin_LED_G, 0);
  // Blue LED gets bright.
  for(brightness=0; brightness<256; brightness++)
  {
    analogWrite(pin_LED_B, brightness);
    delay(20);
  }
  // Blue LED off
  digitalWrite(pin_LED_B, 0);
}
```

3.6. Practice

3-1. Let's write a program that repeats ON-OFF of the entire LED infinitely at interval of 1 second.

3-2. Let's write a program that repeats the LED infinitely at 0.7 second interval as shown below.

LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
○	○	○	○	○	○	●	●

⇨

LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
○	○	○	○	●	●	○	○

⇨

LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
○	○	●	●	○	○	○	○

⇨

LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
●	●	○	○	○	○	○	○

⇨

LED 7	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1	LED 0
○	○	○	○	○	○	●	●

⇨ . . .

3-3. Let's write a program that makes RBG LED infinitely repeats the operation of getting darker from the brightest at the same time.

Chapter 4

Buzzer Control

Learning Objectives

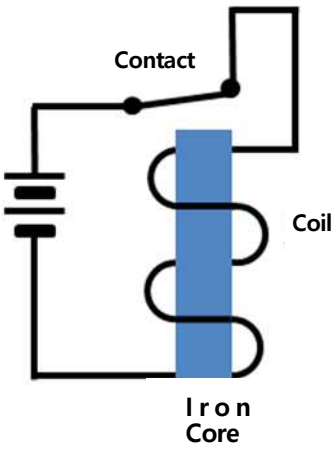
1. Understand the principle of buzzer sound.

4. Buzzer Control

In this chapter, we will practice generating sound with buzzer using the HBE ARDUINO KIT board. The buzzer will be turned ON/OFF by sending a signal to the buzzer using the Arduino Leonardo on HBE ARDUINO KIT board. When the program starts, the buzzer turns on and off indefinitely every second.

4.1. Structure of Buzzer

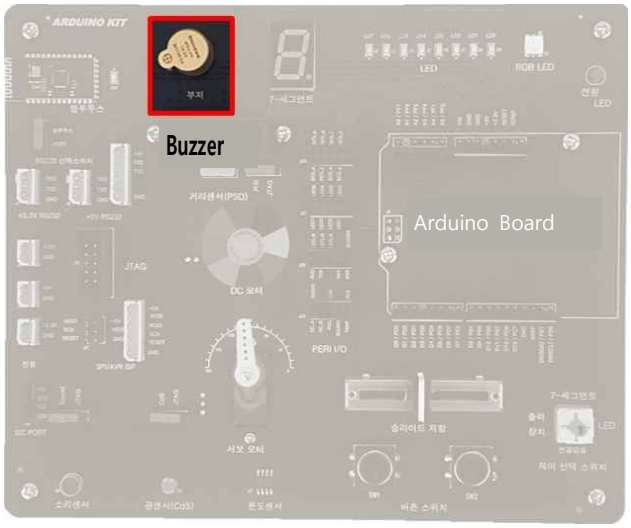
Buzzer is a module that generates sound when a signal is given. This module is used to make the user aware of a specific situation. The buzzer contains an iron core with a coil wound inside, so when a user gives an electrical signal, like a relay module, current flows and becomes magnetized causing the contact to be disconnected. However, if the contact is disconnected, the current stops flowing and the contact is reconnected. This is repeated rapidly to produce a sound.



[Figure 4-1] Inside Buzzer

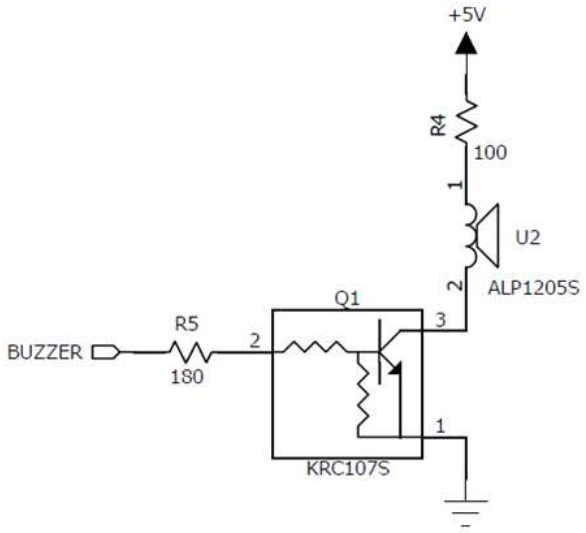
4.2. Location and Circuit of Buzzer

The following shows the location of Buzzer.



[Figure 4-2] Location of Buzzer

To operate the buzzer, Q1(TR) is controlled. Therefore, when HIGH is input to BUZZER, Q1 turns ON and the buzzer generates a sound, and when LOW is input, Q1 turns OFF and the buzzer does not generate a sound.



[Figure 4-3] Circuit of Buzzer Module

4.3. Buzzer Control Program Experiment

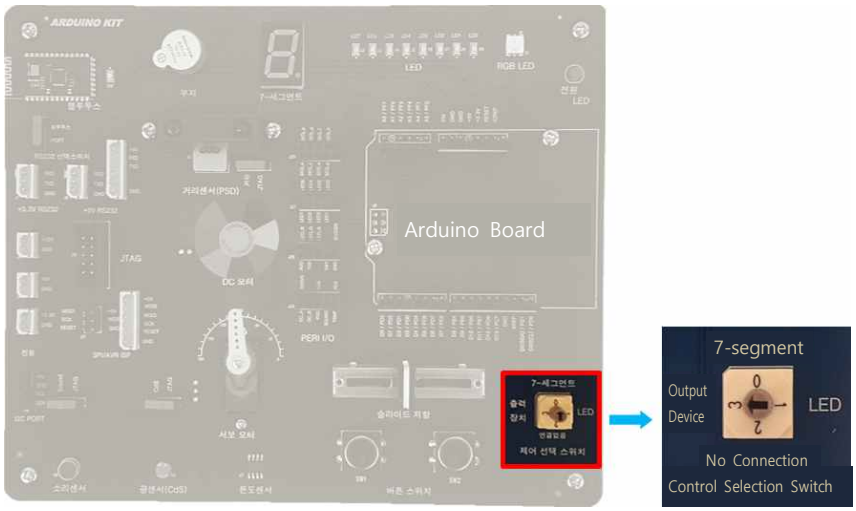
◉ To make a buzzer sound - Let's try to control the buzzer.

◆ Set control selection switch of the ARDUINO KIT board to output device and write a program that turns the buzzer ON (1second)-OFF (1 second) every 2 seconds.

● The following table shows the pin map where the Arduino Leonardo and buzzer on the ARDUINO KIT board are connected.

Arduino Pin Number	Buzzer	Remarks
12	BUZZER	Control selection switch should be set to the output device

For buzzer control, set the control selection switch on the ARDUINO KIT board to the output device.



[Figure 4-4] Set Control Selection Switch to Output Device

ARDUINO_KIT_BUZZER.ino

```
// Save pin number of Buzzer
int pin_BUZZER = 13;

void setup() {
  // Set Buzzer pin as output pin
  pinMode(pin_BUZZER, OUTPUT);
}

void loop() {
  // Output HIGH to Buzzer pin(Buzzer Sound ON)
  digitalWrite(pin_BUZZER, 1);
  delay(1000);

  // Output LOW to Buzzer pin(Buzzer Sound OFF)
  digitalWrite(pin_BUZZER, 0);
  delay(1000);
}
```

4.4. Practice

4-1. Let's write a program that repeats the buzzer ON in 1.2 seconds and OFF in 0.8 seconds indefinitely.

4-2. Let's write a program that repeats the buzzer ON or OFF in 1 second, 2.5 second, 0.7 second, 1.2 second, 3 second indefinitely.
ON(1second) -> OFF(1second) -> ON(2.5second) -> OFF(2.5second) -> ON(0.7second) -> OFF(0.7second) -> ON(1.2second) -> OFF(1.2second) -> ON(3second) -> OFF(3second)

Chapter 5

7-Segment Control

Learning Objectives

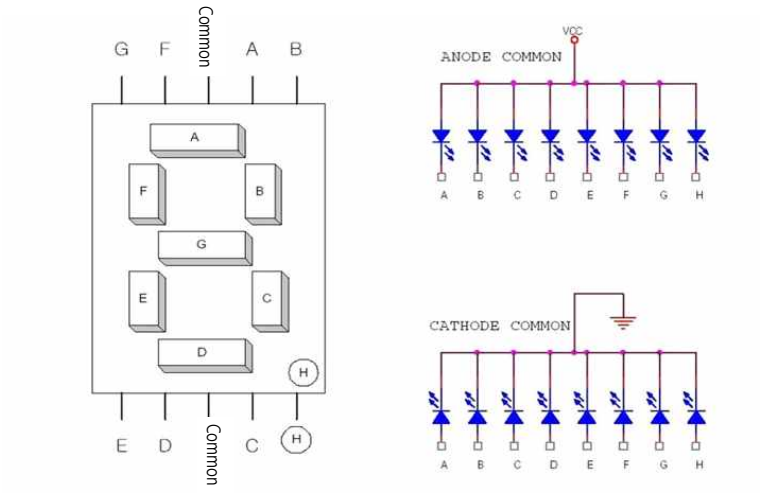
1. Understand types of 7-segment.
2. Understand how to control 7-segment.

5. 7-Segment Control

In this chapter, we will try to write a program to drive 7-segment using HBE ARDUINO KIT board.

5.1. Structure of 7-Segment

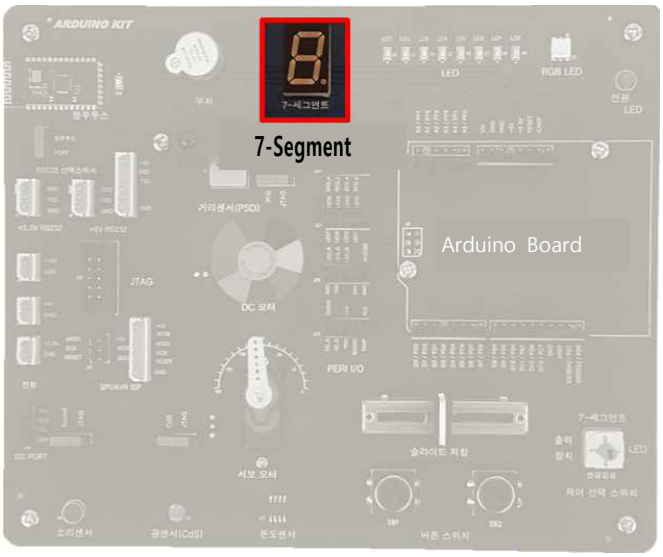
7-segment is an array of 8 LEDs as shown in [Figure 5-1], and is often used for expressing number or simple symbol. 7-segment is classified into Common Anode (+common) and Common Cathode (-common) depending on the power applied to the common terminal. That is, it is classified with Anode(+) common type where the corresponding LED lights up when VCC (+5V) is connected to the common terminal and 0V is applied to the input terminal and Cathode (-) common type where the corresponding LED lights up when ground (0V) is connected to the common terminal and +5V is applied to the input terminal. Common Cathode (-common) circuit is used in this equipment.



[Figure 5-1] Structure of 7-Segment LED

5.2. Location of 7-Segment

The following shows the location of 7-Segment.



[Figure 5-2] Location of 7-Segment

5.3. 7-Segment Control Program Experiment

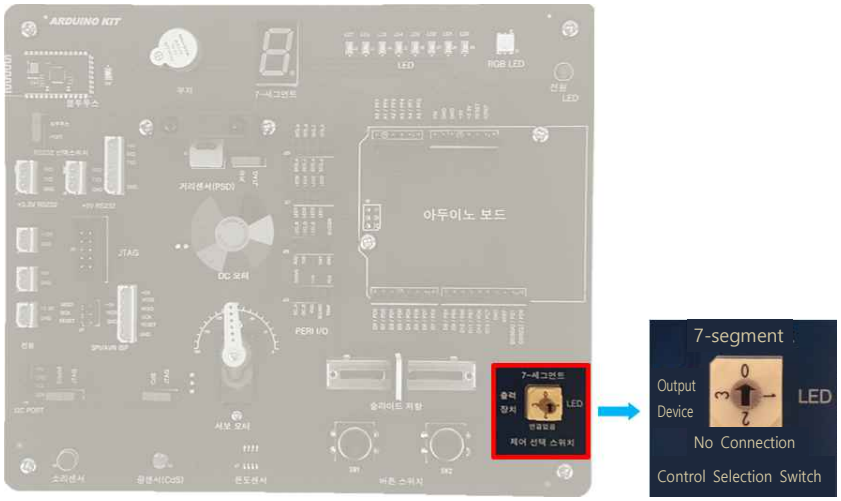
◉ Operate 7-Segment

◆ Set the control selection switch on the ARDUINO KIT board to the 7-segment and let's write a program that turns on LED one by one to find out which bit controls each LED of the 7-segment.

● The following table shows the pin map where the Arduino Leonardo and 7-segment on the ARDUINO KIT board are connected.

Arduino Pin Number	7-Segment	Remarks
5	A	Control selection switch must be set to 7-segment
6	B	
8	C	
9	D	
10	E	
11	F	
12	G	
13	H	

For 7-segment control, set the control selection switch of the ARDUINO KIT board to 7-segment.



[Figure 5-3] Set Control Selection Switch to 7-segment

ARDUINO_KIT_FND1.ino

```
// Save A ~ H pin numbers
int pin_A = 5;
int pin_B = 6;
int pin_C = 8;
int pin_D = 9;
int pin_E = 10;
int pin_F = 11;
int pin_G = 12;
int pin_H = 13;

void setup() {
  // Set A ~ H pins to output pins
  pinMode(pin_A, OUTPUT);
  pinMode(pin_B, OUTPUT);
  pinMode(pin_C, OUTPUT);
```

```
  pinMode(pin_D, OUTPUT);
  pinMode(pin_E, OUTPUT);
  pinMode(pin_F, OUTPUT);
  pinMode(pin_G, OUTPUT);
  pinMode(pin_H, OUTPUT);
}

void loop() {
  // Only A pin outputs HIGH(A ON)
  digitalWrite(pin_A, 1);
  digitalWrite(pin_B, 0);
  digitalWrite(pin_C, 0);
  digitalWrite(pin_D, 0);
  digitalWrite(pin_E, 0);
  digitalWrite(pin_F, 0);
  digitalWrite(pin_G, 0);
  digitalWrite(pin_H, 0);
  delay(500);

  // Only B pin outputs HIGH(B ON)
  digitalWrite(pin_A, 0);
  digitalWrite(pin_B, 1);
  digitalWrite(pin_C, 0);
  digitalWrite(pin_D, 0);
  digitalWrite(pin_E, 0);
  digitalWrite(pin_F, 0);
  digitalWrite(pin_G, 0);
  digitalWrite(pin_H, 0);
  delay(500);

  // Only C pin outputs HIGH(C ON)
  digitalWrite(pin_A, 0);
  digitalWrite(pin_B, 0);
  digitalWrite(pin_C, 1);
  digitalWrite(pin_D, 0);
  digitalWrite(pin_E, 0);
```

```
digitalWrite(pin_F, 0);
digitalWrite(pin_G, 0);
digitalWrite(pin_H, 0);
delay(500);

// Only D pin outputs HIGH(D ON)
digitalWrite(pin_A, 0);
digitalWrite(pin_B, 0);
digitalWrite(pin_C, 0);
digitalWrite(pin_D, 1);
digitalWrite(pin_E, 0);
digitalWrite(pin_F, 0);
digitalWrite(pin_G, 0);
digitalWrite(pin_H, 0);
delay(500);

// Only E pin outputs HIGH(E ON)
digitalWrite(pin_A, 0);
digitalWrite(pin_B, 0);
digitalWrite(pin_C, 0);
digitalWrite(pin_D, 0);
digitalWrite(pin_E, 1);
digitalWrite(pin_F, 0);
digitalWrite(pin_G, 0);
digitalWrite(pin_H, 0);
delay(500);

// Only F pin outputs HIGH(F ON)
digitalWrite(pin_A, 0);
digitalWrite(pin_B, 0);
digitalWrite(pin_C, 0);
digitalWrite(pin_D, 0);
digitalWrite(pin_E, 0);
digitalWrite(pin_F, 1);
digitalWrite(pin_G, 0);
digitalWrite(pin_H, 0);
```

```
  delay(500);

  // Only G pin outputs HIGH(G ON)
  digitalWrite(pin_A, 0);
  digitalWrite(pin_B, 0);
  digitalWrite(pin_C, 0);
  digitalWrite(pin_D, 0);
  digitalWrite(pin_E, 0);
  digitalWrite(pin_F, 0);
  digitalWrite(pin_G, 1);
  digitalWrite(pin_H, 0);
  delay(500);

  // Only H pin outputs HIGH(H ON)
  digitalWrite(pin_A, 0);
  digitalWrite(pin_B, 0);
  digitalWrite(pin_C, 0);
  digitalWrite(pin_D, 0);
  digitalWrite(pin_E, 0);
  digitalWrite(pin_F, 0);
  digitalWrite(pin_G, 0);
  digitalWrite(pin_H, 1);
  delay(500);
}
```

Now, we can create the following 7-segment font combination through [Figure 5-1] and Experiment 5-1.

7-Display Number	Bit Value							
	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	1	1
1	0	0	0	0	0	1	1	0
2	0	1	0	1	1	0	1	1
3	0	1	0	0	1	1	1	1
4	0	1	1	0	0	1	1	0
5	0	1	1	0	1	1	0	1
6	0	1	1	1	1	1	0	0
7	0	0	0	0	0	1	1	1
8	0	1	1	1	0	1	1	1
9	0	1	1	0	1	1	1	1
A	0	1	1	7	0	1	1	1
b	0	1	1	1	1	1	0	0
C	0	0	1	1	1	0	0	1
d	0	1	0	1	1	1	1	0
E	0	1	1	1	1	0	0	1
F	0	1	1	1	0	0	0	1
_	0	0	0	0	1	0	0	0
.	1	0	0	0	0	0	0	0

EXPERIMENT 5-2

◉ Operate 7-segment.

◆ Let's write a program where 0→ 1→ 2→ ⋯ → 9→ 0 → ⋯ repeats infinitely every 0.5 second using 7-segment.

● Control to output 0 to 9 on 7-segment.

ARDUINO_KIT_FND2.ino

```
// Save the pin number of A, B, C, D, E, F, G, H in an array
int pin_FND[8] = {5, 6, 8, 9, 10, 11, 12, 13};

void setup() {
  unsigned char  i;
  // Set A ~ H pins as output pins
  for(i=0; i<8; i++)
  {
    pinMode(pin_FND[i], OUTPUT);
  }
}

void loop() {
  // Output 0 to 7-segment.
  FND_out(0);
  delay(500);
  // Output 1 to 7-segment.
  FND_out(1);
  delay(500);
  // Output 2 to 7-segment.
  FND_out(2);
  delay(500);
  // Output 3 to 7-segment.
  FND_out(3);
  delay(500);
  // Output 4 to 7-segment.
  FND_out(4);
```

```
  delay(500);
  // Output 5 to 7-segment.
  FND_out(5);
  delay(500);
  // Output 6 to 7-segment.
  FND_out(6);
  delay(500);
  // Output 7 to 7-segment.
  FND_out(7);
  delay(500);
  // Output 8 to 7-segment.
  FND_out(8);
  delay(500);
  // Output 9 to 7-segment.
  FND_out(9);
  delay(500);
}

void FND_out(unsigned char data)
{
  switch (data) {
    case 0 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;


    case 1 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
```

```
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],0);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],0);
    break ;

  case 2 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],1);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],0);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],1);
    break ;

  case 3 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],1);
    break ;

  case 4 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],0);
```

```
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],0);
    break ;

  case 5 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],0);
    digitalWrite(pin_FND[0],1);
    break ;

  case 6 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],1);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],0);
    digitalWrite(pin_FND[0],1);
    break ;

  case 7 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],0);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],0);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],1);
```

```
      break ;

    case 8 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    case 9 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],0);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    default:
      break;
  }
}
```

5.4. Practice

5-1. Let's modify the function that turns on FND_out(0) ~ FND_out(9) in Experiment 5-2 so that it operates with for statement.

5-2. Based on the practice 5-1 above, let's write a program that infinitely repeats the display from 0 to F at interval of 1.2 seconds on 7-segment.

Chapter 6

Switch Module

Learning Objectives

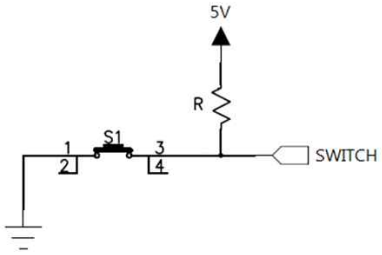
1. Understand the concept of switch input.

6. Switch Module

In this chapter, we will use the switch button of HBE ARDUINO KIT board to write a program that operates LED and 7-segment.

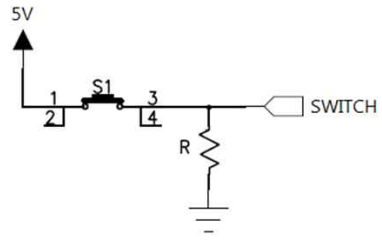
6.1. Operation Method by Switch Circuit

Switch connects a circuit by pressing a button. There are many types of switches, but here we will study how they operate by the button switch circuit. As shown in [Figure 6-1], in pull-up resistor circuit, HIGH (5V) is normally output to SWITCH signal, and LOW (0V) is output when the switch is pressed.



[Figure 6-1] Pull-up Resistor Circuit

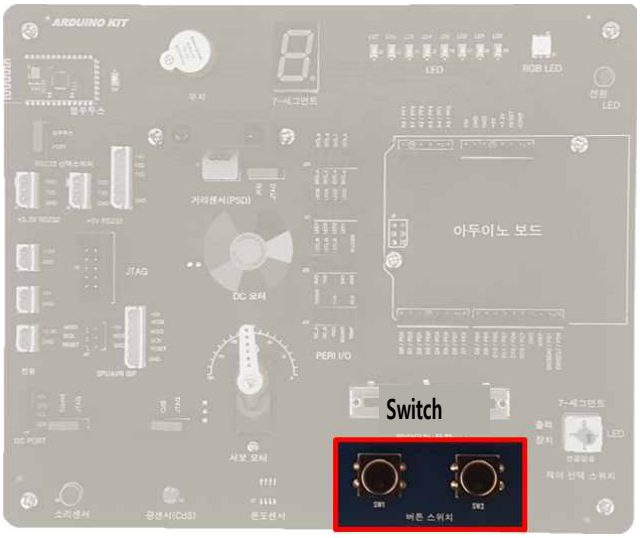
As shown in [Figure 6-2], in pull-down resistor circuit, LOW (0V) is n ormally output and HIGH (5V) is output when the switch is pressed.



[Figure 6-2] Pull-down Resistor Circuit

We will use a Pull-down resistor circuit and HIGH is output when the switch is pressed.

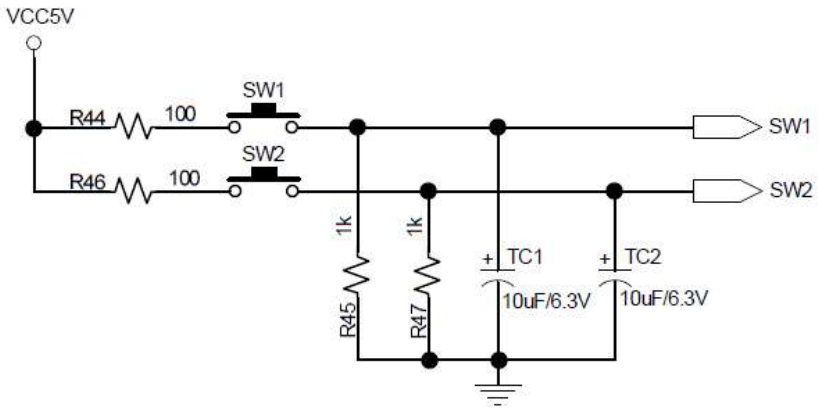
6.2. Location and Circuit of Switch



[Figure 6-3] Switch Location

The following shows the location of switch.

The following figure shows the switch circuit. LOW is output when the switch is not pressed, and HIGH is output when the switch is pressed.



[Figure 6-4] Switch Circuit

6.3. Switch Program Experiment

EXPERIMENT 6-1

◉ Turn on LED corresponding to the switch.

◆ **Let's write a program that turns on a corresponding LED when a switch is pressed.**

Example) When switch 1 is pressed -> LED 0 is ON

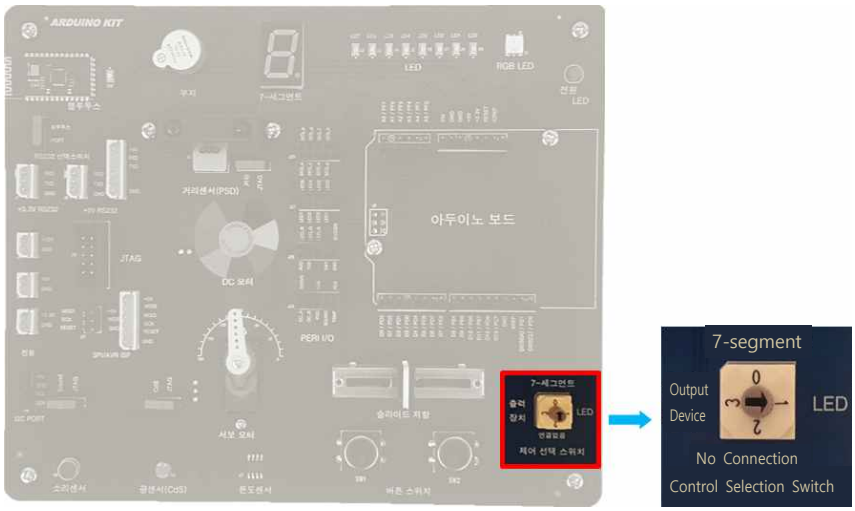
When switch 2 is pressed -> LED 1 is ON

● The following table shows the pin map where Arduino Leonardo, switch and LED of the ARDUINO KIT board are connected.

Arduino Pin Number	Switch	Remarks
3	SW1	
4	SW2	

Arduino Pin Number	LED	Remarks
5	LED0	Control selection switch should be set to LED
6	LED1	

For switch and LED control, set the control selection switch on ARDUINO KIT board to LED.



[Figure 6-5] Set Control Selection Switch to LED

ARDUINO_KIT_SW_LED.ino

```
// Save LED0, 1 pin numbers
int pin_LED0 = 5;
int pin_LED1 = 6;

// Save SW1, SW2 pin numbers
int pin_SW1 = 3;
int pin_SW2 = 4;

void setup() {
  // Set LED0, 1 pins as output pins
  pinMode(pin_LED0, OUTPUT);
  pinMode(pin_LED1, OUTPUT);

  // Set SW1, 2 pins as input pins
  pinMode(pin_SW1, INPUT);
  pinMode(pin_SW2, INPUT);
}
```

```
void loop() {
  int sw1=0, sw2=0 ;

  // Read the value of Switch 1
  sw1 = digitalRead(pin_SW1) ;
  if(sw1==1 )
    digitalWrite(pin_LED0, 1);
  else
    digitalWrite(pin_LED0, 0);

  // Read the value of Switch 2
  sw2 = digitalRead(pin_SW2) ;
  if(sw2==1 )
    digitalWrite(pin_LED1, 1);
  else
    digitalWrite(pin_LED1, 0);
}
```

◉ Display pressed switch number to 7-segment.

EXPERIMENT 6-2

◆ As Experiment 6-1, when a switch is pressed, write a program in which the corresponding switch number is displayed to 7-segment.

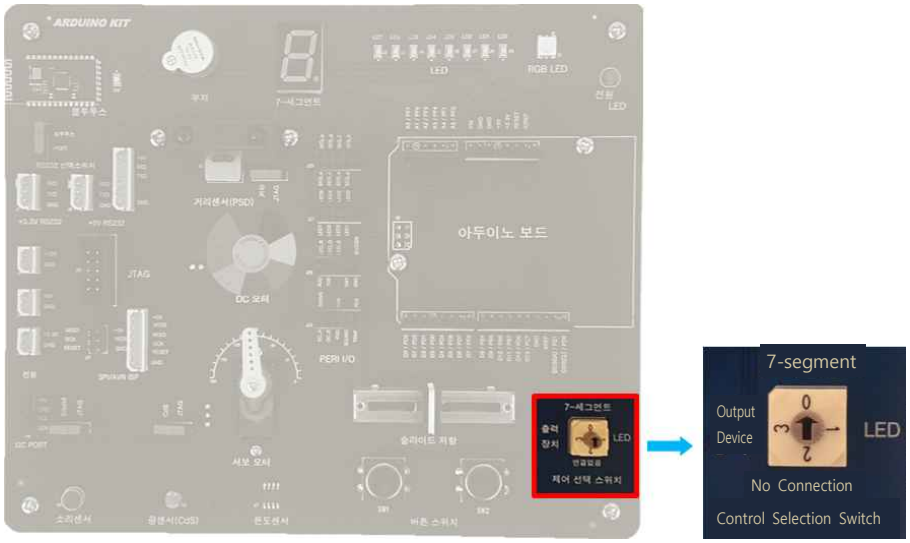
(For example, write a program so that '1' is displayed to 7-segment when switch 1 is pressed, and '2' is displayed when switch 2 is pressed)

● The table below shows the pin map where the Arduino Leonardo, switch and 7-segment of ARDUINO KIT board are connected.

Arduino Pin Number	Switch	Remarks
3	SW1	
4	SW2	

Arduino Pin Number	7-Segment	Remarks
5	A	Control selection switch must be set to 7-segment
6	B	
8	C	
9	D	
10	E	
11	F	
12	G	
13	H	

For switch and 7-segment control, set the control selection switch of th e ARDUINO KIT board to 7-segment.



[Figure 6-6] Set Control Selection Switch to 7-segment

ARDUINO_KIT_SW_FND.ino

```
// Save the pin numbers of A, B, C, D, E, F, G, H in an array
int pin_FND[8] = {5, 6, 8, 9, 10, 11, 12, 13};

// Save pin numbers of SW1, SW2
int pin_SW1 = 3;
int pin_SW2 = 4;

// Allocates the 7-segment font with an array

void setup() {
  unsigned char i;
  // Set A ~ H pins as output pins
  for(i=0; i<8; i++)
```

```
  {
    pinMode(pin_FND[i], OUTPUT);
  }
  // Set SW1, 2 pins as input pins
  pinMode(pin_SW1, INPUT);
  pinMode(pin_SW2, INPUT);
}

void loop() {
  // If there is no switch input, '0' is displayed to the segment
  // If switch '1' is pressed, '1' is displayed to the segment
  // If switch '2' is pressed, '2' is displayed tothe segment
  char sw1 = 0, sw2 = 0;

  sw1 = digitalRead(pin_SW1);
  sw2 = digitalRead(pin_SW2);

  if (sw1==1) FND_out(1);  // 1 is output when SW1 is pressed
  else if(sw2==1) FND_out(2);  // 2 is output when SW2 is pressed
  else FND_out(0);  // 0 is output when the switch is not pressed
}

void FND_out(unsigned char data)
{
  switch (data) {
    case 0 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;
```

```
    case 1 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],0);
      digitalWrite(pin_FND[4],0);
      digitalWrite(pin_FND[3],0);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],0);
      break ;

    case 2 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
      digitalWrite(pin_FND[5],0);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],0);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    default :
      break;
  }
}
```

6.4. Practice

6-1. In Experiment 6-1, write a program that turns on LED 0 to 3 when switch 1 is pressed and hat turns on LED 4 to 7 when switch 2 is pressed.

6-2. In Experiment 6-2, write a program so that '3' is output to the 7-segment when switch 1 and 2 are pressed at the same time.

Chapter 7

DC Motor Control

Learning Objectives

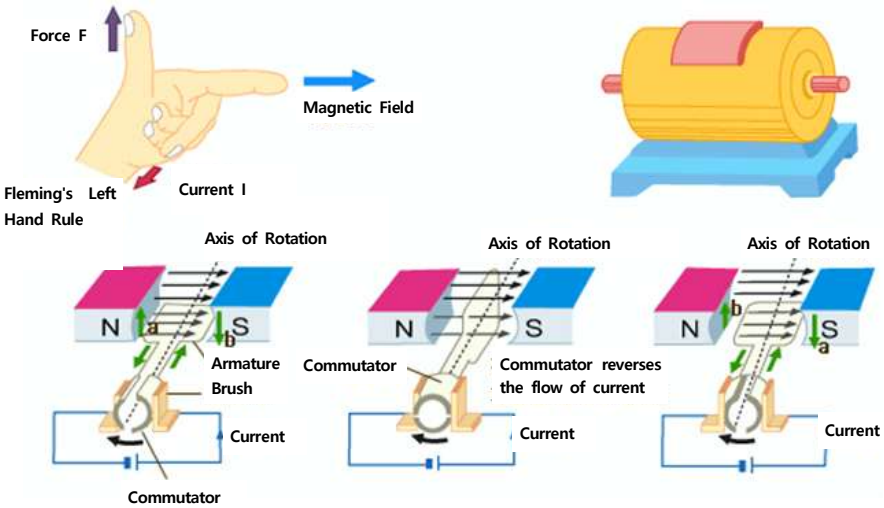
1. Understand the principle of DC motor.
2. Understand how to control DC motor.

7. DC Motor Control

In this chapter, we will try to write a program to control DC motor of HBE ARDUINO KIT board.

7.1. Principle of DC Motor Operation

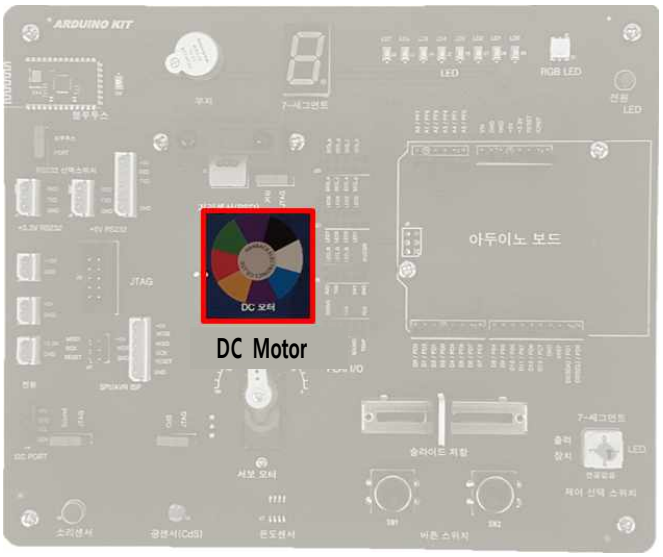
DC Motor is a machine that gets rotational power using electric power. Connect a load such as a wheel to the rotational axis of the motor to make rotation. Currently, DC motors have become a key component in most household appliances such as washer, fan, mixer having moving parts. Using the principle that an electromagnetic force is generated in a direction perpendicular to the direction of the magnetic field, DC motor creates a magnetic field by placing a magnet inside, and when an electric current flows through a wire connected to an axis, an electromagnetic force is generated and rotates. The direction of the electromagnetic force generated at this time is the direction by the Fleming's Left Hand Rule.



[Figure 7-1] Fleming's Left Hand Rule

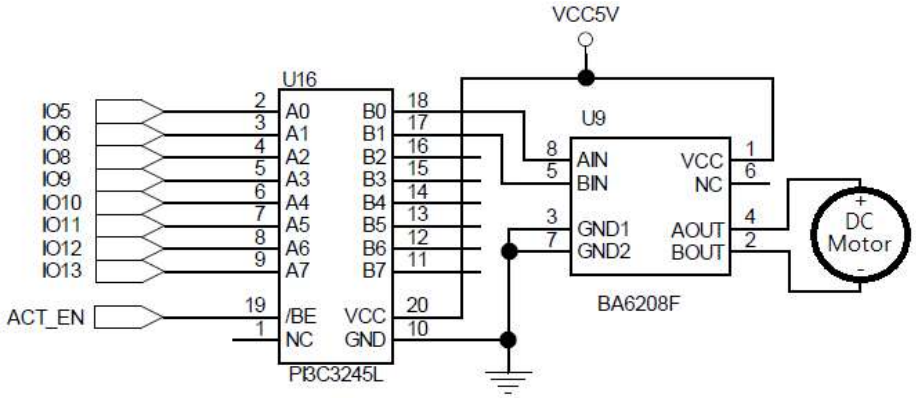
7.2. Location and Circuit of DC Motor

The following shows the location of the DC motor.



[Figure 7-2] Location of DC Motor

The following shows the circuit of the DC motor.



[Figure 7-3] Circuit of DC motor

DC motor is connected to motor IC (BA6208) and operated by the control signals of AIN and BIN pins. AIN/BIN controls the rotation direction of the motor. If PWM signal is input into a signal that inputs HIGH except stop, motor speed can be controlled.

Input		Output		
AIN	BIN	AOUT	BOUT	Operation
L	L	Open	Open	Stop
H	H	L	L	Stop
L	H(PWM)	L	H	Backward rotation
H(PWM)	L	H	L	Forward rotation

7.3. DC Motor Control Program Experiment

EXPERIMENT 7-1

◉ Control DC Motor.

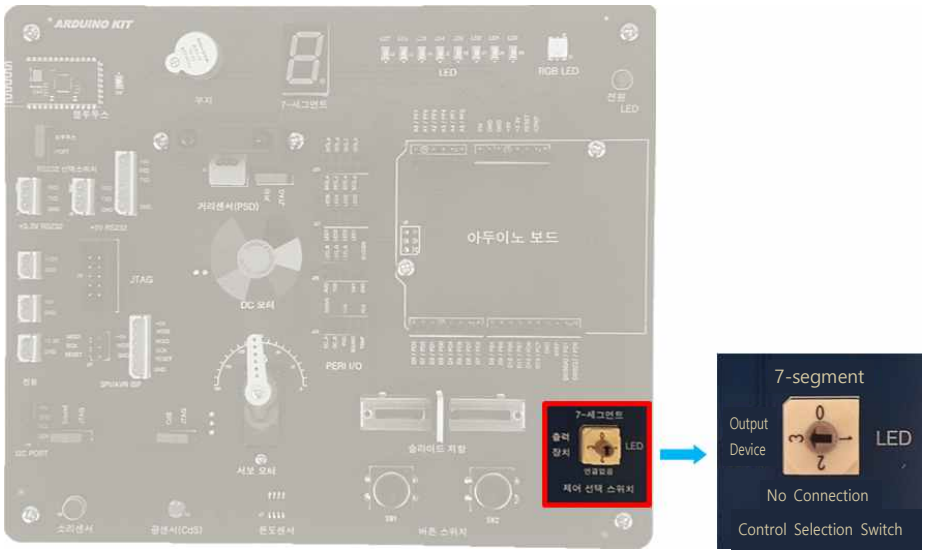
◆ Control the DC motor to write a program in which the motor repeats forward and reverse rotation indefinitely.

(Forward rotation 3 seconds → Stop 3 seconds → Reverse rotation 3 seconds → Stop 3 seconds → Forward rotation 3 seconds → ……)

● The following table shows the pin map where the Arduino Leonardo and DC motor on the ARDUINO KIT board are connected.

Arduino Pin Number	DC Motor	Remarks
5	AIN	Control selection switch should be set to the output device
6	BIN	

To control the DC motor, set the control selection switch on the ARDUI NO KIT board to the output device.



[Figure 7-4] Set Control Selection Switch to Output Device

ARDUINO_KIT_DC_MOTOR1.ino

```
//  Save AIN, BIN pin number of DC motor
int pin_DC_A = 5;
int pin_DC_B = 6;

void setup() {

  // Set DC motor AIN, BIN pins as output pins
  pinMode(pin_DC_A, OUTPUT);
  pinMode(pin_DC_B, OUTPUT);
}

void loop() {
  // The motor rotates clockwise (forward rotation)
  digitalWrite(pin_DC_A, 1);
  digitalWrite(pin_DC_B, 0);
  delay(3000);
```

```
  // The motor stops
  digitalWrite(pin_DC_A, 0);
  digitalWrite(pin_DC_B, 0);
  delay(3000);

  // The motor rotates counterclockwise (reverse rotation)
  digitalWrite(pin_DC_A, 0);
  digitalWrite(pin_DC_B, 1);
  delay(3000);

  // The motor stops
  digitalWrite(pin_DC_A, 0);
  digitalWrite(pin_DC_B, 0);
  delay(3000);
}
```

EXPERIMENT 7-2

◉ Control DC Motor Speed.

◆ **Write a program that controls the speed of DC motor and g radually increases the speed in a clockwise direction**

ARDUINO_KIT_DC_MOTOR2.ino

```
// Save AIN, BIN pin number of DC motor
int pin_DC_A = 5;
int pin_DC_B = 6;

void setup() {
  // Set DC motor AIN, BIN pins as output pins
  pinMode(pin_DC_A, OUTPUT);
  pinMode(pin_DC_B, OUTPUT);
}

void loop() {
  int Speed_value;
  // The motor rotates clockwise (forward rotation)
  // The rotation speed of the motor gradually increases
  for(Speed_value = 0; Speed_value < 256; Speed_value++)
  {
    forwardRotation(Speed_value);
    delay(20);
  }

  // The motor stops
  Stop();
  delay(3000);


  // The motor rotates counterclockwise (reverse rotation)
  // The rotation speed of the motor gradually increases
  for(Speed_value = 0; Speed_value < 256; Speed_value++)
  {
```

```
    reverseRotation(Speed_value);
    delay(20);
  }

// The motor stops
  Stop();
  delay(3000);
}

// DC motor control function
// Rotates the motor clockwise
// Control the speed by the value of Speed
// Speed value can be set from 0 to 255
// If the Speed value is 0, it is the same as stop, and the higher the value,
the faster the rotation speed
void forwardRotation(int Speed)
{
  analogWrite(pin_DC_A, Speed);
  digitalWrite(pin_DC_B, 0);
}

// Rotates the motor counterclockwise (reverse rotation)
// Control the speed by the value of Speed
// Speed value can be set from 0 to 255
// If the Speed value is 0, it is the same as stop, and the higher the value,
the faster the rotation speed
void reverseRotation(int Speed)
{
  analogWrite(pin_DC_B, Speed);
  digitalWrite(pin_DC_A, 0);
}

// Stops the motor
void Stop(void)
{
  digitalWrite(pin_DC_A, 0);
```

```
  digitalWrite(pin_DC_B, 0);
}
```

7.4. Practice

7-1. In Experiment 7-2, write a program in which the DC motor rotates at maximum speed and then slowly decreases.
(Stop -> Gradually increase speed -> Full speed -> Gradually decrease speed -> Stop -> Gradually increase speed -> ……)

7-2. Modify 7-1 program so that it operates in reverse rotation.

7-3. Referring to the program of 7-1 and 7-2, write a program to control the speed while repeating forward/reverse rotation infinitely.
(Stop -> Gradually increase speed in forward rotation -> Maximum speed in forward rotation -> Gradually decrease speed in forward rotation -> Stop -> Gradually increase speed in reverse rotation -> Maximum speed in reverse rotation -> Gradually decrease speed in reverse rotation -> Stop -> Gradually increase speed in forward rotation -> ……)

Chapter 8

Servo Motor Control

Learning Objectives

1. Understand the control principle of servo motor.

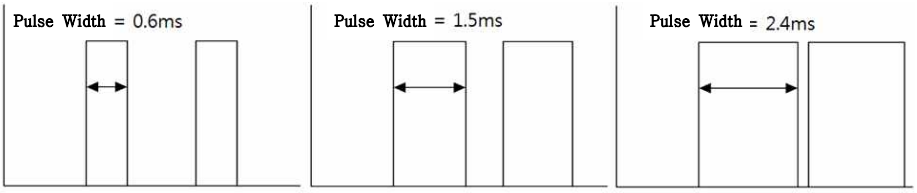
8. Servo Motor Control

In this chapter, let's write a program to control the servo motor of the HBE ARDUINO KIT board.

8.1. Control Principle of Servo Motor

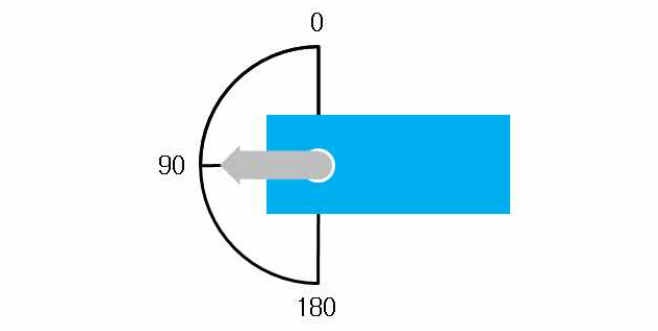
Servo motor is an important element used as actuator in automation system. The servo motor has the features of light weight, small size, easy installation, high efficiency, accurate controllability, and easy maintenance. RC servo motor maintains a specific angle by sending a control signal that matches the motor, PWM. However, unlike DC motor and step motor, servo motor usually has an angle limit of 180°. Most operate between 4V and 5V and consist of Vcc, GND, and PWM control lines. PWM signal should be input as shown in the figure. The cycle of servomotor PWM is 20 ms, and the pulse width in High shows 0° at

0.6 ms, 90° at 1.5 ms, and 180° at 2.4 ms.



[Figure 8-1] Servo Motor Control by Pulse Width

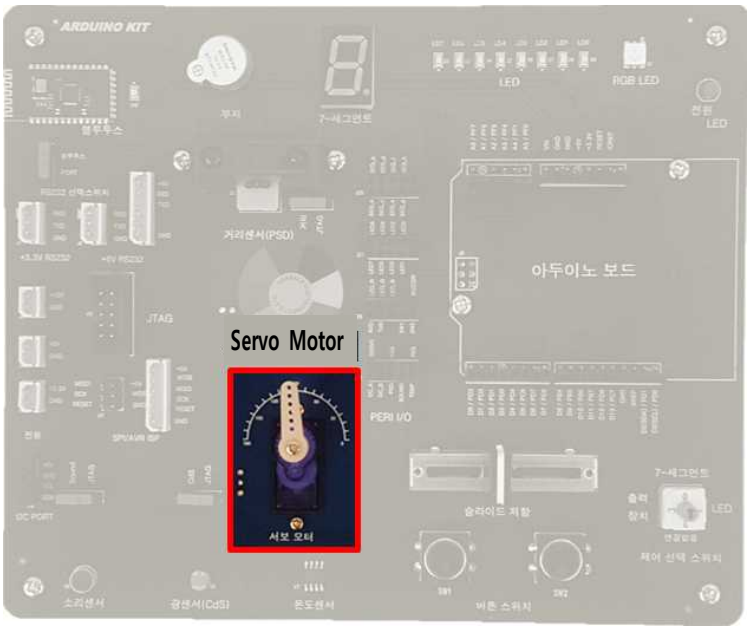
If the servo motor is controlled, the arrow rotates at the indicated angle. When controlled at 90°, it moves as shown below.



[Figure 8-2] Servo Motor Control by Pulse Width

8.2. Location of Servo Motor

The following shows the location of the servo motor.



[Figure 8-3] Location of Servo Motor

8.3. Servo Motor Control Program Experiment

EXPERIMENT 8-1

◉ Control the Servo Motor.

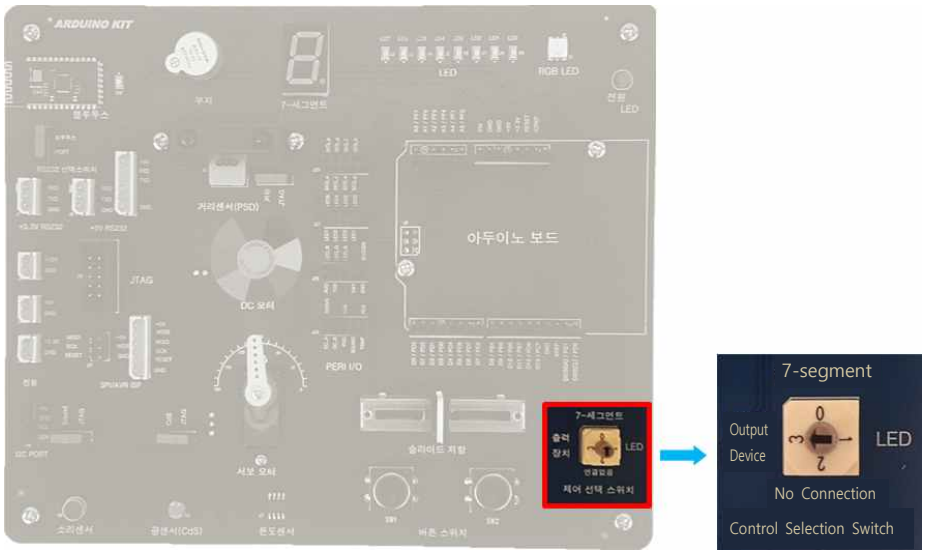
◆ **Write a program that controls the servo motor and repeats the rotation angle of the motor at 0°, 90°, 180°, 90°, and 0° infinitely.**

(0° → 90° →180° → 90° → 0° ……)

● The following table shows the pin map where the Arduino Leonardo and servo motors on the ARDUINO KIT board are connected.

Arduino Pin Number	Servo Motor	Remarks
13	SERVO	

To control the servo motor, set the control selection switch on ARDUINO KIT board to the output device.



[Figure 8-4] Set Control Selection Switch to Output Device

ARDUINO_KIT_SERVO1.ino

```
#include <Servo.h>

// Save pin number of Servo Motor
int pin_SERVO = 13;

// Create an object with SERVO that controls servo motor
Servo SERVO;

void setup() {
  // Connect SERVO object to pin 13
  SERVO.attach(pin_SERVO);

   // Initialize the angle of the servo motor to 0°
  SERVO.write(0);
  delay(2000);
}

void loop() {
  // Set the angle of the servo motor to 0°
  SERVO.write(0);
  delay(2000);

 // Set the angle of the servo motor to 90°
  SERVO.write(90);
  delay(2000);



  // Set the angle of the servo motor to 180°
  SERVO.write(180);
  delay(2000);

   // Set the angle of the servo motor to 90°
  SERVO.write(90);
  delay(2000);
}
```

EXPERIMENT 8-2

◉ Control the Servo Motor.

◆ Write a program that controls the servo motor so that the r otation angle of the motor gradually moves from 0° to 180°.

(0° -> gradually moves 180° -> 180° -> 0° ……)

ARDUINO_KIT_SERVO2.ino

```
#include <Servo.h>

// Save the pin number of Servo Motor
int pin_SERVO = 13;

// Create an object with SERVO that controls servo motor
Servo SERVO;

void setup() {
  // Connect the servo to pin 13 of the SERVO object
  SERVO.attach(pin_SERVO);

 // Initialize the angle of the servo motor to 0°
  SERVO.write(0);
  delay(2000);
}

void loop() {
  unsigned char  angle;
  // Control the angle of servo motor to move gradually from 0° to 180°
  for(angle = 0; angle <= 180; angle++)
  {
    SERVO.write(angle);
    delay(20);
  }
  delay(1000);
```

```
  // Set the angle of the servo motor to 0°
  SERVO.write(0);
  delay(1000);
}
```

EXPERIMENT 8-3

◉ Read servo motor angle.

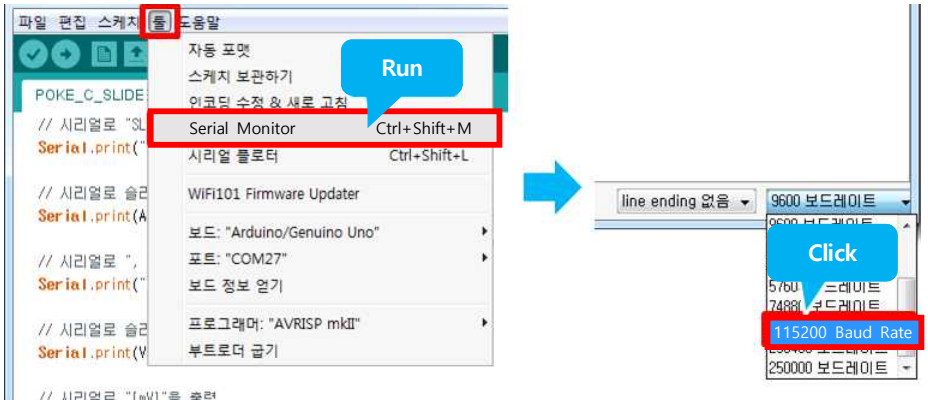
◆ **Specify the position of the servo motor by giving a serial command, write a program that reads the current value of the servo motor and outputs it to the monitor.**

ARDUINO_KIT_SERVO3.ino

```
#include <Servo.h>

// Save the pin number of Servo Motor
int pin_SERVO = 13;

// Create an object with SERVO that controls servo motor
Servo SERVO;

void setup() {
  // Serial Setting : Baud rate 115200, data 8bit, no parity, stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Connect the servo to pin 13 of the SERVO object
  SERVO.attach(pin_SERVO);
  //Wait until the serial port is connected
  while(!Serial);
}

void loop() {
  unsigned char  Command_Angle, Read_Angle;
```

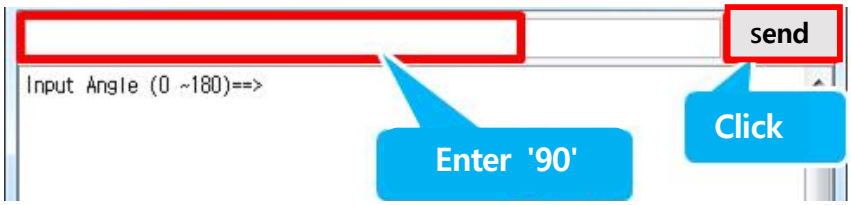
```
  Serial.print("Input Angle (0 ~180)==> ");
  while( Serial.available()==0);

  // Save and output servo motor control angle
  Command_Angle = Serial.parseInt();
  Serial.println(Command_Angle);

  // ---- part that receives serial data and controls the position of servo
motor
  // Specify the servo motor angle
  SERVO.write(Command_Angle);
  delay(1000);
  // Output "Command Angle: " serially
  Serial.print("Command Angle : ");

  // Outputs the angle controlled by the servo motor in serial
  Serial.println(Command_Angle);

  //------------ part that reads the angle of servo motor ------------
  // Read current servo motor angle.
  Read_Angle = SERVO.read();

  // Outputs "Read Angle : "serially
  Serial.print("Read Angle : ");

  // Outputs current servo motor angle serially
  Serial.println(Read_Angle);
  delay(500);
}
```

After uploading is completed, execute the serial monitor in the tool as s hown in the following figure to check the operation with the serial moni tor. After execution, set the baud rate to 115200.



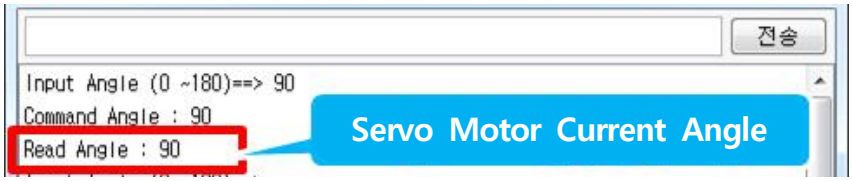
[Figure 8-5] Execute Serial Monitor and Set Baud Rate

Enter the angle to control the servo motor as 90 and click 'Send'.



[Figure 8-6] Enter Control Angle

After controlling the servo motor with the input angle, the result value is output as shown below.



[Figure 8-7] Output Result Value

8.4. Practice

8-1. Write a program so that the servo motor is 180° to 0° as opposed to Experiment 8-2.
(180° -> gradually moves to 0° -> 0° -> 180° -> ……)

8-2. Write a program that makes the servo motor slowly go back and forth between 0° and 180°.
(0° -> gradually moves to 180° -> 180° -> gradually moves to 0° -> 0° -> ……)

Chapter 9

Slide Resistance

Learning Objectives

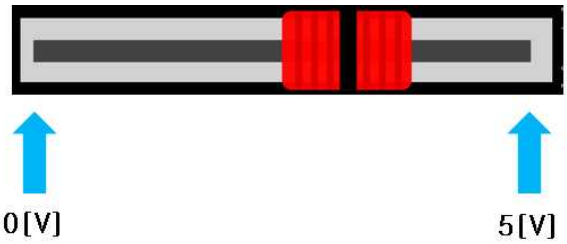
1. Understand the operating principle of slide resistance.

9. Slide Resistance

In this chapter, we write a program that controls a servo motor using the slide resistance of HBE ARDUINO KIT board.

9.1. Principle of Slide Resistance

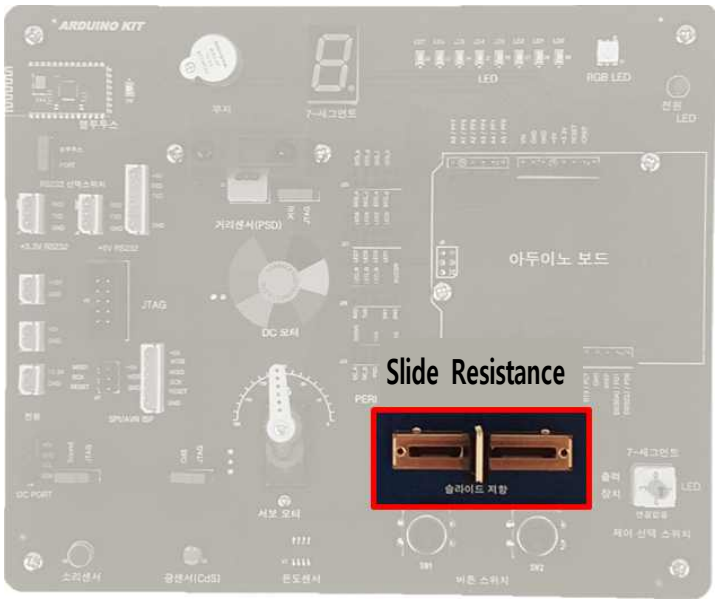
The principle of slide resistance is same as that of the variable resistance. 0V to 5V (input voltage) is output depending on the position of the slide resistor as shown below.



[Figure 9-1] Voltage Output depending on the Position of Slide Resistor

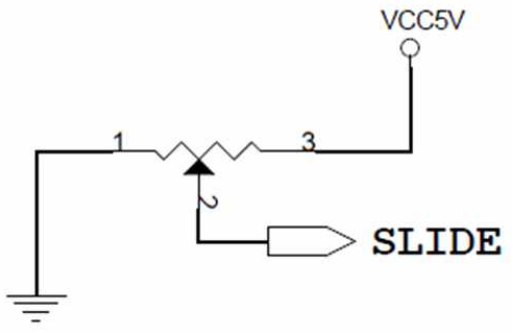
9.2. Location and Circuit of Slide Resistance

The following shows the location of the slide resistance.



[Figure 9-2] Location of Slide Resistance

The following shows the circuit of the slide resistor. 5V power is applie d and a voltage of 0V to 5V is output to SLIDE pin.



[Figure 9-3] Circuit of Slide Resistance

9.3. Slide Resistance Program Experiment

EXPERIMENT 9-1

◉ Output the voltage of the slide resistor to serial monitor.

◆ **Write a program that measures the voltage output from the slide resistor and outputs it to the serial monitor.**

● The following table shows the pin map where the Arduino Leonardo and slide resistors on the ARDUINO KIT board are connected.

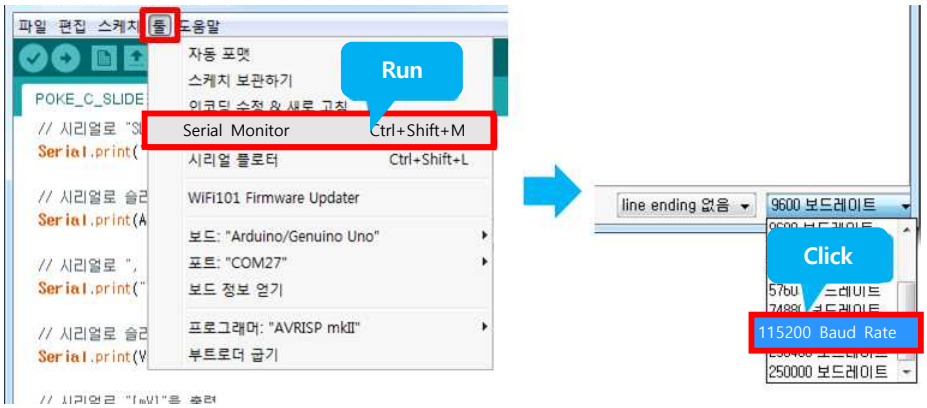
Arduino Pin Number	Slide	Remarks
A5	SLIDE	

ARDUINO_KIT_SLIDE.ino

```
// Save the pin number of slide resistor
int pin_SLIDE = A5;

void setup() {
  // Serial Setting : Baud rate 115200, data 8bit, no parity, stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of the slide resistor as an input pin
  pinMode(pin_SLIDE, INPUT);
  //Wait for the serial port to be connected
  while(!Serial);
}

void loop() {
  unsigned long  Volt, ADC_data;
  // Read analog value of slide resistance as digital value
  ADC_data = analogRead(pin_SLIDE);

  // Convert digital value to voltage value. The unit value is [mV]
  // voltage  = ADC_data * input voltage / ADC value of input voltage
```

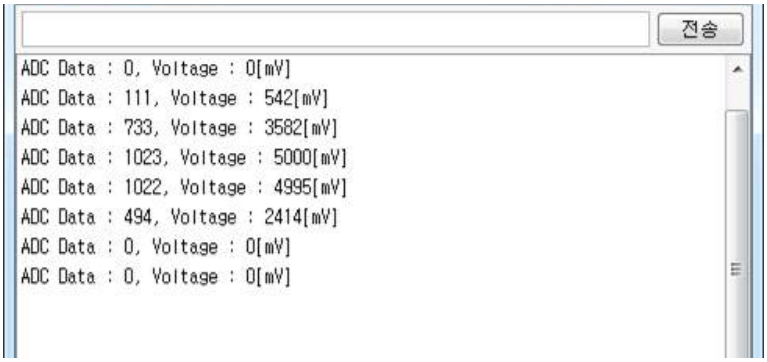
```
  Volt = ADC_data * 5000 / 1023;

  // Outputs "SLOPE : " serially
  Serial.print("ADC Data : ");

  // Outputs ADC value of the slide resistance serially
  Serial.print(ADC_data);

  // Outputs ", Voltage : " serially
  Serial.print(", Voltage : ");

  // utputs voltage value of the slide resistance serially
  Serial.print(Volt);

  // Outputs "[mV]" serially
  Serial.println("[mV]");
  delay(500);
}
```

After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial mo nitor. After running, set the baud rate to 115200.



[Figure 9-4] Execute Serial Monitor and Set Baud Rate

Outputs ADC value and voltage value to the serial monitor as show below (1[V] is 1000[mV].



[Figure 9-5] Check the Operation

EXPERIMENT 9-2

◉ Control Servo motor using slide resistance.

◆ **Write a program that reads the slide resistance and controls the servo motor.**

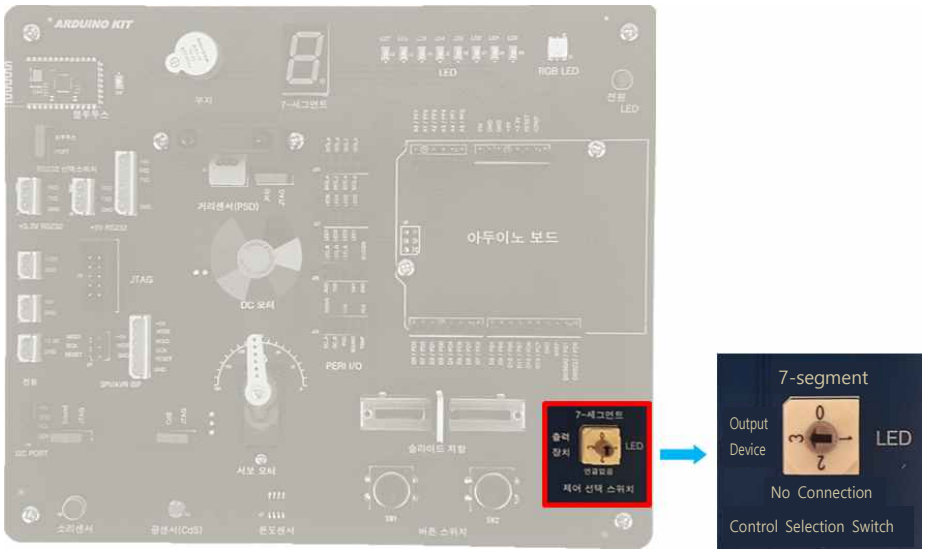
(The range of slide resistance is 0 ~ 1023. Make angle of 0° ~ 180° to control the servo motor)

● The following table shows the pin map where the Arduino Leonardo, the slide resistor and the servo motor on the ARDUINO KIT board are connected.

Arduino Pin Number	Slide	Remarks
A5	SLIDE	

Arduino Pin Number	Servo Motor	Remarks
13	SERVO	

To control the servo motor, set the control selection switch on the ARDUINO KIT board to the output device.



[Figure 9-6] Set the control selection switch to output device

ARDUINO_KIT_SLIDE_SERVO.ino

```
#include <Servo.h>

// Save the pin number of slide resistor
int pin_SLIDE = A5;

// Save the pin number of servo motor
int pin_SERVO = 13;

// Create an object with SERVO that controls servo motor
Servo SERVO;

void setup() {
  // Connect the servo to pin 13 of the SERVO object
  SERVO.attach(pin_SERVO);


  // Set the pin of the slide resistor as an input pin
```

```
  pinMode(pin_SLIDE, INPUT);
}

void loop() {
  long Angle, ADC_data;
  // Read analog value of slide resistance as digital value
  ADC_data = analogRead(pin_SLIDE);

  // Convert digital value to voltage value. The unit is [mV]
  // Angle = ADC_data * Maximum angle / ADC value of input voltage
  Angle = ADC_data * 180 / 1023;

  // Control the angle of the servo motor to move with the Angle value
  SERVO.write(Angle);
  delay(100);
}
```

9.4. Practice

9-1. Write a program that outputs in the range of 0 to F to 7-segment by the slide resistance value as Experiment 9-2.

Chapter 10

Light Sensor

Learning Objectives

1. Understand the principle of operation of the light sensor.

10. Light Sensor

In this chapter, we will write a program to control the relay with HBE ARDUINO KIT board's light sensor.

10.1. What is Light Sensor?

CdS sensor refers to a device that uses a phenomenon in which electrons and holes increase when light hits a semiconductor, and the conductivity changes in proportion to the irradiated light energy, and the electrical resistance changes accordingly. This device is relatively inexpensive because its manufacturing process is simple and suitable for mass production. CdS sensor is widely used in camera exposure meter, street light automatic blinking device, and TV set automatic brightness control device. CdS sensor has photoconductor CdS (cadmium sulfide) inserted between two electrodes, so its internal resistance changes according to the intensity of light. It is an optical variable resistor whose internal resistance becomes small in bright light. Resistance is

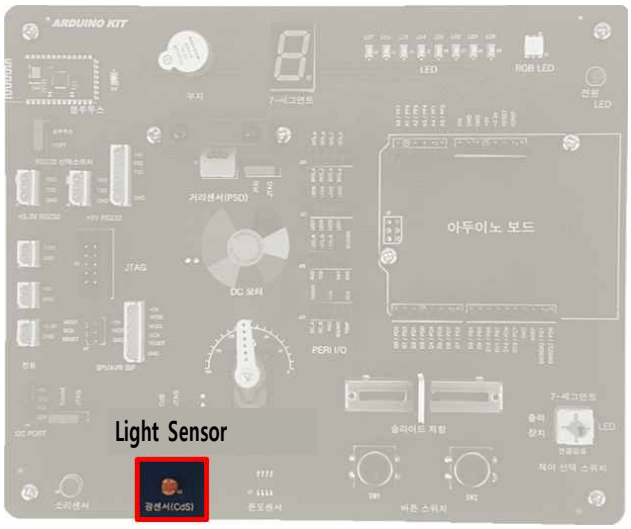
proportional to voltage and inversely proportional to current (R = V/I) according to Ohm's law. Therefore, in a bright place, the resistance decreases and the current increases, and in a dark place, the current decreases.

[Figure 10-1] Image of
Light Sensor

The light detection sensor (CdS) has high sensitivity, is compact, and has a simple structure, so it is easy to use, but its electrical conductivity tends to saturate with the increase in light. It has the drawback that the change in conductivity is delayed for rapid changes in light, so it can be applied only to relatively gentle changes in illuminance. Therefore, rather than being used for applications requiring precise illuminance values, it is widely used in light-sensing switches that operate elements by detecting light.

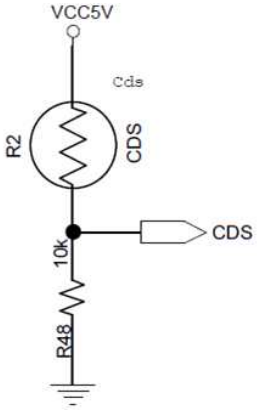
10.2. Location and Circuit of Light Sensor

The following shows the location of the light sensor.



[Figure 10-2] Location of Light Sensor

The following shows the circuit of the light sensor.



[Figure 10-3] Circuit of Light Sensor

The light sensor changes according to brightness. In the circuit above, when it gets dark, the resistance value of the light sensor increases, so the voltage output to CDS decreases. And when it becomes brighter, the resistance value of the light sensor decreases, so the voltage output to CDS increases.

10.3. Light Sensor Program Experiment

EXPERIMENT 10-1

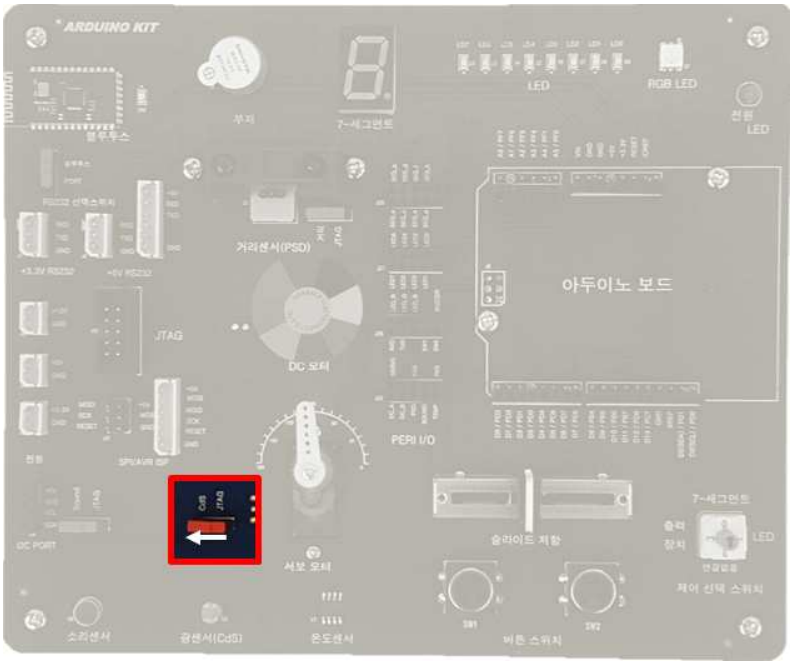
◉ Outputs the value measured with light sensor to the serial monitor

◆ **Write a program that measures the value output from the light sensor and outputs it to the serial monitor.**

● The following table shows the pin map where the Arduino Leonardo and light sensor on ARDUINO KIT board are connected .

Arduino Pin Number	Light Sensor	Remarks
A1	CDS	

● To measure the light sensor, turn the switch on ARDUINO KIT board in the direction of the arrow (left) as shown below.

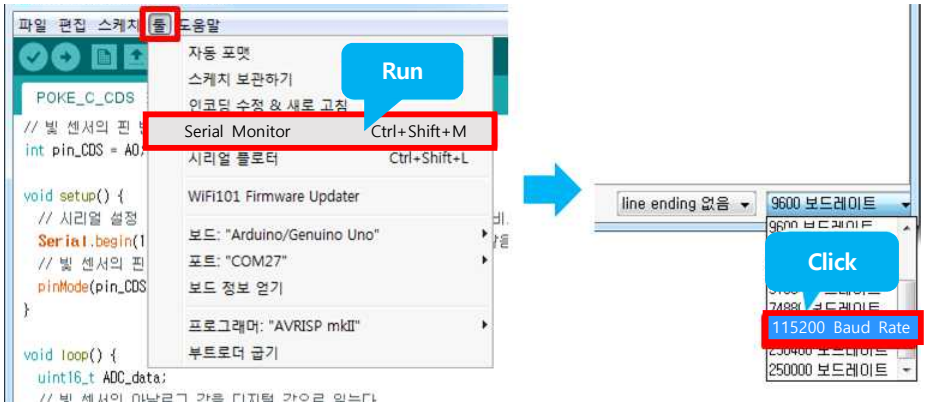


[Figure 10-4] Switch Operation to Measure Light Sensor

ARDUINO_KIT_CDS.ino

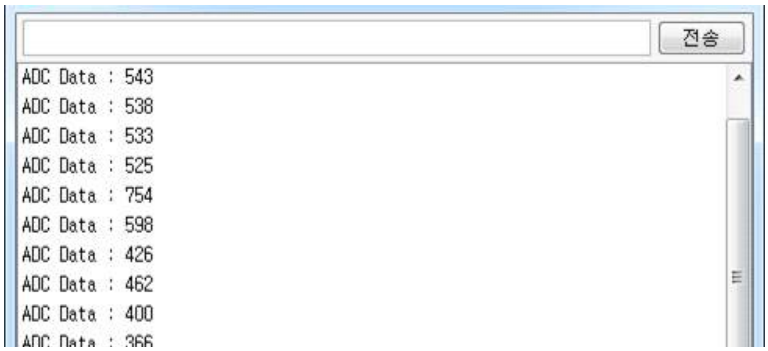
```
// Save the pin number of Light Sensor
int pin_CDS = A1;

void setup() {
  // Serial Setting : Baud rate 115200, data 8bit, no parity, stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of the light sensor as an input pin
  pinMode(pin_CDS, INPUT);
  //Wait until the serial port is connected
  while(!Serial);
}

void loop() {
  unsigned int  ADC_data;
  // Read the analog value of the light sensor as a digital value
  // The higher the digital value, the brighter the light
  ADC_data = analogRead(pin_CDS);

  // Outputs "ADC Data : " serially
  Serial.print("ADC Data : ");

  // Outputs the ADC value in serial
  Serial.println(ADC_data);
  delay(500);
}
```

● After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 10-5] Execute Serial Monitor and Set Baud Rate

● Outputs ADC value to the serial monitor as shown below.



[Figure 10-6] Check Operation

EXPERIMENT 10-2

◉ Control Relay using light sensor

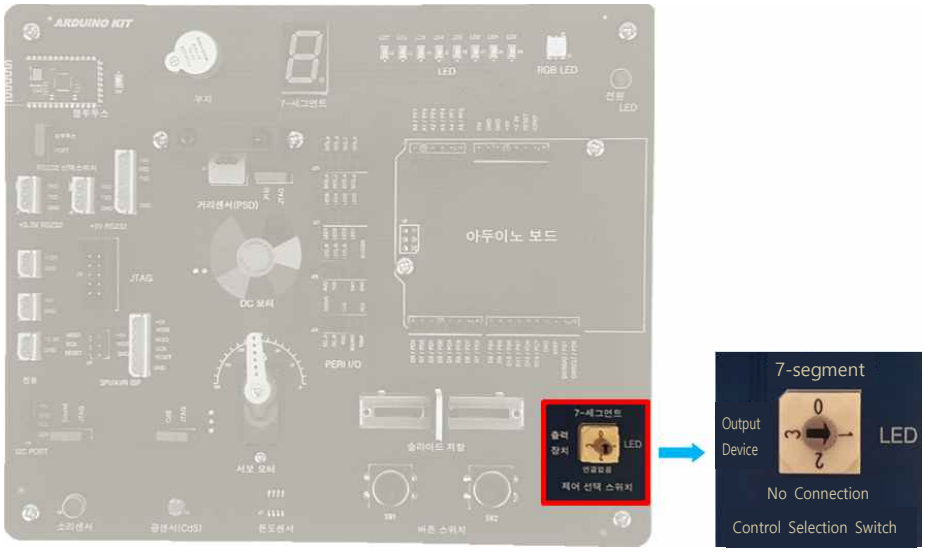
◆ **Write a program that reads the light sensor and turns on the relay if it is less than 512 and turns off the relay if it is more than 512.**

● The following table shows pin map where Arduino Leonardo, light sensor and LED on the ARDUINO KIT board are connected.

Arduino Pin Number	Light Sensor	Remarks
A1	CDS	

Arduino Pin Number	LED	Remarks
5	LED0	Control selection switch should be set to LED

● To control the LED, set the control selection switch of the ARDUINO KIT board to LED.



[Figure 10-7] Select the control selection switch to LED

ARDUINO_KIT_CDS_LED.ino

```
// Save the pin number of Light Sensor
int pin_CDS = A1;

// Save the pin number of LED0
int pin_LED0 = 5;

void setup() {
  // Set the pin of the light sensor as an input pin
  pinMode(pin_CDS, INPUT);
  // Set LED0 pin as output pin
  pinMode(pin_LED0, OUTPUT);
}

void loop() {
  uint16_t ADC_data;
  // Read the analog value of the light sensor as a digital value
  // The higher the digital value, the brighter the light
  ADC_data = analogRead(pin_CDS);

  // If the value measured by light sensor is less than 512, turn on LED0
  if(ADC_data < 512)
  {
    // Output HIGH to LED0 pin (LED0 ON)
    digitalWrite(pin_LED0, 1);
  }
  // If the value measured by light sensor is greater than or equal to 512,
turn off LED0
  else
  {
    // Output LOW to LED0 pin (LED0 OFF)
    digitalWrite(pin_LED0, 0);
  }
  delay(100);
}
```

10.4. Practice

10-1. Write a program that controls the brightness of RGB LED according to the measured values of the light sensor (If the measurement value is large, make Red LED dark, and if the measurement value is small, make Red LED bright. Measured value and brightness are in inverse proportion).

Chapter 11

Temperature Sensor

Learning Objectives

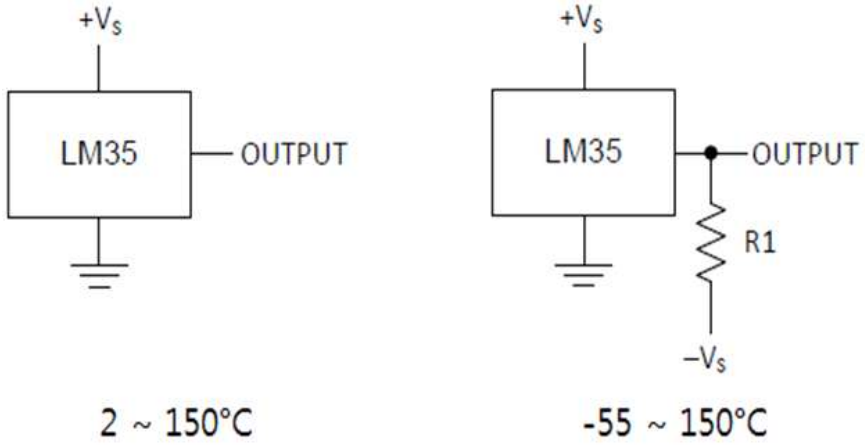
1. Understand the temperature sensor.
2. Check the temperature measurement range by the circuit design.

11. Temperature Sensor

In this chapter, we write a program to control DC motor using the temperature sensor of HBE ARDUINO KIT board.

11.1. What is Temperature Sensor?

Temperature sensor detects temperature and converts it into an electrical signal, such as a change in voltage or resistance. Temperature sensor is very diverse depending on the installation location, processor, physical features of a specific object. The principle is that the resistance value changes according to the temperature by thinly coating the metal oxide. It is widely used for temperature compensation of transistor circuits, temperature measurement, temperature control, and automatic control of communication device.



[Figure 11-1] Temperature Measurement Range by Circuit Design

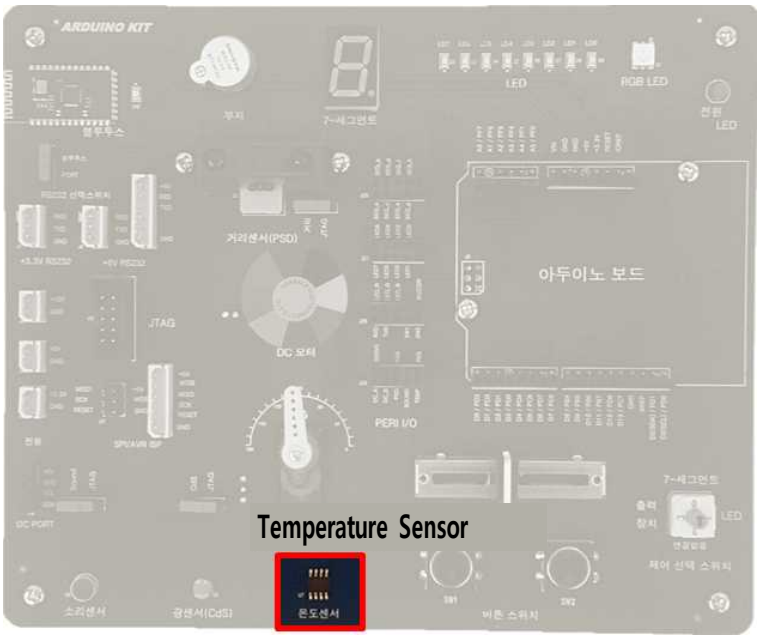
Depending on the circuit design, the temperature range that can be measured differs, and here, left circuit above is designed. OUTPUT value is output as a voltage and 10mV per 1°C is output. For example, if 250mV is output, the temperature is 25°C.

$$\text{Temperature} = \text{Voltage[V]} \times 100$$

The circuit used here is the left circuit in [Figure 11-1].

11.2. Location of Temperature Sensor

The following shows the location of temperature sensor.



[Figure 11-2] Location of Temperature Sensor

11.3. Temperature Sensor Program Experiment

EXPERIMENT 11-1

◉ Output the value measured by the temperature sensor to serial monitor.

◆ **Write a program that measures the value output from the temperature sensor, convert it to temperature, and output it to the serial monitor.**

● The following table shows the pin map where the Arduino Leonardo and temperature sensor on ARDUINO KIT board are connected.

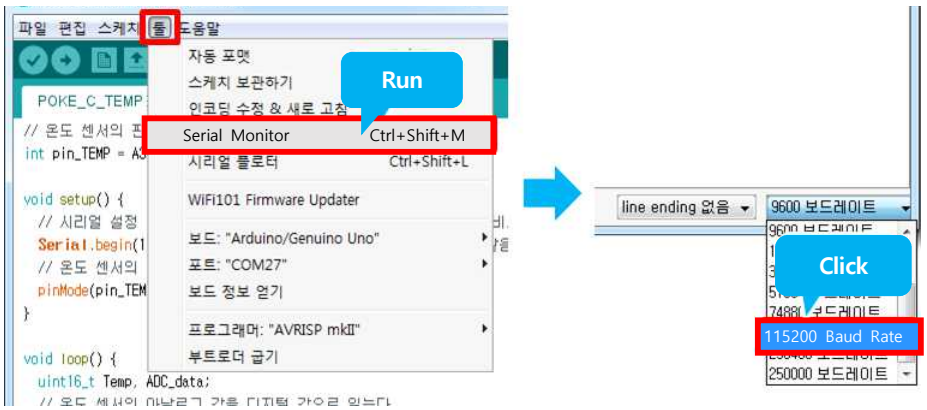
Arduino Pin Number	Temperature Sensor	Remarks
A4	TEMP	

ARDUINO_KIT_TEMP.ino

```
// Save the pin number of Temperature Sensor
int pin_TEMP = A4;

void setup() {
  // Serial Setting : Baud rate 115200, Data 8 bit, No Parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of the temperature sensor as an input pin
  pinMode(pin_TEMP, INPUT);
  //Wait until the serial port is connected
  while(!Serial);
}

void loop() {
    long int Temp, ADC_data;
  // Reads the analog value of temperature sensor as 10-bit digital value
  ADC_data = analogRead(pin_TEMP);

  // Convert digital value to temperature value
```

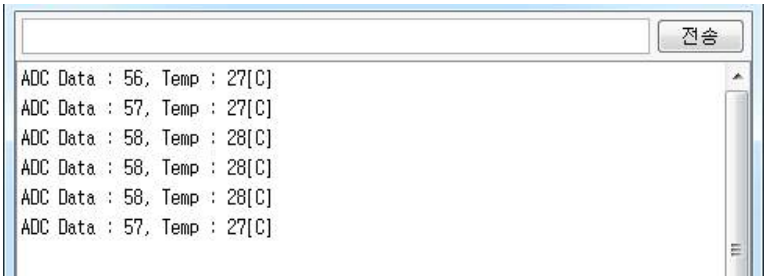
```
  // Temp = Voltage[V] * 100
  Temp = ADC_data * 100 * 5 / 1023;

  // Outputs "ADC Data : " serially
  Serial.print("ADC Data : ");

  // Output the measured value of temperature sensor serially
  Serial.print(ADC_data);

  // Outputs ", Temp : " serially
  Serial.print(", Temp : ");

  // Outputs temperature value serially
  Serial.print(Temp);
  // Outputs "[C]" serially
  Serial.println("[C]");
  delay(500);
}
```

● After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 11-3] Execute Serial Monitor and Set Baud Rate

● Output ADC value and temperature value to the serial monitor as sh own below.



[Figure 11-4] Check Operation

◉ DC motor control using temperature sensor

EXPERIMENT 11-2

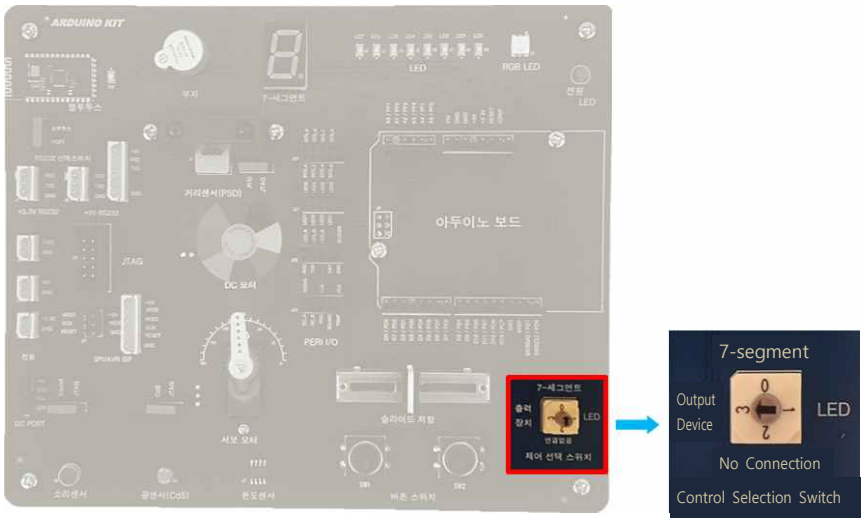
◆ **Write a program that reads the temperature sensor and stops and rotates the DC motor forward/reverse according to the temperature value (When the temperature is less than 25°, make DC motor rotate forward, and when the temperature is greater than 28°, make DC motor rotate reverse. When it is 25° to 28°, make DC motor stop).**

● The following table shows the pin map where the temperature senso r, Arduino Leonardo and DC motor on ARDUINO KIT board are connec ted.

Arduino Pin Number	Temperature Sensor	Remarks
A4	TEMP	

Arduino Pin Number	DC Motor	Remarks
5	AIN	Control selection switch should be set to the output device
6	BIN	

● To control the relay, set the control selection switch on the ARDUIN O KIT board to the output device.



[Figure 11-5] Set Control Selection Switch to Output Device

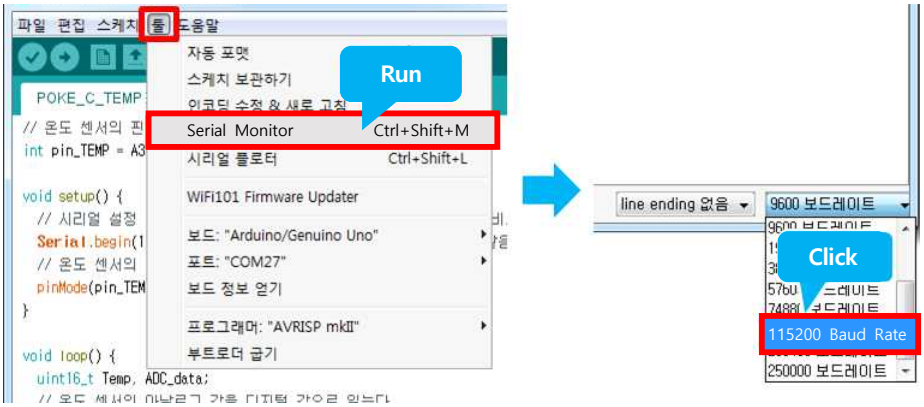
ARDUINO_KIT_TEMP_DC.ino

```
// Save the pin number of Temperature Sensor
int pin_TEMP = A4;

// Save pin number of DC motor AIN, BIN
int pin_DC_A = 5;
int pin_DC_B = 6;

void setup() {
 // Serial Setting : Baud rate 115200, data 8bit, no parity, stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of the temperature sensor as an input pin
  pinMode(pin_TEMP, INPUT);
  // Set DC motor AIN, BIN pins as output pins
  pinMode(pin_DC_A, OUTPUT);
  pinMode(pin_DC_B, OUTPUT);
  pinMode(pin_TEMP, INPUT);
  //Wait until the serial port is connected
```

```
  while(!Serial);
}

void loop() {
  long int Temp, ADC_data;
  // Read the analog value of the temperature sensor as a digital value
  ADC_data = analogRead(pin_TEMP);

  // Convert digital value to temperature value
  // Temp = 전압[V] * 100
  Temp = ADC_data * 100 * 5 / 1023;

  // Outputs ", Temp : " serially
  Serial.print("Temp : ");

  // Outputs temperature value serially
  Serial.print(Temp);
  // Outputs "[C]" serially
  Serial.println("[C]");
  delay(1000);

  if(Temp < 25)
  {
    // Forward rotation when the temperature is less than 25°
    forwardRotation(100);
  }
  else if(Temp > 28)
  {
    // Reverse rotation when the temperature is greater than 28°
    reverseRotation(100);
  }
  else
  {
    // Stop when the temperature is between 25° and 28°
    Stop();
  }
```

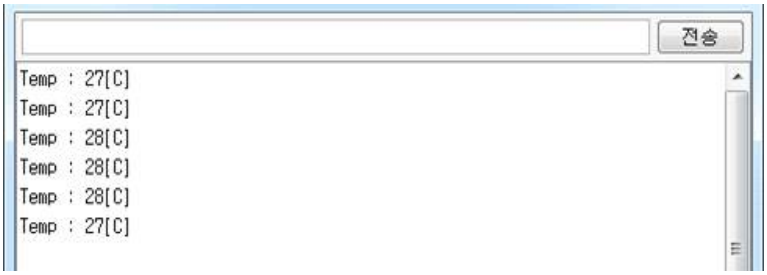
```
  delay(100);
}

// DC motor control function
// Makes DC motor rotate clockwise
// Control the speed according to the value of Speed.
// Speed value can be set from 0 to 255
// If Speed value is 0, it is the same as stop , and the higher the value,
the faster the rotation speed
void forwardRotation(int Speed)
{
  analogWrite(pin_DC_A, Speed);
  digitalWrite(pin_DC_B,0);
}

// Makes DC motor rotate counterclockwise
// Controls the speed by Speed value
// Speed value can be set from 0 to 255
// If tSpeed value is 0, it is the same as stop, and the higher the value,
the faster the rotation speed
void reverseRotation(int Speed)
{
  analogWrite(pin_DC_B, Speed);
  digitalWrite(pin_DC_A, 0);
}

// Stops DC motor
void Stop(void)
{
  digitalWrite(pin_DC_A, 0);
  digitalWrite(pin_DC_B, 0);
}
```

● After uploading is completed, execute the serial monitor in the tool as shown below to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 11-6] Execute Serial Monitor and Set Baud Rate

● Outputs ADC value and temperature value to the serial monitor as s hown below.



[Figure 11-7] Check Operation

11.4. Practice

11-1. Write a program that controls the servo motor with the measured value of the temperature sensor (If the temperature value is 10°, the angle is also controlled by 10°).

Chapter 12

Distance Sensor

Learning Objectives

1. Understand the principle of PSD sensor.
2. Understand how to convert distance according to voltage.

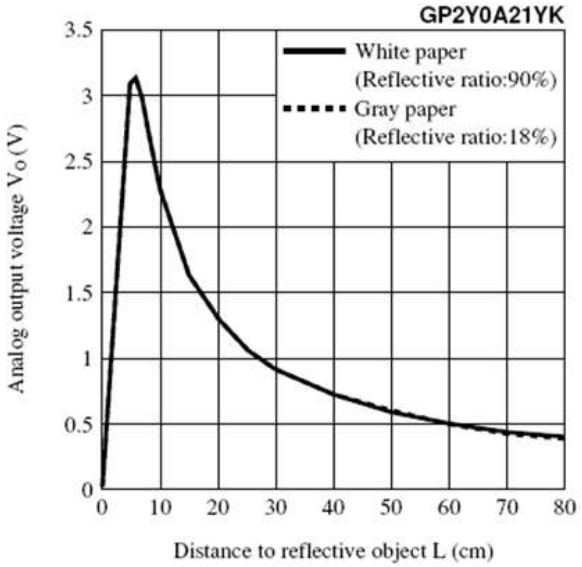
12. Distance Sensor

In this chapter, we will measure the distance using the distance sensor of HBE ARDUINO KIT board and write a program that controls LED according to the distance.

12.1. Principle of PSD Sensor

Distance measurement sensor used in PSD is a distance measurement sensor using an optical sensor by Sharp's PSD. GP2Y0A21YK, a typical distance measurement sensor, is a module with LED (Infrared Light Emitting Diode) and PSD element (Position Sensitive Detector) built-in, and converts the distance to an object placed in front of the sensor into DC voltage and outputs it. This sensor is divided into Ir emitting part and a light receiving part. When the infrared rays emitted from the emitter are reflected on an object, they are focused through the receiver lens and the angle is measured. This angle is calculated using triangulation method. The measurement distance of the used PSD can

be measured from a minimum of 10 cm to 80 cm.



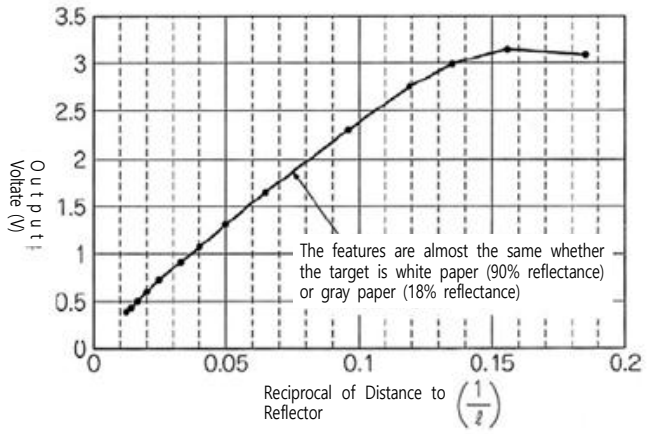
[Figure 12-1] Distance by Voltage

PSD Sensor is an optical sensor that applies the principle of triangulation. Output light of the LED is projected onto the reflector (target) through the light emitting lens as shown below. When light is irradiated to the reflector, a part of the reflected light is incident on the light receiving lens and is irradiated onto the light receiving element (PSD). The light emitting lens in front of the LED is designed so that the output light of the LED is parallel. The light receiver lens is designed to condense the reflected light to a single point on the light receiver. Both the light emitting lens and the light receiving lens are double-sided aspherical convex lenses.

12.2. Distance Conversion

Looking at the curve of [Figure 12-1], when the distance of the object

is 80cm, the output voltage is 0.4V, and when the distance is 10cm, the output voltage is 2.3V. If the output voltage output from PSD sensor is measured using ADC, distance away from the object by the above curve is determined. If the distance shown in the characteristic curve is the reciprocal number, the relationship between the distance and the output voltage becomes linear as shown in the following curve (Calculation of the distance becomes easy).



[Figure 12-2] Reciprocal Value of Distance by Voltage

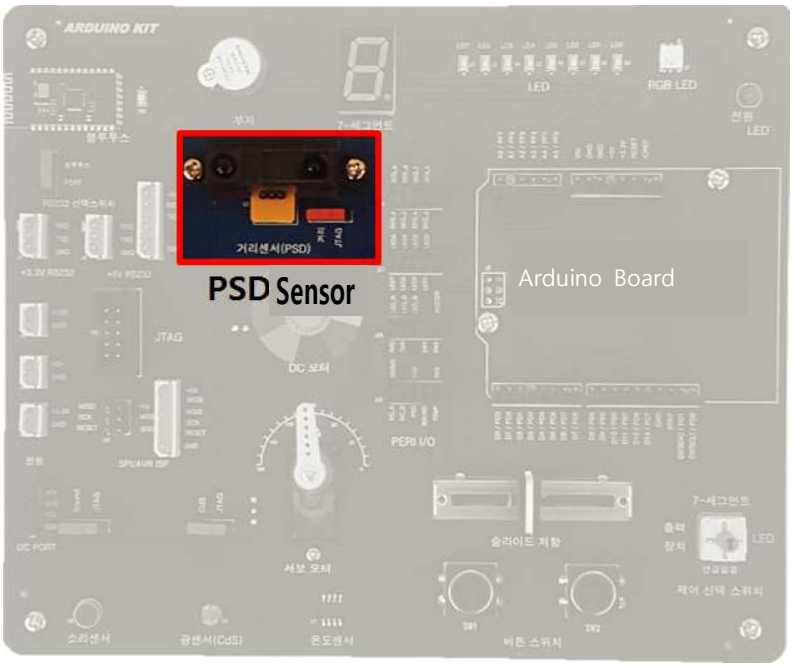
Therefore, based on the above graph, 1/distance becomes measured voltage * 0.0434 - 0.0046.

Taking the reciprocal of 1/distance calculates the distance value. The distance becomes 1/(measured voltage * 0.0434 -0.0046). If recalculating this ;

Distance= 10000 / (Measured Voltage * 434 - 46) [Cm]

12.3. Location of Distance Sensor

The following shows the location of the distance sensor.



[Figure 12-3] Location of Distance Sensor

12.4. Distance Sensor Program Experiment

EXPERIMENT 12-1

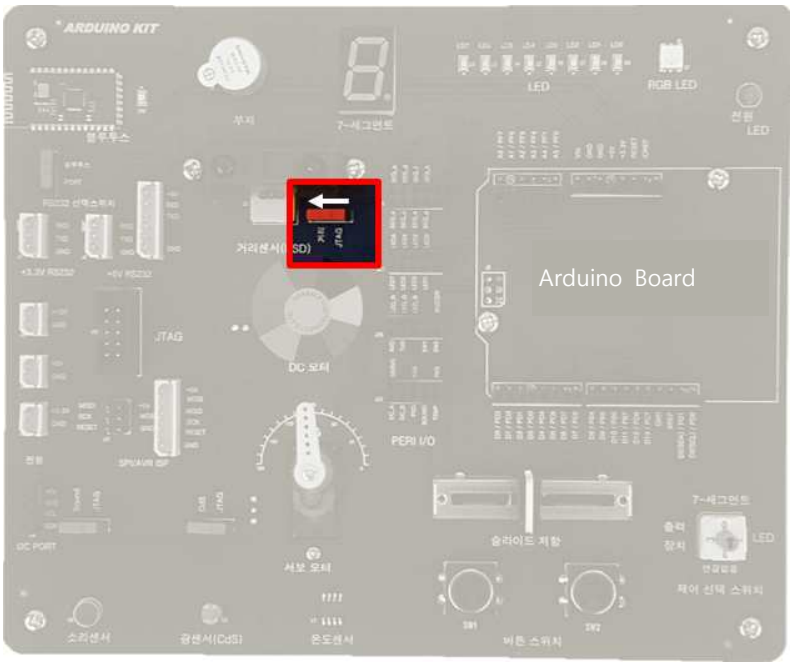
◉ Output the value measured by the distance sensor to serial monitor.

◆ **Write a program that measures the value output from the distance sensor, convert it into distance, and output it to the serial monitor.**

● The following table shows the pin map where the distance sensor is connected to the Arduino Leonardo on the ARDUINO KIT board.

Arduino Pin Number	Distance Sensor	Remarks
A0	PSD	

● To measure PSD sensor, turn the switch on the ARDUINO KIT boar d in the direction of the arrow (left) as shown below.



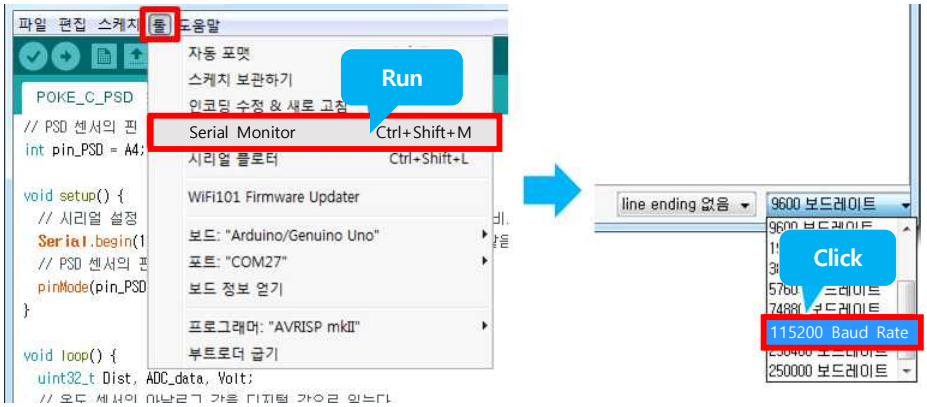
[Figure 12-4] Switch Operation for PSD Sensor Measurement

ARDUINO_KIT_PSD.ino

```
// Save the pin numbers of PSD Sensor
int pin_PSD = A0;

void setup() {
  // Serial Setting : Baud rate 115200, Data 8bit, no parity, stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of PSD sensor as input pin
  pinMode(pin_PSD, INPUT);
  //Wait for the serial port to be connected
  while(!Serial);
}

void loop() {
  long int Dist, ADC_data, Volt;
```

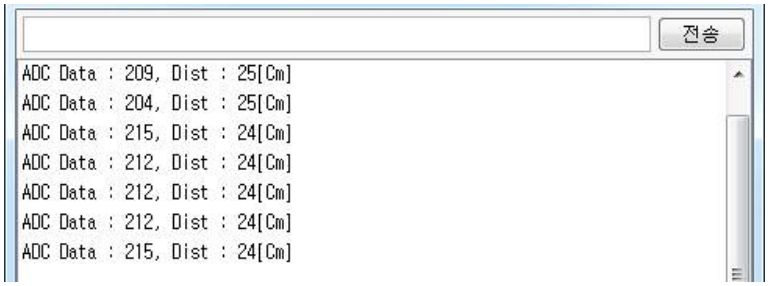
```
  // Read analog value of temperature sensor as digital value
  ADC_data = analogRead(pin_PSD);

  // Convert digital value to distance value
  // Dist = 10000 / (Voltage[V] * 434 - 46)
  // Voltage = ADC+Data * 5 / 1023
  Dist = 10000 / (ADC_data * 434 * 5 / 1023 - 46);

  // Specify range of distance value
  if(Dist > 80)
  {
    // Maximum 80cm
    Dist = 80;
  }
  else if(Dist < 10)
  {
    // Minimum 10cm
    Dist = 10;
  }

  // Outputs "ADC Data : " serially
  Serial.print("ADC Data : ");

  // Outputs the measured value of PSD Sensor serially
  Serial.print(ADC_data);

  // Outputs ", Dist : "serially
  Serial.print(", Dist : ");

  // Outputs distance value serially
  Serial.print(Dist);
  // Outputs "[Cm]" serially
  Serial.println("[Cm]");
  delay(500);
}
```

● After uploading is completed, execute the serial monitor in the tool as shown in the following figure to check the operation with the serial monitor. After running, set the baud rate to 115200.



[Figure 12-5] Execute Serial Monitor and Set Baud Rate

● Outputs the ADC value and distance value to the serial monitor as s hown below.



[Figure 12-6] Check Operation

◉ Control LED using Distance Sensor

EXPERIMENT 12-2

◆ Write a program that reads the distance sensor and controls the LED by the distance value.

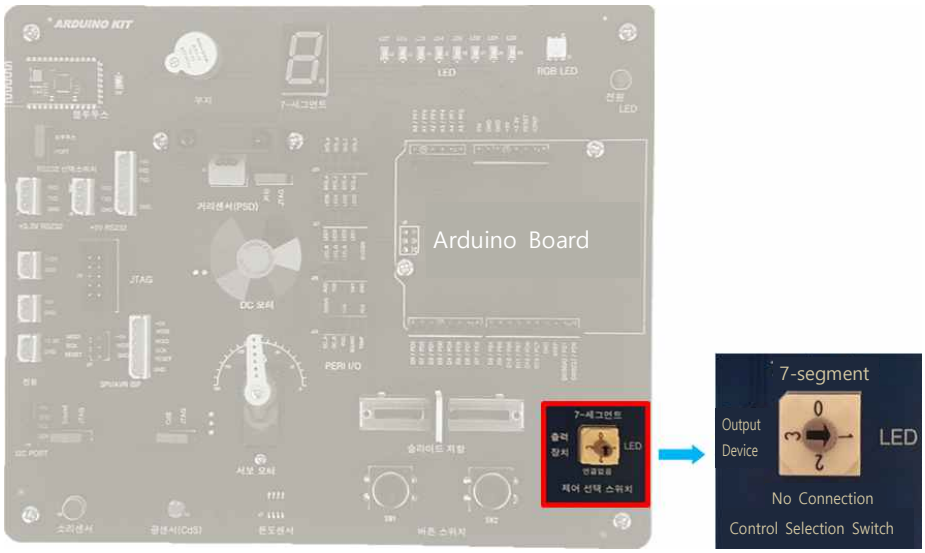
(If distance is less than 19, LED0 ON, if less than 29, LED0, 1 ON, if less than 39, LED0, 1, 2 ON, if less than 49, LED0, 1, 2, 3 ON, if less than 59, LED0, 1, 2, 3, 4 ON, if less than 69, LED0, 1, 2, 3, 4, 5 ON, if less than 79, LED0, 1, 2, 3, 4, 5, 6 ON, if more than 80, write a program so that 8 LEDs are ON)

● The following table shows the pin map where the Arduino Leonardo , distance sensor and LED on the ARDUINO KIT board are connected.

Arduino Pin Number	Distance Sensor	Remarks
A0	PSD	

Arduino Pin Number	LED	Remarks
5	LED0	The control selection switch should be set to LED
6	LED1	
8	LED2	
9	LED3	
10	LED4	
11	LED5	
12	LED6	
13	LED7	

● To control LED, set the control selection switch of the ARDUINO KIT board to LED.



[Figure 12-7] Control Selection Switch of ARDUINO KIT board to LED

ARDUINO_KIT_PSD_LED.ino

```
// Save the pin numbers of PSD Sensor
int pin_PSD = A0;

// Save the pin numbers of LED0~7
int pin_LED0 = 5;
int pin_LED1 = 6;
int pin_LED2 = 8;
int pin_LED3 = 9;
int pin_LED4 = 10;
int pin_LED5 = 11;
int pin_LED6 = 12;
int pin_LED7 = 13;

void setup() {
  // Set the pin of the PSD sensor as an input pin
```

```
  pinMode(pin_PSD, INPUT);
  // Set the pins of LED0~7 as output pins
  pinMode(pin_LED0, OUTPUT);
  pinMode(pin_LED1, OUTPUT);
  pinMode(pin_LED2, OUTPUT);
  pinMode(pin_LED3, OUTPUT);
  pinMode(pin_LED4, OUTPUT);
  pinMode(pin_LED5, OUTPUT);
  pinMode(pin_LED6, OUTPUT);
  pinMode(pin_LED7, OUTPUT);

  // Since the distance value is greater than 10, LED0 is always on
  digitalWrite(pin_LED0, 1);
}

void loop() {
  long int  Dist, ADC_data, Volt;
  // Read the analog value of the PSD sensor as a digital value
  ADC_data = analogRead(pin_PSD);

  // Convert the digital value into the distance value
  // Dist = 10000 / (Voltage[V] * 434 - 46)
  // Voltage = ADC+Data * 5 / 1023
  Dist = 10000 / (ADC_data * 434 * 5 / 1023 - 46);

  // Specify range of distance value
  if(Dist > 80)
  {
    // Maximum 80cm
    Dist = 80;
  }
  else if(Dist < 10)
  {
    // Minimum 10cm
    Dist = 10;
  }
```

```
// Turn off LED 1 ~ 7
LED_OFF();

// Control  LED by the distance value
if(Dist > 19)
{
  // Turn on LED1
  digitalWrite(pin_LED1, 1);
}
if(Dist > 29)
{
  // Turn on LED2
  digitalWrite(pin_LED2, 1);
}
if(Dist > 39)
{
  // Turn on LED3
  digitalWrite(pin_LED3, 1);
}
if(Dist > 49)
{
  // Turn on LED4
  digitalWrite(pin_LED4, 1);
}
if(Dist > 59)
{
  // Turn on LED5
  digitalWrite(pin_LED5, 1);
}
if(Dist > 69)
{
  // Turn on LED6
  digitalWrite(pin_LED6, 1);
}
if(Dist > 79)
```

```
  {
    // Turn on LED7
    digitalWrite(pin_LED7, 1);
  }
  delay(100);
}

// Turn off all LEDs except LED0
void LED_OFF(void)
{
  digitalWrite(pin_LED1, 0);
  digitalWrite(pin_LED2, 0);
  digitalWrite(pin_LED3, 0);
  digitalWrite(pin_LED4, 0);
  digitalWrite(pin_LED5, 0);
  digitalWrite(pin_LED6, 0);
  digitalWrite(pin_LED7, 0);
}
```

12.5. Practice

12-1. Write a program that controls the buzzer from the measured value of the PSD sensor.
(If the distance value is less than 20 cm, makes a buzzer sound)

Chapter13

Bluetooth Communication

Learning Objectives

1. Understand the concept of Bluetooth
2. Understand how to communicate via Bluetooth

13. Bluetooth Communication

In this chapter, we will write a program that communicates with smar tphone using the Bluetooth of HBE ARDUINO KIT board.

13.1. What is Bluetooth?

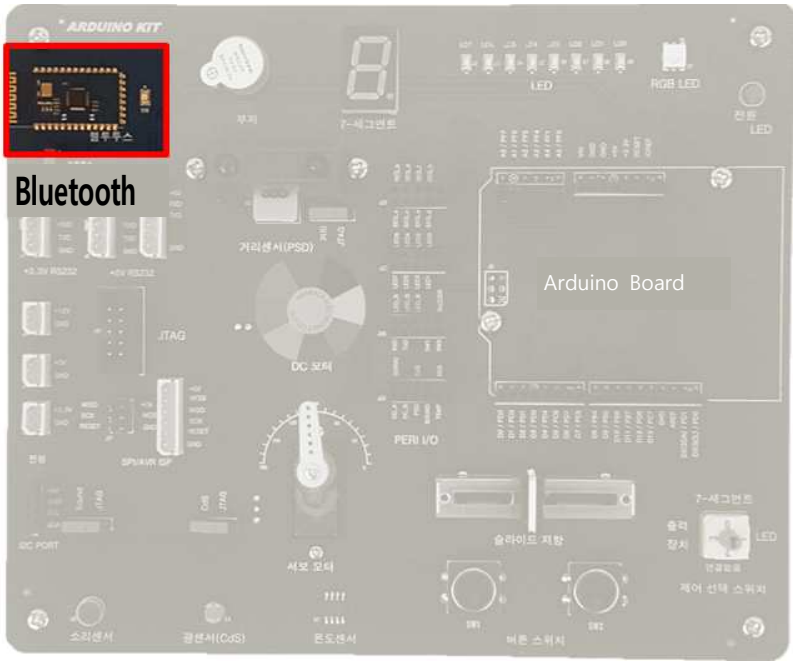
Bluetooth refers to a short-distance wireless technology that communicates data by connecting mobile devices such as mobile phone, laptop, and earphone/headphone to each other. It is mainly used when a low-power wireless connection is required in an ultra-short distance of around 10 meter. For example, a Bluetooth-enabled headset allows you to listen to music from a media player in your pocket without cable. Based on wireless technology research started by Ericsson, a mobile phone supplier in 1994, Bluetooth has been developed since 1998 through the 'Bluetooth SIG (Special Interest Group)' consisting of Ericsson, Nokia, IBM, Toshiba, and Intel. Since then, the number of Bluetooth SIG members has increased rapidly, reaching over 13,000 members worldwide

as of the end of 2010.

In the early days, the transmission speed of Bluetooth was only 1 Mbps. This speed was not good for transferring large data such as high-quality music or video. However, as time passed and a new version of Bluetooth appeared, the speed improved remarkably. The speed of Bluetooth 2.0 (2004) increased up to 3 Mbps and Bluetooth 3.0 (2009) up to 24 Mbps. In 2010, even Bluetooth 4.0 was developed, which reduced power consumption enough to be used for several years with a coin battery for a watch while maintaining a speed of 24 Mbps.

13.2. Location of Bluetooth

The following shows the location of Bluetooth.



[Figure 13-1] Location of Bluetooth

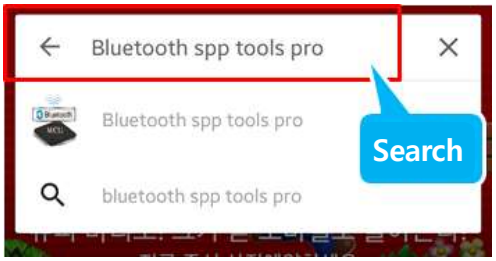
13.3. Install Serial Program for Bluetooth

The process of installing the program for serial communication with Android OS-based smartphones is explained.

Run the Play Store as shown below.

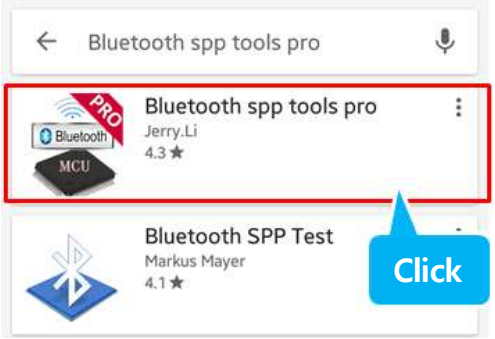
[Figure 13-2] Install Serial Program 1

After running, type "bluebootspp tools pro" as shown below and search.



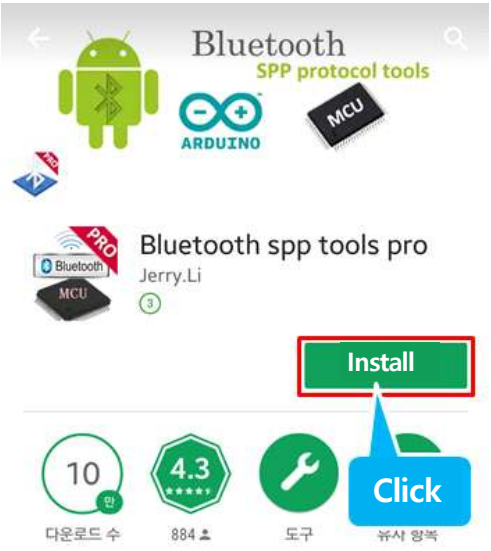
[Figure 13-3] Install Serial Program 2

When the search is complete, click 'Bluetooth spp tools pro' as shown b elow.



[Figure 13-4] Install Serial Program 3

Install Bluetooth spp tools pro by clicking as shown below.



[Figure 13-5] Install Serial Program 4

If it changes as shown below, the installation of the Bluetooth tools pro is completed.

[Figure 13-6] Install Serial Program 5

13.4. Bluetooth Program Experiment

EXPERIMENT 13-1

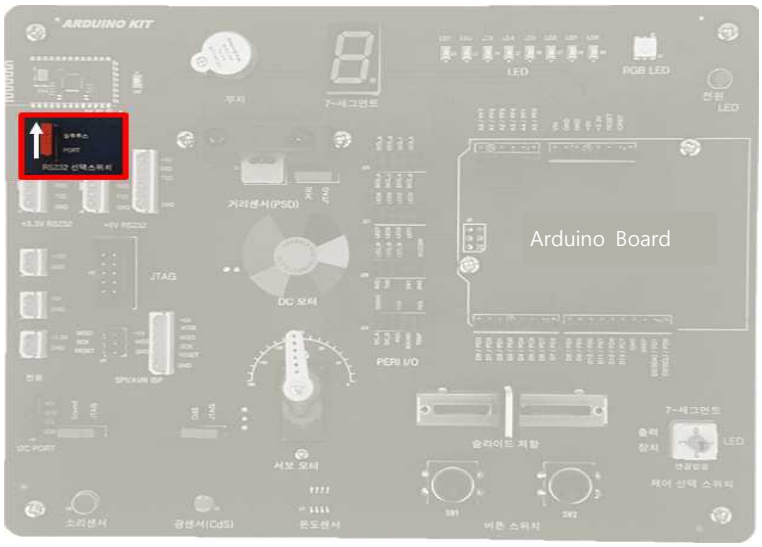
◉ Communicate with a smartphone using Bluetooth.

◆ **Write a program that sends messages having received from a smartphone back to the smartphone using Bluetooth.**

● The following table shows the pin map where Arduino Leonardo and Bluetooth on the ARDUINO KIT board are connected.

Arduino Pin Number	Bluetooth	Remarks
0	BT_RXD	RXD
1	BT_TXD	TXD

● For Bluetooth operation, turn the switch on the ARDUINO KIT boar d toward the arrow (upside) as shown below.



[Figure 13-7] Switch Operation for Bluetooth

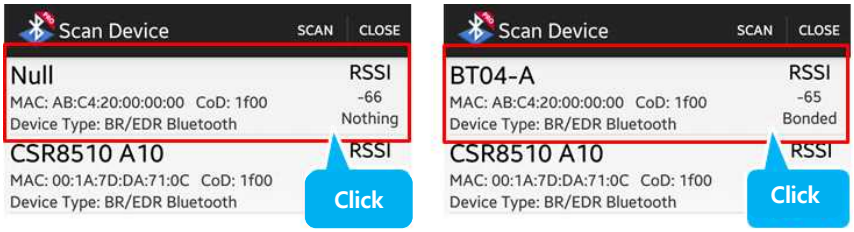
ARDUINO_KIT_BLUETOOTH.ino

```
void setup() {
  // Bluetooth Setting : Baud Rate 9600, Data 8bit, No Parity, Stop 1bit
  Serial1.begin(9600);  // same as Serial1.begin(9600, SERIAL_8N1)
}

void loop() {
  char recv;
  // Check if there is received data via Bluetooth
  if(Serial1.available() > 0)
  {
    // Save the received data to recv
    recv = Serial1.read();
    // Send received data back
    Serial1.write(recv);
  }
}
```

● Run the "Bluetooth spp pro" program.

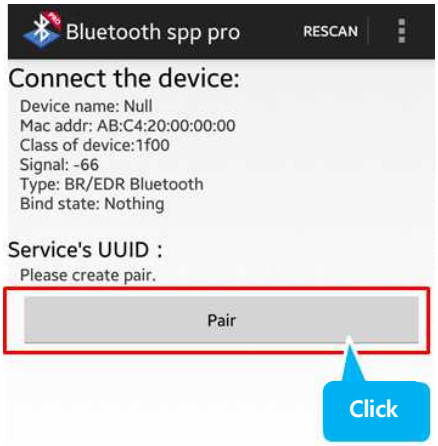
[Figure 13-8] Run Bluetooth Spp Pro

● When the Bluetooth spp pro program runs, the device is searched as shown below, and click the searched Null or BT04-A.



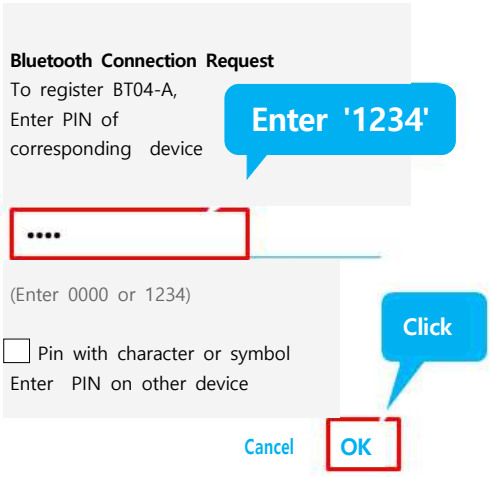
[Figure 13-9] Connect Bluetooth 1

● Click 'Pair' as shown below.



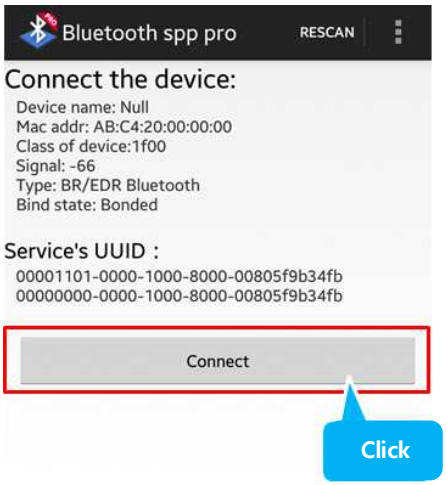
[Figure 13-10] Connect Bluetooth 2

● Enter the PIN as "1234" as shown below and click OK.



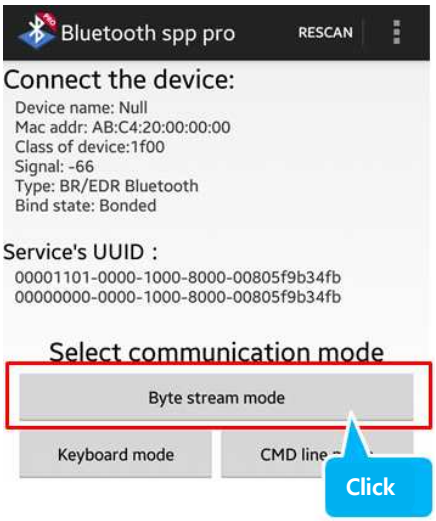
[Figure 13-11] Connect Bluetooth 3

● When pairing is complete, click 'Connect' as shown below.



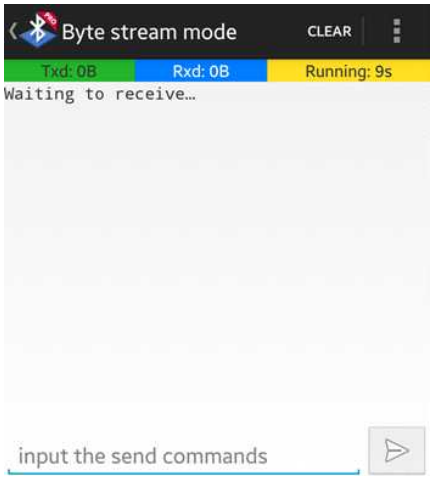
[Figure 13-12] Complete Bluetooth Connection

● Execute 'Byte stream mode' as shown below.



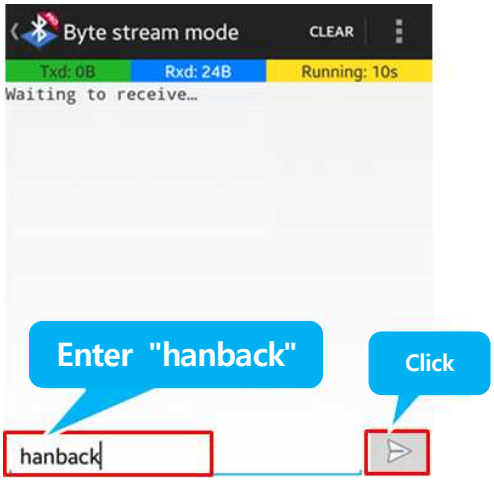
[Figure 13-13] Program Operation 1

● When it is executed as shown in the following figure, it is ready to communicate with Bluetooth.



[Figure 13-14]Program Operation 2

● Enter “hanback” as shown in the following figure and click .



[Figure13-15] Program Operation 3

● The entered characters are output as shown below

[Figure13-16] Program Operation 4

EXPERIMENT 13-2

◉ Control 7-segment using Bluetooth

◆ Let's write a program that controls 7-segment according to the data transmitted from the smartphone to Bluetooth.

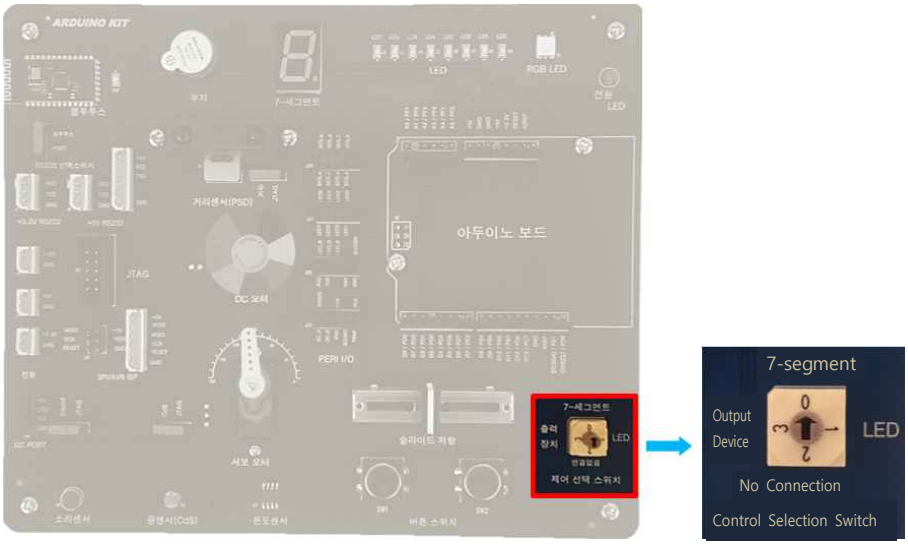
(Transmitted data is 0 ~ 9, and the corresponding value is displayed on 7-segment)

● The following table shows the pin map where Arduino Leonardo, Blu etooth and 7-segment on the ARDUINO KIT board are connected.

Arduino Pin Number	Bluetooth	Remarks
0	BT_RXD	RXD
1	BT_TXD	TXD

Arduino Pin Number	7-Segment	Remarks
5	A	Control selection switch must be set to 7-segment
6	B	
8	C	
9	D	
10	E	
11	F	
12	G	
13	H	

● For 7-segment control, set the control selection switch of ARDUINO KIT board to 7-segment.



[Figure 13-17] Set Control Selection Switch to 7-Segment

ARDUINO_KIT_BLUETOOTH_FND.ino

```
// Save the pin numbers of A, B, C, D, E, F, G, H in array
int pin_FND[8] = {5, 6, 8, 9, 10, 11, 12, 13};

void setup() {
  char  i;
  // Bluetooth Setting : Baud rate 9600, Data 8bit, no parity, stop 1bit
  Serial1.begin(9600);  // same as Serial1.begin(9600, SERIAL_8N1)
  // Set A ~ H pins as output pins
  for(i=0; i<8; i++)
  {
    pinMode(pin_FND[i], OUTPUT);
  }
}

void loop() {
```

```
  char recv;
  // Check if there is received data via Bluetooth
  if(Serial1.available() > 0)
  {
    // Save the received data in recv
    recv = Serial1.read();
    // Send received data back
    Serial1.write(recv);

    switch(recv)
    {
      // Process values from 0 to 9
      case '0': FND_out(0); break;
      case '1': FND_out(1); break;
      case '2': FND_out(2); break;
      case '3': FND_out(3); break;
      case '4': FND_out(4); break;
      case '5': FND_out(5); break;
      case '6': FND_out(6); break;
      case '7': FND_out(7); break;
      case '8': FND_out(8); break;
      case '9': FND_out(9); break;
      default:              break;
    }
  }
}

void FND_out(unsigned char data)
{
  switch (data) {
    case 0 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
```

```
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],1);
    break ;


  case 1 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],0);
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],0);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],0);
    break ;

  case 2 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],1);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],0);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],1);
    break ;

  case 3 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],0);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
```

```
    digitalWrite(pin_FND[0],1);
    break ;

  case 4 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],0);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],1);
    digitalWrite(pin_FND[0],0);
    break ;

  case 5 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],0);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],0);
    digitalWrite(pin_FND[0],1);
    break ;

  case 6 :
    digitalWrite(pin_FND[7],0);
    digitalWrite(pin_FND[6],1);
    digitalWrite(pin_FND[5],1);
    digitalWrite(pin_FND[4],1);
    digitalWrite(pin_FND[3],1);
    digitalWrite(pin_FND[2],1);
    digitalWrite(pin_FND[1],0);
    digitalWrite(pin_FND[0],1);
    break ;
```

```
    case 7 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],0);
      digitalWrite(pin_FND[3],0);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    case 8 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    case 9 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],0);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    default:
      break;
  }
}
```

● The method of operation after upload is complete is same as Experiment 13-1, excluding Bluetooth connection 2 and Bluetooth connection 3. And enter only one character among 0 ~ 9 characters of program operation 3, and check it.

13.5. Practice

13-1. Write a program that controls the buzzer using Bluetooth with a smartphone ('0' - buzzer off, '1' - buzzer on).

Chapter 14

Sound Sensor

Learning Objectives

1. Understand the concept of sound.
2. Understand how to measure the sound.

14. Sound Sensor

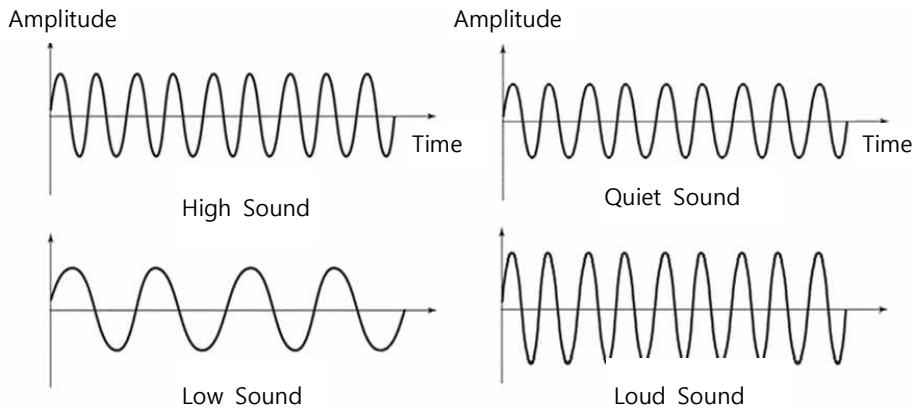
In this chapter, we will write a program that controls 7-segment using the sound sensor of HBE ARDUINO KIT board.

14.1. Definition and Types of Sound

Sound is a kind of wave generated by the vibration of an object or the flow of air. The sound produced by the vibration of an object is called solid sound, different sounds are produced depending on the shape and type of vibrating object.

The sound produced by the flow of air is called air current, different sounds are produced depending on the speed and shape of the air current. Loudness, Pitch, Timbre are three elements of sound. The loudness of a sound is determined by the amplitude of the sound. The higher the amplitude, the louder the sound, and the smaller the amplitude, the quieter the sound. In general, the range that humans can

hear is called Audible Frequency, and it is approximately a range of frequencies between 16 and 20000 Hz.

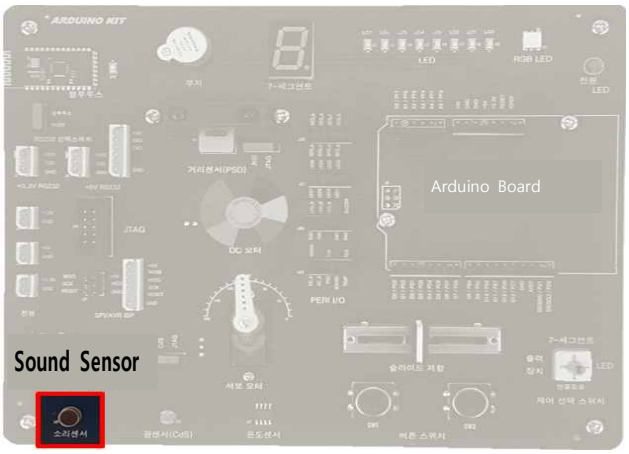


[Figure 14-1] Sound Comparison by Frequency and Amplitude

As shown in the figure above, the left coordinate is divided into high and low sound according to frequency, and the right coordinate is divided into loud and quiet sound according to amplitude.

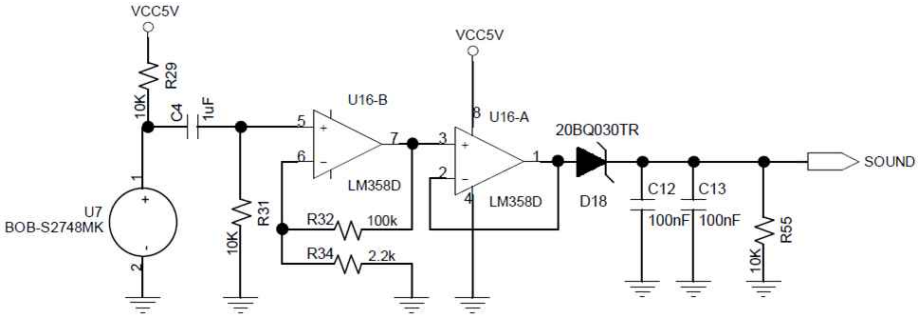
14.2. Location and Circuit of Sound Sensor

The following shows the location of the sound sensor.



[Figure 14-2] Location of Sound Sensor

The following shows the circuit diagram of the sound sensor.



[Figure 14-3] Sound Sensor Circuit

The signal received from the sound sensor circuit diagram is amplified and output through Zener Diode. If no sound is generated, 0 is output to SOUND, and if sound is generated, 1 is output.

14.3. Sound Sensor Program Experiment

EXPERIMENT 14-1

◉ Output the value measured by the sound sensor to the serial monitor.

◆ **Write a program that measures the value output from the sound sensor and outputs it to the serial monitor.**

● The following shows the pin map where the Arduino Leonardo and s ound sensor on the ARDUINO KIT board are connected.

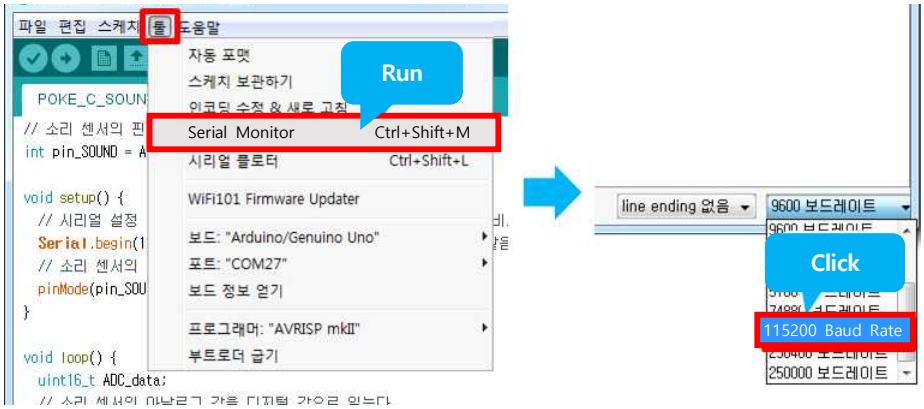
Arduino Pin Number	Sound Sensor	Remarks
A2	SOUND	

ARDUINO_KIT_SOUND.ino

```
// Save the pin number of the sound sensor
int pin_SOUND = A2;

void setup() {
  // Serial Setting : Baud Rate 115200, Data 8bit, no parity, Stop 1bit
  Serial.begin(115200);  // same as Serial.begin(115200, SERIAL_8N1)
  // Set the pin of the sound sensor as an input pin
  pinMode(pin_SOUND, INPUT);
}

void loop() {
  unsigned int  ADC_data;
  // Read analog value of sound sensor as digital value.
  ADC_data = analogRead(pin_SOUND);

  // Outputs "ADC Data : " serially
  Serial.print("ADC Data : ");

  // Output the measured value of sound sensor serially
```

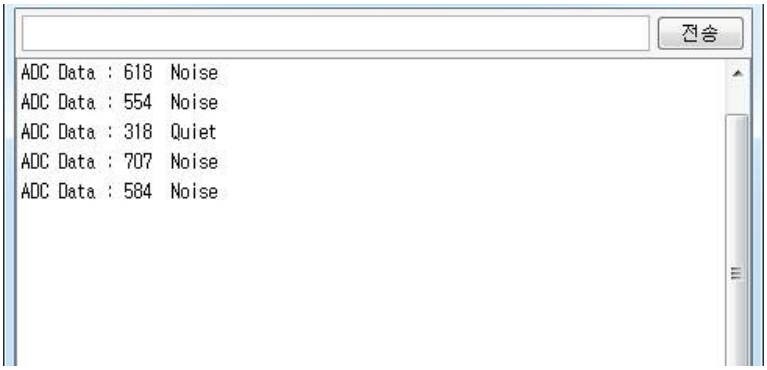
```
  Serial.print(ADC_data);
  if(ADC_data < 512)
  {
    // No sound detection
    Serial.println("   Quiet");
  }
  else
  {
    // Sound detection
    Serial.println("   Noise");
  }
  delay(500);
}
```

● After uploading is completed, execute the serial monitor in the tool as shown below to check the operation with the serial monitor. After execution, set the baud rate to 115200.



[Figure 14-4] Execute Serial Monitor and Set Baud Rate

● Outputs the ADC value and sound detection status to the serial monitor as shown below.



[Figure 14-5] Check Operation

EXPERIMENT 14-2

◉ 7-segment Control using Sound Sensor

◆ **Write a program that controls the 7-segment according to the value measured by reading the sound sensor.**

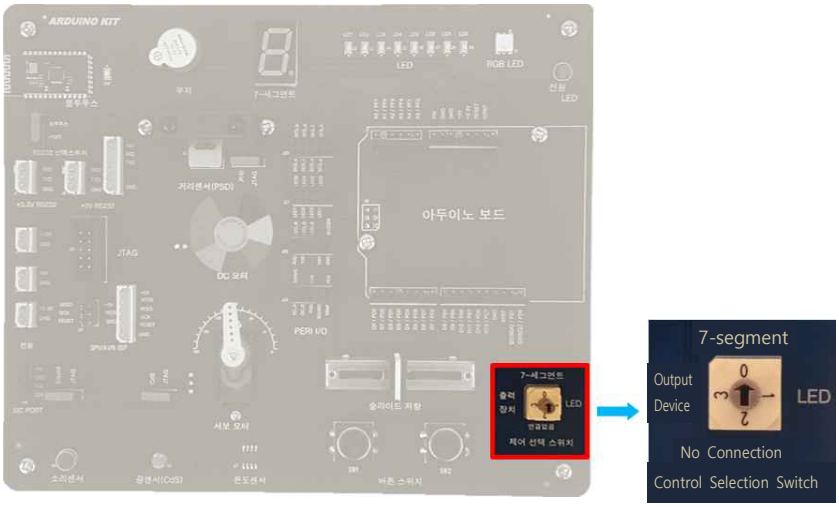
(If the measured value is 512 or more, 8 is output to the 7-segment, and if it is less than 512, 0 is displayed)

● The following table shows the pin map where Arduino Leonardo, sou nd sensor and 7-segment on the ARDUINO KIT board are connected.

Arduino Pin Number	Sound Sensor	Remarks
A2	SOUND	

Arduino Pin Number	7-segment	Remarks
5	A	The control selection switch must be set to 7-segment
6	B	
8	C	
9	D	
10	E	
11	F	
12	G	
13	H	

● For 7-segment control, set the control selection switch on the ARDUINO KIT board to 7-segment.



[Figure 14-6] Set Control Selection Switch to 7-segment

ARDUINO_KIT_SOUND_FND.ino

```
// Save the pin number of the sound sensor
int pin_SOUND = A2;

// Save the pin numbers of A, B, C, D, E, F, G, H in an array
int pin_FND[8] = {5, 6, 8, 9, 10, 11, 12, 13};

void setup() {
  char i;
  // Set the pin of the sound sensor as an input pin
  pinMode(pin_SOUND, INPUT);
  // Set the pin of A ~ H as an output pin
  for(i=0; i<8; i++)
  {
    pinMode(pin_FND[i], OUTPUT);
  }
}
```

```
void loop() {
  int ADC_data;
  // Read the analog value of the sound sensor as a digital value
  ADC_data = analogRead(pin_SOUND);

  if(ADC_data < 512)
  {
    // Display 0 in 7-segment
    FND_out(0);
  }
  else
  {
    // Display 8 in 7-segment
    FND_out(8);
  }
  delay(500);
}

void FND_out(unsigned char data)
{
  switch (data) {
    case 0 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],0);
      digitalWrite(pin_FND[5],1);
      digitalWrite(pin_FND[4],1);
      digitalWrite(pin_FND[3],1);
      digitalWrite(pin_FND[2],1);
      digitalWrite(pin_FND[1],1);
      digitalWrite(pin_FND[0],1);
      break ;

    case 8 :
      digitalWrite(pin_FND[7],0);
      digitalWrite(pin_FND[6],1);
```

```
        digitalWrite(pin_FND[5],1);
        digitalWrite(pin_FND[4],1);
        digitalWrite(pin_FND[3],1);
        digitalWrite(pin_FND[2],1);
        digitalWrite(pin_FND[1],1);
        digitalWrite(pin_FND[0],1);
        break ;

      default:
        break;
   }
}
```

14.4. Practice

14-1. Write a program that controls the LED by the measured value of the sound sensor.
(If the measured value is more than 300, LED0~7 turns on, and if less than 300, LED0~7 turns off).

Arduino Leonard

Learning with HBE ARDUINO KIT

Written by : Technical Research Center, Hanback Electronics Co., Ltd.
Published by : Hanback Electronics Co., Ltd.

Address : #518, Yuseong-daero, Yuseong-gu, Daejeon, Republic of Korea
Tel : +82-42-610-1111 / Fax : +82-42-610-1199
Web Site : http://www.hanback.com

V2.0.3