CODING ARRAY STARTER KIT

THE BEGINNER'S GUIDE
1st Editoin

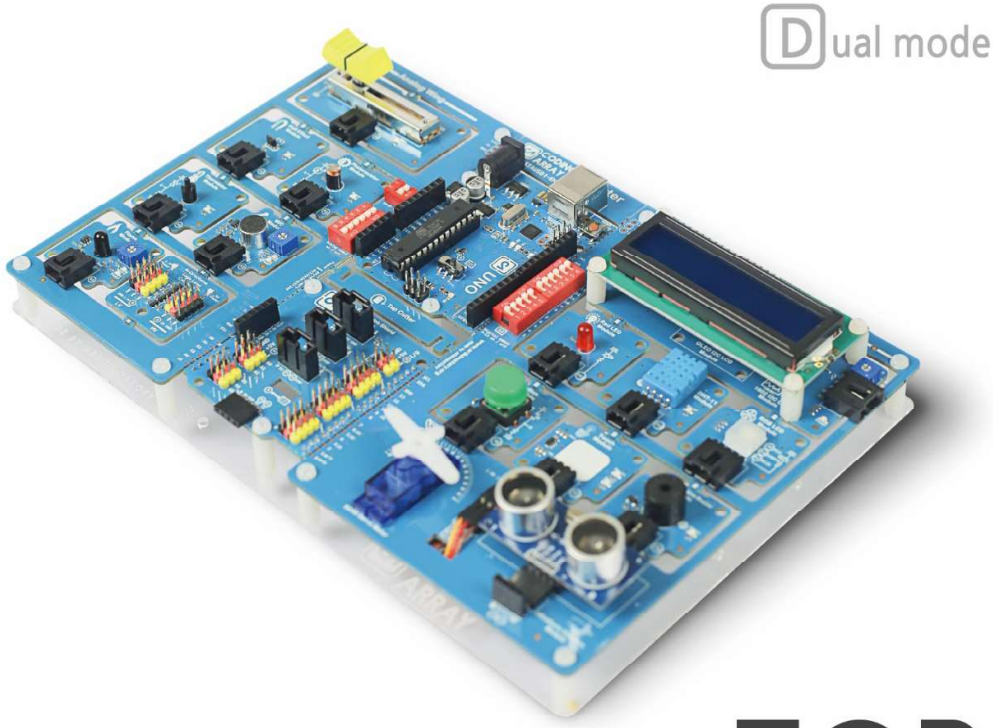
Easier

Faster

Safer

Simple

Dual mode



FOR ARDUINO

Made By Arduino Story

CODING ARRAY KIT

Figure 1

Coding array kit is training toys designed to provide easy, fast and safe access to digital computing without the need for electrical engineering knowledge.

- • Supplied from the coding array kit "Arduino Story" used in this guidebook.
- • This guidebook is version 0.1 modified, translated, and written by Arduino story and owns the Arduino Story.

CODING ARRAY STARTER Kit for Arduino

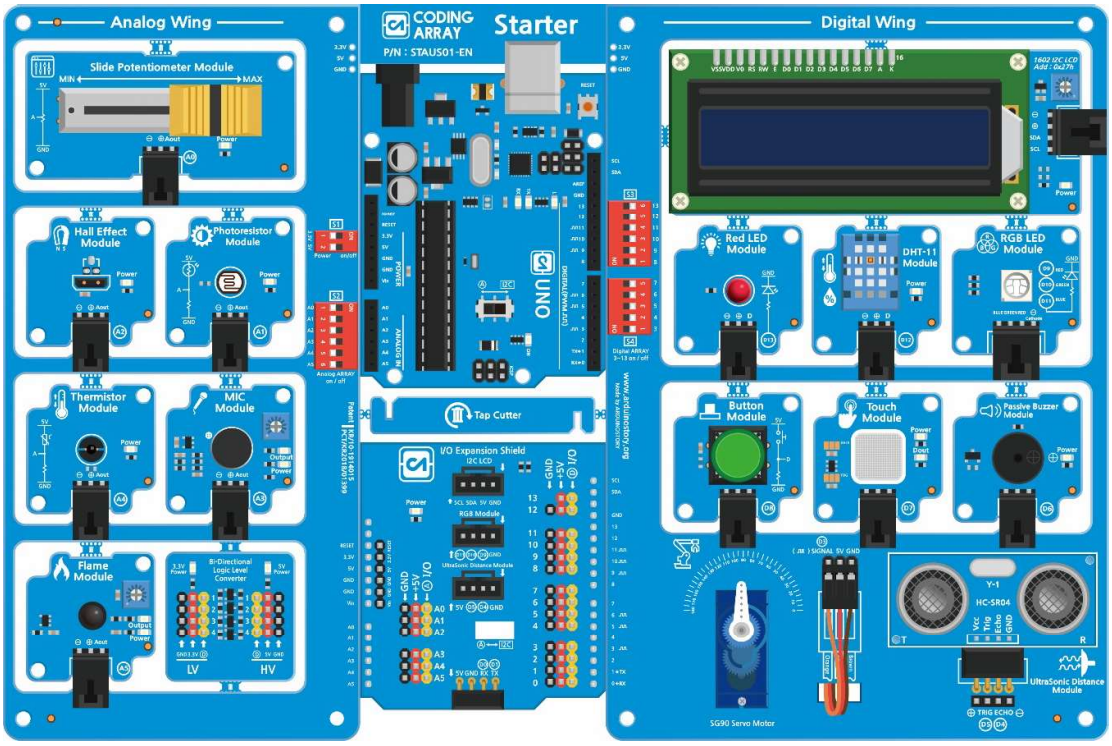


Figure 2

A suitable starter kit for training Arduino beginner is designed to be used without having to configure a separate circuit for 15 input and output units around Arduino Uno.

The modules are arranged in an analog wing on the left side of the Uno Board and a digital wing on the right You can use and control modules at the same time as you upload sketch files.

After learning, modules can be disconnected and have the advantages to use in the project.

CAUTION! Do not disconnect the module using the Tap Cutter before board testing and learning is complete.

Internal Power Circuit Diagram of CODING ARRAY STARTER Kit for Arduino

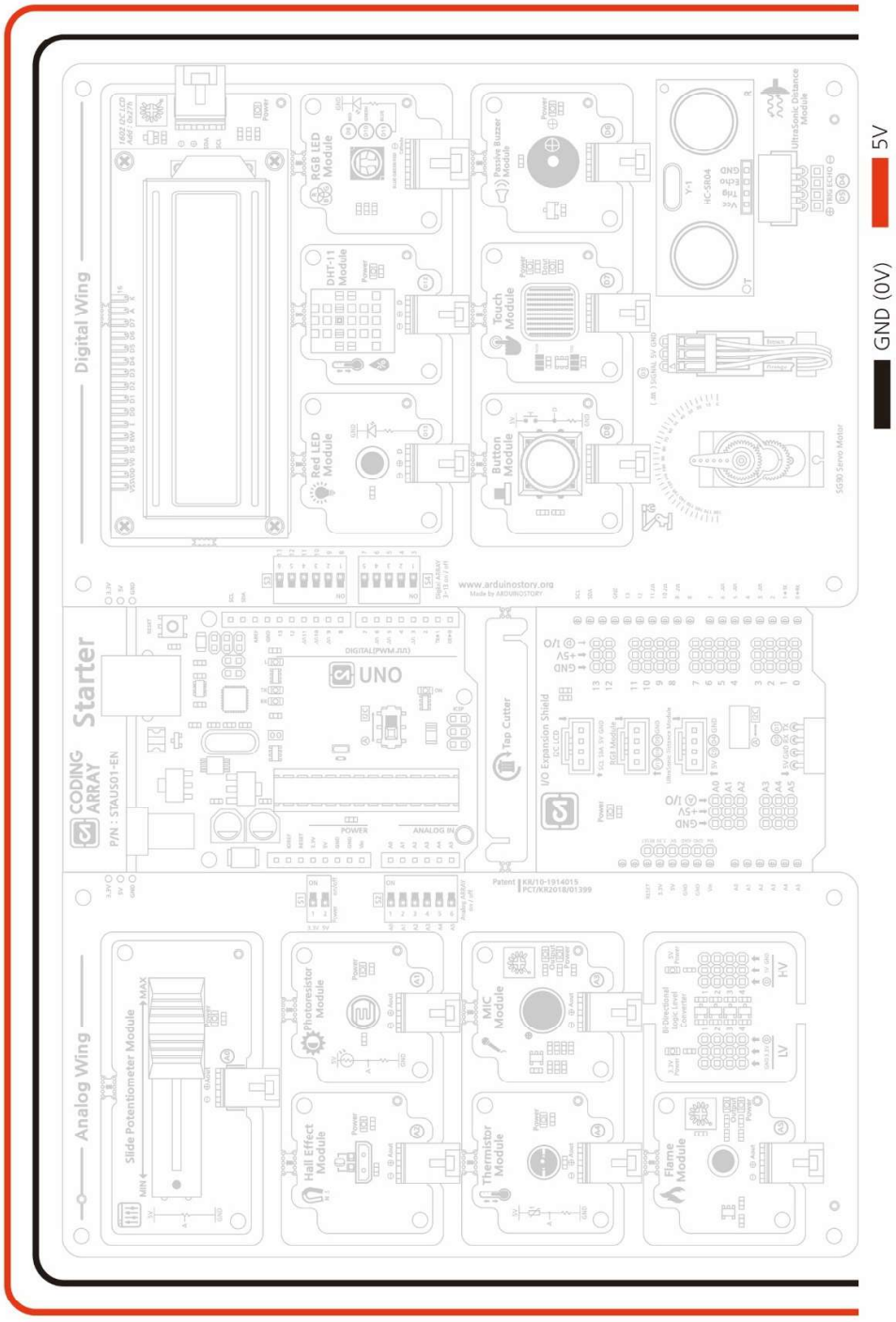


Figure 3

Coding Array Kit PINOUT

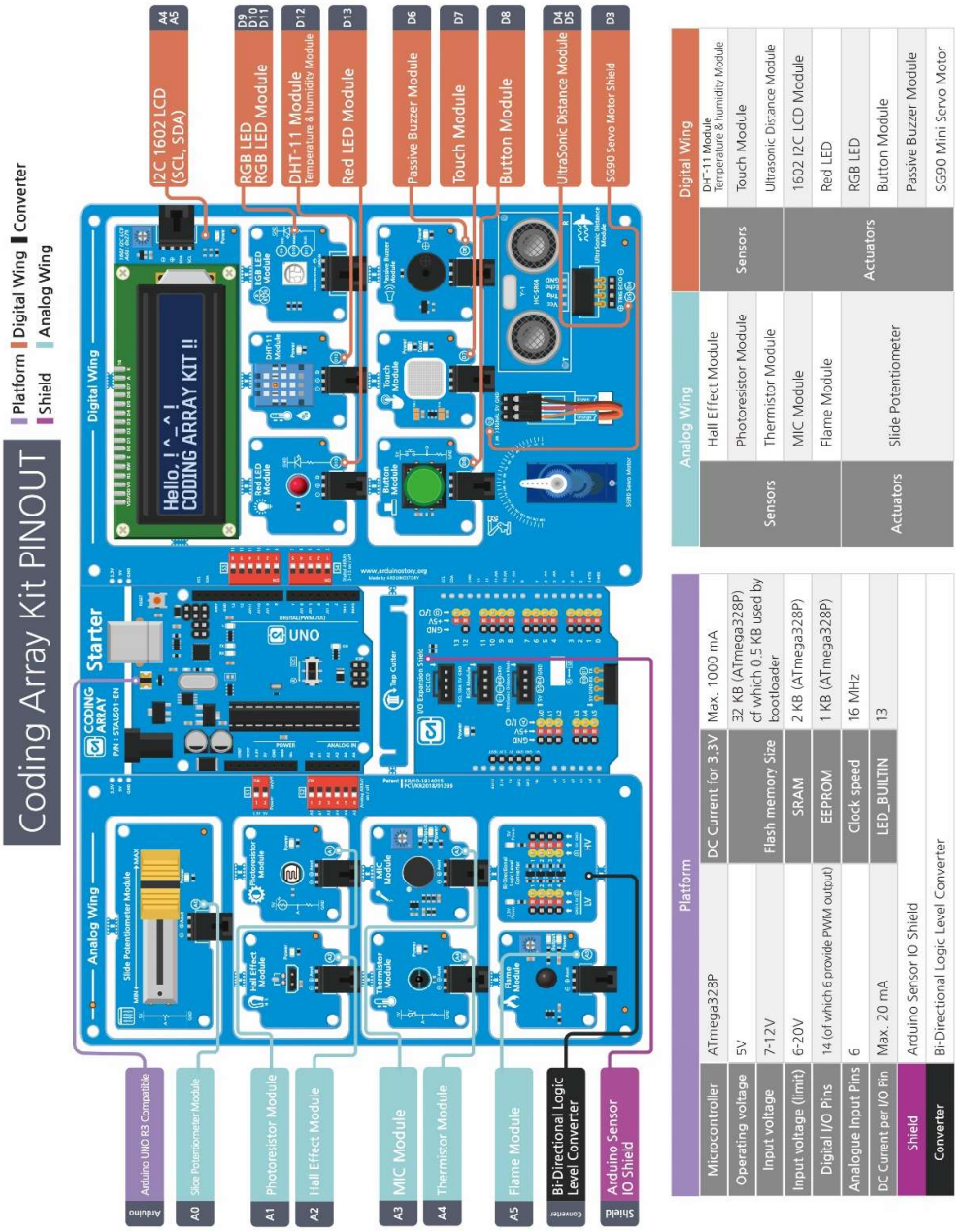
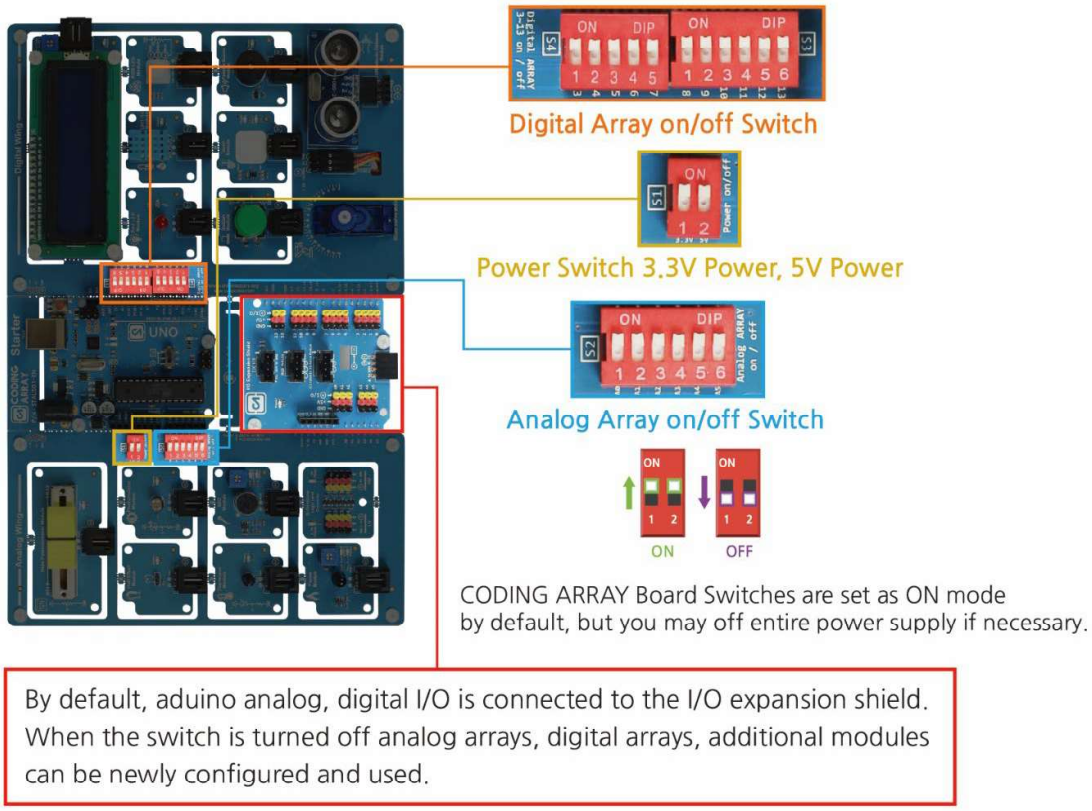
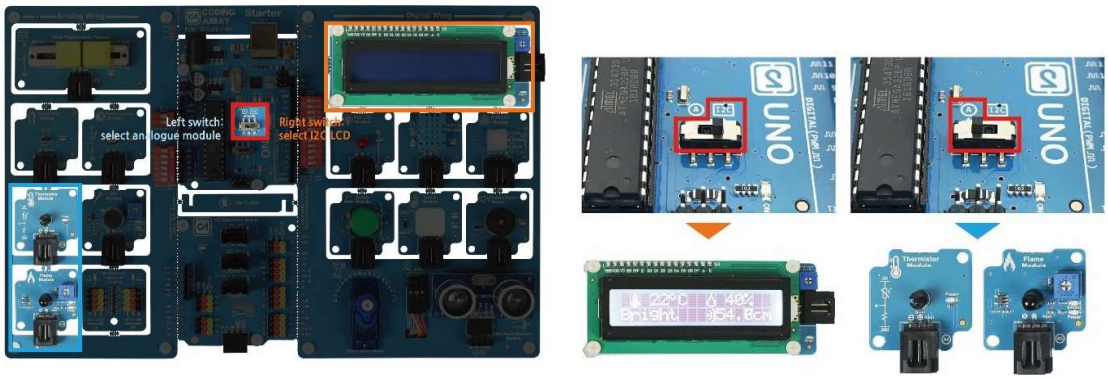


Figure 4

CODING ARRAY Kit's another advantage





When shipping the product, it will be shipped to the 'I2C LCD Selection' position. The switch must be moved as shown in the Figure to use the Analog Temperature Sensor (A4) and Analog Fire Detection Sensor (A5) modules.

To ensure the use of the coding array kit without damage, observe the following

Figure 5

INDEX

Chapter1. Prepare

- Install Arduino IDE and download example code

Install the IDE installation, an integrated development environment, for use the Arduino board, and check use methods and function of the IDE. Learn how to download and open an example file for using a coding array kit..

1. Arduino?
2. Arduino IDE (Integrated Development Environment) Download
3. Arduino Software IDE Open
4. Menu Bar Function
5. For Windows
6. For MAC

1. Arduino?

Arduino is a typical Microcontroller Unit (MCU) of an open source base (hardware + software) that is easy to use.

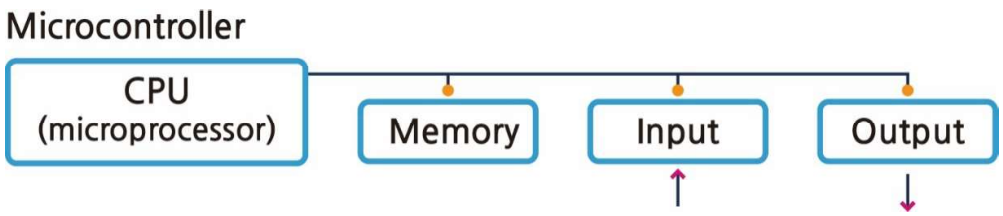


Figure 6

Microcontrollers are made of a single chip to perform a series of tasks in the CPU and memory, and to send the results to the I/O port with an electrical signal.

Arduino was designed in 2005 by Professor Massimo Banzi, who taught interactive design called "physical computing" at Italy's Interaction Design Institute Ivrea (IDII) that taught art and IT. Professors and students used Parllax's Basic Stamp program, but it was expensive and inconvenient to use. The programming language developed by MIT is open-source and allows even those who are not well-programmed to implement graphics in simple codes, starting with Arduino's idea of how to make it easy for students who are not familiar with hardware to control.

The most used and documented board of Arduino is Uno, the best board to start Arduino. Uno means one in Italian and has been chosen to celebrate the release of Arduino IDE 1.0. The USB communication chip is built into the main processor, so it can be connected directly to the PC's USB, and has 14 digital input/output pins (including 6 PWM pins) and six analog input pins.

Table **1**

Microcontroller	ATmega328P
Dynamic voltage	5V
Input voltage (recommended)	7-12V
Input voltage (limit)	6-20V
Digital input/output pin	14 ea (including 6 PWM pins)
Analog input pin	6 ea
DC current per input/output pin	Max. 20 mA
DC current of 3.3V pin	**Max. 1000 mA** (150mA Existing Uno)
Flash memory size	32 KB (ATmega328P) 0.5 KB Include Bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock speed	16 MHz
LED_BUILTIN	13

< Uno-compatible board details used in array kits >

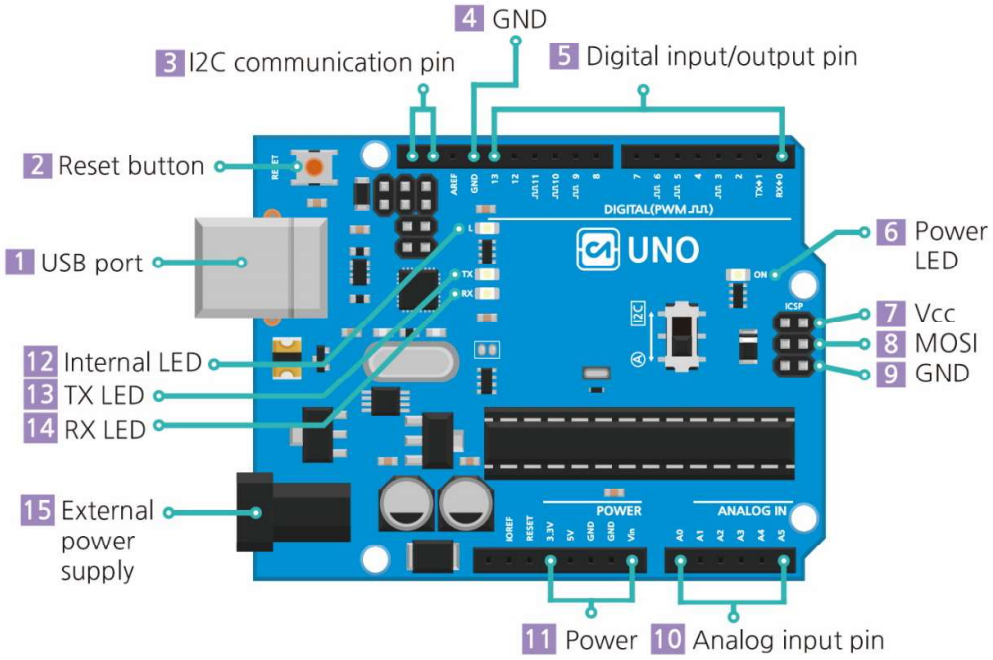


Figure 7

Table 2

1	USB port	It receives a 5V power supply from the computer and performs serial communication.
2	Reset button	Restart button
3	I2C communication pin	SDA, SCLpins for I2C communication
4	GND	Grouding
5	Digital input/output pin	14 Digital input/output pin (including 6 PWM pins)
6	Power LED	Illuminates when power is supplied.
7	Vcc	Grouding
8	MOSI	Master output.
9	GND	5V Power supply
10	Analog input pin	6 Analog pin
11	Power	3.3V, 5V, Grounding, External power supply
12	Internal LED	Connect digital pin 13
13	TX LED	Indicates that the FTDI chip sends data to the computer.
14	RX LED	Indicates that the FTDI chip receives data from the computer.
15	External power supply	7V to 12V DC power supply. (Battery Pack)

2. Arduino IDE (Integrated Development Environment) Download

Step1. Connecting a Board with Computer

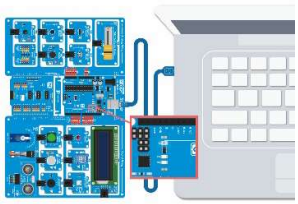


Figure 8

Connecting the computer and Arduino with a USB cable allows users to receive 5V power supply to operate the Arduino as well as send and receive data. (However, separate power supplies shall be used for modules requiring a power supply greater than 5 V.)

Step2. Access the Arduino Software Download Site from the Internet

Figure 1

https://www.arduino.cc/en/Main/Software

step3. Free download of Arduino Software IDE

Figure 2

Select and download the installation file that matches your computer's operating system.

Figure 3

Windows and Mac OS users run the program by selecting Download without Donation or Donation.

Figure 4

When a compressed file such as "ardino-1.8.5-lunux64.tar.xz" is downloaded, Linux users run a terminal to decompress as follows: Run the install.sh file in the extracted folder.

tar xvfz [File name]

3. Arduino Software IDE Open

Let's open the Arduino IDE installed on the computer and look at the configuration.

IDE is largely divided into menu bars, tool bars, code windows, and console windows.

Figure 5

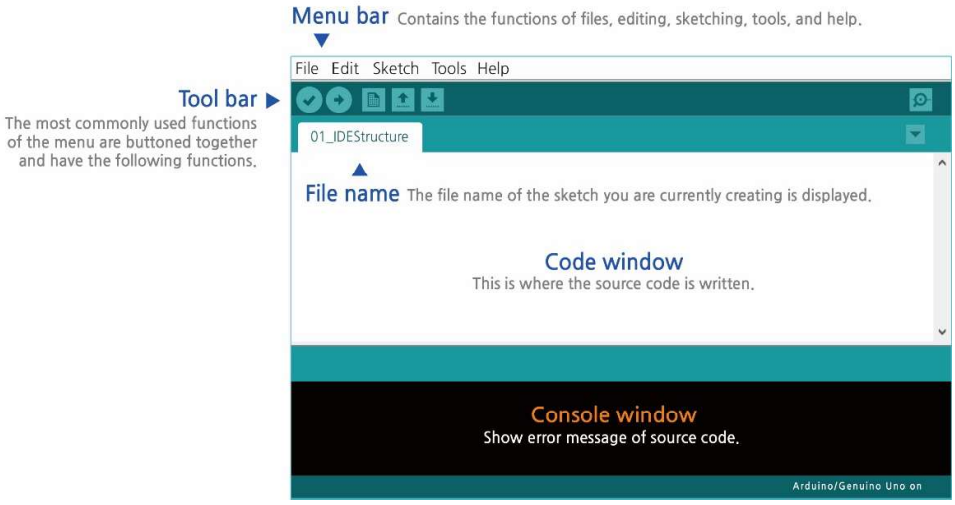


Figure 6

1) Menu bar: Contains the functions of files, editing, sketching, tools, and help.

2) Tool bar: The most commonly used functions of the menu are buttoned together and have the following functions.

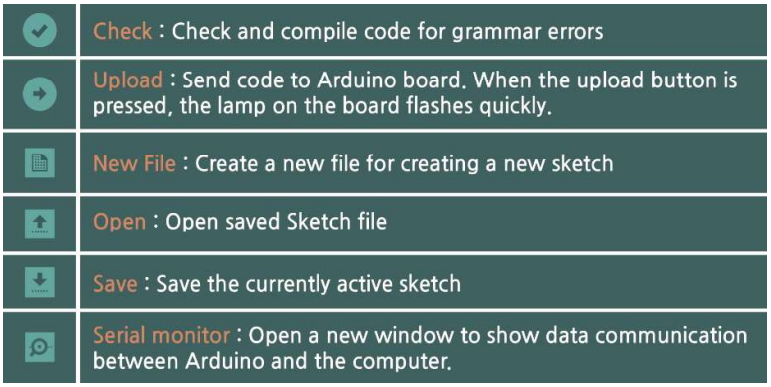


Figure 7

3) File name: The file name of the sketch you are currently creating is displayed.

4) Code window: This is where the source code is written.

5) Console window: Show error message of source code.

4. Menu Bar Function Overview

File	Editing	Sketch	Tool	Help
New file	When creating a new sketch			
Open	When opening a saved sketch			
Open Recent Files	View and open a list of recently opened sketches			
Sketchbook	When you open one of the sketches stored in the Sketchbook folder			
Example	When opening a built-in example, library example, and so on			
Close	Exit sketch			
Save	Save Sketch File			
Save As	Save Sketch Files As other name			
Page Setting	Page Setting window for printing			
Print	Print the current sketch			
Setting	Sketchbook folders, font sizes, compiler warnings, etc.			
Exit	Exit sketch at once, and open all when re-run			

Table 3

File	Editing	Sketch	Tool	Help
Cancel	Revert to Previous			
Redo	Re-run Cancel			
Crop	Cut Selection			
Copy	Copy Selection			
Copy for Forum	form for posting sources on Arduino Formal Forum notice			
Copy to HTML	Copy to Clipboard as HTML when you want to upload to a webpage			
Paste	Paste cut or copied part to cursor position			
Select All	Select All Code			
Go to line...	Shortcut to a specific code line			
Add/Delete Annotations	Process Annotations // When Displaying or Deleting Annotations			
Add indentation	When adding indentation			
Reduce indentation	When reducing indentation			
Increase Font Size	When increasing the size of the editor's font (Ctrl + over the mouse wheel)			
Decrease Font Size	When reducing the size of the editor's font (Ctrl + below the mouse wheel)			
Find	When looking for a specific character			
Find next	When you grow a specific character and find it in a later sentence			
Find previous	When you grow a specific character and find it in this previous sentence			

table 4

File	Editing	Sketch	Tool	Help
Check/comfile	Check for code errors and comfile			
Upload	Send Code to Arduino Board			
Upload using a programmer	Overwrite to a bootloader on the board			
Export comfiled binary	Save as .hex file			
Show Sketch Folder	Open Current Sketch Folder			
Include Libraries	Use the #include to add librarie			
Add File	Add Source Files to Sketch			

table 5

File	Editing	Sketch	Tool	Help
Auto Format	Formatting code to look good			
Archive Sketches	Keep a copy of your sketch in a .zip file			
Modify Encoding & Refresh	Reduce encoding differences between editor and other editors			
Serial monitor	where you see the communication data between Arduino and the computer.			
Serial plotter	Graphical representation of communication data between Arduino and computer			
WiFi101 Firmware Updater	Send Code to Arduino Board			
Board:"Arduino/Genuino Uno"	Select the board you are using			
Port	Choose the computer port to which the Arduino board is connected			
Get board information				
Programmer : "AVRISP mkll"	Used to program boards or chips			
Burning a bootloader	When using IDE for a MCU other than Arduino			

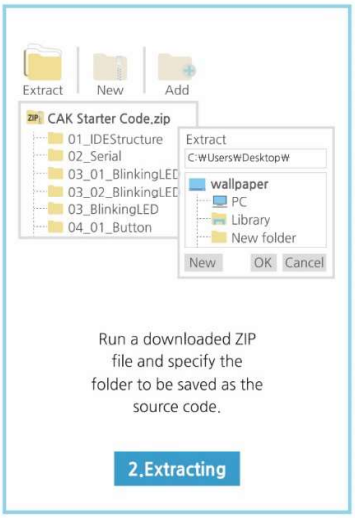
table 6

File	Editing	Sketch	Tool	Help
Getting started				
Environment				
Trouble shooting				
Reference				
Galileo Help				
Getting started				
Trouble shooting	*Access and help with the various documents and other descriptions*			
Edison Help	*provided with Arduino IDE at www.arduino.cc*			
Getting started				
Trouble shooting				
Find in Reference				
FAQ				
Visit Arduino.cc				
About Arduino				

5. For Windows

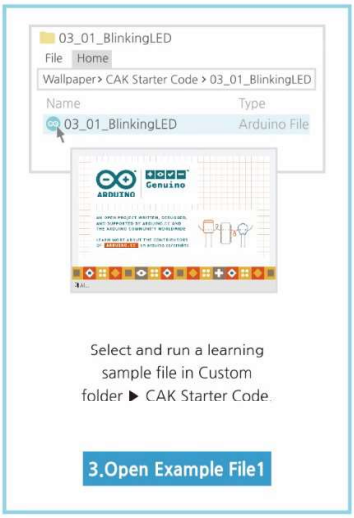
www.arduinostory.org
DOWNLOADS -> SOURCE CODE
Access the link above,
download the sample file

1.Downloading an example file



Run a downloaded ZIP
file and specify the
folder to be saved as the
source code.

2.Extracting

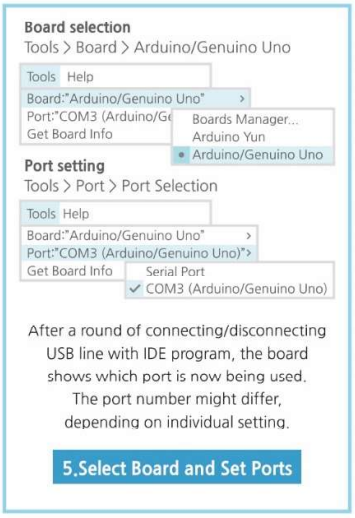


Select and run a learning
sample file in Custom
folder ▶ CAK Starter Code

3.Open Example File1

Once you execute your sample file,
Sketch file is opened as follows.
Please proceed with desired board
selection and port setting on the
top menu bar 'Tools' before
uploading Sketch file into the board.

4.Open Example File2

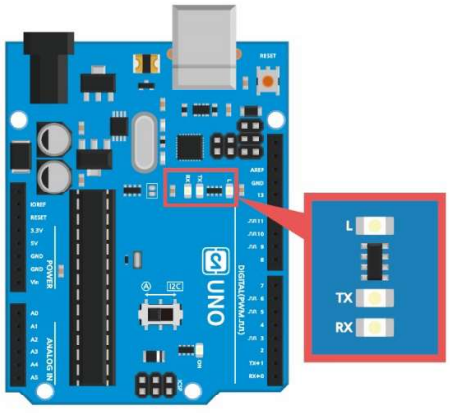


After a round of connecting/disconnecting
USB line with IDE program, the board
shows which port is now being used.
The port number might differ,
depending on individual setting.

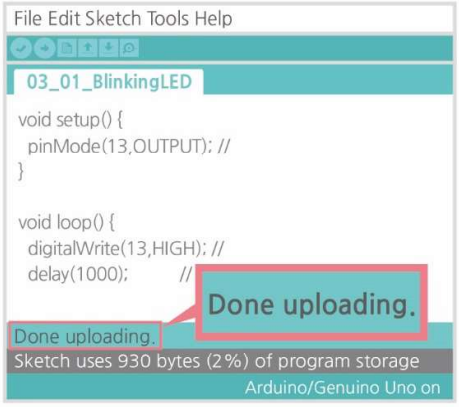
5.Select Board and Set Ports

Click to upload Sketch,
then you may be able to view the outcome.

6.Open Example File3

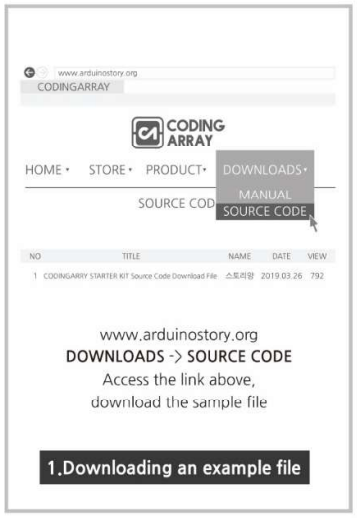


During upload, the TX.RX.LED on the
Unoboard flashes quickly and then turns off.



If a message is displayed at the top
of the IDE console window, it means
that the upload was successful.

6. For MAC



www.arduinostory.org
DOWNLOADS -> SOURCE CODE
Access the link above,
download the sample file

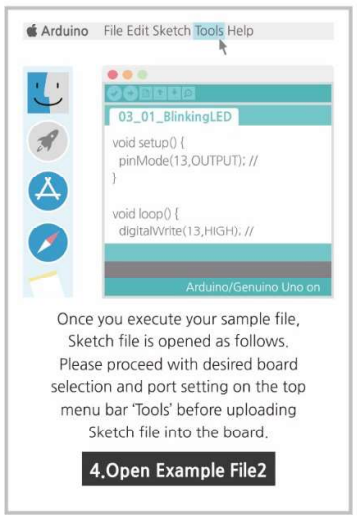
1.Downloading an example file

Specify the folder to be saved
as the source code.

2.Extracting

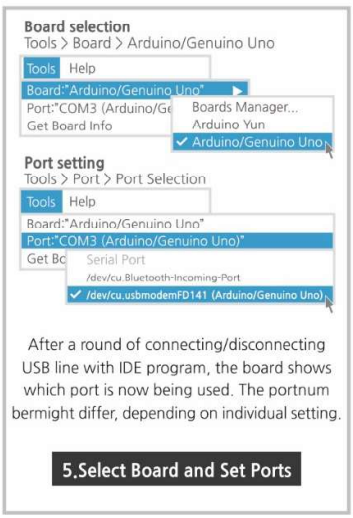
Select and run a learning sample
file in Custom folder ▶ CAK Starter Code.

3.Open Example File1



Once you execute your sample file,
Sketch file is opened as follows.
Please proceed with desired board
selection and port setting on the top
menu bar 'Tools' before uploading
Sketch file into the board.

4.Open Example File2

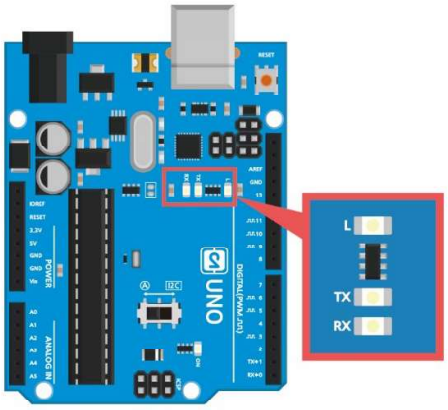


After a round of connecting/disconnecting
USB line with IDE program, the board shows
which port is now being used. The portnum
bermight differ, depending on individual setting.

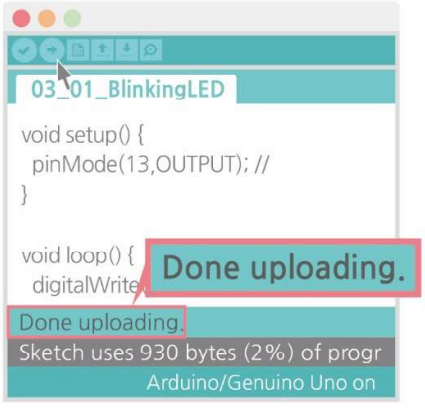
5.Select Board and Set Ports

Click to upload Sketch,
then you may be able to view the outcome.

6.Open Example File3



During upload, the TX.RX.LED on the
Unoboard flashes quickly and then turns off.



If a message is displayed at the top
of the IDE console window, it means
that the upload was successful.

Chapter2.Prepare

Shows the characteristics of the module used in the example file supplied with the board, as well as the default usage code and results.

Example contents

1. IDE structure – setup and loop

2. Serial communication

3. Turn LED on and off with digital output

4. Read button switch values with digital input

5. Change RGB LDE color using digital output and PWM function

6. Implementing moods, etc. with capacitive touch sensors

1. IDE structure – setup and loop

Running Arduino IDE creates a sketch file consisting of two parts: void setup and void loop.

IDE structure – setup and loop

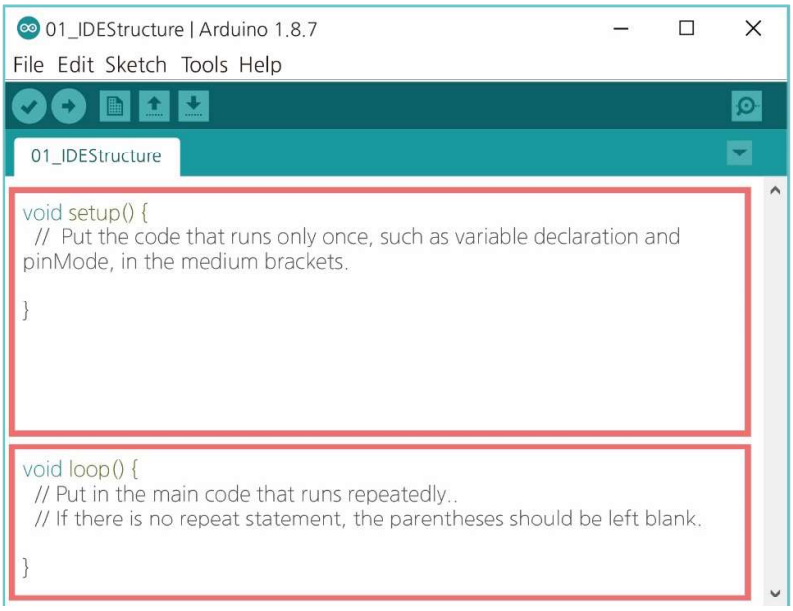


Figure 8

- void setup ( ) { }
  - Runs only once when the program starts.
  - Declares variables between medium brackets, includes pinMode settings, etc.

- void loop ( ) { }
  - The main content of a program that runs repeatedly between medium brackets.
  - Even if there are no repeat statements, the medium brackets shall remain blank.

CAK Starter Code > 01_ IDEStructure

Let's run the IDE on Arduino and open an example above.

There are two main screens, void setup { } and void loop { }.

Powering up Arduino will execute the previously uploaded code, which can act as a clearing of the previous code.

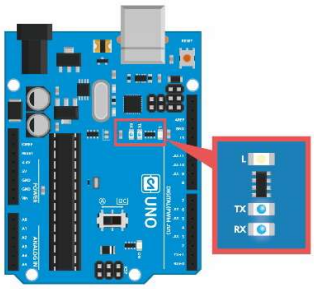
```
1  void setup() {
2    //  Put the code that runs only once, such as variable declaration and pinMode, in the medium brackets.
3
4  }
5
6  void loop() {
7    // Put in the main code that runs repeatedly..
8    // If there is no repeat statement, the parentheses should be left blank.
9
10 }
```

Precautions for creating a sketch file

- When you create a sketch file, make sure to write case-sensitive characters.
- At the end of the command statement, a semicolon (;) should be added at all times.
- Comments are part of the program that does not affect the program
- One-line annotation (// content) and multi-line annotation (/* content */).

2. Serial Communications

Serial Communications



The USB cable allows data to be exchanged between Arduino and the computer, called serial communication (serial communication, UART communication). When uploading sketch files, you can see the RX (:data received) and TX (data sent) lights on the board flickering quickly. Since Arduino's digital No. 0 pin (RX) and Digital No. 1 pin (TX) are used for serial communication, use Pin 2 to connect modules to the digital pin.

Figure 9

Serial communications make it easy to debug the computer to give data to Arduino, check the program results value of Arduino through the computer window, or find and correct errors in the program.

After uploading the sketch file, touch the same icon to the right to display a serial monitor pop-up window.

Caution: do not open the serial window during program upload.

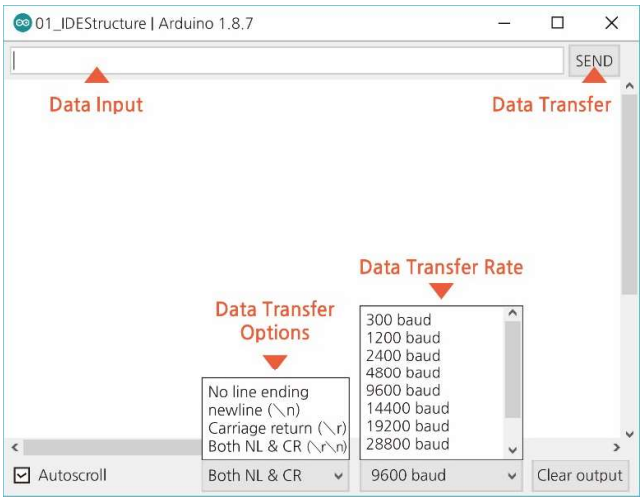


Figure 10

CAK Starter Code > 02_Serial

This sketch shows how to print messages entered on a computer into a serial window.

```
1  void setup() {
2    Serial.begin(9600);              // Prepare serial communication. Set the communication speed to 9600.
3
4  }
5
6  void loop() {
7    Serial.println("Hello Coding Array Kit ~!");    // Print Hello Coding Array Kit~! in the serial window.
8    delay(1000);                                    //Wait for 1000 milliseconds (=1 second).
9  }
```

Serial.begin (communication speed); the Baud rate is between 300 and 115200.
It is usually set to 9600.

Serial.println (value, format); to change the line after printing the value on the serial monitor.
Values can contain both letters and numbers to be printed.
However, the letters must be in between ' or ' .
》Serial.println("A") outputs A
The format specifies an integer or decimal number.
》Serial.println (3.14159, 0) outputs 3
》Serial.println (3.14159, 2) outputs 3.14.

Serial.print (value, format); add values to the serial monitor without changing lines. Format option

delay (milliseconds); delay the command by milliseconds. 1000 milliseconds = 1 second

Observation of results

Figure 11

3. Turn LED on and off with digital output

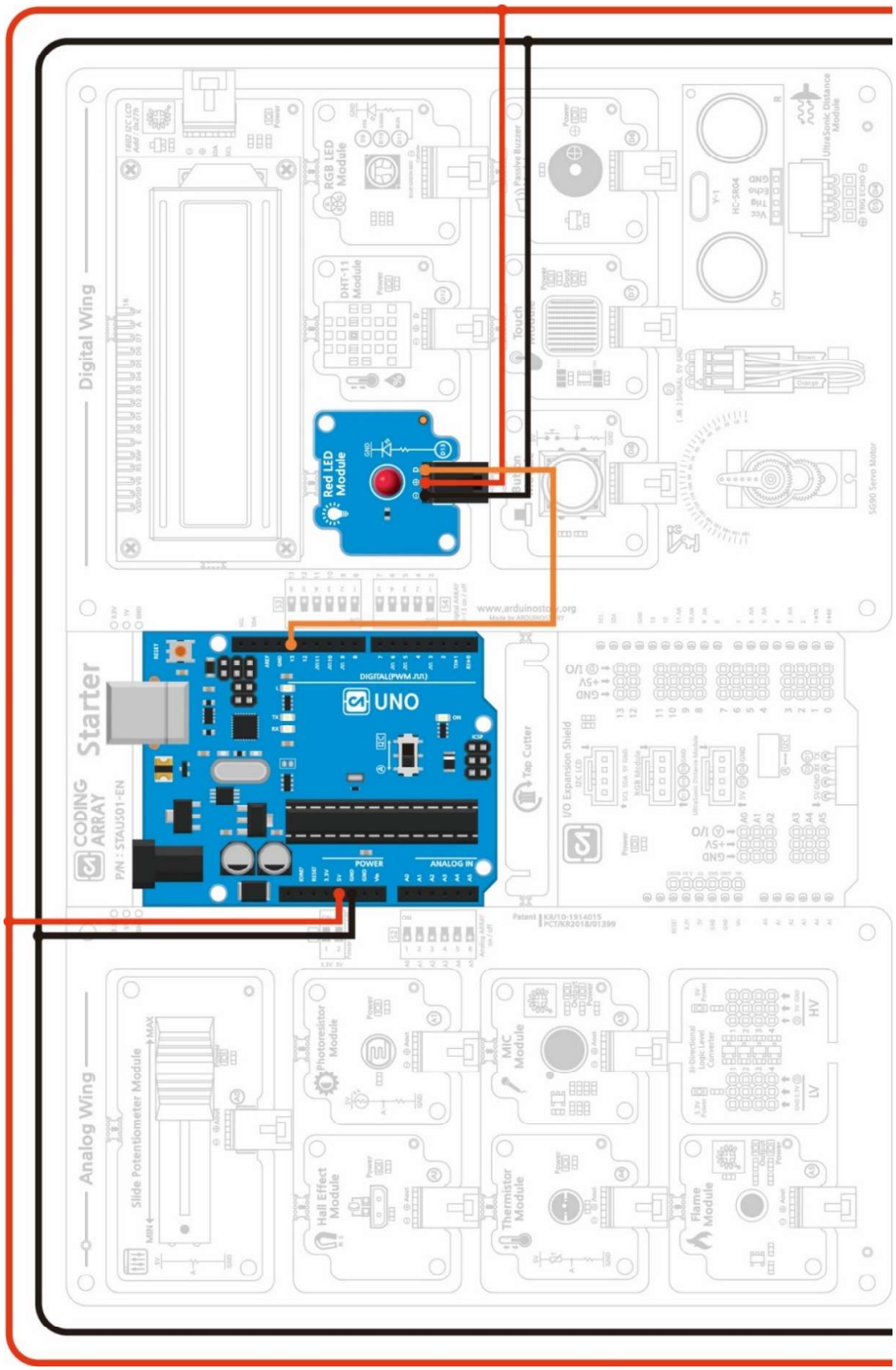
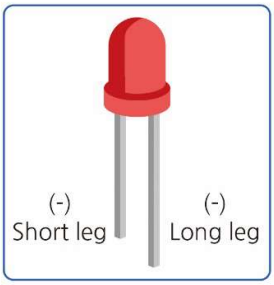


Figure 12

✧ Let's set the red LEDs connected to pin 13 to the output and repeat the execution of the LED turning on for 1 second and off for 1 second, depending on the time settings of the digitalWrite function indicating the digital output and the delay function..

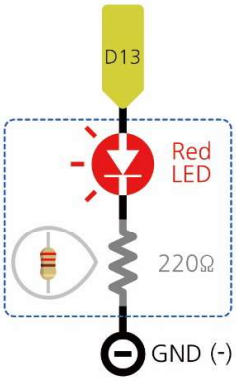
LED (Light Emitting Diode)



LEDs are semiconductor devices that emit light when current flows through LEDs. LEDs are of type lamp (lead) and surface seal (SMD) type. The red LED used in the array kit is a lamp type and has two legs. A long pin (+) connects to Arduino's No. 13 pin, and a short pin (-) connects to Arduino's GND (ground)

Figure 13

Since the operating voltage of the LED used is 1.6 to 2 volts, the module is also equipped with a resistance (220 ohms) that limits the current at 5 volt supply.



CAUTION: ★) LED's do not illuminate when connected with (+) (-) polarity changes.

CAUTION: ★) If the LED is powered without resistance, it will not work or reduce its service life.

Figure 14

LEDs are smaller in size and longer in life compared to light bulbs or fluorescent light bulbs, and use less power, but produce brighter light. It is often used for portable flashlights, lights, billboards, car lights, flat-screen TVs and monitors. Two or more LED lights can be used to implement a beam walker signal lamp or to indicate the device's on/off indicator.

Let's find out about digital output.

- Arduino Uno's digital input/output pin

Arduino's input and output signals are largely divided into two categories: digital and analog.

Digital signals refer to the high voltage of two signals, HIGH / LOW, 1 / 0, True / False , and On /Off..

Figure 15

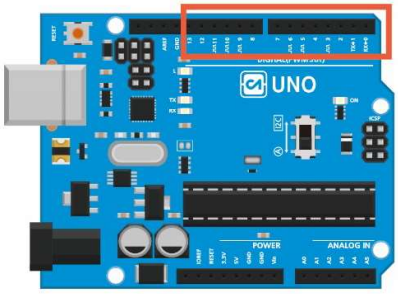


Figure 16

First, let's learn about digital output signals.

Arduino has 14 digital I/O pins from 0 to 13. However, since 0 and 1 are connected to the computer, it is preferable to use pins 2 if possible.

- Digital input **digitalWrite (Pin Number, HIGH);**

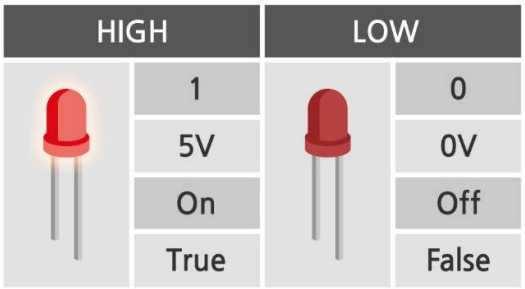


Figure 17

Since digital pins are specified by default as input pins, when used as output pins, the setup declares the output as pinMode (pin number, OUTPUT). digitalWrite (pin number, HIGH) after declaration; can command 5V output to pin number or digitalWrite (pin number, LOW) to 0V output to pin number..

Let's find out about variables..

Figure 18

Variables

Variable means the name or space itself of a space that stores values that can change during a program to process or store data.

In order to define a variable in C programming, the data type for the value that goes into the variable must be declared together. For numeric data types, appropriate numeric data types should be declared according to the size of the data. If the data type is incorrectly declared, the desired result value cannot be obtained.

If a variable is declared at the top of the previous program, it becomes a global variable that can be used in all parts of the program.

Variable data type

Type	Scope	byte	Use
void			Function declaration, used when return value is missing ex) void setup() { } void loop() { }
boolean		1	Use only true or false values ex) boolean state= true ;
char	-128~127	1	Save character values, one character value is enclosed in single quotes, and stored as ASCII code (number) values. The two examples below store the same values. e.g.) char myChar= 'A'; char myChar = 65; Multiple characters are enclosed in double quotes. e.g.) char array[ ]"ardinostory"
unsinged char	0~255	1	Same as byte data type. Byte data type is preferred.
byte	0~255	1	Similar to char, but having a positive integer value.
int	-32768~32767	2	Basic data type for storing integers with symbols If data is out of range, it will result in unexpected values and should be replaced with double or long.
unsinged int	0~65535	2	Use for positive integer values
word	0~65535	2	Use for positive integer values
long	-2147483648 ~ 2147483647	4	Use for integer values in a range greater than int
unsinged long	0~4294967295	4	Use for positive integer values in large ranges
short	-32768~32767	2	Use for integer values
float	-3.4028235E+38 ~3.4028235E+38	4	True (numeric) data type
double	-3.4028235E+38 ~3.4028235E+38	4	In Arduino, the same data type as float

table 7

CAK Starter Code > 03_01_BlinkingLED

```
1  void setup() {
2    pinMode(13,OUTPUT);                    // Set 13 to output pin
3  }
4
5  void loop() {
6   digitalWrite(13,HIGH);                  // Give digital signal 1 (HIGH) to pin 13. LED illuminated
7   delay(1000);                            // Wait for 1000 milliseconds (=1 second).
8   digitalWrite(13,LOW);                   // Give digital signal 0 (LOW) to pin 13. LED Off
9   delay(1000);                            // Wait for 1000 milliseconds (=1 second
10 }
```

pinMode(pin number, value); pin number puts the digital pin number of Arduino, where the (+) pole of the LED is connected. The value specifies the input/output role of the pin. You can give **INPUT** or **OUTPUT** or INPUT_PULLUP.

digitalWrite(pin number, value); can give HIGH (5V) or LOW (0V) digital output value to parts connected to pin number..

CAK Starter Code > 03_02_BlinkingLED2

```
1  int redLED = 13;                  // Red LED to 13
2
3  void setup() {
4    pinMode(redLED,OUTPUT);      // Set No. 13 to output pin
5  }
6
7  void loop() {
8    digitalWrite(redLED, HIGH);       // Give digital signal 1 (HIGH) to pin 13. Red LED illuminated
9    delay(1000);                      // Wait for 1000 milliseconds (=1 second)
10
11   digitalWrite(redLED, LOW);        // Give digital signal 0 (LOW) to pin 13. Red LED Off
12   delay(1000);                      // Wait for 1000 milliseconds (=1 second).
13 }
```

Int variable name = value; int is a data type that stores integers. The redLED variable stores pin number 13 and can be represented using variable names instead of pin numbers.

View Results

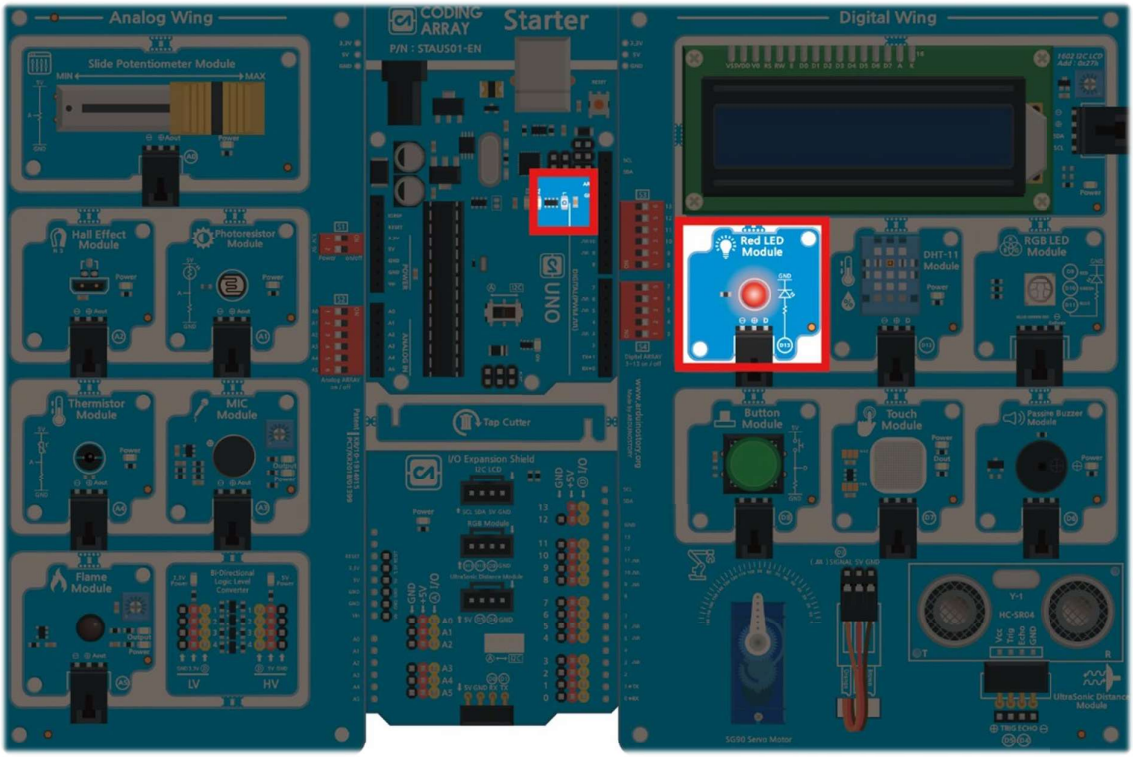


Figure 19

Once the program is uploaded, depending on the digitalWrite function representing the digital output and the time setting of the delay function, the LED can be repeatedly lit for one second and then turned off for one second. At this point, you can see the 'L' LED with built-in on the 13th Uno Board flickering Different delay function times can control the flashing speed of LEDs.

4. Read button switch values with digital input

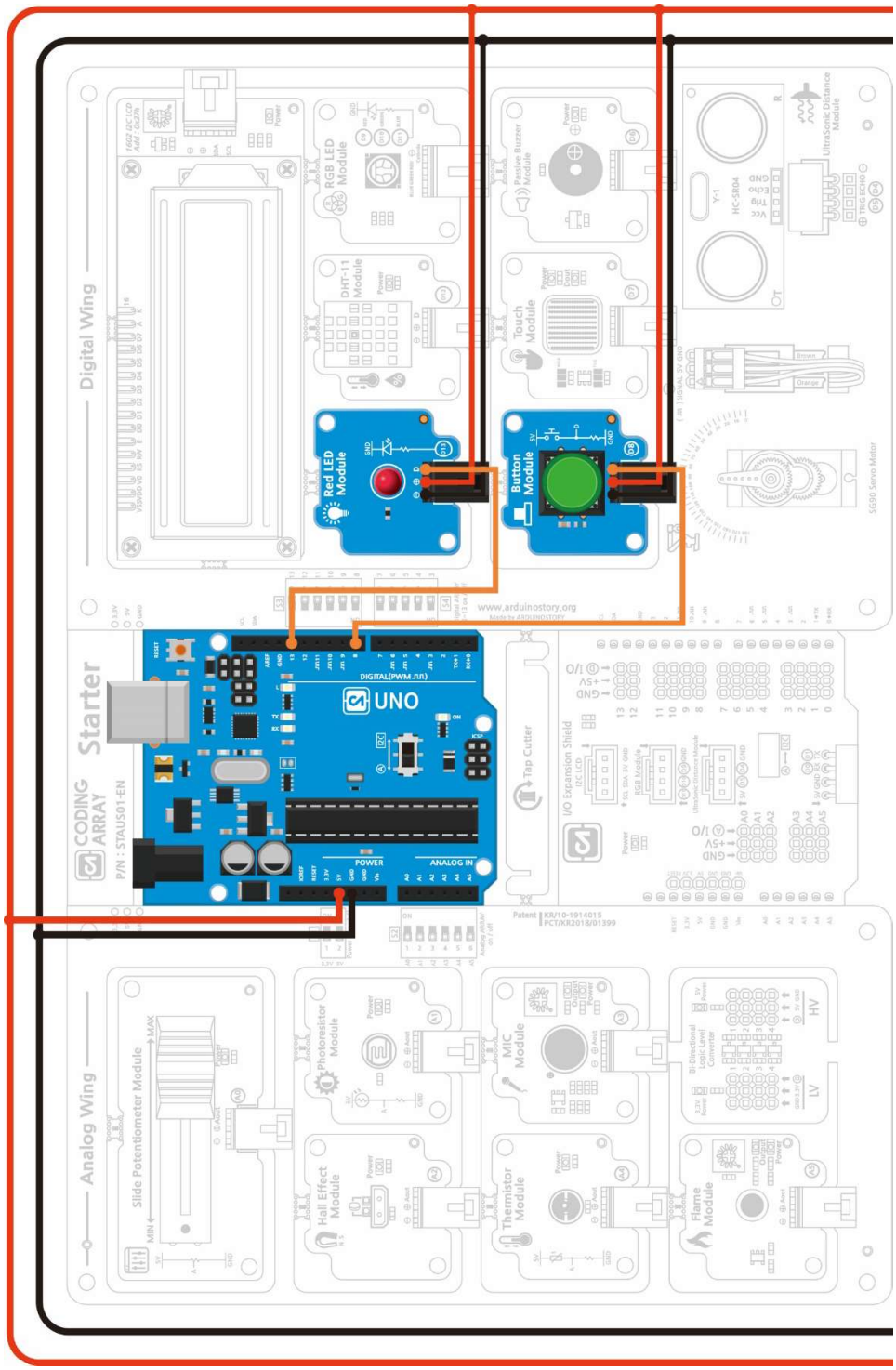


Figure 20