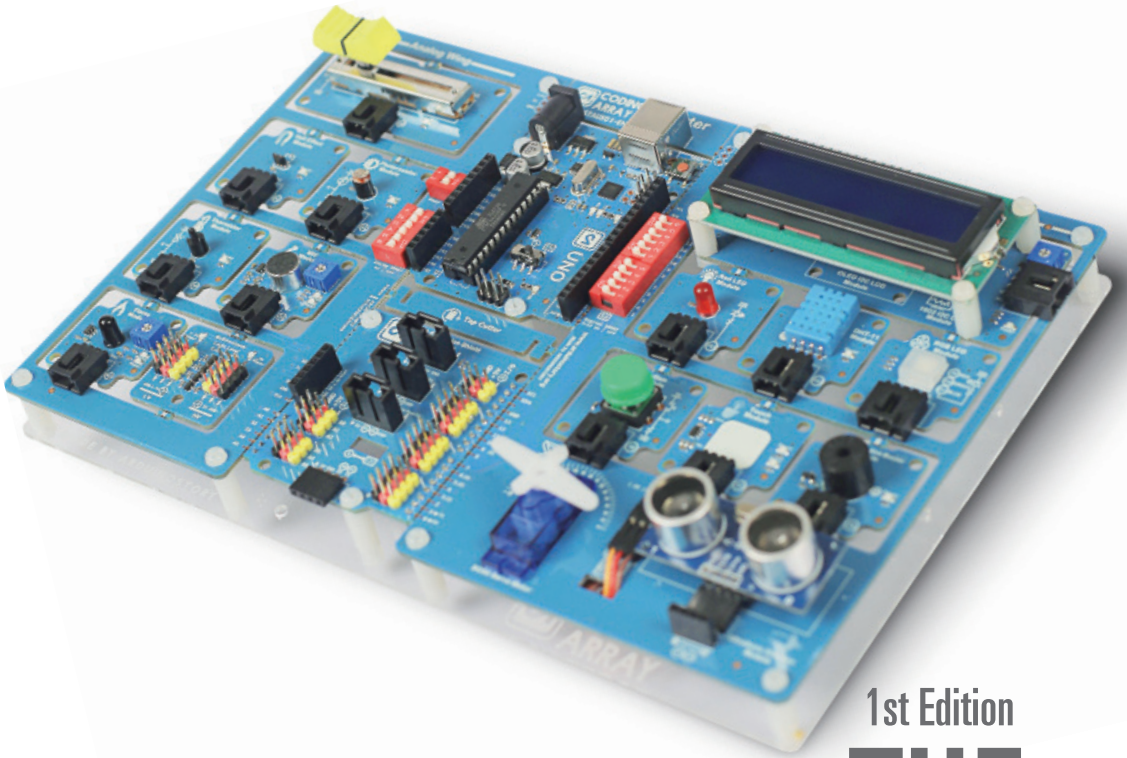


CODING ARRAY

STARTER KIT FOR ARDUINO

- Easier
- Faster
- Safer
- Simple
- Dual mode



1st Edition

THE BEGINNER'S GUIDE

 Made By
Arduino Story

CODING ARRAY

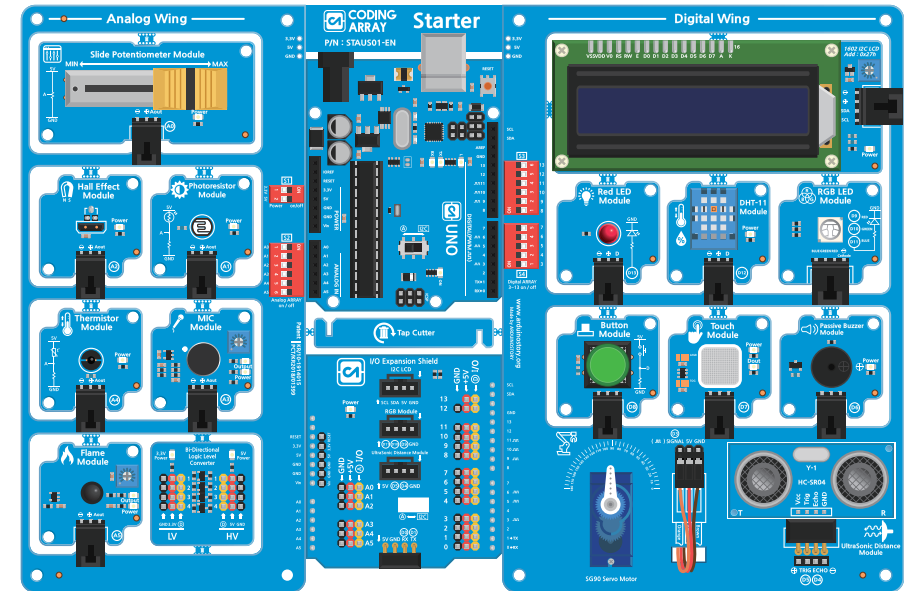
STARTER KIT FOR ARDUINO



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.

※ You may not modify, delete or distribute without permission from the owner, and may be subject to legal punishment in the event of a violation.



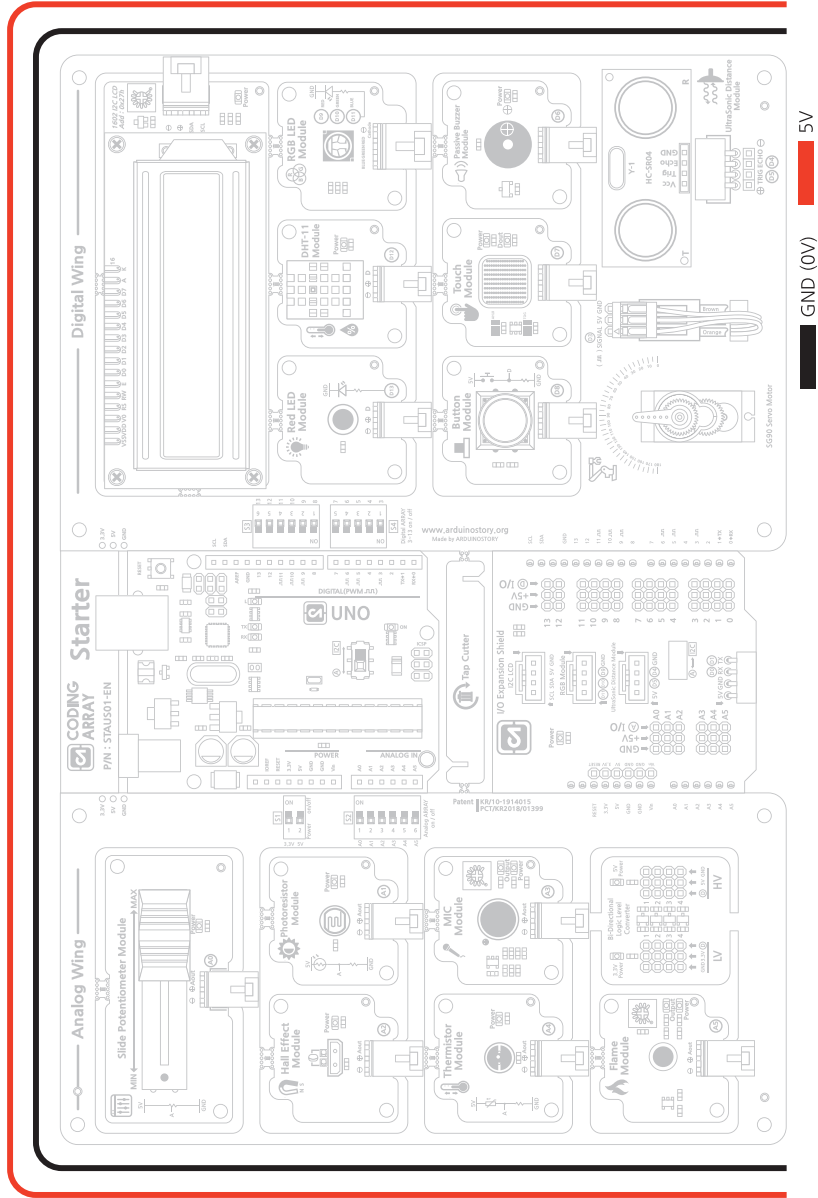
A suitable starter kit for training Arduino beginner is designed to be used without having to configure a separate circuit for 15 input and output units around Arduino Uno.

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.

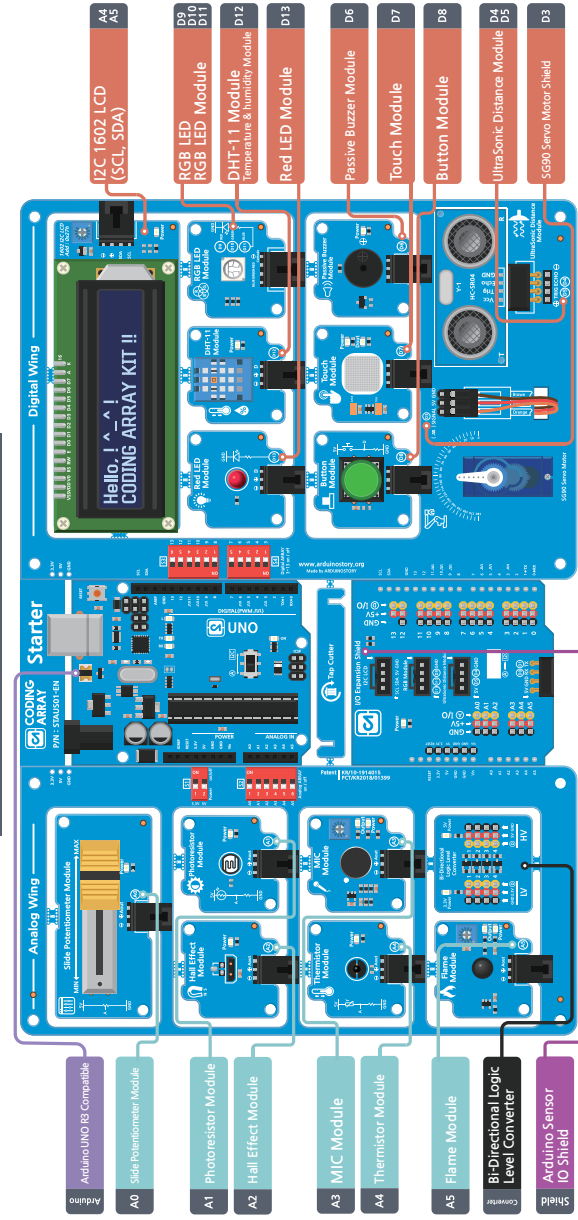
! CAUTION!
Do not disconnect the module using the Tap Cutter before board testing and learning is complete.

INTERNAL POWER CIRCUIT DIAGRAM OF CODING ARRAY STARTER KIT FOR ARDUINO



Coding Array Kit PINOUT

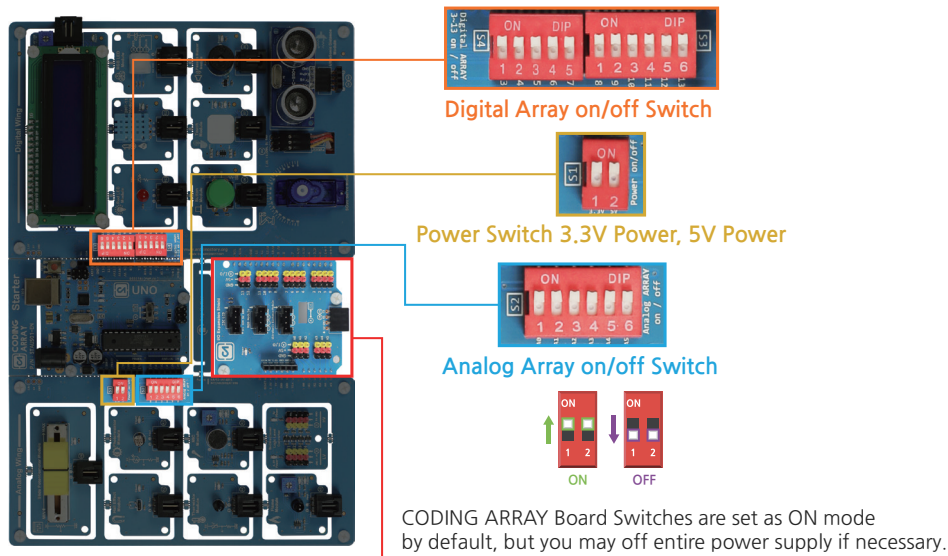
Platform | Digital Wing | Converter
Shield | Analog Wing



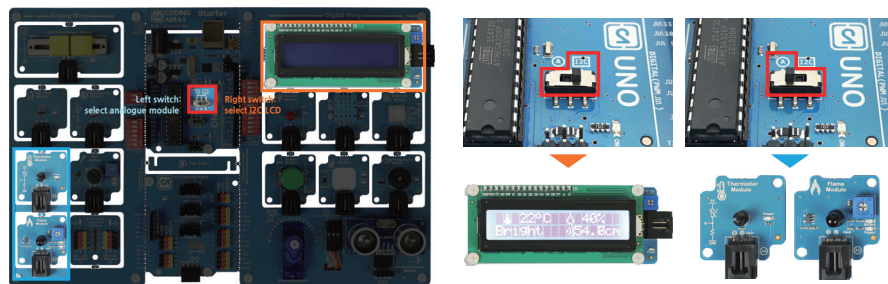
Platform	
Microcontroller	ATmega328P
DC Current for 3.3V	Max. 1000 mA
Operating voltage	5V
Flash memory Size	32 KB (ATmega328P) of which 0.5 KB used by bootloader
Input voltage (limit)	7-12V
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Digital I/O Pins	14 (of which 6 provide PWM output)
Clock speed	16 MHz
Analog Input Pins	6
LED_BUILTIN	13
DC Current per I/O Pin	Max. 20 mA
Arduino Sensor IO Shield	
Shield	Bi-Directional Logic Level Converter

Analog Wing		Digital Wing	
Sensors	Hall Effect Module	Sensors	DHT-11 Module Temperature & Humidity Module
Actuators	Photoresistor Module	Actuators	Touch Module
	Thermistor Module		Ultrasonic Distance Module
	MIC Module		1602 I2C LCD Module
	Flame Module		Red LED
	Slide Potentiometer		RGB LED
			Button Module
			Passive Buzzer Module
			SG90 Mini Servo Motor

CODING ARRAY Kit's another advantage



By default, aduino analog, digital I/O is connected to the I/O expansion shield. When the switch is turned off analog arrays, digital arrays, additional modules can be newly configured and used.



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



Anti-static Packaging

Array kits are packaged in anti-static bags.



Wet Hand

Do not touch the product with wet hands.



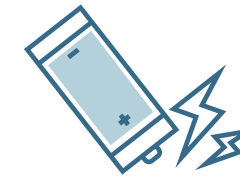
Chemicals

Keep the product away from water or chemicals.



Break

Do not place heavy or sharp objects on the product.



Conductive object

Take care not to contact any conductive object, such as metal, in the circuit. Risk of damage to the circuit or burn or fire.



Separate storage

Be sure to keep the power separate when you are not using the product.



Optimal Temperature

Do not store and use at extreme hot or cold temperatures.



Hand care

The pins are sharp. Be careful of injuries.



Disconnect power

Turn off the power to connect to the board using a separate module.

CODING ARRAY

STARTER KIT FOR ARDUINO

THE BEGINNER'S GUIDE
1ST EDITION

INDEX

Chapter 1. Prepare

1. Arduino?	12
2. Arduino IDE (Integrated Development Environment) Download	14
3. Arduino Software IDE Open	16
4. Menu Bar Function Overview	18
5. For Windows	20
6. For MAC	22

Chapter 2. Follow Example

1. IDE Structure – setup and loop	26
2. Serial communication	28
3. LED On/Off with Digital Output	30
4. Read button switch values with Digital Input	36
5. Change RGB LED Color Using Digital Output and PWM	42
6. Enables mood light with Electrostatic Touch Sensor	52
7. Read slide variable resistance value with Analog Input	60
8. Play melodies with manual buzzer	68
9. Displaying text on 1602 I2C LCD	76
10. Distance measurement with ultrasonic sensor	92
11. Magnet sensing with Hole sensor	98
12. Light intensity detection and calibrate sensor values with light sensing sensors	102
13. Flame Detection Sensor Detects Fire	116
14. Temperature measurement with NTC thermistor	122
15. Sound Detection with Microphone Sensor	130
16. Temperature and humidity measurement with sensors	136
17. Servo motor control	146

FAQ and solutions	153
--------------------------	-----

CHAPTER 1

- 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

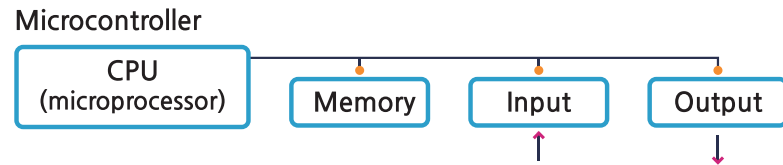
CODING ARRAY STARTER Kit for Arduino

THE BEGINNER'S GUIDE 1st Edition

1 Arduino?

CHAPTER 1

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



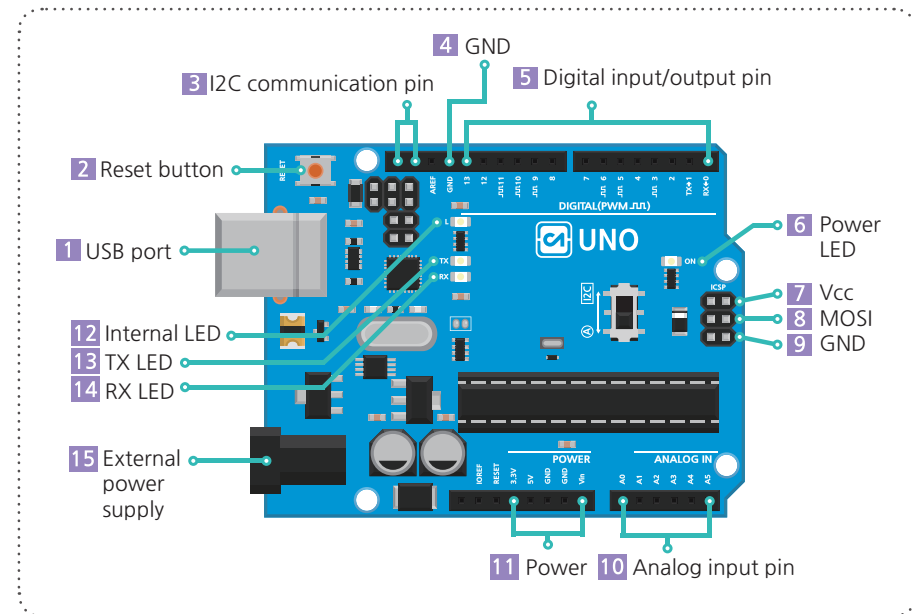
Microcontrollers are made of a single chip to perform a series of tasks in the CPU and memory, and to send the results to the I/O port with an electrical signal.

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 Parallax'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.

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 >



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	Grounding
5	Digital input/output pin	14 Digital input/output pin (including 6 PWM pins)
6	Power LED	Illuminates when power is supplied.
7	Vcc	Grounding
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)

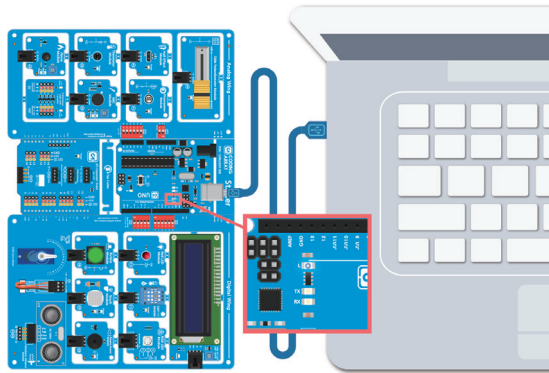
2

CHAPTER 1

Arduino IDE (Integrated Development Environment) Download

STEP 01

Connecting a Board with Computer



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.)

STEP 02

Access the Arduino Software Download Site from the Internet



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

STEP 03

Free download of Arduino Software IDE

Download the Arduino IDE



ARDUINO 1.8.8

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non-admin install

Windows app Requires Win 8.1 or 10

Get

Mac OS X 10.8 MountainLion or newer

Linux 32bits

Linux 64bits

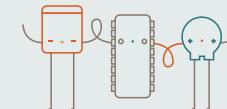
Linux ARM

Release Notes
Source Code
Checksums (sha512)

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

Contribute to the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.



SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **29,216,207** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

\$3

\$5

\$10

\$25

\$50

OTHER

JUST DOWNLOAD

CONTRIBUTE & DOWNLOAD

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



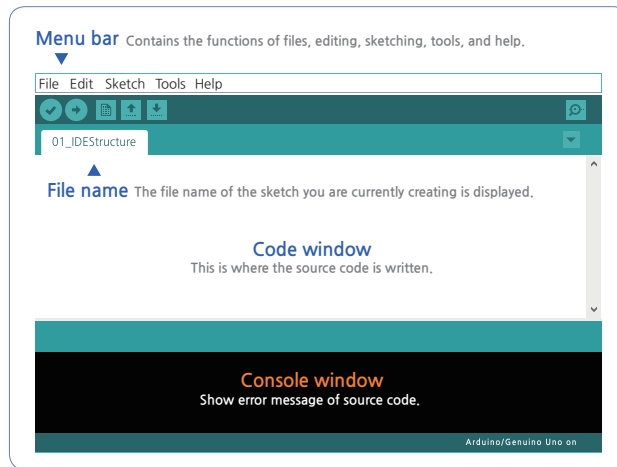
When a compressed file such as "ardino-1.8.5-linux64.tar.xz" is downloaded, Linux users run a terminal to decompress as follows: Run the install.sh file in the extracted folder.

tar xvfz [File name]



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



- 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.

	Check : Check and compile code for grammar errors
	Upload : Send code to Arduino board. When the upload button is pressed, the lamp on the board flashes quickly.
	New File : Create a new file for creating a new sketch
	Open : Open saved Sketch file
	Save : Save the currently active sketch
	Serial monitor : Open a new window to show data communication between Arduino and the computer.

- 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

CHAPTER 1

Menu Bar Function

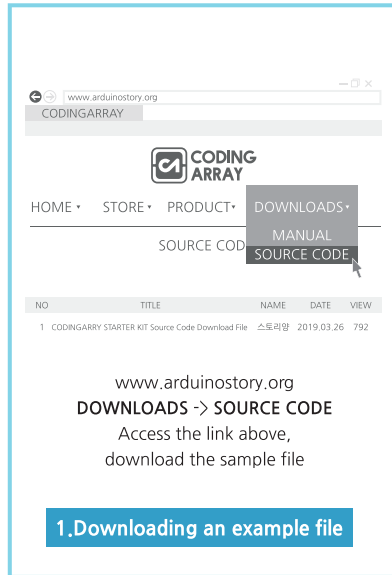
File	Editing	Sketch	Tool	Help
New file	When creating a new sketch			
Open	When opening a saved sketch			
Open Recent Files	View and open a list of recently opened sketches			
Sketchbook	When you open one of the sketches stored in the Sketchbook folder			
Example	When opening a built-in example, library 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			

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			

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 compiled 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			

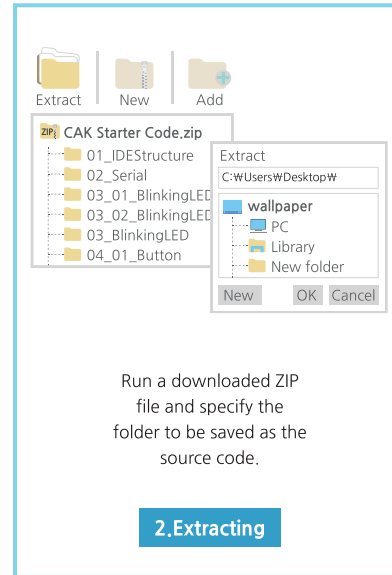
File	Editing	Sketch	Tool	Help
Auto Format	Formatting code to look good			
Archive Sketches	Keep a copy of your sketch in a .zip file			
Modify Encoding & Refresh	Reduce encoding differences between editor and other editors			
Serial monitor	where you see the communication data between Arduino and the 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 mkII"	Used to program boards or chips			
Burning a bootloader	When using IDE for a MCU other than Arduino			

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



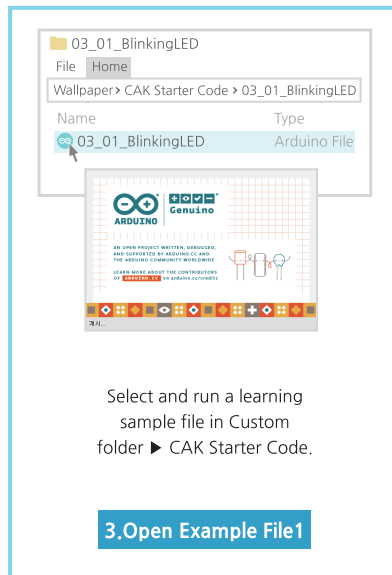
www.arduinoostory.org
DOWNLOADS -> SOURCE CODE
 Access the link above,
 download the sample file

1. Downloading an example file



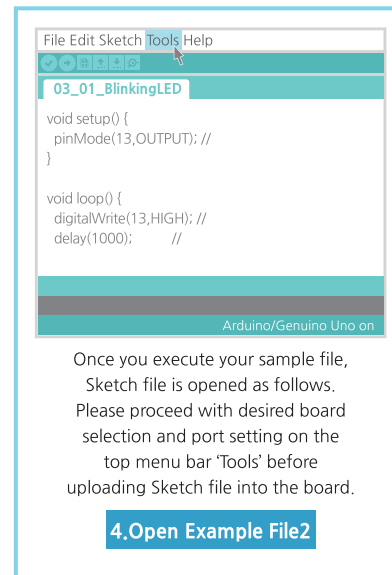
Run a downloaded ZIP
 file and specify the
 folder to be saved as the
 source code.

2. Extracting



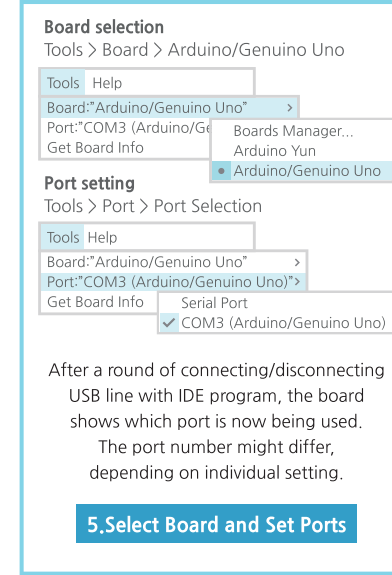
Select and run a learning
 sample file in Custom
 folder ▶ CAK Starter Code.

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

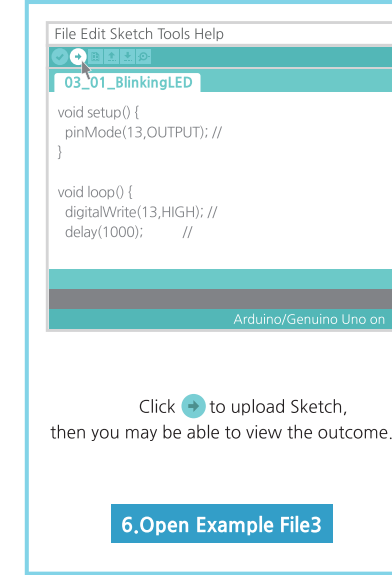


Board selection
 Tools > Board > Arduino/Genuino Uno

Port setting
 Tools > Port > Port Selection

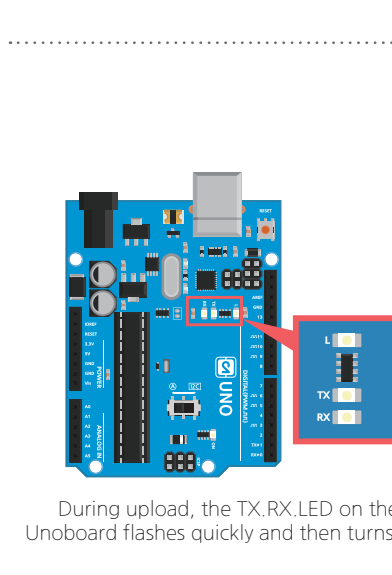
After a round of connecting/disconnecting
 USB line with IDE program, the board
 shows which port is now being used.
 The port number might differ,
 depending on individual setting.

5. Select Board and Set Ports

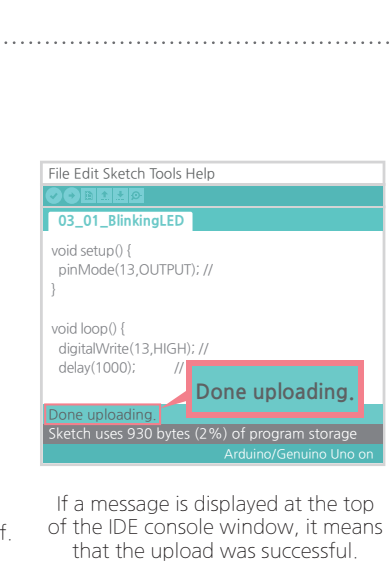


Click → to upload Sketch,
 then you may be able to view the outcome.

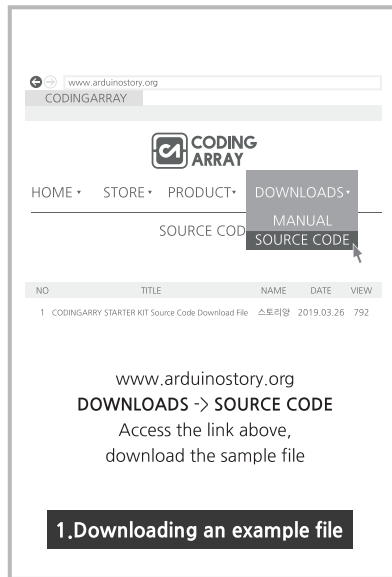
6. Open Example File3



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

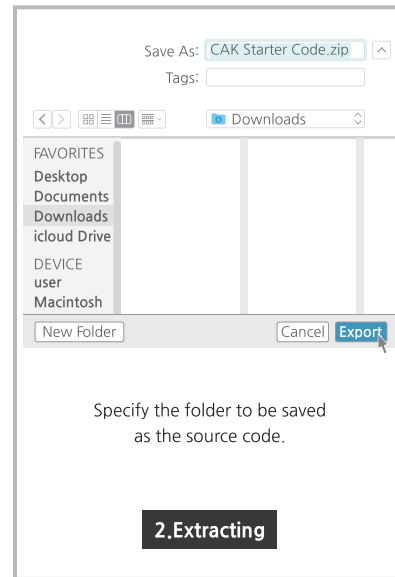


If a message is displayed at the top
 of the IDE console window, it means
 that the upload was successful.



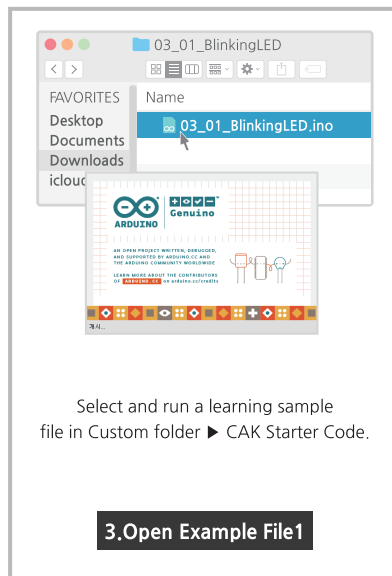
www.arduino.org
DOWNLOADS -> SOURCE CODE
 Access the link above,
 download the sample file

1. Downloading an example file



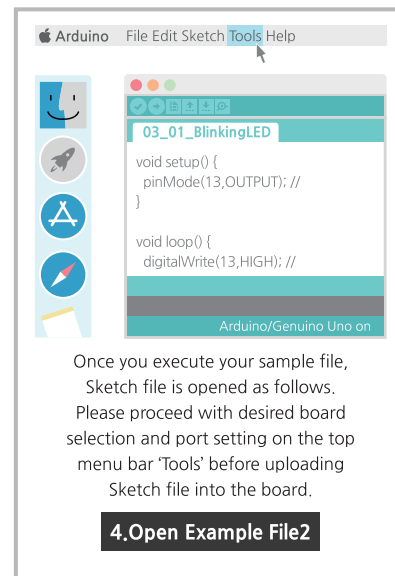
Specify the folder to be saved
 as the source code.

2. Extracting



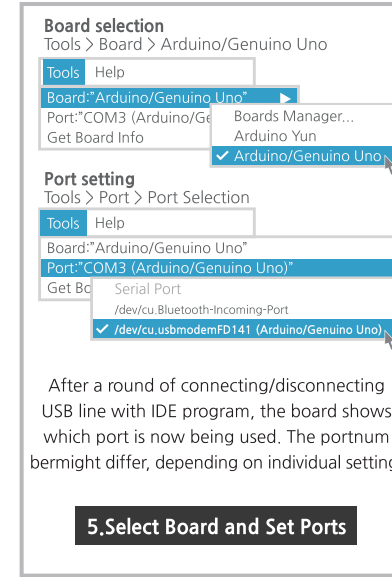
Select and run a learning sample
 file in Custom folder ► CAK Starter Code.

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

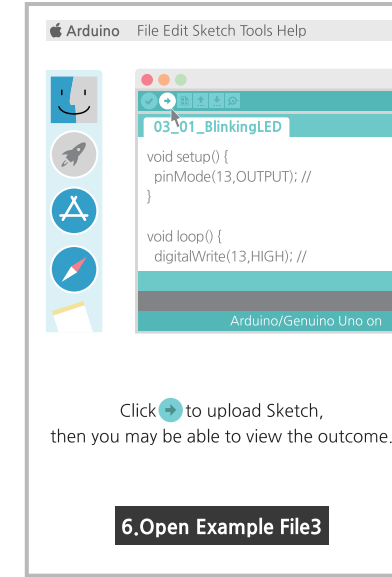



Board selection
 Tools > Board > Arduino/Genuino Uno

Port setting
 Tools > Port > Port Selection

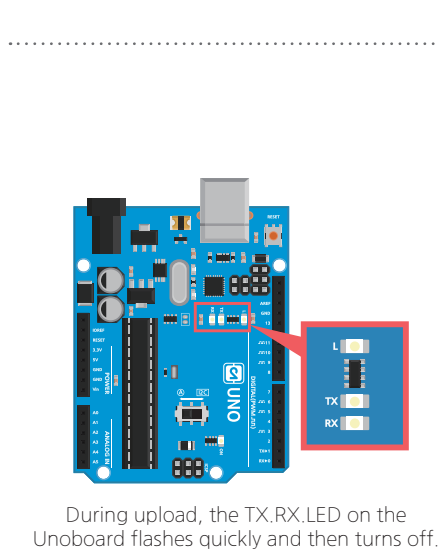
After a round of connecting/disconnecting
 USB line with IDE program, the board shows
 which port is now being used. The portnum
 might differ, depending on individual setting.

5. Select Board and Set Ports

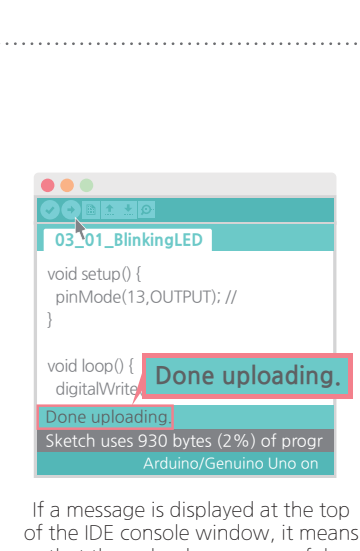


Click  to upload Sketch,
 then you may be able to view the outcome.

6. Open Example File3



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



Done uploading.
 Sketch uses 930 bytes (2%) of progr
 Arduino/Genuino Uno on

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

CHAPTER 2

- FOLLOW EXAMPLE

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

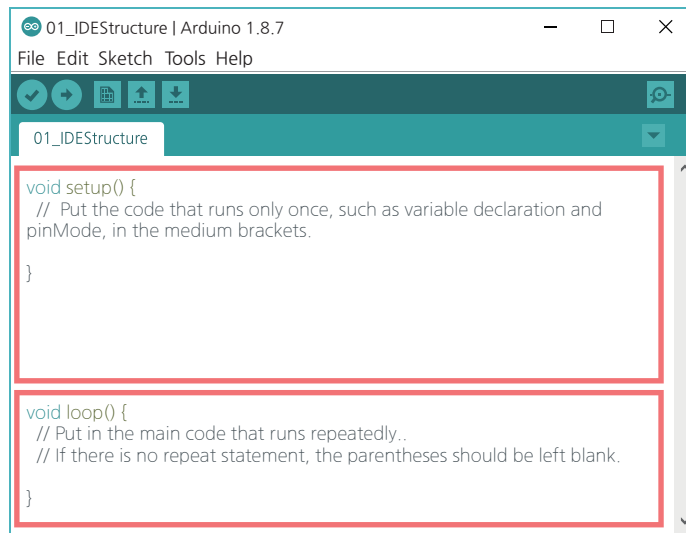
1. IDE structure – setup and loop
2. Serial communication
3. Turn LED on and off with digital output
4. Read button switch values with digital input
5. Change RGB LDE color using digital output and PWM function
6. Implementing moods, etc. with capacitive touch sensors

CODING ARRAY STARTER Kit for Arduino

THE BEGINNER'S GUIDE 1st Edition

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

IDE structure – setup and loop



■ 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.

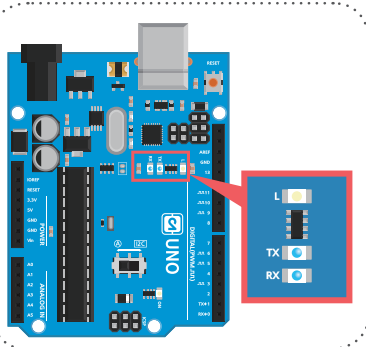
```

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 */).

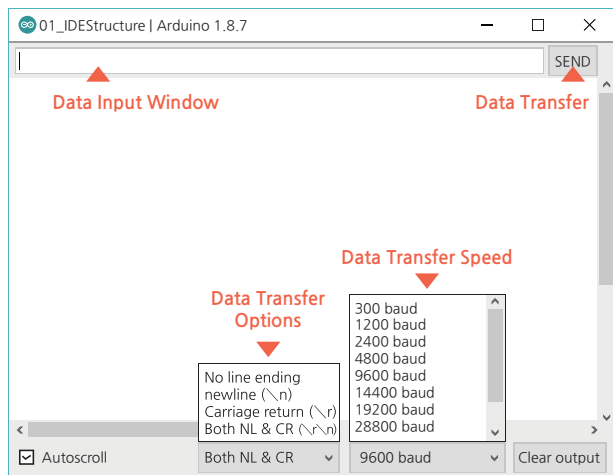


The USB cable allows data to be exchanged between Arduino and the computer, called serial communication (serial communication, UART communication). When uploading sketch files, you can see the RX (:data received) and TX (data sent) lights on the board flickering quickly. Since Arduino's digital No. 0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

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.



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 }

```

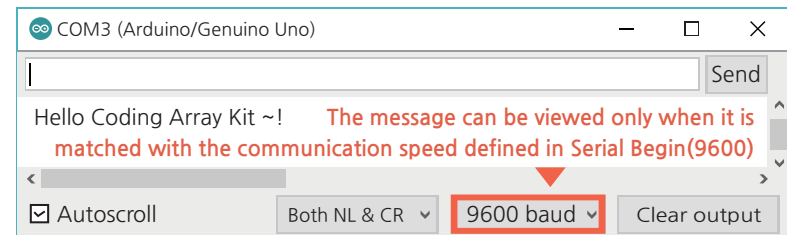
Serial.begin (communication speed); the Baud rate is between 300 and 115200. It is usually set to 9600.

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 ' ' .
 > 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.

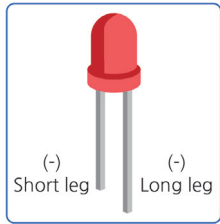
Serial.print (value, format); add values to the serial monitor without changing lines. Format option

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

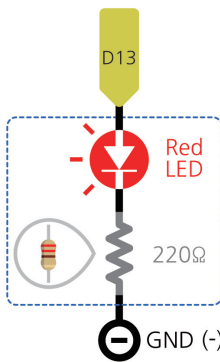
Observation of results



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)



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.

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.

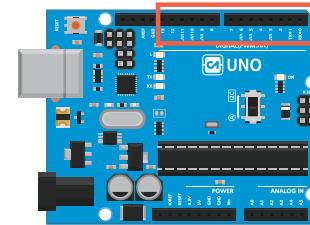
Let's find out about digital output

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.



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);*

	HIGH		LOW
	1		0
	5V		0V
	On		Off
	True		False

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

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[]"ardinstory"
unsigned 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.
unsigned 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
unsigned 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

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

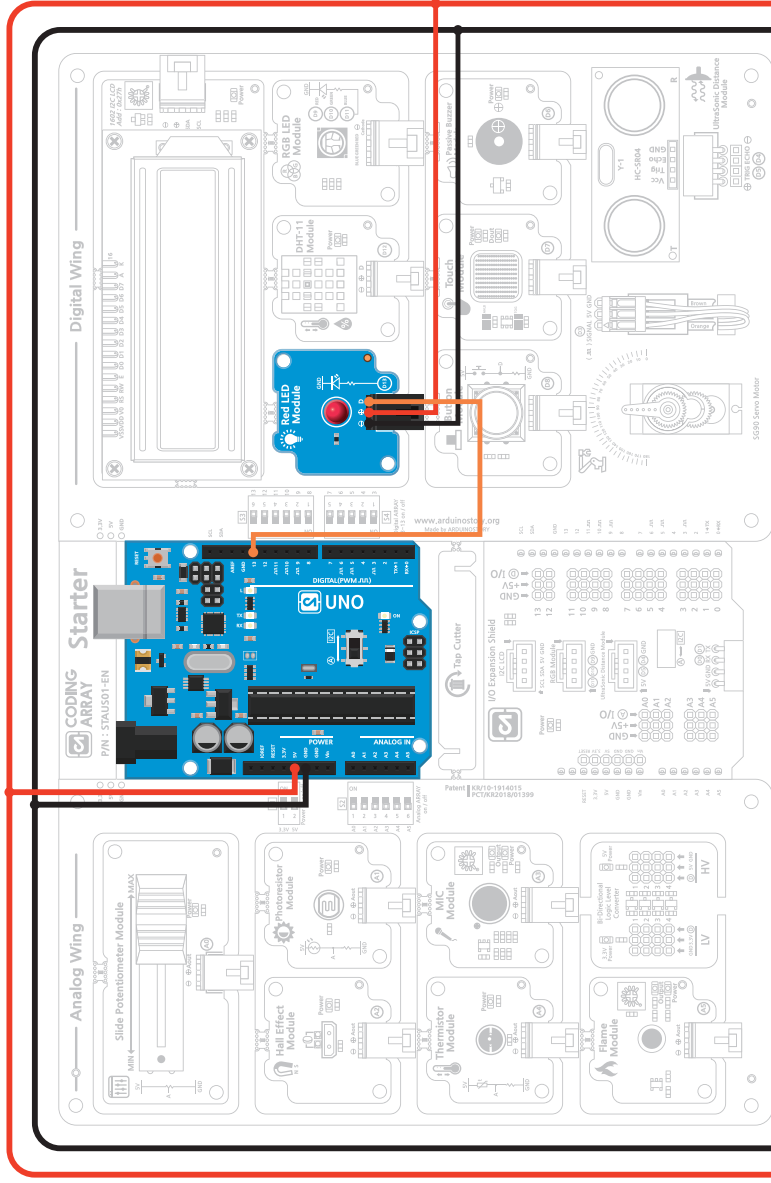
```

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 }

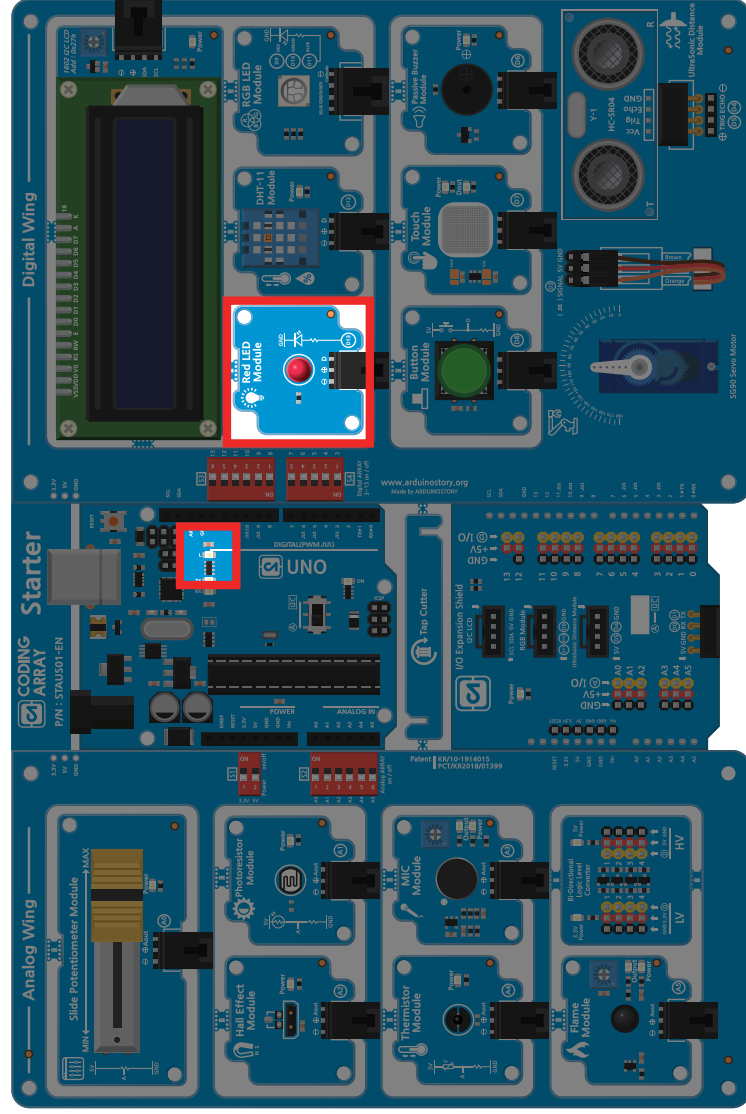
```

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.



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..

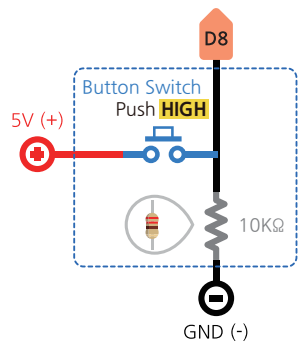
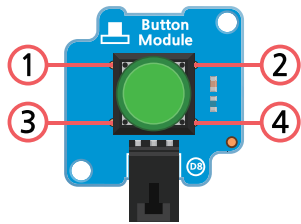


Once the program is uploaded, depending on the digitalWrite function representing the digital output and the time setting of the delay function, the LED can be repeatedly lit for one second and then turned off for one second. At this point, you can see the 'L' LED with built-in on the 13th Uno Board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

View Results

Push Button

The button switch is used to open and close the circuit by pressing the button at the top. When a button is pressed, a circuit is connected and an electrical current flows between 1,2 and 3,4. Therefore, when connecting wires, select one from 1,2 and one from 3,4. Button switches are used in everyday life such as game consoles, bus exit notification buttons, keyboard buttons, etc. as well as control of opening and closing circuits.



When the circuit of the button switch is open, the Arduino board cannot logically predict the HIGH, LOW for the pin that is not connected, resulting in a floating phenomenon that moves high and low fast. To prevent floating, a button switch can be received as a digital input after a pull-up or pull-down resistance is hung.

The coding array starter kit is connected to a button switch using a pull down resistance that defines the LOW voltage in normal situations without a drive signal. Therefore, the HIGH (1) value is entered when the button switch is pressed and the LOW (0) value is entered if the button switch is not pressed.

Pull-up circuit	Pull-Down circuit
Connect the normal state of the input pin to HIGH. With the resistance connected to power 5V, HIGH returns 1 in digitalRead .	Connect the normal state of the input pin to LOW. With the resistance connected to GND 0V, HIGH returns 1 in digitalRead .

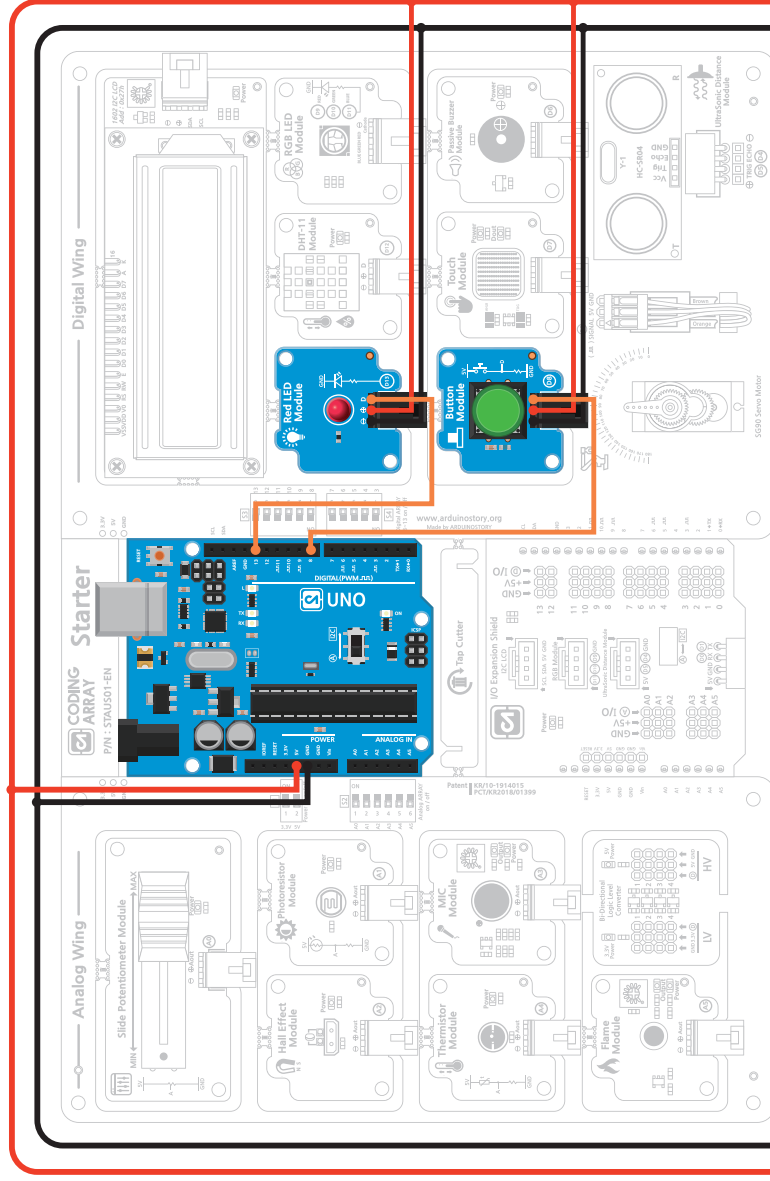
```

1  /* If condition statement is used
2  * Press button switch to connect circuit and return HIGH (1) value to light LED
3  * If the button switch is not pressed, the circuit will open and the LOW (0) value will be returned
4  * to turn off the LED.
5  */
6
7  int redLED = 13; // Set red LED pin to 13.
8  int Button = 8; // Button switch pin set to 8
9
10 void setup() {
11   pinMode(Button, INPUT); // Set button pin to input
12   pinMode(redLED, OUTPUT); // Set redLED pin to output
13
14   Serial.begin(9600); // Starts serial communication at 9600 speed
15
16 }
17
18 void loop() {
19
20   int sensorVal = digitalRead(Button); // Receive button input value for sensorVal variable.
21   // Variables inserted in void function are regional variables
22
23   Serial.println(sensorVal); // Mark the value of the button one line in the serial window.
24   // If the pull-down resistance is connected, give LOW (0) when the button is open and HIGH (1)
25   // when pressed.
26   // give HIGH (1) value when the button is open and LOW (0) when pressed.
27
28   if (sensorVal == LOW) { // If the button is open,
29     digitalWrite(redLED, LOW); // RedLED OFF
30   }
31
32   else { // If the button is pressed,
33     digitalWrite(redLED, HIGH); // Turn on the redLED.
34   }
35   delay(10);
36 }

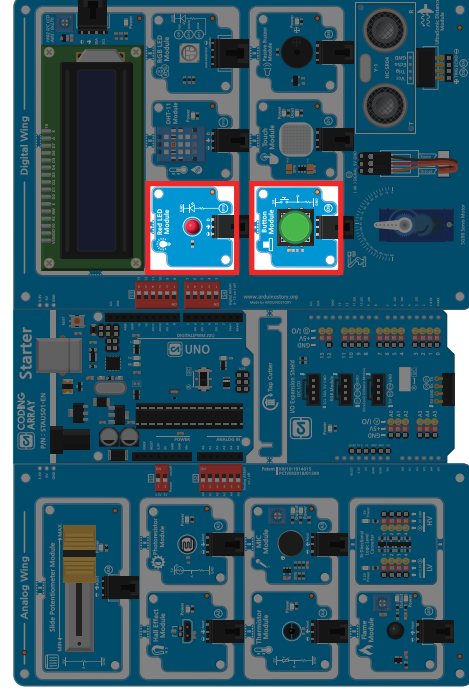
```

digitalRead(pin number, value); The voltage entering the pin is read by HIGH (1) as a LOW (0) digital input value.

== It means that the left and right values are the same.



Use a button switch connected to pin 8 using a pull-down resistance inside the module. Use the digitalRead function to receive the digital input value of the button switch and to issue a digitalWrite command to the LED. Let's also find out how button switches divaunhings..



View Results

While the sketch is uploaded and the button switch is pressed, the HIGH value is entered to illuminate the LED.

While the button is not pressed, the LOW value is entered and the LED is turned off.

The pull-up resistance circuit may be configured separately on the button switch, but it may also be used to use a 20K Ω pull-up resistance inside the Uno Board using INPUT_PULLUP. When connecting a sensor to a pin consisting of INPUT_PULLUP, the other end must be connected to the GND (0V). Uno board does not have INPUT_PULLDOWN function.

pinMode(pin number, INPUT_PULLUP); Set to input pin using pull-up resistance inside (available from arduino 1.0.1)

When a button switch is opened and closed, it can often be caused by mechanical and physical problems. This situation can be avoided by pressing and reading multiple times in a very short time when a program can be deceiving. Learn more about divaunhings in 04_02



CAK Starter CODE > 04_02_Button_Debounce

```

1  /* When a button switch is opened and closed, it often generates incorrect signals due to mechanical and
2  physical problems.
3  * Avoid this situation by pressing and reading multiple times in a very short time that can fool a program
4  * This process is called divauning.
5  */
6  const int Button =8;    // Set button switch pin to 8
7  const int redLED = 13;  // Set LED pin to 13
8
9  int ledState =HIGH;    // Set output pin to HIGH
10 int buttonState;      // Variables that read and store the current button switch status
11 int lastButtonState =HIGH; // Read and save the previous button switch status, reset to LOW
12
13 unsigned long lastDebounceTime =0; // Save the last time the output pin was switched.
14 unsigned long debounceDelay =50; // Time to wait for steady state (milliseconds)
15
16 void setup() {
17   pinMode(redLED, OUTPUT); // Set red LED pin to output
18   pinMode(Button, INPUT);
19   digitalWrite(redLED,ledState); // Turn the LED on and off according to the ledState.
20 }
21
22 void loop() {
23   int reading = digitalRead(Button); // Read button status and save to reading variable
24   if(reading != lastButtonState) { // If the status of the button changes to Noise or Press,
25     lastDebounceTime =millis(); // Reset the debouncing timer,
26   }
27
28   // Whatever value you've read, if it's longer than the debounce delay,
29   if((millis() -lastDebounceTime) > debounceDelay) {
30     if(reading != buttonState) { // If the status of the button changes,
31       buttonState =reading; // Save the status of the button.
32       if(buttonState ==LOW) { // If the new button status is HIGH
33         ledState =!ledState; // Change the status of the LED.
34       }
35     }
36   }
37   digitalWrite(redLED,ledState); // LEDs are on/off with values stored in ledState
38   // Store the value of the reading variable in lastButtonState (used in the next loop)
39   lastButtonState =reading;
40 }

```

const A keyword representing constants. It can be used with other variables but makes it impossible to change the value of a variable.

millis() Returns the number of milliseconds after the Arduino board executes the current program (unsigned long)

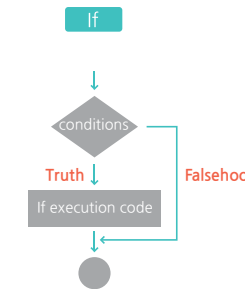
RELATIONAL OPERATOR

A < B	If A is less than B
A > B	If A is greater than B
A == B	If A equals B
A <= B	If A is less than or equal to B
A >= B	If A is greater than or equal to B
A != B	If A is not equal to B

ARITHMETIC OPERATOR

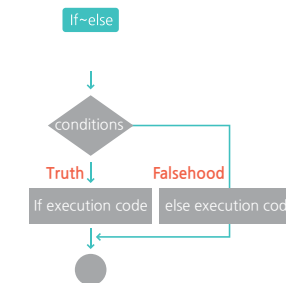
+	Addition
-	Subtraction
*	Multiplication
/	Value of division
%	Rest of division.

If / if ~else / Multiple if condition statements



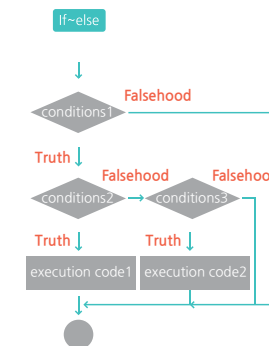
If (conditions) { If execution code }

If the conditional statement is true, perform the if execution code and if not move on to the next execution statement.



If (conditions) {If execution code;} else { else execution code; }

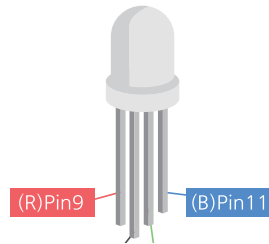
If the condition is true, perform the if execution code, and if the condition is false, execute the else execution code.



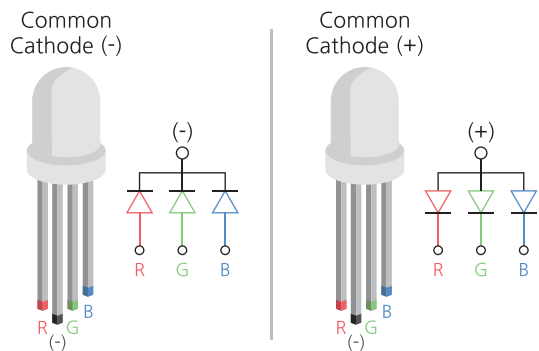
If (conditions1) { If (conditions2) {execution code1 } If (conditions3) {execution code2 } }

If condition 1 is satisfied and condition 2 is satisfied at the same time, process execution code 1 and execute code 2 if condition 1 is satisfied and condition 3 is satisfied at the same time. You can also put another if statement inside the if statement. If a statement is executed inside, then indentation is required.

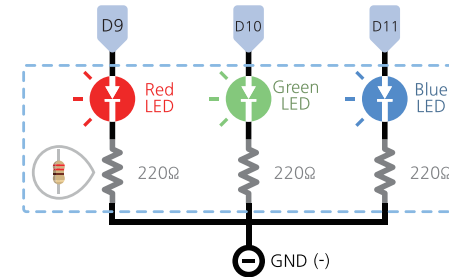
RGB LED



RGB LEDs are LEDs that use three different color combinations: red, green and blue. In modules, red is connected to pin 9 digital, green to pin 10 and blue to pin 11. The longest pin is the common pin..



There are two types of RGB LEDs: common cathodes that connect the longest pins to the GND and common anodes that connect the longest pins to 5 V. In the coding array kit, the common cathode type SMD (surface mounted device) type was used in the module.



Since the operating voltage of the RGB LED used is approximately 2V, the module is equipped with a resistance (220 Ohms) that limits the current at 5V power supply.

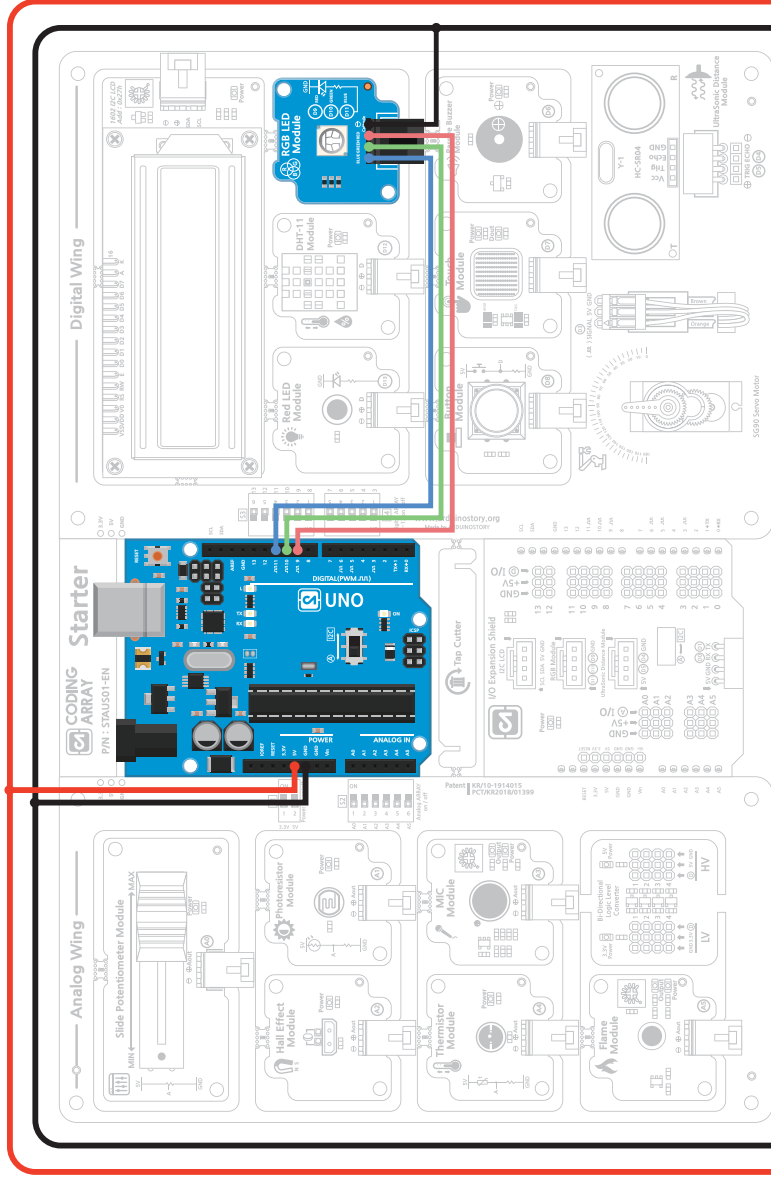
Using RGB LEDs, a variety of lighting effects can be obtained by producing different colors from a single LED. It is often used to decorate the computer's main case with colorful lights or change the color of billboards..

**CODING
ARRAY**
STARTER KIT FOR ARDUINO

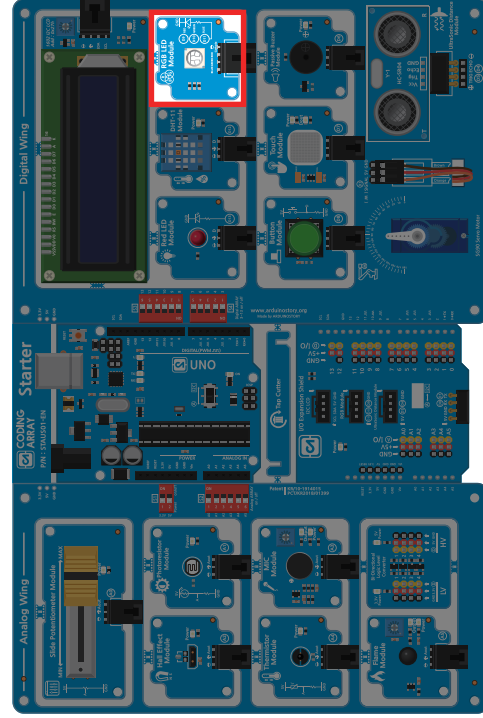


```
1  /* RGB LEDs combine the three primary colors of red, green and blue to release a variety of
2  colors.
3  * Pin 9,10 and 11 are connected to pins that control red, green and blue LEDs respectively.
4  * In this example, we will use a common cathode RGB LED to find out the tri-circular mixture of
5  light from the digital output.
6  */
7
8  const int redPin = 9;    // Red LED No. 9
9  const int greenPin = 10; // Green LED No.10
10 const int bluePin = 11;  // Blue LED No.11
11
12 void setup() {
13     pinMode(redPin, OUTPUT); // Set pin 9 to output
14     pinMode(greenPin, OUTPUT); // Set pin 10 to output
15     pinMode(bluePin, OUTPUT); // Set pin 11 to output
16     Serial.begin(9600);      // 9600-speed serial communication start
17 }
18
19 void loop() {
20     Serial.println("RED on"); // Red LED illuminated
21     digitalWrite(redPin,HIGH);
22     digitalWrite(greenPin,LOW);
23     digitalWrite(bluePin,LOW);
24     delay(1000);             // for a second
25
26     Serial.println("GREEN on"); // Green LED illuminated
27     digitalWrite(redPin,LOW);
28     digitalWrite(greenPin,HIGH);
29     digitalWrite(bluePin,LOW);
30     delay(1000);             // for a second
31
32     Serial.println("BLUE on"); // Blue LED illuminated
```

```
31     digitalWrite(redPin,LOW);
32     digitalWrite(greenPin,LOW);
33     digitalWrite(bluePin,HIGH);
34     delay(1000);             // for a second
35
36     Serial.println("Yellow on"); // Yellow LED illuminated
37     digitalWrite(redPin,HIGH);
38     digitalWrite(greenPin,HIGH);
39     digitalWrite(bluePin,LOW);
40     delay(1000);             // for a second
41
42     Serial.println("Magenta on"); // Magenta illuminated
43     digitalWrite(redPin,HIGH);
44     digitalWrite(greenPin,LOW);
45     digitalWrite(bluePin,HIGH);
46     delay(1000);             // for a second
47
48     Serial.println("Cyan on"); // Cyan LED illuminated
49     digitalWrite(redPin,LOW);
50     digitalWrite(greenPin,HIGH);
51     digitalWrite(bluePin,HIGH);
52     delay(1000);             // for a second
53
54     Serial.println("White on"); // White LED illuminated
55     digitalWrite(redPin,HIGH);
56     digitalWrite(greenPin,HIGH);
57     digitalWrite(bluePin,HIGH);
58     delay(1000);             // for a second
59 }
```

Use RGB LEDs that are connected with red, green, and blue LEDs in digital 9, 10 and 11 that can use PWM function. Three LEDs are simultaneously controlled by digital outputs to achieve a three-circular mixture of light. Also learn how to adjust the brightness of LEDs using the PWM function.

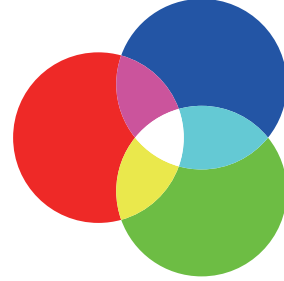


View Results

When a program is uploaded, it uses digital outputs to implement a three-way color mixture of light



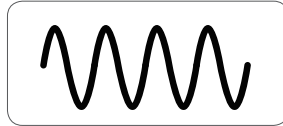
**RGB LED Digital Output
Control Table of Common
Negative Type for
Implementing the Three
Circular Color of Light**



	R	G	B
Red	HIGH	LOW	LOW
Green	LOW	HIGH	LOW
Blue	LOW	LOW	HIGH
Yellow	HIGH	HIGH	LOW
Magenta	HIGH	LOW	HIGH
Cyan	LOW	HIGH	HIGH
White	HIGH	HIGH	HIGH

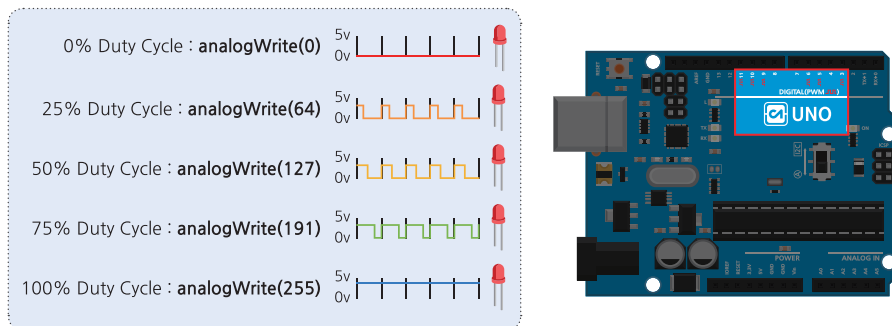
Let's learn about analog output / pulse width modulation (PWM).

Unlike turning on and off LEDs, opening and closing circuits, many values such as light intensity, temperature, distance, sound size adjustment, and light intensity are made up of continuous analog signals..



The digitally operated Arduino does not contain a DAC (Digital-Analog Converter) and cannot output analog signals. Instead, **PWM (Pulse Width Modulation)** is used to **output signals that make them look analog digitally**. If the digital signal ON (5 V) OFF (0 V) signal changes the time portion of the duration (change the pulse width) and this pattern is repeated at a speed that is not recognized by the eye, it appears to be a voltage between 0 and 5 V. This is a logic that feels like 24 frames per second of animation show a series of movements.

The PWM brightness measurement is described by the term duty cycle (assuming a duration of 5 V voltage). The duty cycle is the percentage of the time the circuit is switched on versus the total run time, with 100% representing the maximum brightness and the low percentage representing the low light output. The PWM output can be adjusted to a number between **0 and 255** via `analogWrite()`.



Among Arduino's 0-13, the six pins marked (3, 5, 6, 9, 10, 11) are PWM pins. **These pins do not need a pinMode () setting.**

CAK Starter Code > 05_02_RGB_LED_Fading

```
1  /* In this example, we use a common cathode RGB LED
2  * Pin 9/10 and 11 are connected to pins that control red, green and blue LEDs respectively.
3  * Use the PWM (Pulse-Width Modulation) function to adjust the brightness of the LEDs.
4  * The 'random' command adjusts the brightness of the three colors of RGB to enable a variety of color blends.
5  */
6
7  const int redPin = 9;    // Red LED No. 9
8  const int greenPin = 10; // Green LED No. 10
9  const int bluePin = 11; // Blue LED No. 11
10
11 int delayTime=30;      // Delay time setting
12
13 int redV;              // Set red LED analog value [0-255]
14 int greenV;            // Set green LED analog value [0-255]
15 int blueV;             // Set blue LED analog value [0-255]
16
17 int fadeAmount = 5;    // fade storage variable
18
19 void setup() {
20   pinMode(redPin, OUTPUT); // Set pin 9 to output
21   pinMode(greenPin, OUTPUT); // Set pin 10 to output
22   pinMode(bluePin, OUTPUT); // Set pin 11 to output
23 }
24
25 void loop() {
26   // Red LED brightness adjustment
27   greenV=0;
28   blueV=0;
29   for(redV =0; redV <=255;redV +=5) {
30     // Increase the value by 5 times from 0 to 255.
31     analogWrite(redPin,redV); // Turn on the LED more and more and more.
32     analogWrite(greenPin,greenV);
33     analogWrite(bluePin,blueV);
34     delay(delayTime); // 30-millisecond wait
35   }
36
37   for(int redV=255 ;redV >=0; redV -=5) {
38     // Reducing the value by 5 times from 255 to 0.
39     analogWrite(redPin,redV); // Turn on the darker LEDs.
40     analogWrite(greenPin,greenV);
41     analogWrite(bluePin,blueV);
42     delay(delayTime); // 30-millisecond wait
```

```

43 }
44
45 // Green LED brightness adjustment
46 redV=0;
47 blueV=0;
48 for(greenV=0 ; greenV <=255; greenV +=5) {
49     // Increase the value by 5 times from 0 to 255.
50     analogWrite(redPin,redV);    // Turn on the LED more and more and more.
51     analogWrite(greenPin,greenV);
52     analogWrite(bluePin,blueV);
53     delay(delayTime);    // 30-millisecond wait
54 }
55
56 for(int greenV =255 ; greenV >=0; greenV -=5) {
57     // Reducing the value by 5 times from 255 to 0.
58     analogWrite(redPin,redV);    // Turn on the darker LEDs.
59     analogWrite(greenPin,greenV);
60     analogWrite(bluePin,blueV);
61     delay(delayTime);    // 30-millisecond wait
62 }
63
64 // Blue LED brightness adjustment
65 redV=0;
66 greenV=0;
67 for(blueV=0 ; blueV <=255; blueV +=5) {
68     // Increase the value by 5 times from 0 to 255.
69     analogWrite(redPin,redV);    // Turn on the LED more and more and more.
70     analogWrite(greenPin,greenV);
71     analogWrite(bluePin,blueV);
72
73     delay(delayTime);    // 30-millisecond wait
74 }
75
76 for(int blueV=255 ;blueV >=0; blueV -=5) {
77     // Reducing the value by 5 times from 255 to 0.
78     analogWrite(redPin,redV);    // Turn on the darker LEDs.
79     analogWrite(greenPin,greenV);
80     analogWrite(bluePin,blueV);
81     delay(delayTime);    // 30-millisecond wait
82 }
83
84 // 20 Random colors
85 for (int i=0; i<20; i++){
86     analogWrite(redPin,random(0,255));
87     analogWrite(greenPin,random(0,255));

```

```

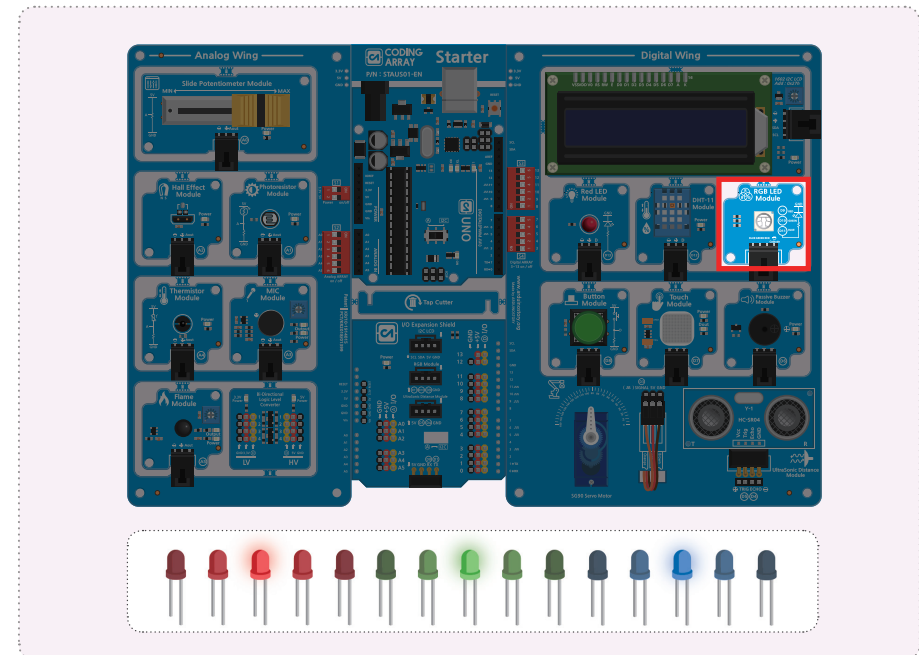
88     analogWrite(bluePin,random(0,255));
89     delay(1000);
90 }
91 }

```

random (Min, Max); The random function sets the range and returns the random integer values within the maximum value-1. Maximum value: Maximum value of random number (optional), Maximum value: Maximum value of random number

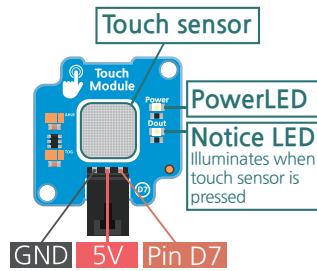
analogWrite (Pin number , Value); Only PWM pin numbers 3, 5, 6, 9, 10, 11 are available. Values can be expressed as analog outputs with integers of 0 to 255.

View Results

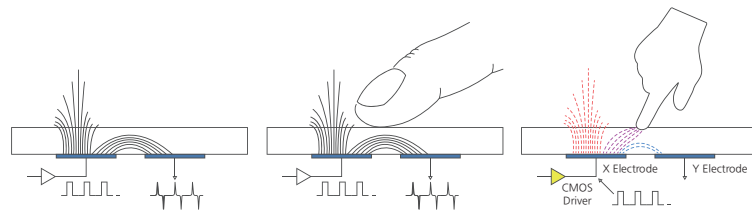


Red LEDs connected to pin 13 are not only lit/off outputs, but three-color LEDs connected to pins with PWM function can also be brightness controlled as well as on and off. Once the RGB_LED_Fading program is uploaded, you can see that it is getting brighter and darker in the order of red, green and blue LEDs. In addition, a random mix of RGB colors using the random function shows 20 colors..

Capacitive Touch Sensor



The touch module used in the coding array starter kit used a capacitive touch sensor. A touch-piece consisting of metal from a capacitive touch sensor has a small amount of current between the outgoing and incoming electrodes and is (operating standby). When a body such as a finger touches a touch surface, part of the electrical power flowing toward the receiving electrode moves to the body, which weakens the electric field detected by the receiving electrode. A slight touch of the human body can detect a slight change in the capacitance and indicate a HIGH or LOW value.



The operating voltage of the touch sensor is 2.0 to 5.5 V and the response time is 60 milliseconds to 220 milliseconds. When the module is energized, the power LED turns on. The notification LED turns on while the body is in contact with the touch sensor and continues to read the HIGH value. If there is no physical contact, the notification LED turns off and reads the LOW value.

Touch sensors are often used for hand-touch smartphone screens without using touch pens.



```

1  /* The touch sensor is a sensor that returns a digital input value when the body touches it.
2  * Can be used as a button switch.
3  * Short delay time will not count the number of touch accurately.
4  * This sketch will learn how to use the touch sensor by default and how to count the number of times the sensor
   has been pressed.
5  */
6
7  #define Touch 7      // Electrostatic Touch Sensor to 7
8
9  int touchCounter = 0; // Variables that store the number of times a touch sensor is pressed
10 int lastTouchState = 0; // Read and save the previous button switch status
11
12 void setup() {
13   pinMode(Touch, INPUT); // Set the touch sensor connected to pin 7 to input
14   Serial.begin(9600); // Starts serial communication at 9600 speeds
15 }
16
17 void loop() {
18   int touchState = digitalRead(Touch); // Read touch sensor switch values and store them in touchState
19
20   if (touchState != lastTouchState) { // Touch sensor status has changed
21     if (touchState == HIGH) { // When the touch sensor is pressed,
22       touchCounter++; // Increase the number of touch sensors pressed
23       Serial.println("TOUCHED"); // Write "TOUCHED" in the serial window and replace lines
24       Serial.print("number of touch sensor pushes: "); // "-" to the cereal window.
25       Serial.println(touchCounter); // Connect and press and replace touch sensor
26     } else { // If the touch sensor has changed from TOUCHED to not touched
27       Serial.println("not touched"); // Write "not touched" in the serial window and replace lines
28     }
29     delay(100);
30   }
31   lastTouchState = touchState; // Use current touchState as lastTouchState in the next loop
32 }

```

#define Constant name value : One of the pre-processing statements processed before program compilation is named constant value (you cannot change the data value while the program is running). Constants created in Define are compiled with all the constants of the source code replaced with values, so they do not take up memory. Caution)Do not insert '=' between constant life and value. Don't use a semicolon at the end

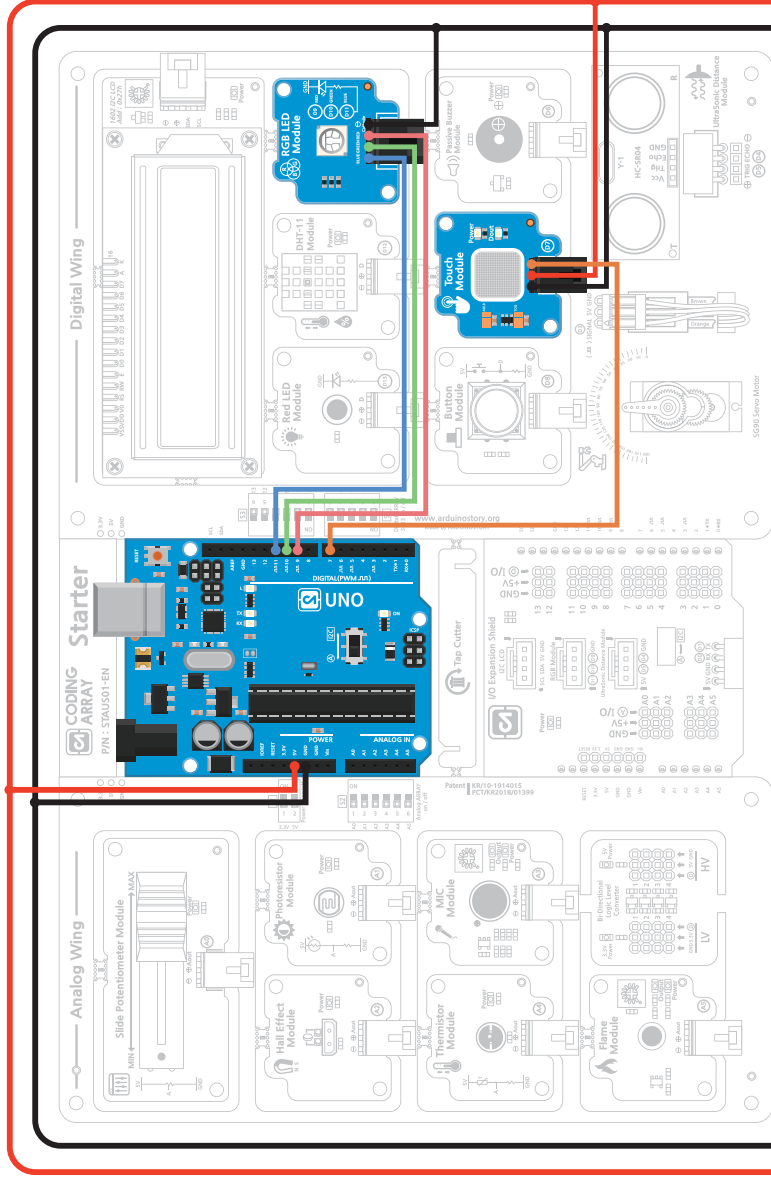
void function: Variables declared within { } are recognized as regional variables only in brackets.

```
void loop() { int touchState = digitalRead(Touch);
```

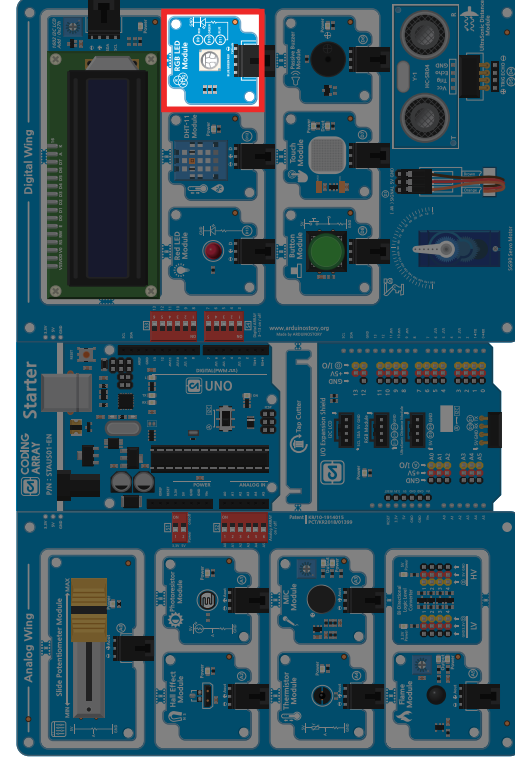
Caution) If a variable is declared before setting the void, use the variable in all parts of the program. :

global variable

touchCounter++; Increase the value of the touchCounter variable by 1.



Touch sensors connected to pin 7 digital can be used like button switches as sensors that return HIGH (1) digital input values when the body touches them and LOW (0) digital input values if the body does not touch them. Continue to return the HIGH input value while the body touches it, but by adjusting the delay time, you can count the number of touches.



View Results



After you upload the sketch file, open the serial monitor. Each time you touch a touch sensor, you can see in the serial window that the number of clicks increases with the message "TOUCHED." When the touch sensor is released, a "not touched" message will be displayed.



```

1  /* This sketch shows a touch sensor
2   Touch 1, 2 and 3 to light up the green LED.
3   Touch four times to show the LED is off.
4   This allows users to set the desired brightness mood using the touch sensor.
5  */
6
7  #define Touch 7      // Connect the capacitive touch sensor to 7.
8
9  const int greenPin = 10 ;    // Green LED to No.10
10
11 int touchCounter = 0;    // Variables that store the number of times a touch sensor is pressed
12 int lastTouchState = 0;    // Read and save the previous Touch Sensor status
13 int alanlogValue;    // Set analog value of LED (0-255)
14
15 void setup() {
16   pinMode(Touch, INPUT);    // Set the touch sensor connected to pin 7 to input
17   pinMode(greenPin, OUTPUT); // Set pin 11 to output
18   Serial.begin(9600);    // Starts serial communication at 9600 speeds
19 }
20
21 void loop() {
22   int touchState = digitalRead(Touch); // Read touch sensor switch values and store them in touchState
23
24   if (touchState != lastTouchState) { // Touch sensor status has changed
25     if (touchState == HIGH) {    //
26       touchCounter ++;          // Increase the number of touch sensors pressed
27       Serial.println("TOUCHED"); // Write "TOUCHED" in the serial window and replace lines
28       Serial.print("number of touch sensor pushes: "); // "-:" to the cereal window
29       Serial.println(touchCounter); // Connect and press and replace touch sensor
30
31       // Use current touchState as lastTouchState in the next loop
32       lastTouchState = touchState;
33
34       if (touchCounter % 4 == 0) { // Touch sensor presses 0
35         alanlogValue = 0;
36         Serial.println("OFF");

```

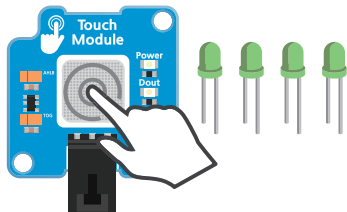
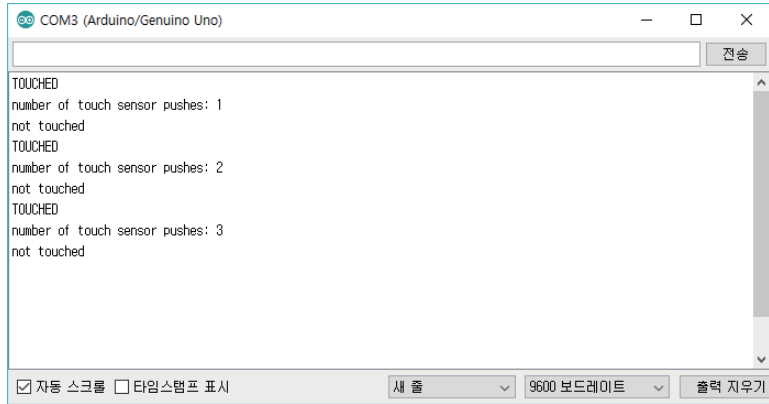
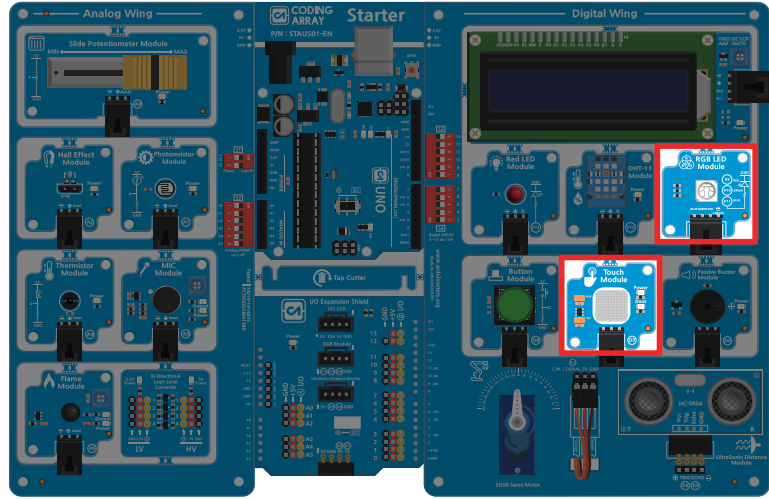
```

37   }
38
39   if (touchCounter % 4 == 1) { // Touch sensor presses 1
40     alanlogValue = 85;
41     Serial.println("First level");
42   }
43
44   if (touchCounter % 4 == 2) { // Touch sensor presses 2
45     alanlogValue = 170;
46     Serial.println("Second level");
47   }
48
49   if (touchCounter % 4 == 3) { // Touch sensor presses3
50     alanlogValue = 255;
51     Serial.println("Third level");
52   }
53
54   // Press the touch sensor to turn on the changed analog value.
55   analogWrite(greenPin, alanlogValue);
56
57   } else {          // If the touch sensor has changed from TOUCHED to not touched
58     Serial.println("not touched"); // Write "not touched" in the serial window and replace lines
59   }
60   delay(100);
61 }
62
63 // Use current touchState as lastTouchState in the next loop
64 lastTouchState = touchState;
65 }

```

Using the remainder of the touchCounter divided by four values, the remaining values are only displayed in four different levels, so you can set the number of touchings divided by four levels.

View Results



After you upload the sketch file, open the serial monitor. You can verify that the green LED's brightness increases when you press the touch sensor 1st, 2nd, and 3rd, and that the LED turns off when you press the fourth time. Using this method, it is possible to implement brightness adjustment such as mood using touch sensor.

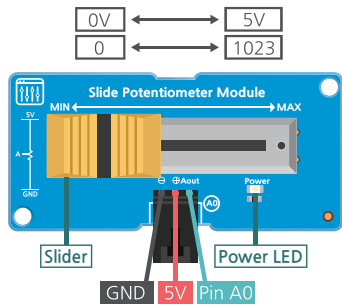
**CODING
ARRAY**
STARTER KIT FOR ARDUINO

7

Read slide variable resistance value with analog input

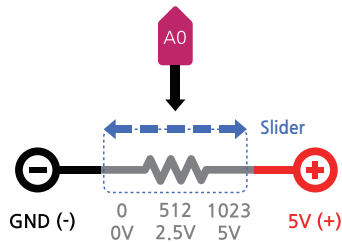
CHAPTER 2

Slide Potentiometer



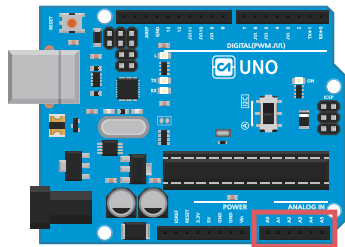
Resistance is enough to interfere with the flow of electric charges.

Unlike a typical set of values for resistance, variable resistance is also called potentiometer and voltage divider and is either turned around the center fireplace, or adjusted by pushing the slider left and right. The coding array start kit uses a slide variable resistance module

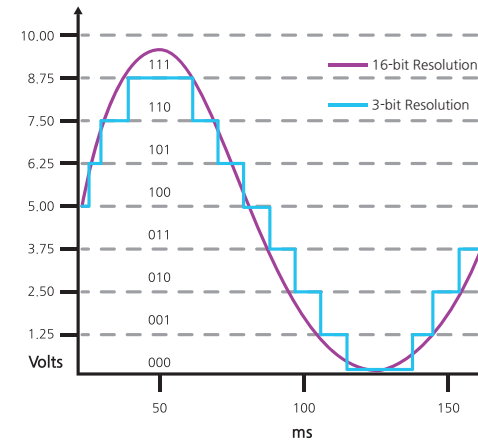


As you push the slider left and right, the resistance values change according to the position, and you return the value to the analog input pin connected to Aout by converting the changing voltage values between 0 V and 5 V to the analog input values. Analog values are read using the `analogRead()` function. Examples of slide variable resistance, such as volume control and boiler temperature control, are found in everyday life..

Let's learn about analog input.



Arduino contains 6 analog to digital converters (ADC) so that analog values can be read from A0 to A5 pins using the `analogRead` function



If the analog-to-digital converter has a 3-bit resolution, it means that it is distinguished by converting input voltages from 0 to 5 V into 2^3 digital signal stages. The higher the disassembly ability, the higher the accuracy, the better the analog value can be measured.

Arduino's analog-to-digital converter converts analog signals into 10-bit disassembly capabilities ($2^{10} = 1024$). This means that the input voltage of 0 to 5 V is converted into a digital signal and returned as an integer between 0 and 1023 and the voltage can be distinguished in units of 4.9 mV.

0V	5V
0	1023

Analog-to-digital transducers require a certain amount of time (Conversion Time) to change the analog input value to digital. Arduino takes about 100 microseconds (0.0001 seconds), so you can read up to 10,000 analog input values per second. To read analog values, `analogRead` can be declared as a table. If there is no analog input, the A0 to A5 analogue input pins can be used the same as the digital pins (pins 14 to 19)...

<code>analogRead(0)</code>	<code>analogRead(A0)</code>	<code>analogRead(14)</code>
<code>analogRead(1)</code>	<code>analogRead(A1)</code>	<code>analogRead(15)</code>
<code>analogRead(2)</code>	<code>analogRead(A2)</code>	<code>analogRead(16)</code>
<code>analogRead(3)</code>	<code>analogRead(A3)</code>	<code>analogRead(17)</code>
<code>analogRead(4)</code>	<code>analogRead(A4)</code>	<code>analogRead(18)</code>
<code>analogRead(5)</code>	<code>analogRead(A5)</code>	<code>analogRead(19)</code>

Let's find out about voltage distribution.

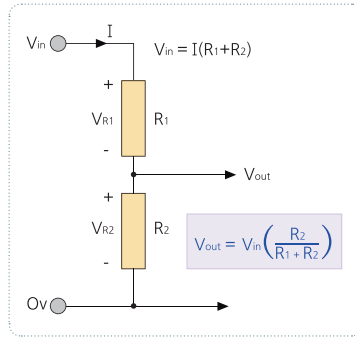
Calculate the Voltage Divider as follows:

When a resistance is connected in series, the total resistance value is the sum of each resistance, and the current flowing to each resistor is equal to the supply current.

$$I = I_{R_1} = I_{R_2}$$

In addition, the sum of the voltages applied to each resistor is equal to the supply voltage..

$$V_{in} = V_{R_1} + V_{R_2}$$



According to Ohm's Law $V=IR$,

$$V_{in} = I(R_1 + R_2)$$

$$I = I_{R_2} = \frac{V_{in}}{R_1 + R_2}$$

$$V_{R_2} = \frac{V_{in}}{R_1 + R_2} \times R_2$$

$$V_{R_2} = \frac{R_2}{R_1 + R_2} \times V_{in}$$

The size of the resistance increases as the length of the resistance increases, and the larger the cross-sectional area becomes smaller.

CAK Starter Code > 07_01_SlidePotentiometer

```

1  /* Change the resistance value by pushing the variable resistance from side to side.
2  As the resistance changes, 0 to 1023 analog values are entered as A0 pins.
3  Analog values are converted to voltages and expressed as values on the serial monitor.
4  Can use a serial plotter to express it in graphs.
5  */
6
7  const int potentiometerPin = A0; // Output value of variable resistance is read from A0
8  const int redLED = 13;
9  const int threshold = 400;
10
11 void setup() {
12   pinMode(redLED, OUTPUT); // Set red LED to output
13   Serial.begin(9600); // Starts serial communication at 9600 speed

```

```

14 // Analog input/output pins need not be declared..
15 }
16
17 void loop() {
18   int analogValue = analogRead(potentiometerPin); // Read the analog value of variable resistance and
19   // store it in analogValue
20   // AnalogValue stores integer values in the range 0 to 1023.
21
22   float voltage = analogValue * [5.0 / 1023.0]; // Convert Analog Readings to Volts 0 to 5 V
23   // Store in float variable because the math result value is real
24
25   //Serial.print("Analog Value : ");
26   //Serial.println(analogValue); // Indicate the value of analogValue one line in the serial window.
27   Serial.print("Voltage : ");
28   Serial.println(voltage); // Print the converted voltage value one line in the serial window
29
30   if (analogValue > threshold) { // If the value stored in analogValue is greater than 400
31     digitalWrite(redLED, HIGH); // Turn on the LED.
32   } else { // If the value stored in analogValue is less than or equal to 400
33     digitalWrite(redLED, LOW); // Turn off the LED.
34
35     delay(1); // Wait 1 millisecond to read reliably
36   }

```

analogRead (Pin); Read the analog value from the specified pin (read only about 10,000 times per second). Respond to 0 - 5 V voltage with an integer value of 0 - 1023.

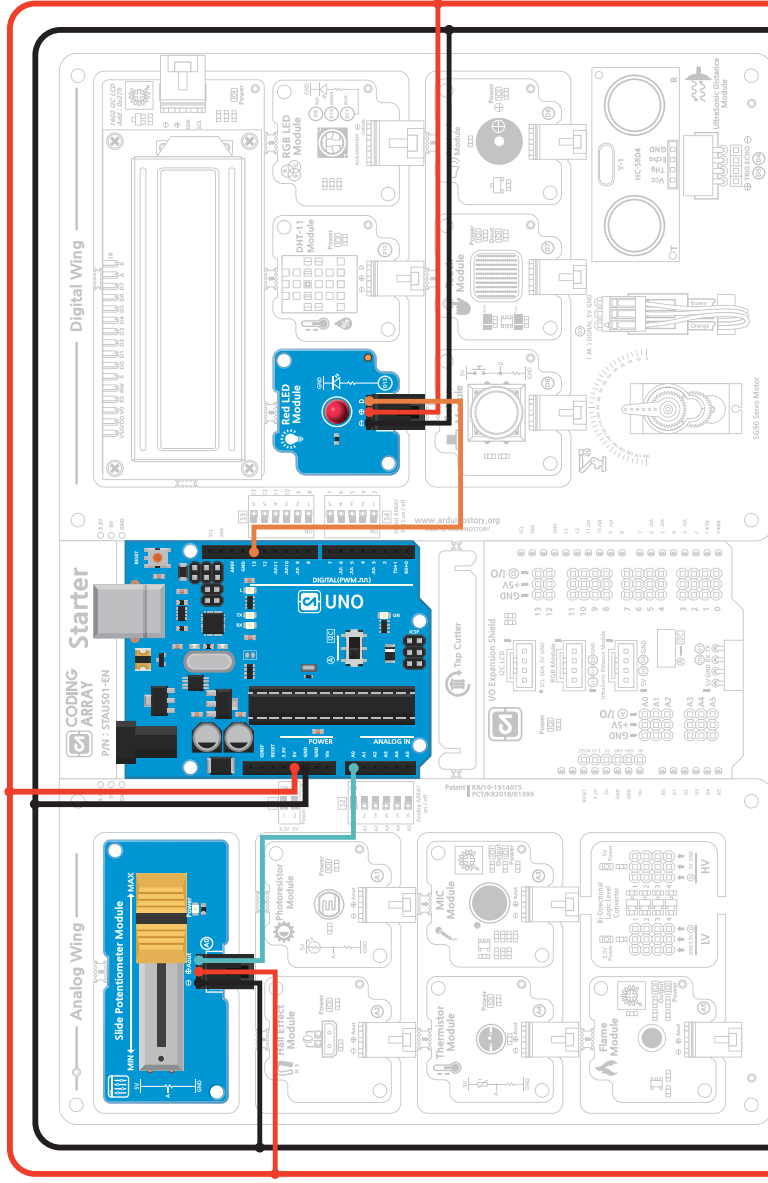
float Variables; declared for the purpose of storing decimal mistakes. When performing math operations with float, you must add a decimal point. (Example: 5.0 /1023.0) Otherwise (e.g. 5 /1023= 0) treated as an integer int.

CAUTION!

- The analog input/output does not have to be set in pinMode separately.
- Click Menu Bar > Tools > Serial Plotter (Ctrl+Shift+L) to open the serial plotter. During the output of the serial plotter, the serial monitor window does not open simultaneously..

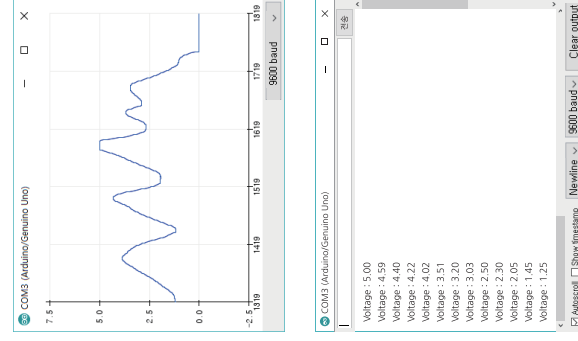
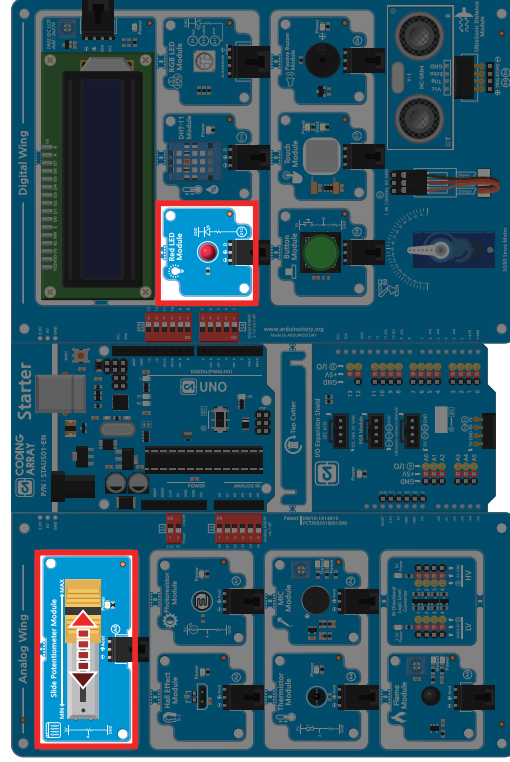
Tools Help	
Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L

CODING ARRAY CIRCUIT



Touch sensors connected to pin 7 digital can be used like button switches as sensors that return HIGH (1) digital input values when the body touches them and LOW (0) digital input values if the body does not touch them. Continue to return the HIGH input value while the body touches it, but by adjusting the delay time, you can count the number of touches.

View Results



After uploading a sketch file, moving the slider of variable resistance to the left and right changes the analog input value, and you can graphically represent the changing voltage values in the serial plotter. Also, the red LED value turns on when the miniset value is higher than the set value.

CAK Starter Code > 07_02_SlidePotentiometer2

```

1  /* As the variable resistance value increases, the color of the RGB LED changes to red->green->blue... */
2
3  int potPin = A0;    // Output of variable resistance connected to analogue pin A0
4  int potVal = 0;    // Variables that store analog [0-1023] values read from variable resistance
5
6  const int redPin = 9;    // Red LED No. 9
7  const int greenPin = 10; // Green LED No. 10
8  const int bluePin = 11; // Blue LED No. 11
9
10 int redV = 0;        // Set red LED analog value [0-255]
11 int greenV = 0;     // Set green LED analog value [0-255]
12 int blueV = 0;     // Set blue LED analog value [0-255]
13
14 void setup() {
15   pinMode(redPin, OUTPUT); // Set pin 9 to output
16   pinMode(greenPin, OUTPUT); // Set pin 10 to output
17   pinMode(bluePin, OUTPUT); // Set pin 11 to output
18 }
19
20 void loop() {
21   potVal = analogRead(potPin); // The analog output value of variable resistance is read from the A0 pin [0-1023].
22   int ledLevel = map(potVal, 0, 1023, 0, 255); // Converts the analog input value to the analog output value [0-255]
23
24   if (potVal < 341){ // When the value of variable resistance is 1 divided into three stages [0-340]
25     redV = 255 - ledLevel; // The red is getting lighter
26     greenV = ledLevel;    // The green is getting darker
27     blueV = 1;           // Blue has no effect
28   }
29
30   else if (potVal < 682) { // When the value of variable resistance is 2 divided into three stages [341-681]
31     redV = 1;            // Red has no effect
32     greenV = 255 - ledLevel; // The green is getting lighter
33     blueV = ledLevel;    // The blue is getting darker
34   }
35
36   else { // When the value of variable resistance is 3 divided into three stages [682-1023]
37     redV = ledLevel;     // The Red is getting darker
38     greenV = 1;         // Green has no effect

```

```

39   blueV = 255 - ledLevel; // The blue is getting lighter
40 }
41
42 analogWrite(redPin, redV); // Write values to red pins
43 analogWrite(greenPin, greenV); // Write values to green pins
44 analogWrite(bluePin, blueV); // Write values to blue pins
45 }

```

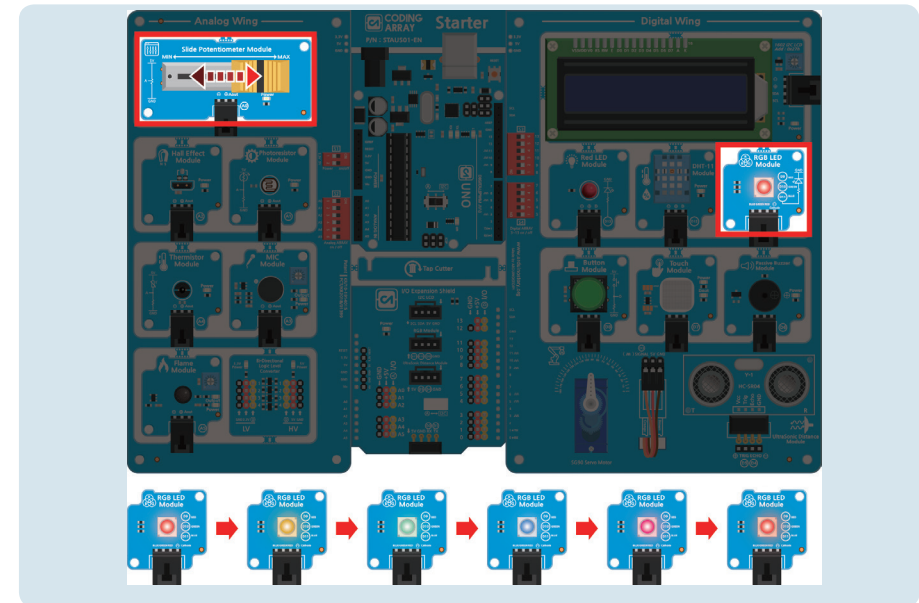
map (number to convert, current maximum value, current maximum value, current maximum value to be converted, maximum value to be converted)

A function that simply represents the value that is converted using a proportional expression. The value being converted is expressed as an integer (including negative numbers), and the decimal number is not rounded up.

```
int ledLevel= map(potVal,0,1023,0,255);
```

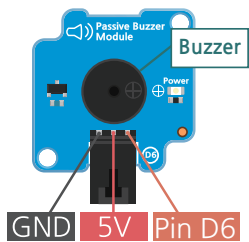
Converts the number of 0 to 1023 accepted by portVal to a value between 0 and 255.

View Results



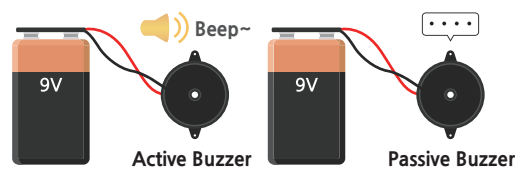
After uploading the sketch, pushing the slider of variable resistance left and right changes the analog input value. As the resistance value increases, you can see that the color of the RGB LED changes to red -> green-> blue..

Passive Buzzer



Piezo buzzer is a small speaker that makes sound using piezo effect that creates vibrations when electricity is released. The downside is that the sound isn't loud, but you can also play music if you manipulate it carefully. The piezo buzzer is polar and should be connected to the (+) pole by the side that reads (+) on the top or has a small groove dug. The piezo buzzer is largely divided into active buzzer and passive buzzer. The active buzzer has built-in circuits, so it is often used to alert people as it makes only one sound of a certain frequency when current flows. A manual buzzer is a buzzer that makes sound through a tone function that can produce frequencies between 31 and 65535 Hz.

The best way to distinguish between these two buzzers is to connect the two pins of the buzzer to the GND and 5V of the Arduino board to the active buzzer if a constant sound occurs and the manual buzzer if there is no sound. The coding array kit uses a manual buzzer module for digital No. 6 pin, so you can play melodies..



To perform melodies with a manual buzzer, use the tone (pin number, negative frequency, negative duration) function, which uses integer values as follows:

Frequency by octave and pitches (in Hz)

octave pitches	1	2	3	4	5	6	7	8
C (do)	33	65	131	262	523	1047	2093	4186
C#	35	69	139	277	554	1109	2217	4335
D (re)	37	73	147	294	587	1175	2349	4699
D#	39	78	156	311	622	1245	2489	4987
E (mi)	41	82	165	330	659	1319	2637	5274
F (fa)	44	87	175	349	698	1397	2794	5588
F#	46	93	185	370	740	1480	2960	5920
G (sol)	49	98	196	392	784	1568	3136	6272
G#	52	104	208	415	831	1661	3322	6645
A (La)	55	110	220	440	880	1760	3520	7040
A#	58	117	233	466	932	1865	3729	7459
B (Si)	62	123	247	494	988	1976	3951	7902

Let's find out how to give a constant name to a frequency value with #define..

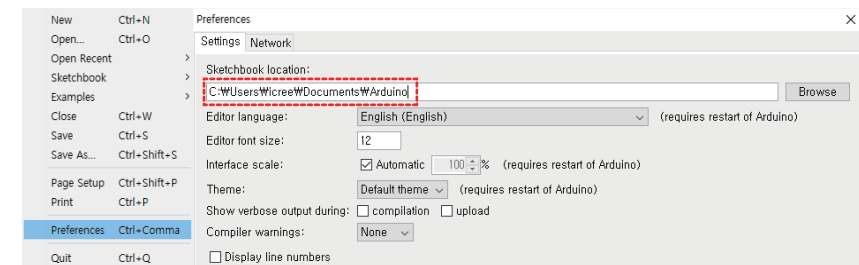
Each sound meter has its own frequency of shaking and should be predefined so that the frequency (in units Herz HZ) value of the sound meter. Write the frequency required to play the melody by defining it as the #define constant name value (e.g. #define NOTE_C1 33) before {}. #define is a function that gives a name to a constant value before a program compilation. Be careful not to put "=" between the constant and the value, and not to use the semicolon at the end..

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
```

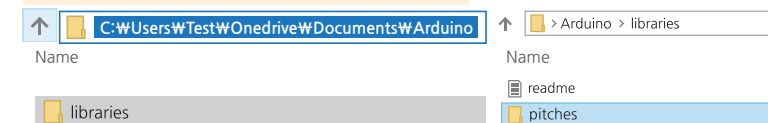
Let's learn how to save the pitches.h header file..

If you find it difficult to put a sound frequency value into a program each time, you can also create a separate library of files for the sound frequency value. Let's learn how to create and store a sound frequency file in a file called "pitches.h".

Way 1. File > Preferences > Open the folder in the Sketchbook location..



Create pitches folder under libraries folder.

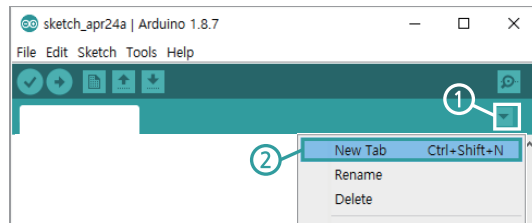


When you open the file > Example > 02. Digital > ToneMelody in the pitches folder, copy the pitches.h on the right tab and save it as a file and make it a library.

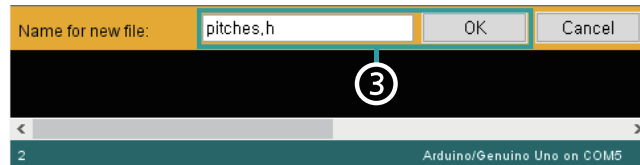


You can also edit and use it as needed. Bring up header file to #include "pitches.h" before void setup {}.

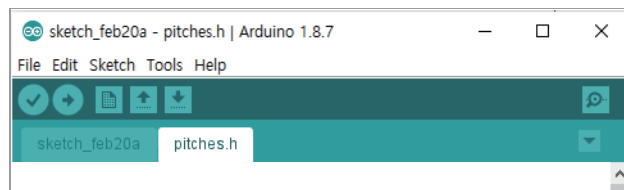
Way 2. New File > New Tab



Create a new tab by tapping the bottom of the serial monitor icon..



Type pitches.h in the name for the new file and click OK..



When you open the **file > Example > 02. Digital > ToneMelody**, copy and paste the pitches.h on the right tab to save. The pitches.h header file is stored in the same folder as the program. This file can also be edited and used as needed. Stored header file is called #include "pitches. h" before void setup {}.

CAK Starter Code > 08_PassiveBuzzer

```

1  /* This sketch plays a note through a buzzer connected to pin 6.
2  * The * tone() function calls up the pitches.h header file to give a given frequency sound.
3  * Play a familiar 'school bell' song to us.
4  * In this sketch, the code is inserted in the setup part and played only once.
5  * If the loop part is coded, the performance may be repeated.
6  */
7
8  #include "pitches.h"
9
10 int buzzer=6;    // Connect the Piezo buzzer to No. 6.
11                // the pitch of playing
12 int melody[]={NOTE_G7,NOTE_G7,NOTE_A7,NOTE_A7,NOTE_G7,
13              NOTE_G7,NOTE_E7,NOTE_G7,NOTE_G7,NOTE_E7,
14              NOTE_E7,NOTE_D7,0,NOTE_G7,NOTE_G7,NOTE_A7,
15              NOTE_A7,NOTE_G7,NOTE_G7,NOTE_E7,NOTE_G7,
16              NOTE_E7,NOTE_D7,NOTE_E7,NOTE_C7,0
17              };
18                // Sound length, 4 = crotchet, 2 = minim
19 int noteDurations[]={4,4,4,4,4,4,2,4,4,4,4,3,1,4,4,4,4,4,2,4,4,4,4,3,1};
20
21 void setup() {
22   for(int thisNote =0; thisNote <26; thisNote++)
23   {
24     int noteDuration =1000 /noteDurations[thisNote];
25     tone(buzzer, melody[thisNote], noteDuration);    // Connect the Piezo buzzer to No. 6
26     int pauseBetweenNotes =noteDuration *1.30;    // phonetic delimiting
27     delay(pauseBetweenNotes);    //delay
28     noTone(buzzer);    // Stop playing music
29   }
30 }
31
32 void loop() {
33 }

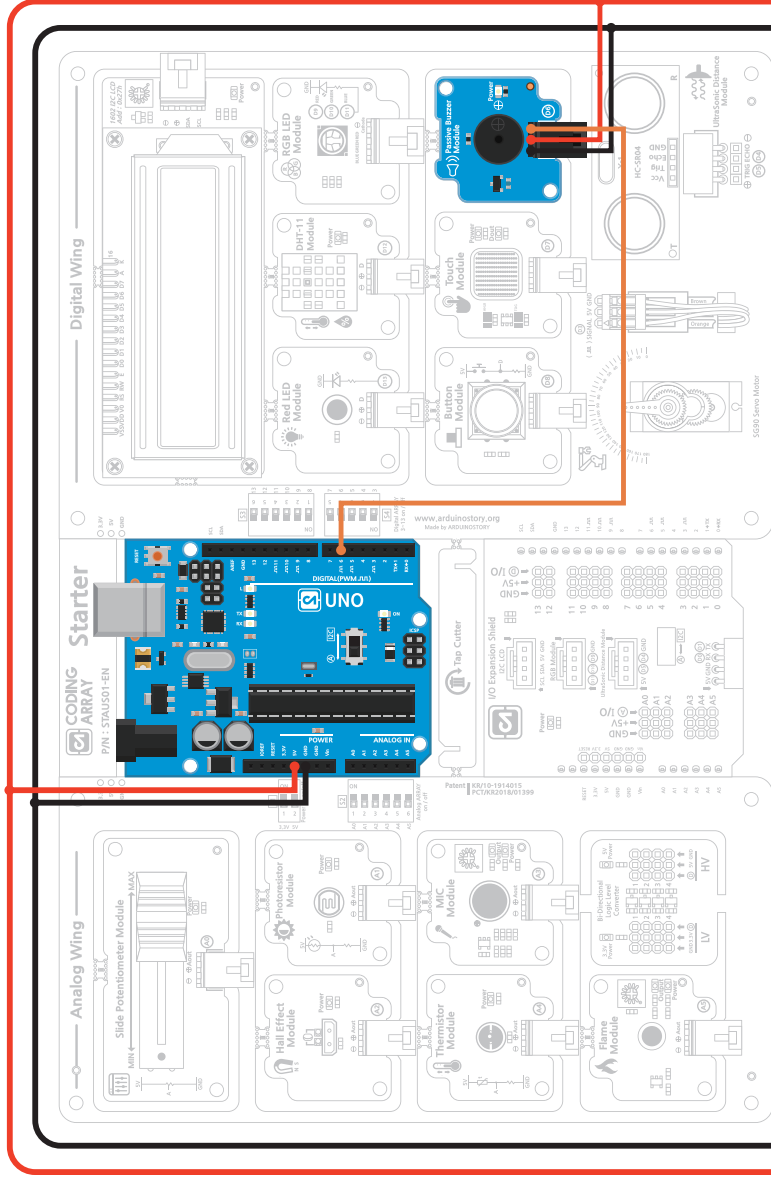
```



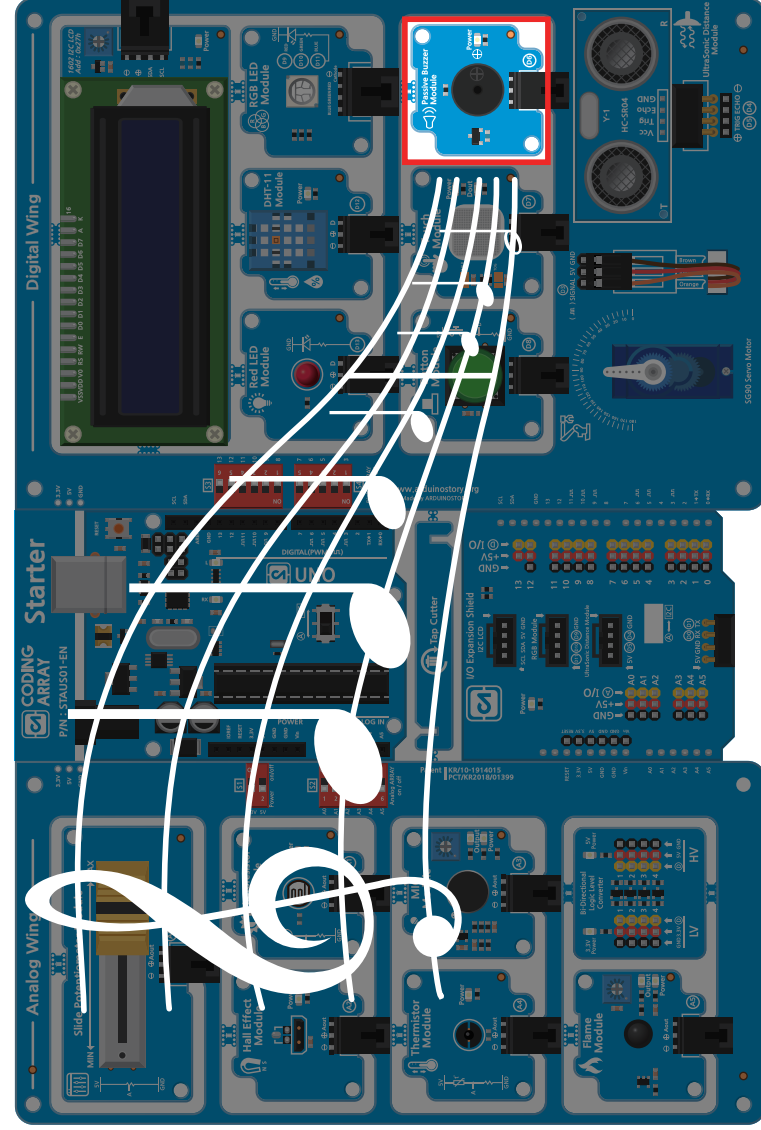
tone (pin number, sound frequency, sound duration); To make a sound of a specific frequency.
 Sound frequency: Hertz unit (Hz), rounded to the standard frequency to add an integer.
 Sound duration: Milliseconds, unsigned long type, and optionally.
 Only one note can be generated at a time.
 The tone() function prevents PWM output on pins 3 and 11.



noTone (Pin Number); Stop the waveform generated by tone().
 In order to play different scales on different pins, you must call noTone before calling the next pin.



Play the melody using the tone() function on the manual buzzer connected to digital pin 6.



Once the sketch is uploaded, you can check out the 'school bell' song. If you want to repeat the performance, you can insert the code into the void loop.
Caution: passive buzzer can't produce frequency specific sounds.

View Results

Let's find out about voltage distribution.

The array allows you to declare as many variables as in []. If the length of the array is omitted in [], the compiler will determine the length of the array by referring to the number of values in the list. You can also initialize the values with the array declaration at the same time..

int array name [collection length]

The number of int variables is declared side by side.

Array name[0]=value1; store value1 on first element of array

Array name[1]=value2; store value2 on second element of array

A number in [] is called an index, and the index value begins at zero..

int array name [] = {value1, value2, ... }

If the length of the array is omitted in [], the compiler will refer to the number of values in the list.

Determine the length of the array.

You can also initialize the values with the array declaration at the same time..

```

      Index  0    1    2    3    4
      ↓      ↓    ↓    ↓    ↓
Array Name melody[] = { NOTE_G7, NOTE_G7, NOTE_A7, NOTE_A7, NOTE_G7 }
  
```

```

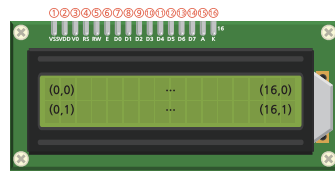
melody[0] ↔ NOTE_G7
melody[1] ↔ NOTE_G7
melody[2] ↔ NOTE_A7
melody[3] ↔ NOTE_A7
melody[4] ↔ NOTE_G7
  
```

**CODING
ARRAY**
STARTER KIT FOR ARDUINO

LCD (Liquid Crystal Display)

Piezo buzzer is a small speaker that makes sound using piezo effect that creates vibrations when electricity is released. The downside is that the sound isn't loud, but you can also play music if you manipulate it carefully. The piezo buzzer is polar and should be connected to the (+) pole by the side that reads (+) on the top or has a small groove dug.

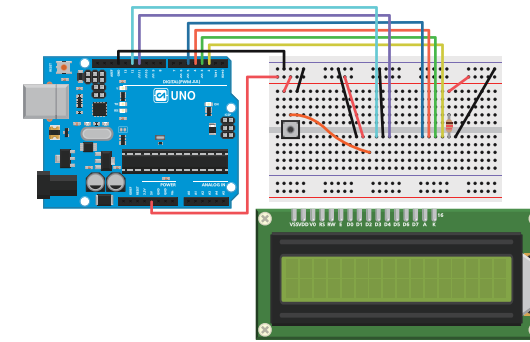
The piezo buzzer is largely divided into active buzzer and passive buzzer. The active buzzer has built-in circuits, so it is often used to alert people as it makes only one sound of a certain frequency when current flows. A manual buzzer is a buzzer that makes sound through a tone function that can produce frequencies between 31 and 65535 Hz.



1	V _{SS}	GND connection	power
2	V _{DD}	5V connection	
3	V ₀	the darkening of letters	Setting
4	RS	Select space to store LCD values	
5	RW	Select a value for the LCD in read/write mode	
6	E	To write a value to the LCD	data
7	DB0	Data input/output pin Used when LCD and Arduino exchange prices.	
8	DB1		
9	DB2		
10	DB3		
11	DB4		
12	DB5		
13	DB6		
14	DB7		
15	A	5V connection	Background brightness control. No pins when no background lights or no background brightness is required
16	K	GND connection	

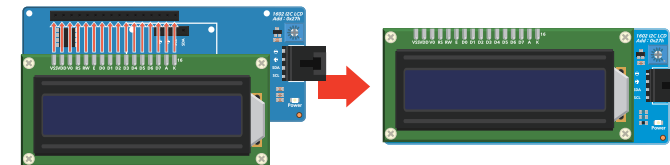
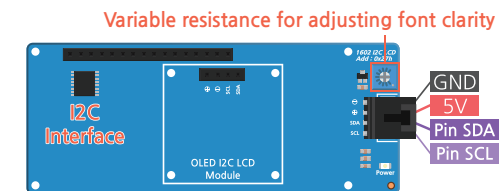
Inter-Integrated Circuit Interface

The basic wiring for using a 1692 LCD uses a lot of digital pins, as shown in Figure below..

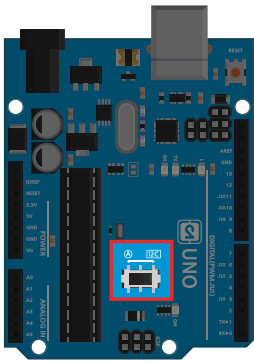


But Arduino Uno has 14 digital input/output pins and six analog input pins. If you want to connect other parts to Uno with 1602 LCD, there may not be enough connecting ports because it requires a lot of pin connections.

To address these issues, the coding array starter kit uses the I2C interface module. The I2C interface module has variable resistance that adjusts the clarity of the writing, so it does not have to be connected to variable resistance..



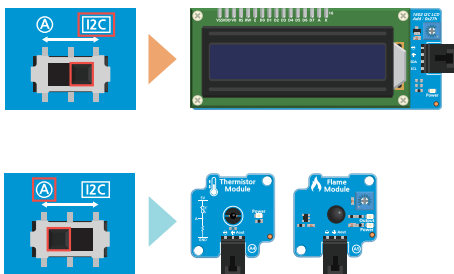
The I2C interface module and 1602 LCD module are connected as shown in Figure above, and the LCD can be removed and used as a normal 1602 LCD..



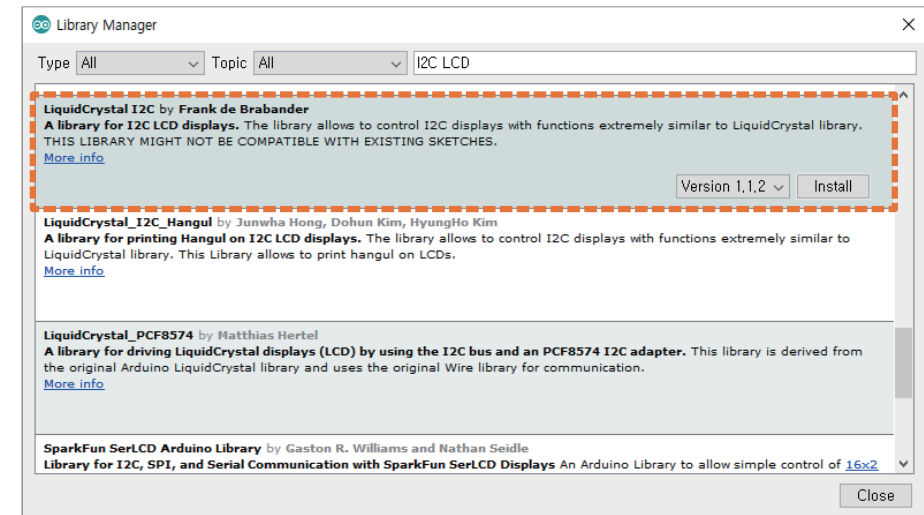
I2C (Inter-Integrated Circuit) is an NFC that can connect a 1 master (Arduino) : multiple slave (sensor modules) in one direction using a SCL (Serial Clock) pin and SDA (Serial Data) pin with full-up resistance connected. Arduino can use SDA, SCL pins as I2C communication pins or analog A4, A5 pins as functions of SDA and SCL respectively. In the starter kit, the SDA and SCL pins of the Uno board were placed for easy selection with the slide switch in the center of the Arduino Uno board.

CAUTION!

You cannot use the I2C communication interface and the A4, A5 pins at the same time on the Arduino board. Therefore, the thermistor module and flame sensor module cannot be used simultaneously with the I2C LCD in the starter kit. Therefore, when using I2C LCD, define the module of use by placing the slide switch left and right as shown in Figure below..



For I2C communication, the Wire library must be added. To add a library, click **Sketch > Include Library > Manage Libraries** to run the Library Manager. Search for "I2C LCD" in the search box and install "LiquidCrystal I2C by Frank de Brabander."



The default address is 0x27 (hexadecimal value), but sometimes an error occurs when using a module, the address must be checked. Address scanning is recommended first to find the address of the I2C LCD interface module you want to use.

**CODING
ARRAY**
STARTER KIT FOR ARDUINO



```

1  /* This sketch is designed to use 1602 LCDs using I2C communication.
2  * Scans the address to show the results on the serial monitor.
3  */
4
5  #include <Wire.h>      // Includes the Wire Library for I2C communication.
6
7  void setup() {
8    Wire.begin();
9    Serial.begin(9600);    // Starts serial communication at 9600 speeds
10   while(!Serial);      // Wait for the serial monitor.
11   Serial.println("\nI2C Scanner");
12 }
13
14 void loop() {
15   byte error,address;
16   int nDevices;
17   Serial.println("Scanning...");
18   nDevices =0;
19
20   for (address =1; address <127; address++) {
21     // The i2c_scanner has determined whether the device has approved the address.
22     // Use the Wire.endTransmission return value to know.
23     Wire.beginTransmission(address);
24     error =Wire.endTransmission();
25
26     if (error ==0) {
27       Serial.print("I2C device found at address 0x");
28       if(address<16)
29         Serial.print("0");
30       Serial.print(address,HEX);
31       Serial.println(" !");
32       nDevices++;
33     }
34     else if(error==4) {
35       Serial.print("Unknown error at address 0x");
36       if (address<16)
37         Serial.print("0");
38       Serial.println(address,HEX);
39     }
40   }
41   if (nDevices ==0)
42     Serial.println("No I2C devices found\n");
43   else

```

```

44   Serial.println("done\n");
45
46   delay(5000);    // Wait five seconds for the next scan..
47 }

```

Let's learn about the method used by LiquidCrystal_I2C.h..

LiquidCrystal_I2C lcd (0x27,16,2); The method under I2C communication address, 16-cand 2-line lcd object creation is for example if the object was named lcd.

lcd.init();	Initialize LCD.
lcd.backlight();	Turn on the LCD backlight.
lcd.noBacklight();	Turn off the LCD backlight.
lcd.setCursor(ROWS);	Sets the position of the cursor.
lcd.print(data, BASE);	Print the text on the LCD screen. Data : Data to be printed, char, byte, int, long, stringable. Characteristic data is displayed in "ln" BASE (Optional) : The standard by which a number is printed. BIN (binary), DEC (decimal), OCT (8 decimal), HEX (16 decimal)
lcd.scrollDisplayLeft();	Scroll text and cursors one space to the left
lcd.scrollDisplayRight();	Scroll text and cursors one space to the right
lcd.noDisplay();	Turn off the screen. The string on the face disappears, but the contents remain in internal memory. lcd.display(); when a function is called, the string is revived.
lcd.display();	Turn on the screen. lcd.noDisplay(); and lcd.display(); two functions can have a flicker effect on the entire screen.
lcd.autoscroll();	Scroll text and cursors one space to the left
lcd.noAutoscroll();	Scroll text and cursors one space to the right
lcd.clear();	Clear the LCD screen and place the cursor in the upper left corner.
lcd.cursor();	Indicates the underscore of the position in which the following characters are written
lcd.noCursor();	Hide the LCD cursor.
lcd.rightToLeft();	Set the direction of the string used for the LCD from right to left. The default value is left to right. It does not affect previously printed text.
lcd.leftToRight();	Set left to right direction of string written to LCD. It does not affect previously printed text.
lcd.write(data);	Text is written on the LCD.
lcd.createChar(number,data);	Create a custom character to use for LCD. Up to 8 characters 5*8 pixels are supported. Numeric : Numbers from 0 to 7. Specify the number of characters to create. Data: Pixel Data for Text
lcd.noBlink();	Turn off the blinking LCD cursor.
lcd.blink();	Turn on the blinking LCD cursor.
lcd.home();	Position the cursor in the upper left corner of the LCD and output the string.

Learn how to print characters on an LCD using method and how it works..



CAK Starter Code > 09_02_1602LCD_HelloCAK

```
1  /* This sketch shows Hello! ^_^ ! Coding Array Kit for 1602 LCDs using I2C communication
2  * Show the string, using scrollDisplayRight and scrollDisplayLeft methods
3  * Scrolls to the left and right.
4  * Flashes the screen using noDisplay and display method to indicate that one loop is complete.
5  */
6
7  #include <Wire.h>
8  #include <LiquidCrystal_I2C.h>
9
10 LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16-kan2 line LCD.
11           // Put the scanned address instead of 0x27.
12 void setup(){
13   lcd.init();
14   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
15   lcd.setCursor(0,0); // first line first column
16   lcd.print("Hello! ^_^!");
17   lcd.setCursor(0,1); // 2nd line first column
18   lcd.print("Coding Array Kit");
19   delay(1000);
20 }
21
22 void loop(){
23   // Scroll 16 length of string to the left
24   for(int positionCounter = 0; positionCounter < 16; positionCounter++) {
25     lcd.scrollDisplayLeft(); // Scroll to the left by one position
26     delay(200); // for 200 milliseconds
27   }
28
29   // String length 16+ Rows 16 Scroll to the right at a total of 32 locations
30   for(int positionCounter = 0; positionCounter < 32; positionCounter++) {
31     lcd.scrollDisplayRight(); // Scroll to the left by one position
32     delay(200); // for 200 milliseconds
33   }
34
35   // Scroll to the left 16 positions to center
36   for(int positionCounter = 0; positionCounter < 16; positionCounter++) {
37     lcd.scrollDisplayLeft(); // Scroll to the left by one position
38     // wait a bit:
39     delay(200); // for 200 milliseconds
40   }
41
```

```
42 // noDisplay and display
43 lcd.noDisplay(); // Turn off the screen
44 delay(500); // for 0.5 second
45 lcd.display(); // Turn on the screen
46 delay(500); // for 0.5 second
47 }
```



CAK Starter Code > 09_03_1602LCD_Autoscroll

```
1  /* This sketch shows that 1602 LCD uses I2C communication.
2  * Numbers from 0 to 9 are displayed on the screen.
3  * Use the autoscroll() and noAutoscroll() methods.
4  * Show how to move all strings left or right.
5  */
6
7  #include <Wire.h>
8  #include <LiquidCrystal_I2C.h>
9
10 LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16 Space 2-line LCD.
11
12 void setup(){
13   lcd.init();
14   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
15 }
16
17 void loop() {
18   lcd.setCursor(0,0); // Position the cursor [0,0] in the upper left corner.
19
20   for(int thisChar = 0; thisChar < 10; thisChar++) {
21     lcd.print(thisChar); // The numbers from 0 to 9 are shown on the LCD.
22     delay(500); // For 0.5 Second
23   }
24
25   lcd.setCursor(16,1); // Position the cursor in the lower right hand corner.
26   lcd.autoscroll(); // Set to auto-scroll
27
28   for(int thisChar = 0; thisChar < 10; thisChar++) {
29     lcd.print(thisChar); // The numbers from 0 to 9 are shown on the LCD..
30     delay(500); // For 0.5 Second
31   }
32
33   lcd.noAutoscroll(); // Automatic Scroll Revocation
34   lcd.clear(); // Clear the screen before going to the next loop.
35 }
```



```

1  /* This sketch shows that 1602 LCD uses I2C communication.
2  * Show one alphabet from a to Z.
3  * 'a through l' is written from left to right.
4  * 'm through r' is written from right to left
5  * 's through z' causes text to appear from left to right again.
6  */
7
8  #include <Wire.h>
9  #include <LiquidCrystal_I2C.h>
10
11  LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16 Space 2-line LCD.
12
13  int thisChar ='a';           // thisChar Save
14
15  void setup(){
16    lcd.init();
17    lcd.backlight();          // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
18    lcd.cursor();            // Turn on the cursor.
19  }
20
21  void loop(){
22    if(thisChar == 'm' ) {    // If thisChar is m, change direction.
23      lcd.rightToLeft();     // From the next letter, mark to the left.
24    }
25
26    if(thisChar == 's' ) {    // If thisChar is s
27      lcd.leftToRight();     // From the next letter, mark to the left.
28    }
29
30    if(thisChar > 'z' ) {     // When thisChar is out of z,
31      lcd.home();            // Go to (0,0)
32      thisChar ='a';         // Restart from the beginning
33    }
34
35    lcd.write(thisChar);     // Indicate thisChar value on LCD.
36    delay(1000);             // for a second
37    thisChar++;              // increase thisChar value one by one
38  }

```



```

1  /* This sketch shows that 1602 LCD uses I2C communication.
2  * Set up a special character or Figure to "I Array Kit !" and smile on the first line of the LCD.
3  * In the second row, the shape of a person who raises and lowers his or her arms is displayed.
4  * Learn how to indicate that the value of A0 variable resistance is constant.
5  */
6
7  #include <Wire.h>
8  #include <LiquidCrystal_I2C.h>
9
10  LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16 Space 2-line LCD.
11
12  // Create special characters
13  byte heart[8] = {
14    0b00000,
15    0b01010,
16    0b11111,
17    0b11111,
18    0b11111,
19    0b01110,
20    0b00100,
21    0b00000
22  };
23
24  byte smiley[8] = {
25    0b00000,
26    0b00000,
27    0b01010,
28    0b00000,
29    0b00000,
30    0b10001,
31    0b01110,
32    0b00000
33  };
34
35  byte frownie[8] = {
36    0b00000,
37    0b00000,
38    0b01010,
39    0b00000,
40    0b00000,

```

```

41 0b0000,
42 0b01110,
43 0b10001
44 };
45
46 byte armsDown[8]={
47 0b00100,
48 0b01010,
49 0b00100,
50 0b00100,
51 0b01110,
52 0b10101,
53 0b00100,
54 0b01010
55 };
56
57 byte armsUp[8]={
58 0b00100,
59 0b01010,
60 0b00100,
61 0b10101,
62 0b01110,
63 0b00100,
64 0b00100,
65 0b01010
66 };
67
68 void setup(){
69   lcd.init();
70   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
71
72   // Define New Characters
73   lcd.createChar(0,heart);
74   lcd.createChar(1,smiley);
75   lcd.createChar(2,frownie);
76   lcd.createChar(3,armsDown);
77   lcd.createChar(4,armsUp);
78
79   lcd.setCursor(0,0); // first line first column
80   // Write text to LCD
81   lcd.print("I ");
82   lcd.write(byte[0]); // Use a heart that is stored at 0 bytes..
83   lcd.print(" Array Kit! ");

```

```

84   lcd.write(byte[1]); // Use a smiley stored in one byte..
85 }
86
87 void loop(){
88   int sensorReading = analogRead(A0); // Read the variable resistance value of A0.
89   int delayTime = map(sensorReading,0,1023,200,1000); // Map resistance values from 200 to 1000
90   lcd.setCursor(4,1); // Set the cursor to the bottom 5th position
91   lcd.write(3); // Draw a person with his arm down on number 3.
92   delay(delayTime); // Delay by variable resistance value
93   lcd.setCursor(4,1); // Set the cursor to the bottom 5th position
94   lcd.write(4); // Draw the person with the arm up stored in number 4.
95   delay(delayTime); // Delay by variable resistance value
96 }

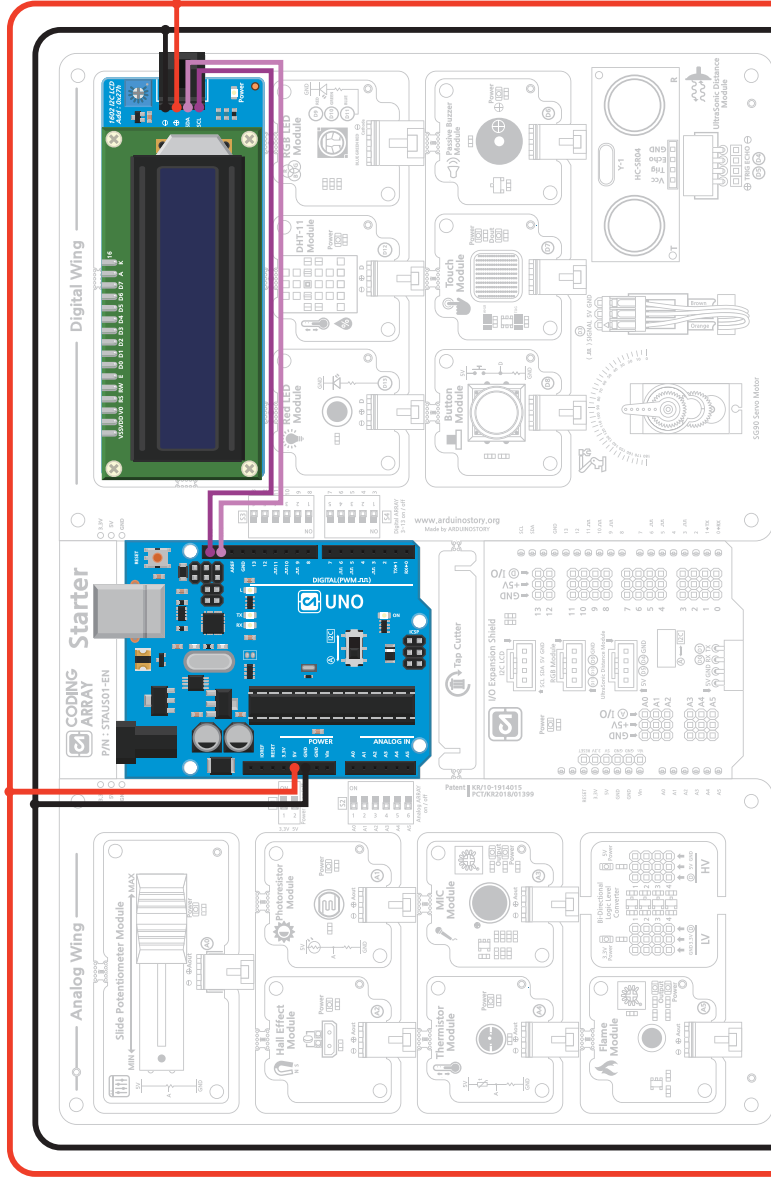
```

`byte` Save the 8-bit signless number from 0 to 255

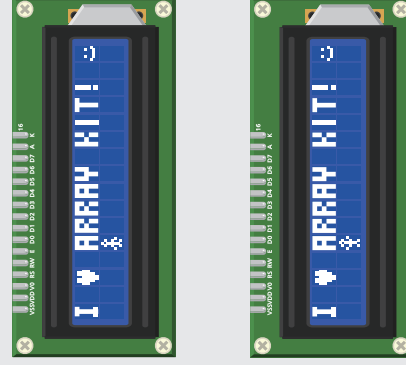
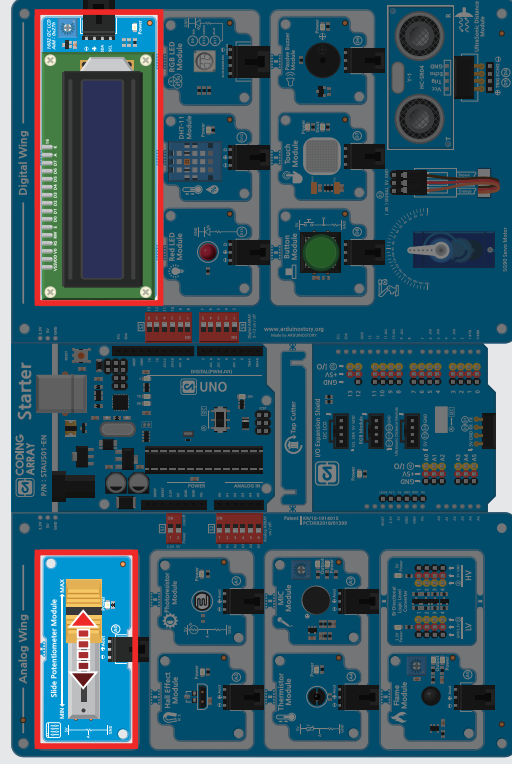
The shape of each user-defined character consists of 5×8 dots. Each row is specified in an array of eight bytes, one by one, and each row is defined as a hexadecimal. Assuming that you make a heart, you can set it as follows:

								00000	0b00000
								01010	0b01010
								11111	0b11111
								11111	0b11111
								11111	0b11111
								01110	0b01110
								00100	0b00100
								00000	0b00000

Special characters specified in the array are defined as `lcd.createChar (number, data)`. The numbers can then be defined by a total of eight characters from 0 to 7, and the data can be named after the array. `lcd` to output user-defined characters to LCD.use a number.



Print out characters using the I2C interface module, which controls 1602 LCDs consisting of 2 rows wide and 16 spaces with SDA (Serial Data, A4) and SCL (Serial Clock, A5), as well as GND, and 5 volt total.



View Results

You can adjust the rate of special character change in the second row by adjusting the variable resistance of the slide

```

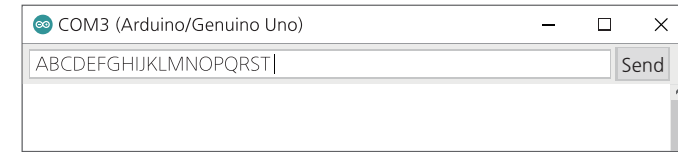
1  /* This sketch shows that 1602 LCD uses I2C communication.
2   Try printing the entered characters in the serial window.
3  */
4
5  #include <Wire.h>
6  #include <LiquidCrystal_I2C.h>
7
8  LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address. Use 16 Space 2-line LCD.
9
10 void setup() {
11   lcd.init();
12   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
13
14   Serial.begin(9600); // Starts serial communication at 9600 speed
15 }
16
17 void loop() {
18   if (Serial.available()) { // When the text arrives on the serial communication,
19     delay(100); // Wait 0.1 seconds for the entire message to arrive.
20     lcd.clear(); // Clear the screen.
21     while (Serial.available() > 0) { // while the text is coming in
22       lcd.write(Serial.read()); // Write the characters you read on the LCD.
23     }
24   }
25 }

```

Serial.available(); returns the number of data when received by serial communication. Returns zero if no data has been received.

Serial.read(); to read data entering the serial by 1 byte and return it to the decimal (constant) ASCII code value. Returns -1 if the receive buffer is empty.

lcd.write(Serial.read()); data entered into the serial are passed in aske code numbers. It is represented by converting it to a letter using lcd.write. If you write lcd.print, the number of Aski codes will be output.

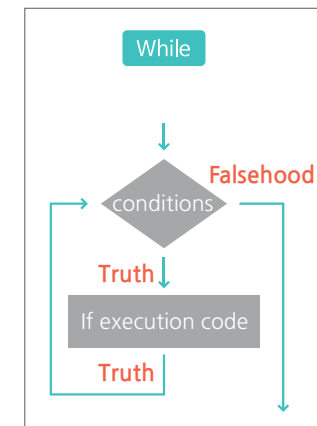


When you enter and transfer data in the serial window, the data is output on the first line of the LCD. If you enter more than 16 data, only 16 will appear in the first line..

Let's find out about While Sentence.

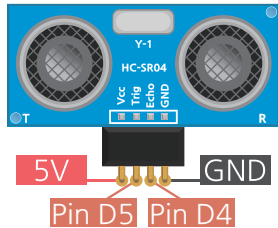
while {execution code; }

'While' is one of the repeats used to give the same command over and over, like 'for'. 'While statement' is a structure that gives a certain repeat condition and repeats the execution code while satisfying the condition. For statement lists the multiplication table under the iterative conditions, but in the while statement, the multiplication ceremony is placed in the execution code.



10 Distance measurement with ultrasonic sensor

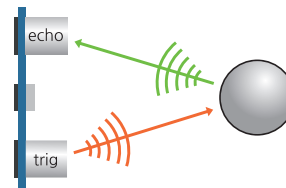
CHAPTER 2



Ultrasonic Distance Module

Ultrasonic is a type of sound wave, which is the sound wave of a frequency (>20KHz) area that is higher than the area that a person can hear. Ultrasonic sensors can calculate the distance using the time it takes for sound waves to return to an obstacle that is close to 3-400 cm..

Ultrasonic sensors have two speakers that look like speakers. One side has to produce ultrasonic waves with digital output HIGH and then stops ultrasonic waves with LOW. Ultrasonic sensors used in the starter kit will use a pulse width of 10 μs and therefore maintain the HIGH for 10 μs. The other is the role (Echo) of detecting ultrasonic waves reflected on an object. To detect the return of ultrasound, make sure that the reflector and the ultrasonic sensor are at right angles



Distance, velocity and time form the following formular

Distance

Speed

Time

velocity = $\frac{\text{Distance}}{\text{time}}$, time = $\frac{\text{Distance}}{\text{velocity}}$, Distance = v elocity × time

Ultrasonic waves can be generated from the trig pin of the ultrasonic sensor and obtained by means of the reciprocating distance, reciprocating time and sound velocity reflected on the barrier..

Round – trip distance = Sound speed × travel time

∴ Distance to object = $\frac{\text{Sound speed} \times \text{the time it takes for a microwave to return}}{2}$

In Arduino, the time it takes for the ultrasound to return is in microseconds (μs), so the distance can be obtained in cm after the unit conversion..

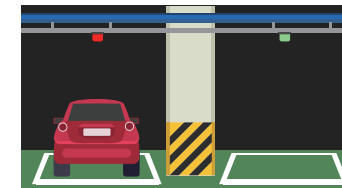
∴ Distance to object = $\frac{\text{Sound speed} \times \text{the time it takes for a microwave to return}(\mu\text{s})}{2}$

= $\frac{1}{2} \times \frac{340 \text{ m}}{1 \text{ s}} \times \frac{100 \text{ cm}}{1 \text{ m}} \times \text{the time it takes for a microwave to return} \mu\text{s} \times \frac{1 \text{ s}}{1000000 \mu\text{s}}$

= 0.017 × the time it takes for a microwave to return(cm)

At this point, the speed of sound is about 340 m/s at 15° C, and as the temperature rises by 1° C, the speed of 0.6 m/s increases. Therefore, it may be used with the following corrections:.

Sound speed=340+0.6×(current temperature-15)



Using the principles of ultrasonic sensors, one can measure a key or measure a distance to an obstacle in front of one. It is also possible to implement a parking system by sensing that the parking space is empty or parked, such as a parking lot in a large shopping mall. Ultrasonic waves are also used to examine the condition of the human organs or to identify deep undersea terrain, as the density of the medium that transmits sound is higher. In vacuum, there is no medium, so distance measurements using ultrasonic waves are not allowed..

**CODING
ARRAY**
STARTER KIT FOR ARDUINO



```

1  /* This sketch uses ultrasonic sensors to measure the distance.
2  * The trigger pin is connected to Arduino's number 5. The trigger pin produces ultrasonic waves.
3  * Echo pins are connected to Arduino's number 4. The echo pin detects reflected ultrasound.
4  * The measured distance should be shown in both the serial and LCD windows.
5  * If the measured distance exceeds the set value, the green LED,
6  * If the measured distance exceeds the set value, turn on the red LED.
7  */
8
9  #include <Wire.h>
10 #include <LiquidCrystal_I2C.h>
11
12 LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16 Space 2-line LCD
13
14 int redLED = 13;      // Red LED No. 13
15 int greenPin=10;     // Green LED No. 13
16 int threshold = 15;  // Set Distance Threshold Value
17
18 void setup() {
19   pinMode(5,OUTPUT); // Trigger pin connection No. 5
20   pinMode(4,INPUT);  // Echopin to Pin 4
21
22   pinMode(redLED,OUTPUT); // Set pin 13 to output
23   pinMode(greenPin, OUTPUT); // Set pin 10 to output
24
25   Serial.begin(9600); // Starts serial communication at 9600 speeds
26
27   // LCD Setting
28   lcd.init();
29   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
30   lcd.clear();
31 }
32
33 void loop() {
34   // Distance measurement with ultrasonic sensor
35   float Duration, Distance;
36   digitalWrite(5, HIGH); // Fire an ultrasound.
37   delayMicroseconds(10); // for 10 microseconds
38   digitalWrite(5, LOW); // Turn off the ultrasound.
39   Duration = pulseIn(4, HIGH); // Save the time the ehopin is held in HIGH
40   Distance =((float)(340 *Duration) /10000) /2; // convert the distance to cm
41

```

```

42 // Measured Distance Output
43 Serial.print(Distance); // Print distance in serial window without changing lines
44 Serial.println("cm"); // unit output
45
46 if (Distance < threshold){ // If the measurement distance is less than the threshold value, turn on the
red LED.
47   digitalWrite(redLED,HIGH);
48   digitalWrite(greenPin,LOW);
49   // Show Distance to LCD Window
50   lcd.clear();
51   lcd.setCursor(0,0); // first line first column
52   lcd.print(Distance); // Measured Distance Output
53   lcd.print("cm"); // unit output
54 }
55 else { // If the measured value is greater than the threshold value, turn on the green LED.
56   digitalWrite(redLED,LOW);
57   digitalWrite(greenPin,HIGH);
58   // Show Distance to LCD Window
59   lcd.clear();
60   lcd.setCursor(0,0); // first line first column
61   lcd.print(Distance); // Measured Distance Output
62   lcd.print("cm"); // unit output
63 }
64 delay(2000); // 2000 millisecond delay to reliably read values
65 }

```

float Duration, Distance; A variable with the same type of data can be declared at once..

pulseIn (Pin number, Value, timeout); Measure the amount of time it takes to return after a pulse occurs.

Pin number : pin number to read the pulse

Value : Type of pulse to read. HIGH or LOW

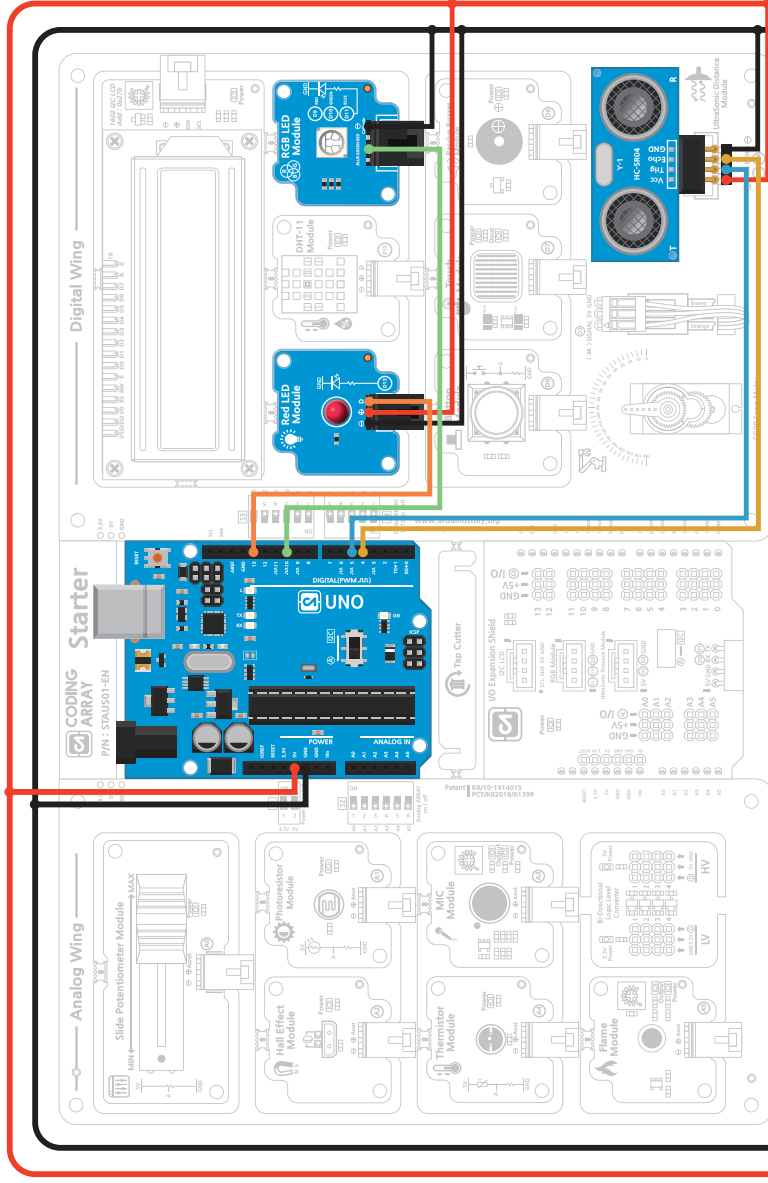
timeout (optional) : The time (microseconds) to wait for the pulse to start, and the length of the pulse (unshinged long) in microseconds.

Returns zero if the pulse does not start within the specified timeout.

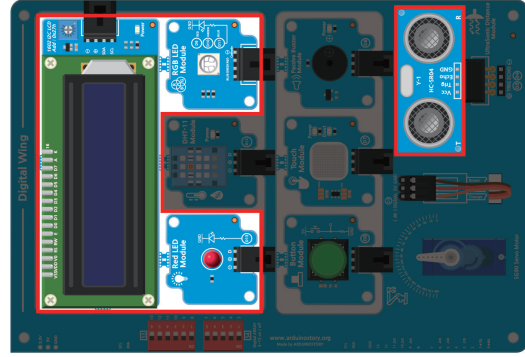
It can operate from 10 microseconds to 3 minutes.

pulseIn (echoPin, HIGH); when the value of echoPin reaches HIGH, start the timer and return the time that HIGH is maintained.

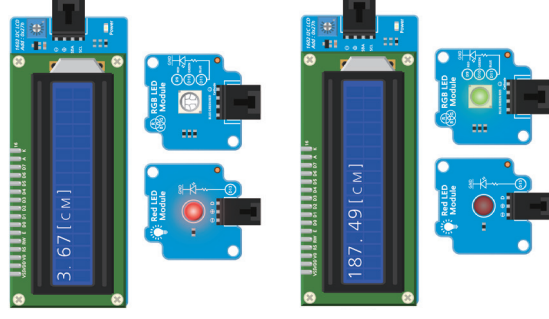
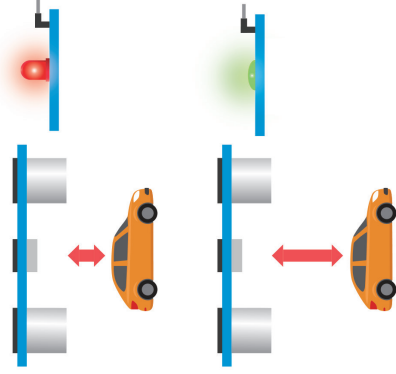
((float)(340 *Duration) /10000) /2; Since the calculation result is a true type with a decimal point, attach the data type as float.



The trigger that produces ultrasonic waves is connected to digital No. 5 and the echo that detects reflected ultrasonic waves is connected to digital No. 4. Using the principles of ultrasonic sensors, measure the distance to the obstacle in front...

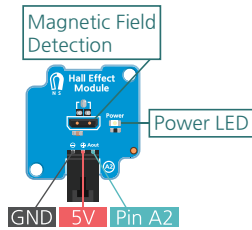


View Results



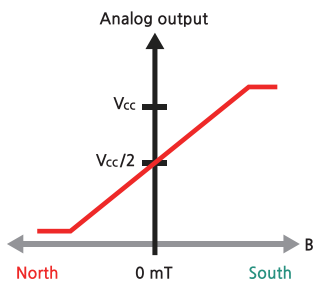
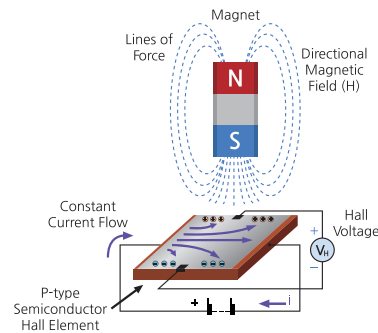
Once the program is uploaded, you can see that the green LED illuminates when the obstacle is more than 15cm apart in front of the ultrasonic sensor, and that the red LED turns on when the obstacle is less than 15cm away.

Magnetic Sensing Hall Sensor (Hall Effect Module)



Magnetic sensing hole sensor is a device that changes internal resistance according to strength of external magnetic field by using Hall Effect principle. The closer and stronger the magnetic field, the higher the output voltage. There are two types of Hall sensors: Digital Hall sensors and Analog Hall sensors.

Hall Effect (Hall Effect) A conductor is placed in a magnetic field, and the flow of current in a direction perpendicular to both the magnetic field and the electric current in the conductor creates electromotive force in that direction, which is published by the potential difference in this direction, was discovered by the American physicist Hall (E. H. Hall).



Digital hall sensors can only detect whether or not a magnetic field exists, and analog hall sensors can detect the magnetic field poles as well as the strength of the magnetic field with linear hall effects. Analog hall sensor module is used in array kit. As the S-pole approaches the front of the hall sensor, the voltage becomes close to 5 V, and the N-pole can be closer to the lock 0 V, indicating the strength and poles of the upcoming magnetic field.

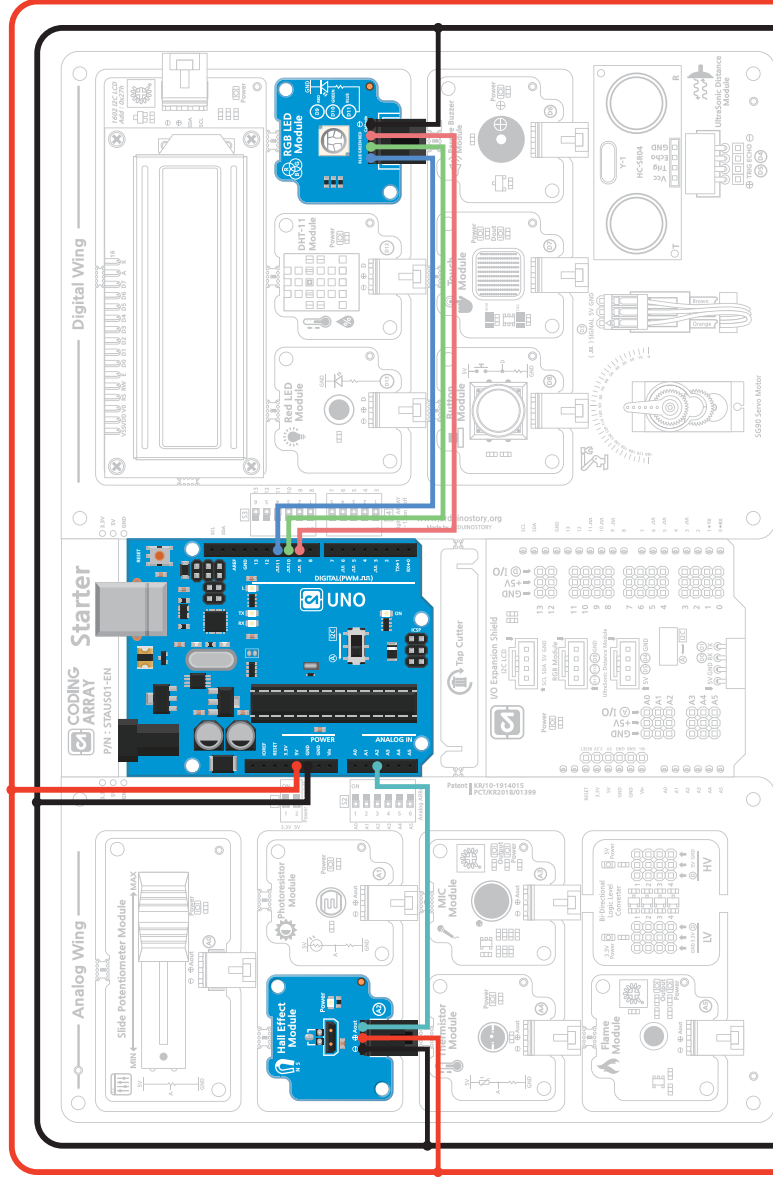
The magnetic sensing sensor can be used to check the magnetic field of the conductor to act as a switch or to detect the rotational speed, position and current of the motor. It is used variously in real life such as the speed measured on a car's instrument panel, the speed measured on a treadmill, and the door switch on a washing machine or refrigerator.

CAK Starter Code > 11_HallEffect

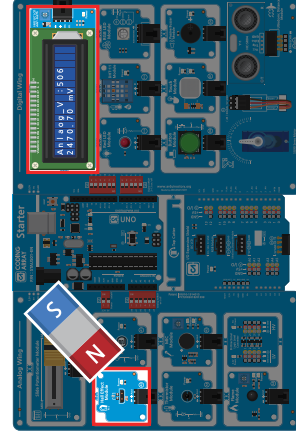
```

1  /* This sketch uses the Hall Effect module connected to Analog A2.
2  * Read the value of the analogue sensor which varies with the surrounding magnetic field
3  * Measure the strength of the magnetic field by converting it to voltage.
4  * This module has a lower output voltage as the N pole approaches and a higher output voltage as the S pole approaches.
5  * Outputs a comma-separated serial message for easy transfer of result values to Excel.
6  * Copy the message from the serial window and paste it into the memo pad. Save as CSV" extension
7  * You can create a file that can be retrieved from Excel.
8  */
9
10 // set up for LCD use
11 #include <Wire.h>
12 #include <LiquidCrystal_I2C.h>
13 LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address. Use 16 Space 2-line LCD
14
15 const int hallPin = A2; // connect hall sensor to A2 pin
16 int sensorReading;      // Store analog sensor values
17 int voltage;           // Store the converted value to voltage
18
19 void setup() {
20   Serial.begin(9600); // Set communication speed to 9600
21   Serial.println("sensorReading, Voltage (mV)"); // Output message for csv file in serial window
22
23   // LCD initialization
24   lcd.init();
25   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
26   delay(1000);
27 }
28
29 void loop() {
30   sensorReading = analogRead(hallPin); // Read and save the analog value of the sensor
31   voltage = sensorReading * (5.0 / 1024.0) * 1000; // Convert Analog Values from Voltage0 to 5000
32
33   // csv (when you want to receive data in comma-separated text format)
34   Serial.print(sensorReading);
35   Serial.print(",");
36   Serial.println(voltage);
37
38   // Output a message in an LCD window
39   lcd.clear();
40   lcd.setCursor(0, 0); // first line first column
41   lcd.print("Anlaog_V :");
42   lcd.print(sensorReading);
43   lcd.setCursor(0, 1); // 2nd line first column
44   lcd.print(voltage);
45   lcd.print(" mV");
46   delay(500);
47 }

```

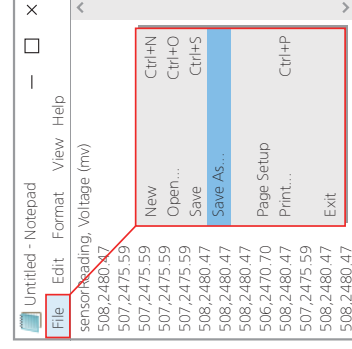
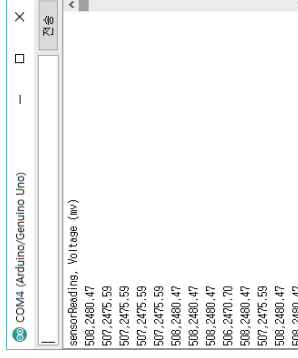


Analog magnetic sensing module is connected to A2 pin. You can check the analog output value that changes as the N pole and S pole of the magnet approach the sensor. It also looks at moving serial output data to Excel.



View Results

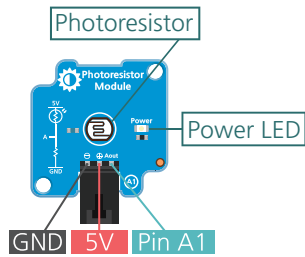
	A	B
1	sensorRea	Voltage (mv)
2	507	2475.59
3	506	2470.7
4	507	2475.59
5	507	2475.59
6	502	2451.17
7	476	2324.22
8	439	2143.55
9	385	1879.88
10	318	1552.73
11	268	1308.59
12	247	1206.05



After the program uploads, the LCD screen outputs analog input values (approximately 506) and conversion voltages (approximately 2470 mV) when no magnetic field is detected. The closer the N pole of the magnet is to the magnetic sensing sensor, the smaller the analog input value, and the closer the S-pole, the larger the analog input value..

In this example, let's move the data that appears in the serial window to Excel. Outputs data separated by commas, drags and copies the results shown in the serial window..

Attach this result value to Notepad and save it with a csv extension, such as file >Save As>result.csv. Open the csv extension file in Excel and use it.



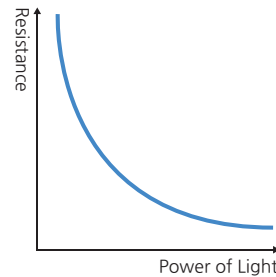
Photoresistor

Light sensing resistance sensors are called by a variety of names such as light sensors, light sensing sensors, photoresistors, LDR (Light Dependent Resistors), Cadmium Sulfide (CdS), CdS Cells, CdS Photoresistors, Photometric Cells and Photocorel. As the intensity of light increases, the resistance value decreases, and the intensity of light can be measured by increasing the resistance value.

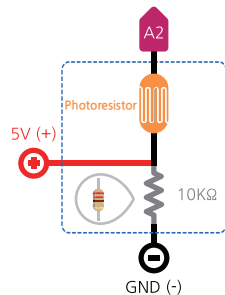
voltage distribution of between 0 and 5 volts between the resistance of the light sensor and the resistance of the 10 KΩ resistance and the intensity of the light..

The other, which utilizes a large resistance of 10KΩ, is that measuring light in a very bright area can cause the resistance value of the light sensor to be very small, allowing over-current to flow to the analog input pin, which can also be prevented.

The resistance values vary depending on the type of light sensor, but usually have a range of 5 KΩ (when light) to 200 KΩ (when dark) and show a nonlinear relationship between resistance and intensity of light, as with the following Figure.



Light sensors can measure the change in value due to the intensity of light at low prices, and have various advantages, such as night lighting or lighting sensing devices that control the speed of the camera shutter. However, resistance may vary with temperature, and there is a time difference between the change in intensity of light and the change in resistance. It also has less light sensitivity than photo diode or photo transistor. Therefore, it is not suitable for use in places where rapid changes in light or intensity of light need to be accurately measured, and is suitable for determining only bright and dark levels..



As Arduino reads voltage values rather than resistance values, the resistance of the light sensor must be calculated using a voltage distribution scheme. In order to calculate the voltage entering the sensor using a voltage distribution scheme, the circuit should be constructed with two resistors that know the resistance and size of the sensor. The light sensor is connected in series with a 10KΩ resistance. This causes a

**CODING
ARRAY**
STARTER KIT FOR ARDUINO



```

1  /* This sketch measures the brightness of light using a light sensing sensor.
2  * A1 pin connected to the light sensing sensor enters an analog input value between 0 and 1023,
3  * depending on the brightness of the light.
4  * Analog input values are divided into 0 - 3 4 steps through map function and used for switch-case.
5  */
6  // Settings for LCD use
7  #include <Wire.h>
8  #include <LiquidCrystal_I2C.h>
9  LiquidCrystal_I2C lcd(0x27,16,2); // Set the LCD I2C address. Use 16 space 2 line LCD.
10
11 const int photoresistorPin=A1; // connect the light sensor to the A1 pin
12 const int sensorMin =0; // Minimum sensor value found by experiment, can be modified.
13 const int sensorMax =700; // Maximum sensor value found by experiment, can be modified.
14
15 void setup() {
16   lcd.init(); // LCD initialization
17   lcd.backlight(); // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
18   lcd.setCursor(0,0); // first line first column
19   lcd.print("Range : "); // Message Output
20
21   Serial.begin(9600); // Starts serial communication at 9600 speeds
22 }
23
24 void loop() {
25   // Read sensor values to map ranges
26   int sensorReading = analogRead(photoresistorPin); // Read the light sensor value from the A2 pin
27   Serial.println(sensorReading);
28   int range = map (sensorReading,sensorMin,sensorMax,0,3); // Map the sensor values from 0 to 3.
29
30   // output messages according to sensor range
31   switch(range) { // according to range 0-3
32     case 0: // touch the sensor and when the sensor value is zero,
33       Serial.println("DARK"); // Darkprint and replace lines in the serial window
34       lcd.setCursor(9,0); // the ninth column of the first line
35       lcd.print(range); // indicate brightness phase on LCD window
36       lcd.setCursor(0,1); // the first column of the second line

```

```

37   lcd.print("DARK"); // DARK Output
38   break;
39
40   case 1: // Put your hands close to the sensor and when the sensor value is 1,
41     Serial.println("DIM"); // Dimprint and replace lines in serial window
42     lcd.setCursor(9,0); // the ninth column of the first line
43     lcd.print(range); // indicate brightness phase on LCD window
44     lcd.setCursor(0,1); // the first column of the second line
45     lcd.print("DIM"); // DIM output
46     break;
47
48   case 2: // Keep your hands away from the sensor and when the sensor value is 2
49     Serial.println("MEDIUM"); // print medium on serial window and replace lines
50     lcd.setCursor(9,0); // the ninth column of the first line
51     lcd.print(range); // indicate brightness phase on LCD window
52     lcd.setCursor(0,1); // the first column of the second line
53     lcd.print("MEDIUM"); // MEDIUM Output
54     break;
55
56   case 3: // When the sensor value is 3 without touching the sensor nearby,
57     Serial.println("BRIGHT"); // print the lightprint on the cereal window and replace the line
58     lcd.setCursor(9,0); // the ninth column of the first line
59     lcd.print(range); // indicate brightness phase on LCD window
60     lcd.setCursor(0,1); // the first column of the second line
61     lcd.print("BRIGHT"); // BRIGHT Output
62     break;
63   }
64   delay(50); // 50 millisecond delay to reliably read values
65 }

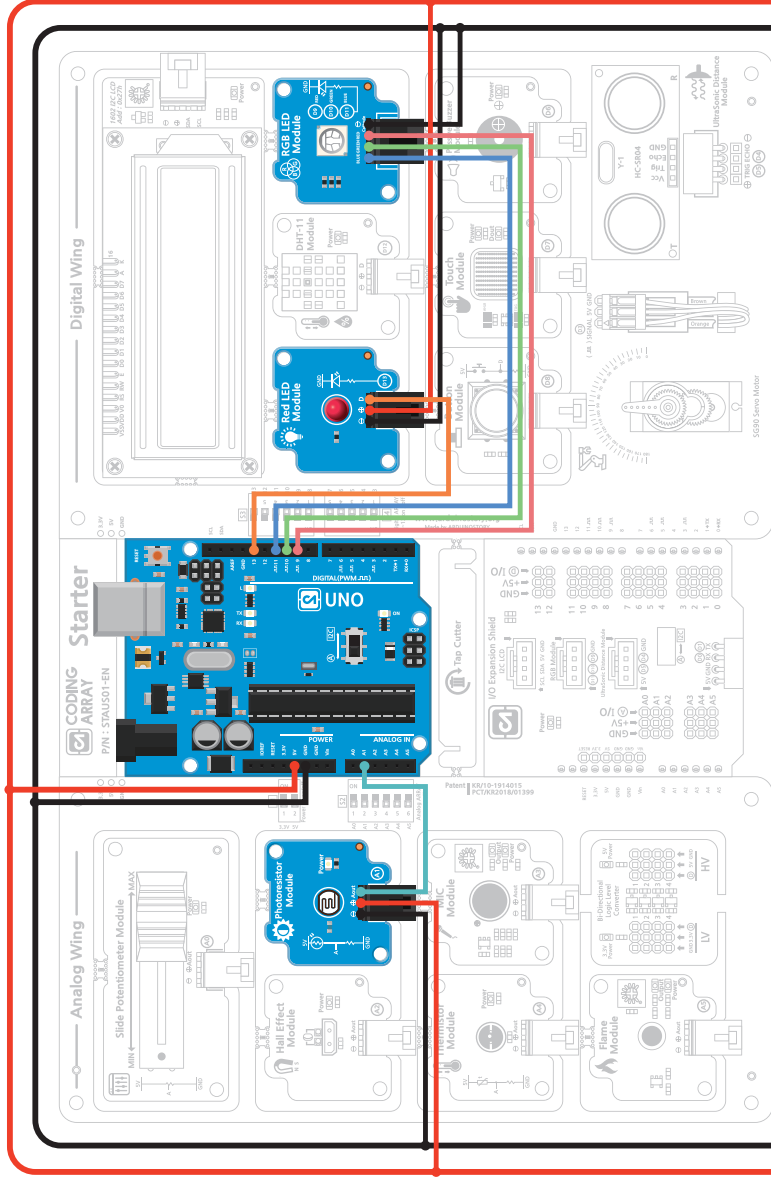
```

constrain(x, a, b); Measure the amount of time it takes to return after a pulse occurs.

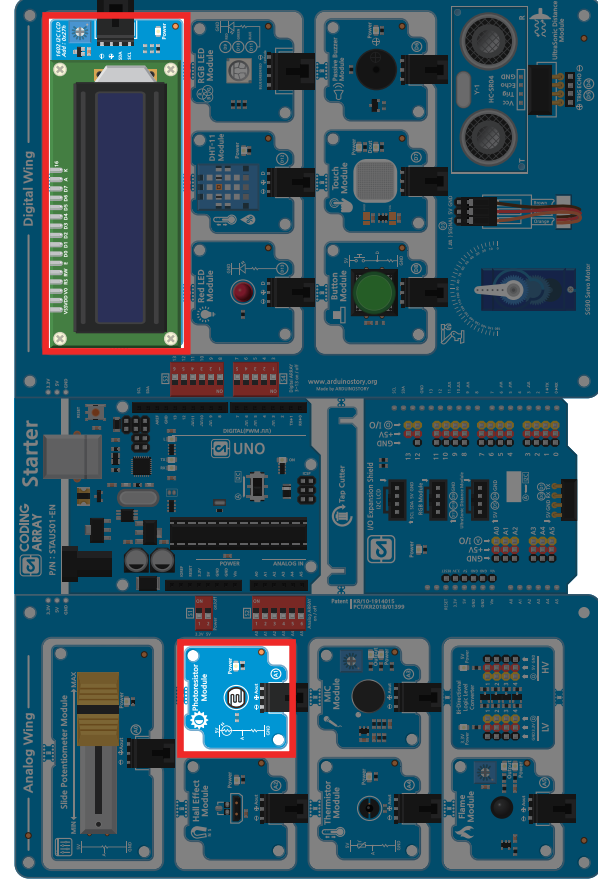
x : number of restrictions, a : lower range, b : upper range, x, a, b are all data types
Return x value if x is a value between a and b, return a value if x is less than a,
Returns the b value if x is greater than b

constrain(sensorValue, 0, 255);

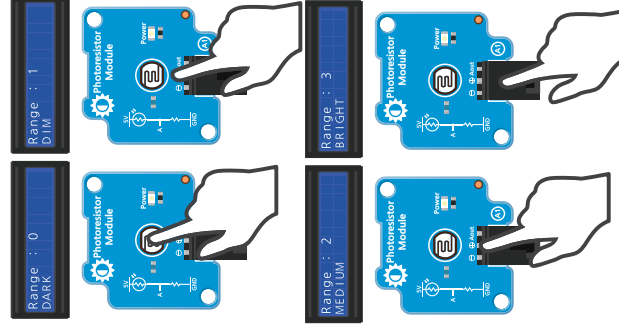
Returns 0 value if sensorValue value is less than 0; returns 255 if sensorValue
value is greater than or equal to 255.
Restrict the scope of 0 and 255.



Using a light sensing resistance sensor connected to the analogue A1 pin, control the LED output according to the intensity of the light, and learn about the calibration of the analog sensor values.



View Results



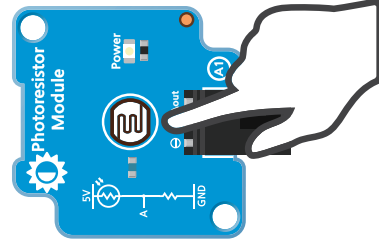
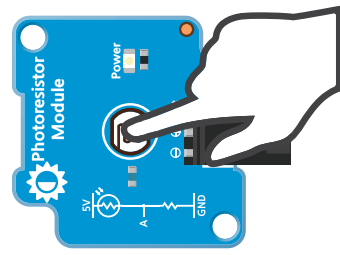
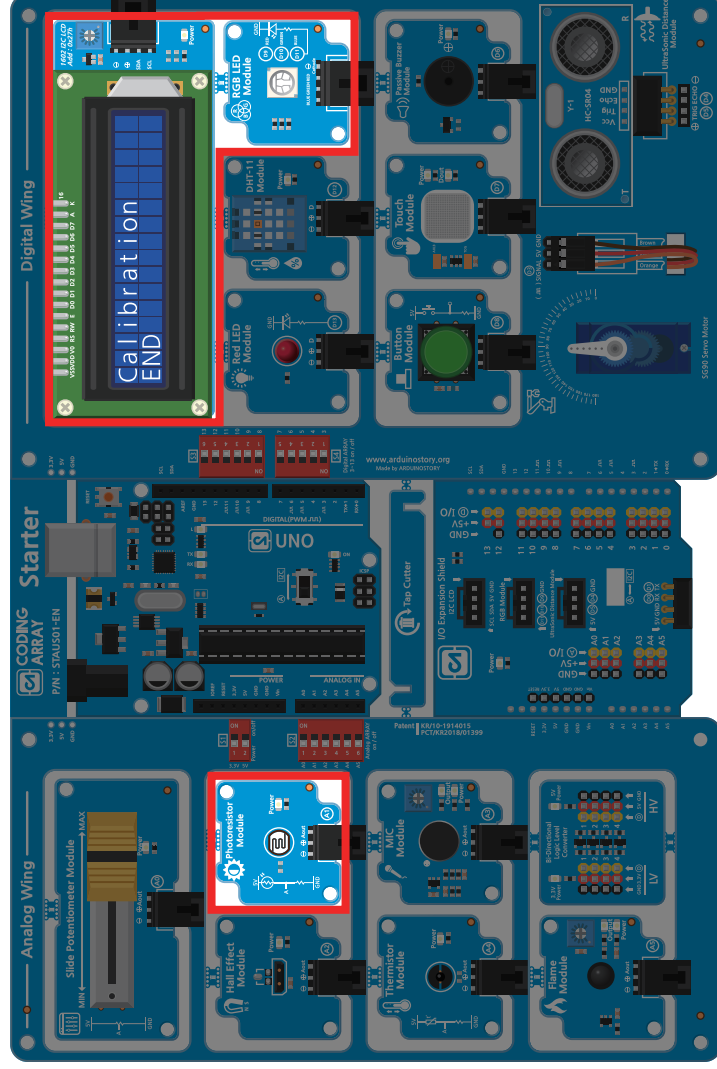
When the program is uploaded, the light sensor detects the light and divides the amount of light into four stages to display the results in an LCD window. Touch the light sensor and observe the results as you keep away from it.



```
1  /* This sketch will learn how to calibrate the light sense sensor input values according to the
2     * surroundings.
3     * Read the value of the analog A1 pin to which the light sensor is connected for 5 seconds to store
4     * the maximum and maximum values.
5     * The maximum value stored is 0 and the maximum value is 255, which is converted through the map
6     * function.
7     * After uploading the code, cover the light sensor with your hands and press the reset button.
8     * Allow the maximum and maximum values to be stored while keeping hands away from the light
9     * sensor for 5 seconds.
10    * Then, move your hands around the light sensor to check the change in the blue brightness of the
11    * RGB LED.
12    */
13
14 // set up for LCD use
15 #include <Wire.h>
16 #include <LiquidCrystal_I2C.h>
17 LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD I2C address. 16 space 2 lines LCD use
18 // Variable setting
19 const int photoresistorPin=A1; // Connect light sensor to A1 pin
20 const int redLED=13; // connect the calibration notification LED to pin 13.
21 const int bluePin=11; // connect LEDs for brightness display according to sensor value to
22 // number 11.
23 int sensorValue =0; // store the light sensor value
24 int sensorMin =1023; // set sensor max to 1023.
25 int sensorMax =0; // set the sensor maximum value to 0
26
27 void setup() {
28 // Calibration Notification LED Output Settings
29 pinMode(redLED, OUTPUT);
30 digitalWrite(redLED, HIGH);
31
32 // LCD setting
33 lcd.init();
34 lcd.backlight(); // turn on the backlight (lcd.noBacklight() turns off the backlight).
35 lcd.clear();
36 lcd.setCursor(0,0); // First line first column
37 lcd.print("Calibration"); // output messages
38 lcd.setCursor(0,1); // First line first column
```

```
33 lcd.print("START"); // output messages
34
35 // Sensor value compensation
36 while(millis() < 5000) { // for 5 seconds
37   sensorValue = analogRead(photoresistorPin); // save sensor values
38   if(sensorValue > sensorMax) { // sensor value is greater than maximum
39     sensorMax = sensorValue; // reset to maximum value
40   }
41
42   if(sensorValue < sensorMin) { // sensor value is less than the maximum value
43     sensorMin = sensorValue; // reset to maximum value
44   }
45 }
46
47 // End calibration. Turn off the LED and output a message to the LCD screen
48 digitalWrite(redLED,LOW);
49 lcd.clear();
50 lcd.setCursor(0,0); // First line first column
51 lcd.print("Calibration"); // output messages
52 lcd.setCursor(0,1); // First line first column
53 lcd.print("END"); // output message
54 }
55
56 void loop() {
57   sensorValue = analogRead (photoresistorPin); // read an analogue sensor value and store it in a
58   // variable
59   // read sensor value and convert to 0-255
60   sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
61
62   // limit if sensor value is outside calibration range
63   sensorValue = constrain(sensorValue, 0, 255);
64
65   // adjust blue LED brightness with sensor value
66   analogWrite(bluePin,sensorValue);
67 }
```


View Results



When the program is uploaded, a "Calibration START" message appears in the LCD window, a calibration task is performed for five seconds and a "Calibration END" message is displayed. Put your hands on the light sensor for calibration and keep away for 5 seconds to create the brightest environment from the darkest. If this operation has not been carried out within 5 seconds, the light sensing sensor may be covered by hand, pressed the "RESET" button next to the power line and recalibrated. The red LED was used as a notification LED indicating that it was being calibrated..

When calibration is finished, the red LED is turned off and the light sensor detects the amount of light from the analog input value between 0 and 1,023. Analog output should be expressed as a value between 0 and 255, so use the map function to convert 0 to 1023 to 0 to 255 to display results with blue LED brightness..



```

1  /* This sketch runs while the button switch connected to the digital pin 8 is pressed.
2  * Call up the calibration () function to find the maximum and maximum values of the analog A1 pin
3  * Return to the main loop if the button is not pressed.
4  * This method can reset the maximum and maximum values of the light sensor when ambient
5  * brightness conditions change..
6  */
7
8  // Set up for LCD use
9  #include <Wire.h>
10 #include <LiquidCrystal_I2C.h>
11
12 LiquidCrystal_I2C lcd(0x27, 16, 2); // set LCD I2C address. 16kans2joules LCD use
13
14 const int photoresistorPin = A1; // Connect the light sensor to the A1 pin
15 const int redLED = 13; // connect the calibration notification LED to pin 13.
16 const int bluePin = 11; // connect an LED for brightness display according to sensor value to
17 No. 11.
18 const int Button = 8; // Connect button switch to pin 2
19
20 int sensorValue = 0; // store the light sensor value
21 int sensorMin = 1023; // set sensor max to 1023.
22 int sensorMax = 0; // set the sensor maximum value to 0
23
24 void setup() {
25   pinMode(redLED, OUTPUT); // Set Calibration Notification LED Output
26   pinMode(bluePin, OUTPUT); // set LED output to indicate brightness
27   pinMode(Button, INPUT); // Enter buttons connected to pull-up resistance
28
29   lcd.init(); // Initialize LCD
30   lcd.backlight(); // turn on the backlight (lcd.noBacklight() turns off the backlight).
31   lcd.clear();
32 }
33
34 void loop() {
35   while (digitalRead(Button) == HIGH) { // when button switch is pressed
36
37     calibrate(); // calibration function.
38   }
39 }

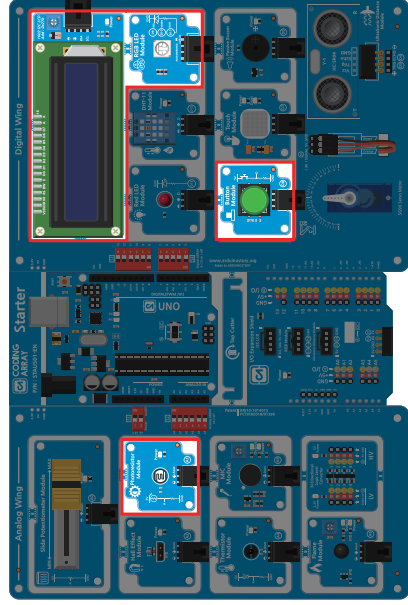
```

```

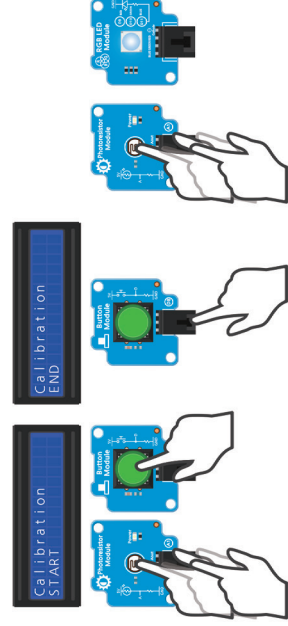
35
36   digitalWrite(bluePin, LOW); // Turn off the blue LED during calibration.
37   lcd.setCursor(0, 0); // First line first column
38   lcd.print("Calibration"); // output messages
39   lcd.setCursor(0, 1); // First line first column
40   lcd.print("START"); // output messages
41 }
42
43   digitalWrite(redLED, LOW); // Turn off the red LED after calibration.
44   lcd.setCursor(0, 0); // First line first column
45   lcd.print("Calibration"); // output messages
46   lcd.setCursor(0, 1); // First line first column
47   lcd.print("END "); // output message
48
49   sensorValue = analogRead(photoresistorPin); // store the light sensor value
50   sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255); // calibrate sensor values to
51   0-255.
52   sensorValue = constrain(sensorValue, 0, 255); // limit if sensor value is outside calibration
53   range
54   analogWrite(bluePin, sensorValue); // adjust the LED brightness with the calibrated value.
55 }
56
57 // calibrate() function setting: Reset the maximum and maximum values of the sensor according to
58 the ambient brightness.
59 void calibrate() {
60   digitalWrite(redLED, HIGH); // Turn on the red LED for calibration notifications.
61   sensorValue = analogRead(photoresistorPin); // Read and save the value of the light sensor
62
63   if (sensorValue > sensorMax) { // the illumination sensor is greater than 1023.
64     sensorMax = sensorValue; // read the sensor value and save it to sensorMax
65   }
66
67   if (sensorValue < sensorMin) { // the light sensor is less than zero
68     sensorMin = sensorValue; // read the sensor value and save it to sensorMin
69   }
70 }

```

View Results



Once the program is uploaded, you can start calibrating the sensor while holding down the button switch. When the button switch is pressed, the "Calibration START" message appears in the LCD window, and a correction function is invoked when you touch the light sensing sensor and then gradually move away. When you release the button switch, the message "Calibration END" appears to end the calibration. You can see that the blue LED's brightness changes depending on the amount of light detected by the light sensing sensor after the calibration operation. The calibration function can be called by pressing the button switch to make it easier to calibrate each time the surrounding environment changes. It is possible to recognize the ambient brightness and to implement a smart street lamp that illuminates when the amount of light entering the light sensor is below a certain value. Smart street lamps not only reduce the effort to turn street lights on and off, but also save electricity..



Let's learn about the use of functions..

When you write a program, you create and write a 'function' to organize the program by performing the same task several times, having to be reused by another program, or creating a modular piece of code. The function is defined as follows

Return type	Name	(Parameter)	{ body }
void	calibrate	()	{ digitalWrite(redLED, HIGH); : : }

Return type : Set type to return result value of function
void : indicates no return value.

Function name : Set as a name that represents the characteristic of the function

(parameter) : Put the factors to be used in the function. If there are multiple parameters, the order must also be observed. If the

parameter is not required, leave () blank.

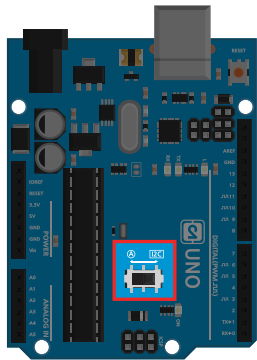
{ **Function body**; **return value**; } : Insert code that runs within the actual function.

return : has the function of ending function and return result value.

If you return the function result value, write the **return** value.

If the **return** type is **void**, return ; can be written or omitted.

Function can be declared above or below the loop() function and called using the function name during code execution..



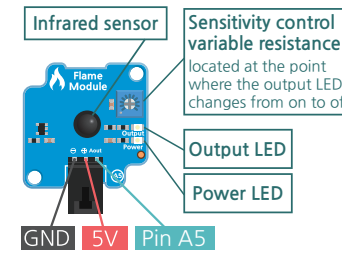
I2C (Inter-Integrated Circuit) is an NFC that can connect a 1 master (Arduino) : multiple slave (sensor modules) in one direction using a SCL (Serial Clock) pin and SDA (Serial Data) pin with full-up resistance connected. Arduino can use SDA, SCL pins as I2C communication pins or analog A4, A5 pins as functions of SDA and SCL respectively. In the starter kit, the SDA and SCL pins of the Uno board are conveniently placed with a slide switch in the center of the Arduino Uno board

CAUTION!

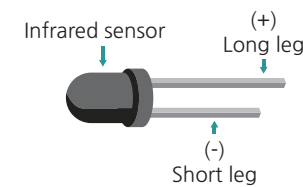
You can not use the I2C communication interface and the A4, A5 pins at the same time on the Arduino board. Therefore, the thermistor module and flame sensor module cannot be used simultaneously with the I2C LCD in the starter kit. Therefore, when using I2C LCD, define the module of use by placing the slide switch left and right as shown in Figure below..



Flame Sensor



There are many types of flame detection sensors connected to analog A5 such as ultraviolet flame detection, infrared flame detection and IR3 flame detection. The infrared flame detection sensor used in the coding array kit detects wavelengths in the infrared LEDs in the range of 760 to 1100nm from flames or light sources within an angle of 60° and converts them into electrical signals. The sensor is also called a collector (Collector, +polar connection), and a short leg is called an emitter (-polar connection) in photo transistor (Phototransistor) and the output voltage increases as the amount of light detected increases. The flame detection sensor has an infrared sensing sensor unit that is covered with a black epoxy that looks like an LED. Because of its polarity, the sensor has long legs (+) poles and short legs (-) connected.



It is connected to A2 pin when module is combined, but can be used as a digital sensor by connecting D and digital pin after module is disconnected. es that control the speed of the camera shutter. However, resistance may vary with temperature, and there is a time difference between the change in intensity of light and the change in resistance. It also has less light sensitivity than photo diode or photo transistor. Therefore, it is not suitable for use in places where rapid changes in light or intensity of light need to be accurately measured, and is suitable for determining only bright and dark levels..

CAUTION!

- The A4, A5 analogue pins cannot be used simultaneously with the SCL and SDA pins of I2C, so they must be fitted with a jumper at "A5 Jumper" to use the flame detection sensor.
- When using flame sensors, it is not possible to display the output on the LCD.

There are many types of fire detectors, such as heat sensing, smoke detection and flame detection. Double flame detection is installed in a space with high ceilings and external cultural properties, which are difficult to detect heat or smoke, to detect and operate infrared or ultraviolet radiation generated from the flame in case of fire. In this example, if a flame is detected within 60° using a flame detection sensor, the piezo buzzer will sound an alarm.



```

1  /* This sketch uses the flame sensor module connected to analogue A5.
2  * Detects flame strength around (prepare lighter and bring flame near module)
3  * If the threshold value set is exceeded, an audible alarm will be
4  */
5
6  #define PI 3.141592 // Set circumference PI value
7
8  int flameSensor =A5; // Connect flame detection sensor to analogue pin A5
9  int Buzzer =6; // connect the piezo buzzer to pin 6.
10 int sensorReading =0; // Variables for storing sensor output values
11
12 void setup() {
13   pinMode(Buzzer, OUTPUT); // Output settings for piezo buzzer pins
14   pinMode(flameSensor, INPUT); // set flame sensor pin to input
15   Serial.begin(9600); // initiate serial communication at 9600 speed
16 }
17
18 void loop() {
19   sensorReading = analogRead(flameSensor); // save an analogue value of the flame detection
20   sensor
21   Serial.println(sensorReading); // Print the value of the flame detection sensor to the serial
22   window
23   if(sensorReading >=1000) { // flame detection sensor value greater than 1000
24     Serial.println("Fire !!"); // Fire! Outputs on screen
25     playTone(); // Alarm negative
26   } else { // If flame detection sensor value is less
27     noTone(Buzzer); // off Peugeot Booger
28   }
29   delay(500); // 0.5 second interval
30 }
31 // set alarm negative function
32 void playTone() {
33   float sinVal; // save the sine wave value
34   int toneVal; // store value for alarm sound generation

```

```

35   for(int i =0 ; i < 180; i++) {
36     sinVal =sin(i * PI/180); // calculate the sin value by changing the angle to the radian
37     value
38     toneVal = 2000+(int(sinVal * 1000)); // translate alarm to frequency
39     tone(Buzzer,toneVal); // frequency generated from Peugeot speakers
40     delay(10); // alarm sound frequency rate adjustment
41   }

```

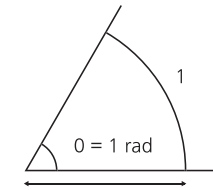
#define PI 3.141592 Define the value of the circumference n.

sin(rad) Calculate the Sin value of the radian angle. $-1 \leq \sin(\text{rad}) \leq 1$ range.

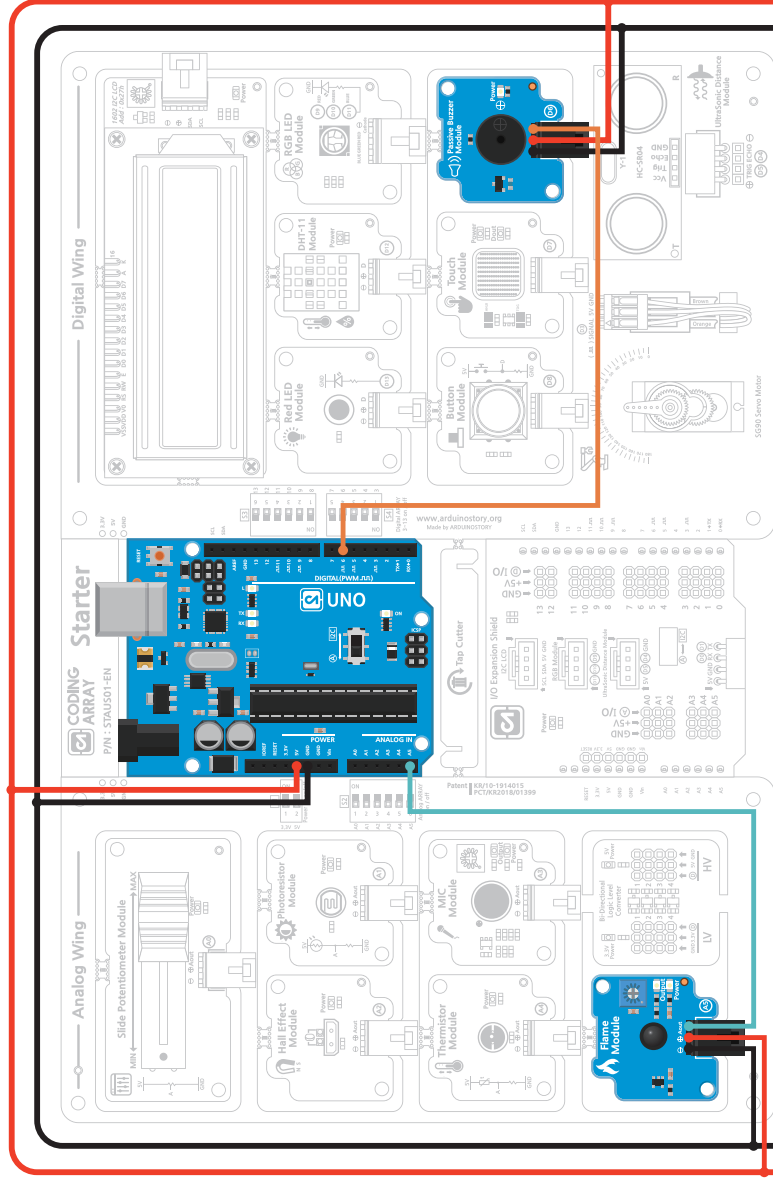
Rad :Radian angle, actual type

Loudness Method: To display angles using arc length

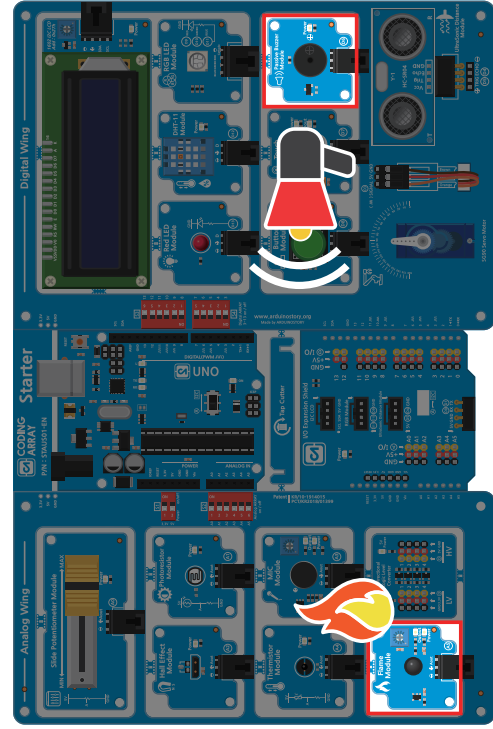
$$1 \text{ radian} = \frac{180^\circ}{\pi} \quad 1^\circ = \frac{\pi}{180} \text{ radian}$$



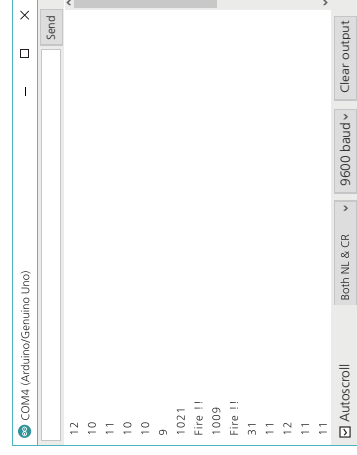
**CODING
ARRAY**
STARTER KIT FOR ARDUINO



If a flame is detected using a flame detection sensor connected to the analogue A5 pin, use the manual buzzer connected to the digital No. 6 pin to sound the alarm..



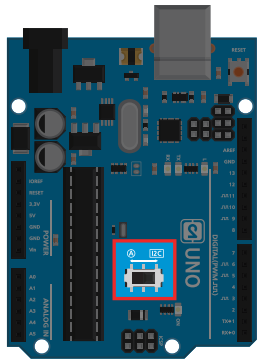
View Results



After uploading the sketch, bring the lighter flame near the flame sensor (if the lighter is not available, replace it with an infrared remote control at home). If a flame is detected within about 10cm, it indicates an analog value of less than 1000 and the manual buzzer produces a sin-wave alert. After module removal, digital pins can be connected to D and used as digital sensors. If the flame is not detected, it will return "0" or "1" if the flame is detected. The sensitivity of the sensor is adjusted by rotating the variable resistance on the board.

14 Temperature measurement with NTC thermistor

CHAPTER 2



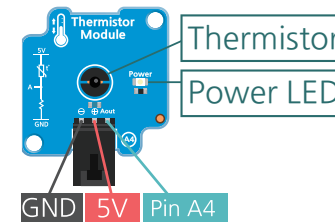
I2C (Inter-Integrated Circuit) is an NFC that can connect a 1 master (Arduino) : multiple slave (sensor modules) in one direction using a SCL (Serial Clock) pin and SDA (Serial Data) pin with full-up resistance connected. Arduino can use SDA, SCL pins as I2C communication pins or analog A4, A5 pins as functions of SDA and SCL respectively. In the starter kit, the SDA and SCL pins of the Uno board are conveniently placed with a slide switch in the center of the Arduino Uno board

CAUTION!

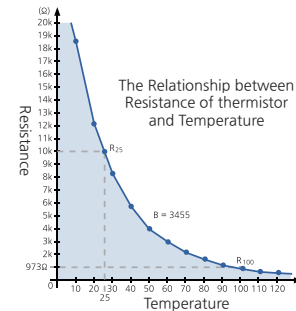
You cannot use the I2C communication interface and the A4, A5 pins at the same time on the Arduino board. Therefore, the thermistor module and flame sensor module cannot be used simultaneously with the I2C LCD in the starter kit. Therefore, when using I2C LCD, define the module of use by placing the slide switch left and right as shown in Figure below..



NTC Thermistor (Negative Temperature Coefficient Thermistor)



There are many types of temperature sensors and they can be divided into analog temperature sensors and digital temperature sensors. The array kit used NCT thermistor as an analogue temperature sensor for analog A4 pins. Thermistor is a composite of Thermal + Resistor, an electrical element with a properties in which the resistance of a substance varies with temperature. A thermistor thermometer is usually used at -50°C to 350°C . Thermistors are divided into NTC (Negative Temperature Coefficient) using properties that reduce resistance values as temperatures rise and PTC (Positive Temperature Efficient) using properties that increase resistance values as temperatures rise..



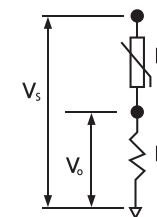
Three steps are taken to indicate the temperature due to the change in thermistor's electrical resistance.

- Step1)** On the analog input A4 pin to which the thermistor is connected, measure the voltage using the voltage distribution method.
- Step2)** Convert voltage to resistance.
- Step3)** Convert resistance to temperature

STEP 01

Voltage distribution

Thermistors indicate temperature due to changes in electrical resistance. However, the analog input pins of the Arduino board measure voltages other than resistance. Therefore, the resistance of the thermistor should be converted to voltage using the voltage distribution method. Series connection between Ohm's law and resistance (current flowing to each resistor is the same. The following expressions can be derived using the addition of two resistors). Use known resistance $R=10\text{K}\Omega$, R_t = thermistor to use voltage divider circuits, V_0 is measured at A4..



- Vs:** Full Voltage
- V0:** Voltage to 10KΩ resistance
- I:** Total Current
- R:** 10KΩ resistance
- Rt:** thermistor resistance

$$V_x = I(R + R_t) \quad I = \frac{V_s}{R + R_t}$$

$$V_0 = IR \quad V_0 = \frac{V_s R}{R + R_t}$$

STEP 02

Convert voltage to resistance.

If you organize the above expression for R_t and indicate the resistance value of the thermistor

$$R_t = \frac{V_s R}{V_0} - R$$

STEP 03

Convert resistance to temperature.

There are two main ways to measure the resistance value of the thermistor and convert it to temperature.

$$R_t = \frac{V_s R}{V_0} - R$$

The first <Shenhart-Hart>

$$\frac{1}{T} = A + B \ln(R) + C \ln(R)^2$$

T: Kelvin temperature (absolute temperature)

R: Resistance value at temperature T

A, B, and C: Known constants derived from resistance values according to the three temperatures

Second <B or β parameter >

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R_t}{R_0}\right)$$

NTC thermistors are inexpensive, small, responsive, and have a large coefficient of resistance to temperature, which can be used for precise temperature measurements. It is used for industrial equipment, home appliances, remote weather observation, and home automation system equipment.



CAK Starter Code >> 14_01_Thermistor_SH

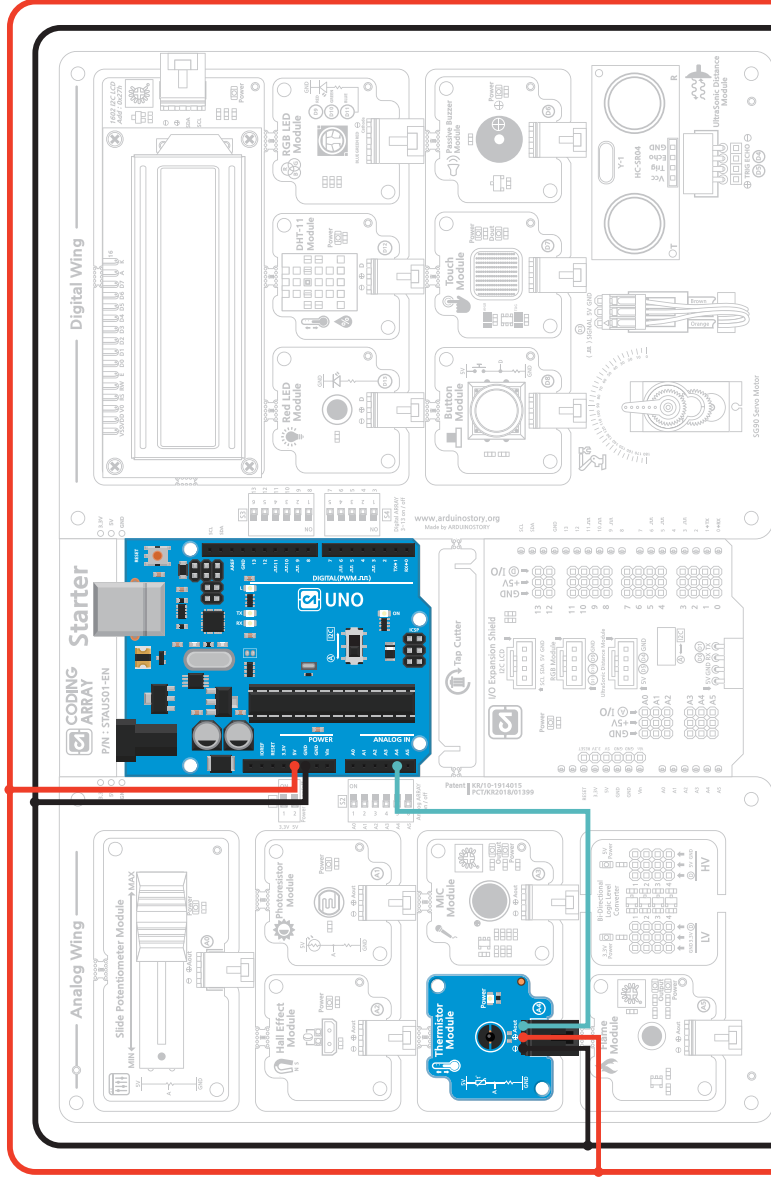
```

1  /* This sketch converts the voltage distributed to the thermistor connected to the A4 into a resistor.
2  One of the ways to change the resistance value to temperature
3  Use the Steinhart-Hart formula to calculate the temperature.
4  The A4 pin and I2C LCD module cannot be used together and must be selected as a jumper.
5  */
6
7  #include <math.h>
```

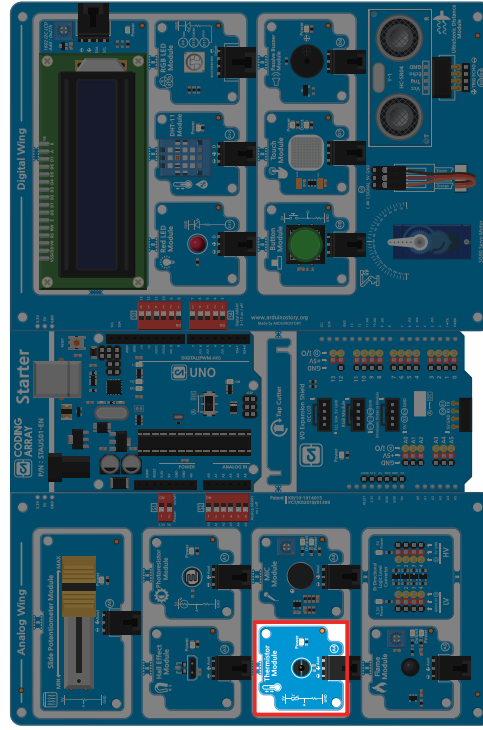
```

8
9  const int thermistorPin = A4;           // connect thermistor to A4 pin
10
11 // parameters can vary the value depending on the module.
12 double ParamA = 0.001129148;
13 double ParamB = 0.000234125;
14 double ParamC = 0.0000000876741;
15
16 void setup() {
17   Serial.begin(9600);                   // initiate serial communication at 9600 speeds
18 }
19 void loop() {
20   int readVal = analogRead(thermistorPin);
21   double temp = Thermistor(readVal);     // recall temperature measurement function
22   double tempC = temp - 273.15;         // Convert Absolute Temperature to Celsius
23   double tempF = (tempC * 9.0) / 5.0 + 32.0; // Convert temperature to Fahrenheit
24
25   // Output Serial Monitor
26   //Serial.println(readVal);
27   Serial.print(tempC);                  // display temperature
28   Serial.println(" C");
29   delay(500);
30 }
31 // set Steinhart-Hart temperature measurement function
32 double Thermistor(int RawADC) {
33   double Temp;
34   Temp = log(10000.0 * ((1024.0 / RawADC) - 1));
35   Temp = 1 / (ParamA + (ParamB + (ParamC * Temp * Temp)) * Temp);
36   return Temp;
37 }
38 return Temp;
39 }
40
```

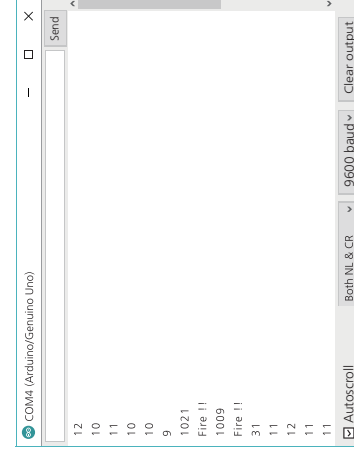
#include <math.h> → Arduino's Math Library is designed to help you manipulate floating point numbers. Contains useful functions. If you need to perform log, root, triangle function, exponential function, absolute value calculation, etc., you must insert this header file to use the desired function function.



Use the NTC thermistor connected to the analog A4 pin to measure the temperature. Read the voltage at the NTC according to temperature and convert it to temperature using the Steinhart-Hart formula and the B parameter formula..



View Results



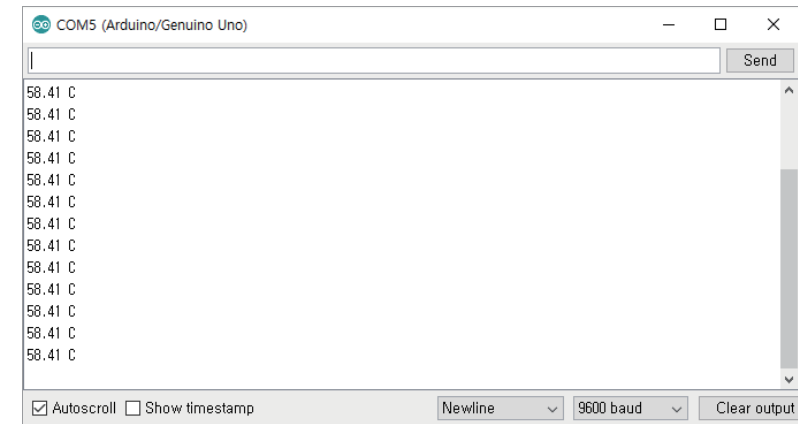
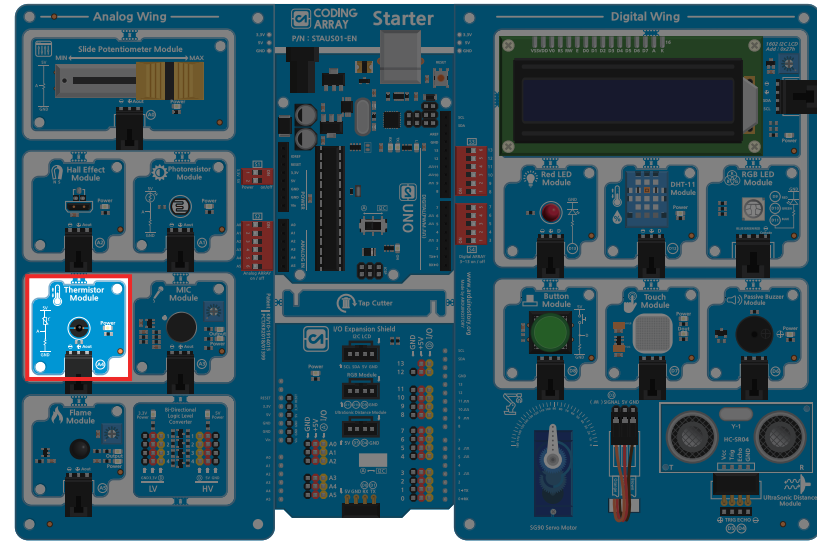
When you upload a sketch, you can see the result of converting the resistance value of the thermistor to temperature using the Steinhart-Hart expression in the serial window..



```

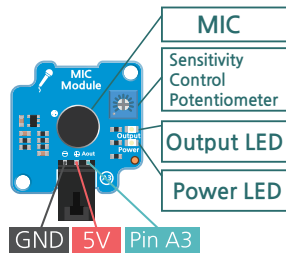
1  /* This sketch converts the voltage distributed to the thermistor connected to the A4 into a resistor.
2  * One of the ways to change the resistance value to temperature
3  * Use the B parameter expression to calculate the temperature.
4  */
5
6  #include <math.h>
7
8  const int thermistorPin=A4; // connect thermistor to A4 pin
9
10 //B parameter
11 float ParmB=3435.0;
12
13 void setup() {
14   Serial.begin(9600); // initiate serial communication at 9600 speeds
15 }
16
17 void loop() {
18
19   float readVal =analogRead(thermistorPin);
20
21   // calculate temperature
22   float resistor =(1023.0*10000)/readVal-10000;
23   float tempC =(ParmB/(log(resistor/10000)+(ParmB/(273.15+25)))) -273.15;
24
25   // Output Serial Monitor
26   // Serial.println(readVal); // output analog values read from Serial.println(readVal); A4
27   Serial.print(tempC); // temperature output
28   Serial.println(" C"); // output units
29
30   delay(1000); //1 second delay
31 }
31 // set Steinhart-Hart temperature measurement function

```



When you upload a sketch, you can see the result of converting the resistance value of the thermistor to temperature using the B parameter expression..

MIC Sensor



Sound sensors are also called sound sensors, microphone sensors, or sound sensors. This is a device that collects sound from the surrounding area through a microphone, enters the LM386 amplifier and measures it in size (db, decibels) regardless of the low (frequency) of the sound. Note, however, that there is no linear relationship between the decibel size of the actual sound and the analog input value.

When energized, the power lamp turns on and the output LED turns on when the sound is detected. Turn the sensitivity controlled variable resistance to adjust it to the moment when the output LED turns into a load

You can use sound sensors to create LED lights that respond to the volume size of your speakers, or to turn the lights on and off by recognizing clapping sounds.



CAK Starter Code > 15_MIC_Clap_ONOFF

```

1  /* This sketch turns on the LED when you clap twice.
2  * Shows the LED illuminates when hit twice again.
3  */
4
5  #include <Wire.h>
6  #include <LiquidCrystal_I2C.h>
7
8  LiquidCrystal_I2C lcd(0x27, 16, 2); // set LCD I2C address. 16kans2joules LCD use
9
10 const int sampleWindow = 125; // sample period milliseconds (125 ms = 8 Hz)
11 int ledPin = 13;           // Red LED connection to pin 13
12

```

```

13 int soundValue = 0; // store the value of the sound sensor
14 int clapCounter = 0; // Save clap count
15 double threshold = 2.0; // set clapping sound threshold voltage value
16
17 void setup() {
18
19   Serial.begin(9600); // initiate serial communication at 9600 speeds
20
21   lcd.init(); // Initialize LCD
22   lcd.backlight(); // turn on the backlight (lcd.noBacklight() turns off the backlight).
23   lcd.clear(); // clear LCD screen
24   lcd.setCursor(0, 0); // First line first column
25   lcd.print("LED OFF"); // message output
26   lcd.setCursor(0, 1); // Line 1st column
27   lcd.print("CLAP TWICE~"); // output messages
28
29   pinMode(ledPin, OUTPUT); // set red LED to output
30 }
31
32
33 void loop() {
34
35   unsigned long start= millis(); // start sampling
36   unsigned int peakToPeak = 0; // Amplitude Value Variables
37   unsigned int signalMax = 0;
38   unsigned int signalMin = 1024;
39
40   // collect data for 125 milliseconds.
41   while (millis() - start < sampleWindow)
42   {
43     soundValue = analogRead(3); // specify analogue pin number 3.
44     if (soundValue < 1024) // Read data up to the ADC maximum (1024=10bit).
45     {

```

```

46 if (soundValue > signalMax)
47 {
48   signalMax = soundValue; // Store the maximum sound value in the signalMax variable.
49 }
50 else if (soundValue < signalMin)
51 {
52   signalMin = soundValue; // store the sound minimum in the variable (signalMin).
53 }
54 }
55 }
56 peakToPeak = signalMax - signalMin; // calculate peak-to-peak amplitude
57 double volts = (peakToPeak * 5) / 1024; // convert the ADC value to a voltage value. Reference Voltage
58 5V
59 Serial.println(volts);
60 if (volts >=threshold)
61 {
62   clapCounter ++;
63   Serial.println(soundValue); // Output soundValue value to serial monitor
64   Serial.println(clapCounter); // Output number of applause to serial monitor
65   delay(50);
66   // turn on the LED in two claps
67   if(clapCounter == 2)
68   {
69     digitalWrite(ledPin, HIGH); // turn on the red LED.
70     Serial.println("LED ON"); // Output a message to the serial monitor
71     // LCD output
72     lcd.clear();
73     lcd.setCursor(0, 0); // First line first column
74     lcd.print("LED ON"); // message output
75     lcd.setCursor(0, 1); // Line 1st column
76     lcd.print("CLAP TWICE "); // output message
77   }
78   // turn off the LED in four claps

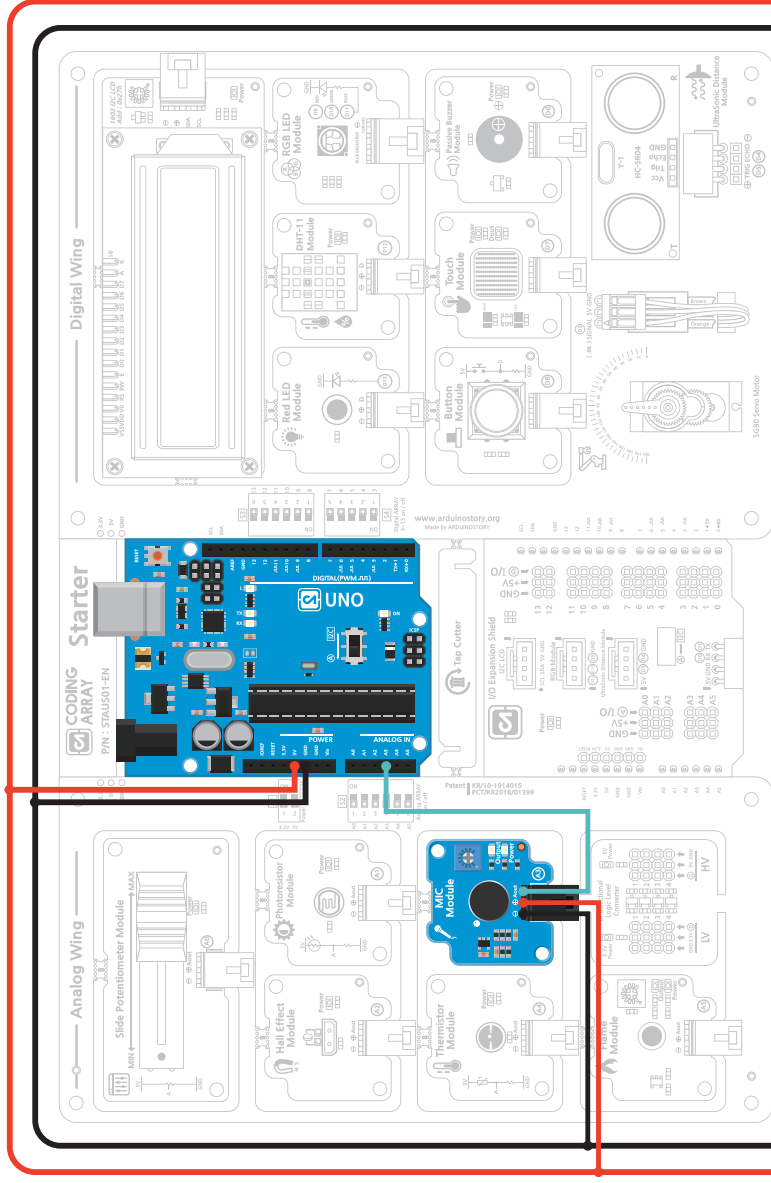
```

```

79   if (clapCounter == 4)
80   {
81     digitalWrite(ledPin, LOW); // Turn off the red LED.
82     Serial.println("LED OFF"); // output messages to serial monitors
83     // LCD output
84     lcd.clear();
85     lcd.setCursor(0, 0); // First line first column
86     lcd.print("LED OFF"); // message output
87     lcd.setCursor(0, 1); // Line 1st column
88     lcd.print("CLAP TWICE "); // output message
89     clapCounter = clapCounter % 2; // Save remaining 0 (initialize clap count)
90   }
91 }
92 }

```

**CODING
ARRAY**
STARTER KIT FOR ARDUINO



Using a sound sensor connected to the analog A3 pin, the ambient sound is received as an analog input value.

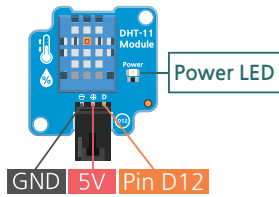
View Results

After you upload the sketch, open the serial window. If you clap, the serial window will often clap, and the LED will turn on when the clapping is detected twice. If the clapping sound is detected four times, the LED goes off and the next clapping count goes back to 1..

16 Measure the temperature and humidity with the sensor

CHAPTER 2

Humidity & Temperature Sensor



Hot-humidity sensors are sensors that can help people understand humidity at the same time. The temperature sensor has a positive temperature coefficient type in which resistance increases with increasing temperature and a negative temperature coefficient in which resistance decreases with increasing temperature.

The temperature sensor used in the eray kit is a DHT-11 model, which includes a fractional temperature sensor whose resistance decreases as the temperature increases, and a capacitive humidity sensor whose resistance varies with humidity. Temperature measurements range from 0° C to 50° C, with error of ±2° C. Humidity is represented by relative humidity and the humidity measurement range is 0 to 100%, with ±2% of the error. Relative humidity means the ratio of the amount of water vapor contained in the atmosphere at a given temperature and the amount of saturated water vapor (the higher the temperature, the greater the value of saturated water vapor) as a percentage).

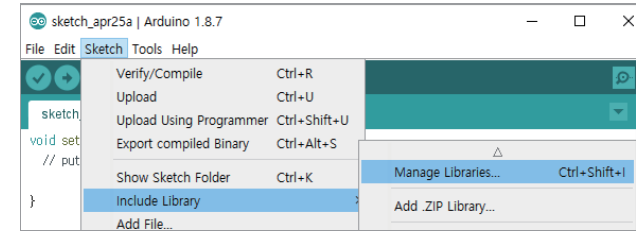
$$\text{relative humidity} = \frac{\text{actual water vapor quantity}}{\text{Current Temperature Saturated Water Vapor Volume}} \times 100$$

DHT11 Pins	
1	VCC
2	DATA
3	NC
4	GND

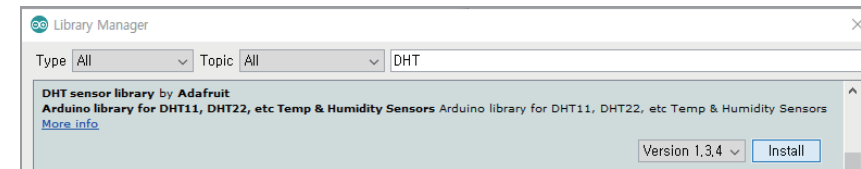
The DHT11 sensor consists of four pins, but the third pin is not used. Connects pin 1 to 5 V, pin 2 to data input/output, and pin 4 to GND (0 V), and does not require resistance. The module has a sampling rate of less than 1 Hz, i.e. not more than once per second.

The hot-humidity sensor provides both temperature and humidity at the same time, so the code is complex, but it can be used easily using a library. A DHT library is required to use the DHT** sensor.

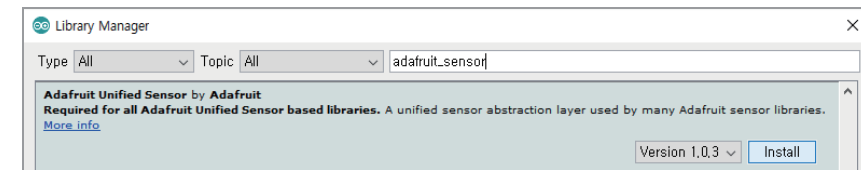
To add a library, click **Sketch > Include Library > Manage Libraries** to run the Library Manager.



Search the search box for "DHT" and install "DHT sensor library by Adafruit".



Install "Adafruit Unified Sensor by Adafruit" in the search box



- Adafruit Unified Sensor Library: https://github.com/adafruit/Adafruit_Sensor
- DHT Sensor Library: <https://github.com/adafruit/DHT-sensor-library>

Using a temperature sensor, the temperature and humidity of the room can be measured to make IoT products as well as a thermometer.



```

1  /* This sketch uses the temperature sensor DHT11 module connected to digital pin 12.
2  * Measure the ambient temperature and humidity
3  * Outputs result values to I2C LCD.
4  */
5
6  // Library for DHT11 Module Use
7  #include <Adafruit_Sensor.h>
8  #include <DHT.h>
9  #include <DHT_U.h>
10
11 // library for I2C LCD use
12 #include <Wire.h>
13 #include <LiquidCrystal_I2C.h>
14
15 // set the temperature sensor
16 #define DHTPIN    12    // DHT sensor to pin 12.
17 #define DHTTYPE   DHT11 // DHT 11.
18
19 DHT_Unified dht(DHTPIN,DHTTYPE); // form a dht object.
20
21 uint32_t delayMS;
22
23 // LCD settings
24 LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD I2C address. 16kans2joules LCD use
25
26 void setup() {
27   Serial.begin(9600); // initiate serial communication at 9600 speed
28
29   // Initialize LCD
30   lcd.init();
31   lcd.backlight(); // turn on the backlight (lcd.noBacklight() turns off the backlight).

```

```

32   lcd.setCursor(0,0); // First line first column
33   lcd.print("Hello~~~");
34   lcd.setCursor(0,1); // the first column of the second line
35   lcd.print("DHT Sensor Start");
36   delay(1000);
37   lcd.clear();
38   // start DHT sensor
39   dht.begin();
40   Serial.println("DHT11 Unified Sensor Example");
41
42   // print temperature sensor information
43   sensor_t sensor;
44
45   dht.temperature().getSensor(&sensor);
46   Serial.println("-----");
47   Serial.println("Temperature");
48   Serial.print ("Sensor: "); Serial.println(sensor.name);
49   Serial.print ("Driver Ver: "); Serial.println(sensor.version);
50   Serial.print ("Unique ID: "); Serial.println(sensor.sensor_id);
51   Serial.print ("Max Value: "); Serial.print(sensor.max_value); Serial.println(" *C");
52   Serial.print ("Min Value: "); Serial.print(sensor.min_value); Serial.println(" *C");
53   Serial.print ("Resolution:"); Serial.print(sensor.resolution); Serial.println(" *C");
54   Serial.println("-----");
55
56   // Print the Humidity Sensor Information
57   dht.humidity().getSensor(&sensor);
58   Serial.println("-----");
59   Serial.println("Humidity");
60   Serial.print ("Sensor: "); Serial.println(sensor.name);
61   Serial.print ("Driver Ver: "); Serial.println(sensor.version);
62   Serial.print ("Unique ID: "); Serial.println(sensor.sensor_id);

```

```

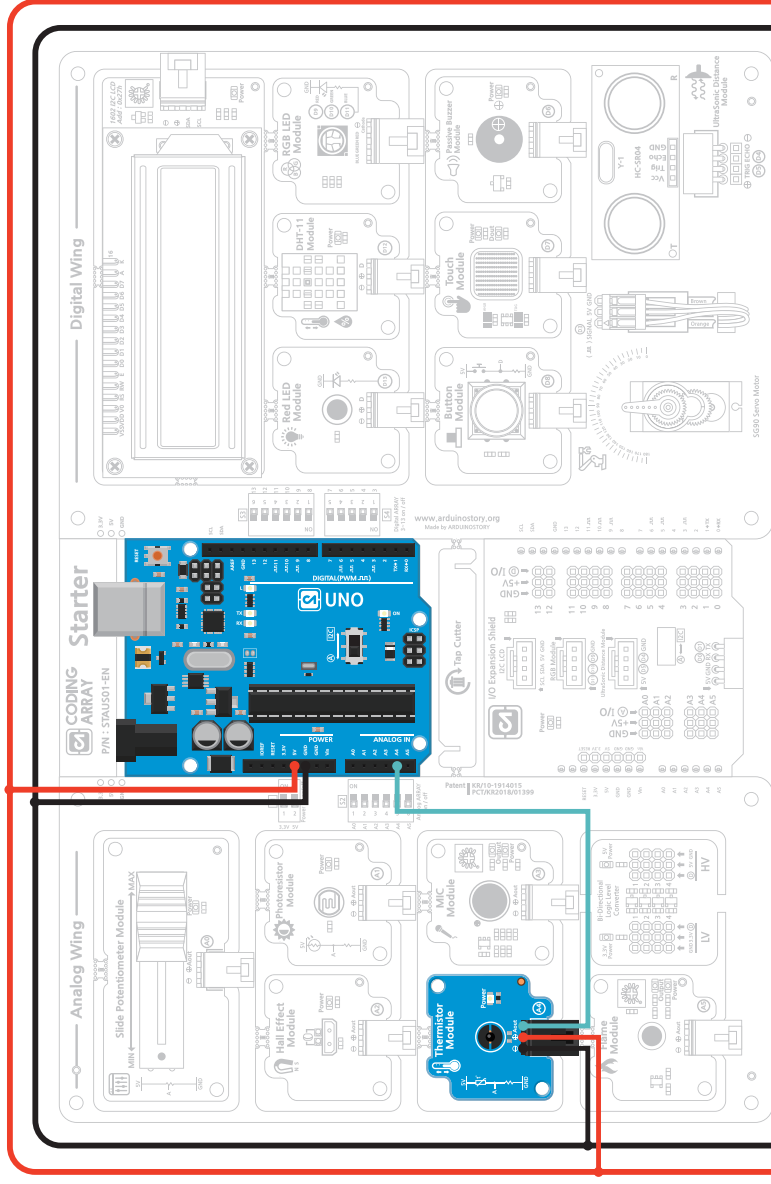
63 Serial.print ("Max Value: "); Serial.print(sensor.max_value); Serial.println("%");
64 Serial.print ("Min Value: "); Serial.print(sensor.min_value); Serial.println("%");
65 Serial.print ("Resolution: "); Serial.print(sensor.resolution); Serial.println("%");
66 Serial.println("-----");
67
68 // Time to read the sensor
69 delayMS =sensor.min_delay/1000;
70 }
71
72 void loop() {
73 // Delay between measurements.
74 delay(delayMS);
75
76 sensors_event_t event;
77 dht.temperature().getEvent(&event); // Get temperature event and print its value.
78 if(isnan(event.temperature)) {
79 Serial.println("Error reading temperature!");
80 }
81
82 else{
83 Serial.print("Temperature: ");
84 Serial.print(event.temperature);
85 Serial.println(" *C");
86
87 lcd.setCursor(0,0); // First line first column
88 lcd.print("Temp : "); // Output message
89 lcd.print(event.temperature,0); // Output without decimal point of measurement
temperature
90 lcd.print(" [C]"); // Output in units
91 }
92
93 dht.humidity().getEvent(&event); // Get humidity event and print its value.
94 if(isnan(event.relative_humidity)) {

```

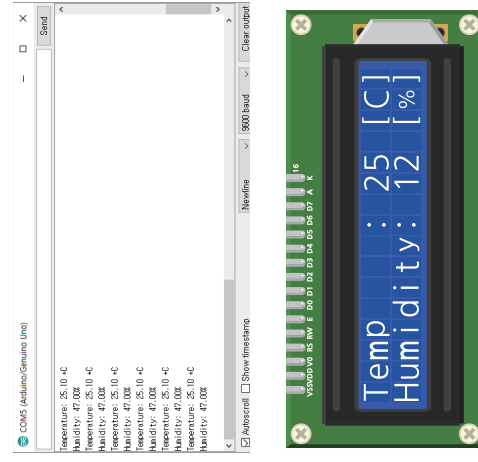
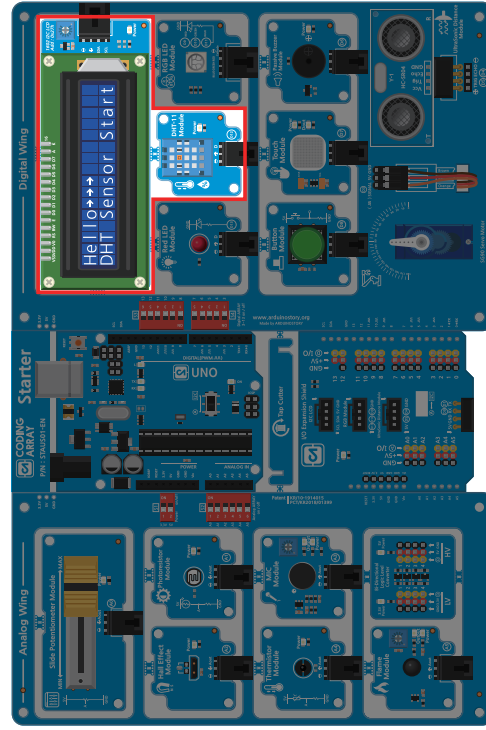
```

95 Serial.println("Error reading humidity!");
96 }
97 else{
98 Serial.print("Humidity: ");
99 Serial.print(event.relative_humidity);
100 Serial.println("%");
101
102 lcd.setCursor(0,1); // the first column of the second line
103 lcd.print("Humidity: "); // Message Output
104 lcd.print(event.relative_humidity,0); // Output measurement humidity
105 lcd.print(" [%]"); // Output in units
106 }
107
108 // Indicate temperature and humidity results on LCD
109 delay(1000); // delay of 1000 milliseconds to reliably read values
110 }

```

Use the NTC thermistor connected to the analog A4 pin to measure the temperature. Read the voltage at the NTC according to temperature and convert it to temperature using the Steinhart-Hart formula and the B parameter formula.

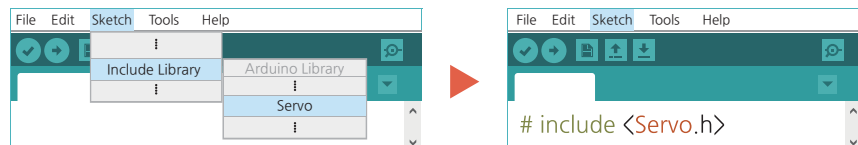


View Results

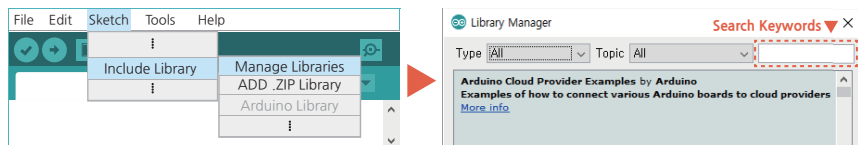
When you upload a program, the DHT-11 Hot and Humidity Module measures the temperature and humidity and shows the result values to the serial monitor and I2C LCD

Let's learn how to add a library..

■ Using the Arduino Library



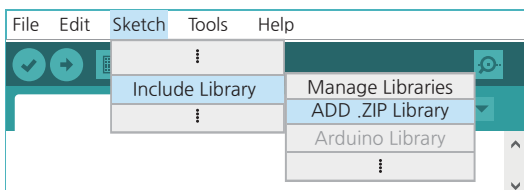
■ Search and install libraries in 'Library Manager'



When you add a library, the example contained in the library in **File > Example** can be used..

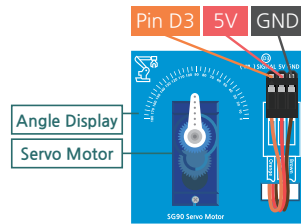
■ Installing the Extended Library

Search for the module keyword you want to use at <https://www.google.com/> or <https://github.com/> Download the library of the ZIP file format. Add a library that you downloaded in the following ways..



**CODING
ARRAY**
STARTER KIT FOR ARDUINO

Servo Motor



The servo motor is a motor that can rotate the axis at the desired angle and was used for small RC toys. It is also the first motor connected to Arduino. The array kit used a 9g servo motor showing the rotation angle between 0 and 180°. The servo motor has a line that can connect three pins, brown to GND, red to 5 V and orange to Arduino's PWM pin. Servo

motors consume significant power and, if more than one movement is required, must be supplied with external power rather than 5V of Arduino..

To control the servo motor, the servo library makes it easy to handle the servo motor. Using this library on the Uno Board disables the analogWrite() PWM function on pins 9 and 10 regardless of whether the pin has a servo motor.

A servo motor larger than the servo motor used in the array kit can move the park breaker or move the robot's hand, arm, etc..

CAK Starter Code > 17_01_Servo_Sweep

```

1  /* This sketch uses servo live
2  * Move the servo motor by 0 -> 180 degrees
3  * Move back to 180 -> 0 degrees.
4  * Note that servo motors cannot be rotated 360 degrees.
5  */
6
7  #include <Servo.h> // Include servo library
8
9  Servo myservo; // create object myservo to control servo
10
11 int position =0; // store the servo's position. Initial value 0
12
13 void setup() {
14   myservo.attach(3); // attach servo motor to pin 3
15 }
16

```

```

17 void loop() {
18   myservo.write(90); // position in the center of the servo motor shaft (90 degrees)
19   delay(1000);
20
21   myservo.write(0); // position 0 degrees on servo motor shaft
22   delay(1000);
23
24   myservo.write(180); // position 180 degrees on servo motor shaft
25   delay(1000);
26
27   myservo.write(90); // located in the center of the servo motor shaft
28   delay(1000);
29
30   for(position = 0; position <= 180; position += 1) { // increase by 1 degree to 0 to 180 degrees.
31     myservo.write(position); // move to a specified angle
32     delay(30); // wait until servo to arrive.
33   }
34
35   for(position = 180; position >= 0; position -= 1) { // decrease by 1 degree to 180 degrees.
36     myservo.write(position); // move to a specified angle
37     delay(30); // wait until servo to arrive.
38   }
39 }

```

myservo.attach(pin number, max value, max value);

recognises servo motor connected to PWM pin.

myservo : servo object

Pin number : Pin number with servo attached

Maximum value (optional) : Pulse width corresponding to the angle of (0 degrees) of the servo (in microseconds), default 544

Maximum value (optional) : Pulse width in microseconds corresponding to the maximum (180 degrees) angle of servo, default 2400

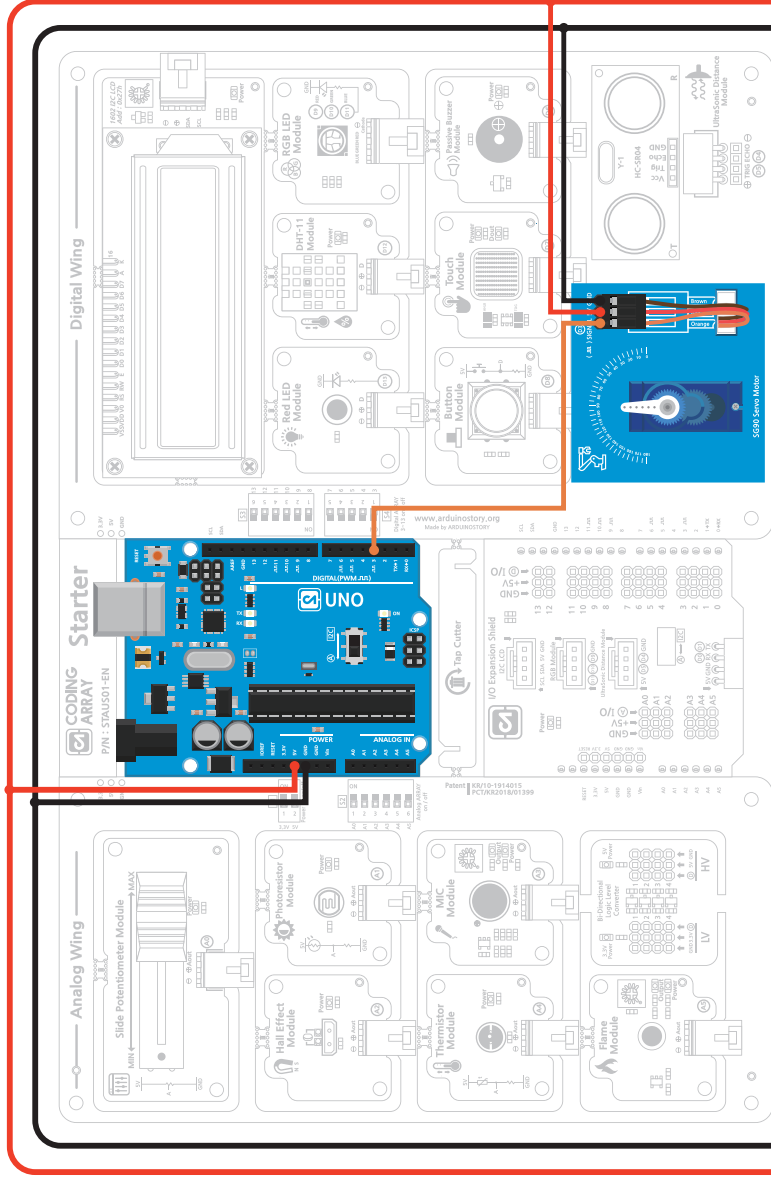
myservo.write; move the axis of the servo motor to the desired angle.

myservo : servo object

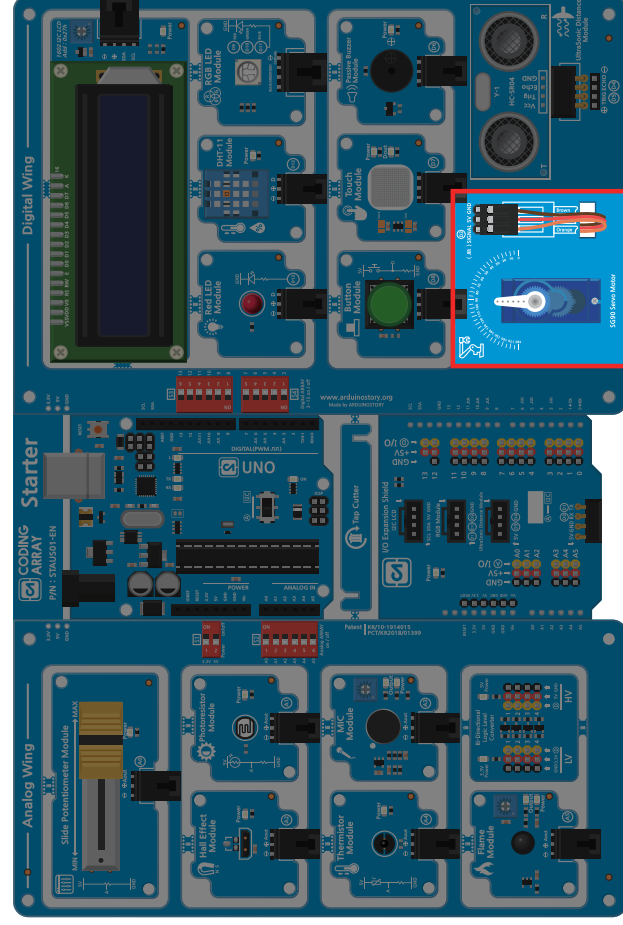
Angle : Value from 0 to 180 for servo to move

Position + = 1 and position = position + 1 and position ++ are all the same expressions to mean increasing position by 1 after executing function.

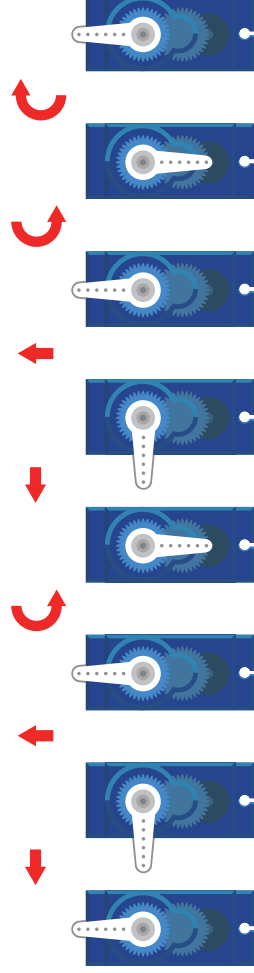
However, **+ position** is a different expression from the above three: 'After increasing position by one, function is executed'.



Using the DHT-11 temperature and humidity module connected to pin 12 digital, measure the temperature and display the results.



Upload the sketch and you can see the axis of the servo motor moving at an angle created by the code.



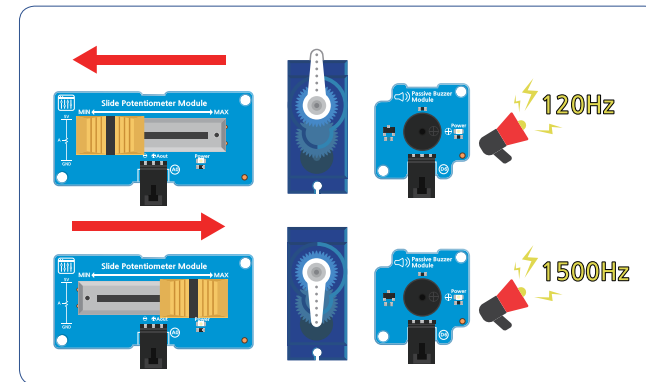
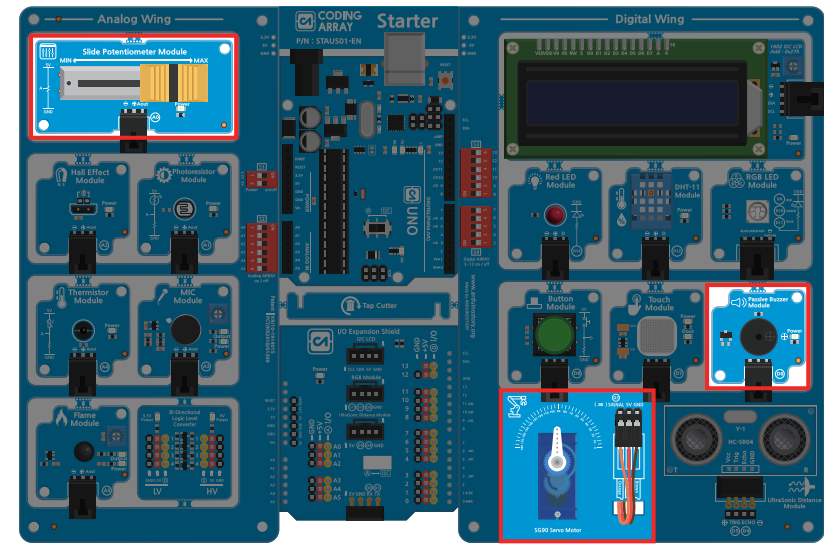
View Results



```

1  /* This sketch uses servo live
2  * The servo motor is between 0 and 180 degrees depending on the variable resistance value
3  * Move the axis.
4  */
5
6  #include <Servo.h>    // Include servo library
7
8  Servo myservo;      // create object myservo to control servo; up to 12 can be created
9
10 int position =0;    // store the servo's position. Initial value 0
11 int potPin=A0;     // Connect variable resistance to A1 pin
12
13 void setup() {
14   myservo.attach(3); // attach servo motor to pin 3
15 }
16
17 void loop() {
18
19   int val =analogRead(potPin); // Read variable resistance value (0-1023)
20   int servoVal =map(val,0,1023,0,180); // map variable resistance values to 0-180 degrees servo motor rotation angle
21   myservo.write(servoVal); // position of mapped servo
22   delay(15); // wait for the servo to arrive.
23
24   int pitchVal=map(val, 0, 1023,120,1500); // map variable resistance values to 120 to 1500 Hz sound frequencies
25   tone(6, pitchVal, 10); // output mapped sound value to manual buzzer connected to pin 6 for 10 milliseconds
26
27   delay (1); // delay of 1 millisecond for safety
28
29 }

```



**CODING
ARRAY**
STARTER KIT FOR ARDUINO

