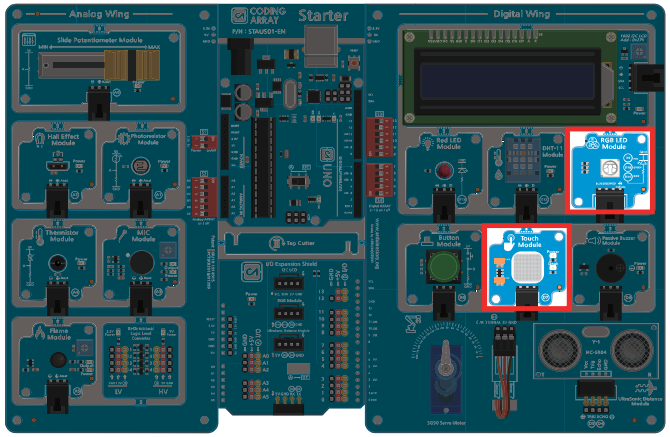
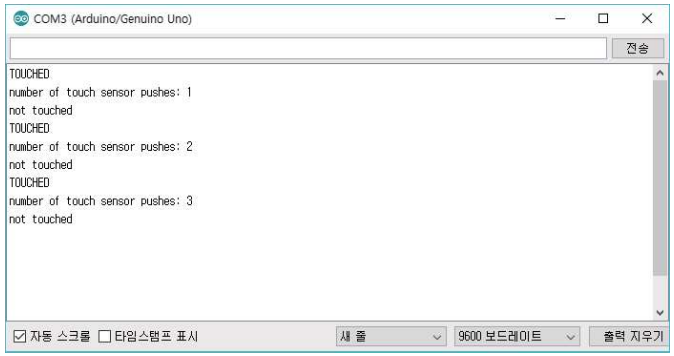
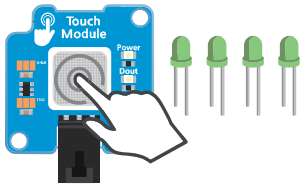
View Results







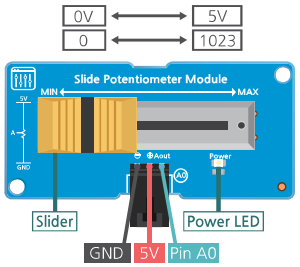
After you upload the sketch file, open the serial monitor. You can verify that the green LED's brightness increases when you press the touch sensor 1st, 2nd, and 3rd, and that the LED turns off when you press the fourth time. Using this method, it is possible to implement brightness adjustment such as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

CODING ARRAY
STARTER KIT FOR ARDUINO

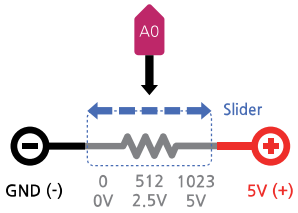
7 Read slide variable resistance value with analog input

CHAPTER 2

Slide Potentiometer

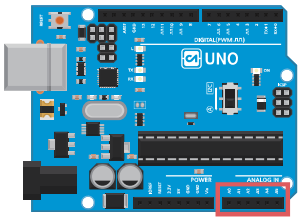


Resistance is enough to interfere with the flow of electric charges.
Unlike a typical set of values for resistance, variable resistance is also called potentiometer and voltage divider and is either turned around the center fireplace, or adjusted by pushing the slider left and right. The coding array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

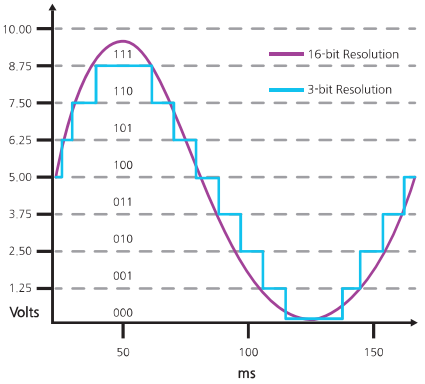


As you push the slider left and right, the resistance values change according to the position, and you return the value to the analog input pin connected to Aout by converting the changing voltage values between 0 V and 5 V to the analog input values. Analog values are read using the anlaogRead( ) function. 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
II	II
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)...

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

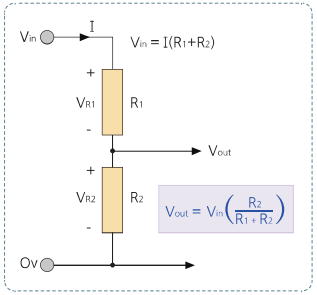
Let's find out about voltage distribution.

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,

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

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

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

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

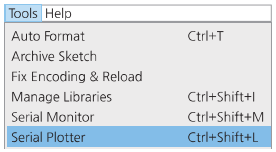
The size of the resistance increases as the length of the resistance increases, and the larger the cross-sectional area becomes smaller.

CAK Starter Code > 07_01_SlidePotentiometer

```
/*  Change the resistance value by pushing the variable resistance from side to side.
    As the resistance changes, 0 to 1023 analog values are entered as A0 pins.
    Analog values are converted to voltages and expressed as values on the serial monitor.
    Can use a serial plotter to express it in graphs.
*/

const int potentiometerPin = A0;  // Output value of variable resistance is read from A0
const int redLED = 13;
const int threshold = 400;

void setup() {
  pinMode(redLED, OUTPUT); // Set red LED to output
  Serial.begin(9600);     // Starts serial communication at 9600 speed
  // Analog input/output pins need not be declared..
}

void loop() {
  int analogValue = analogRead(potentiometerPin);    // Read the analog value of variable resistance and store it in analogValue
  // AnalogValue stores integer values in the range 0 to 1023.

  float voltage = analogValue * (5.0 / 1023.0);  // Convert Analog Readings to Volts 0 to 5 V
  // Store in float variable because the math result value is real

  //Serial.print("Analog Value : ")
  //Serial.println(analogValue);                // Indicate the value of analogValue one line in the serial window.
  Serial.print("Voltage : ");
  Serial.println(voltage);                    // Print the converted voltage value one line in the serial window

  if (analogValue > threshold) {   // If the value stored in analogValue is greater than 400
    digitalWrite(redLED, HIGH);    // Turn on the LED.
  } else {                         // If the value stored in analogValue is less than or equal to 400
    digitalWrite(redLED, LOW);     // Turn off the LED.

    delay(1);                      // Wait 1 millisecond to read reliably
  }
}
```

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.

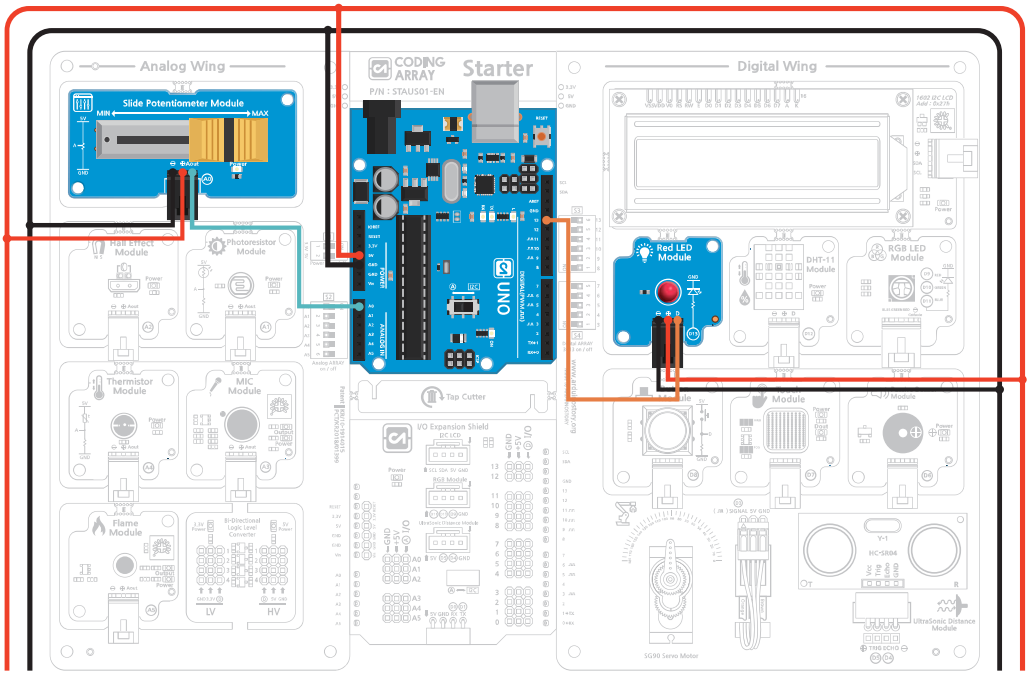
float Variables; declared for the purpose of storing decimal mistakes. When performing math operations with float, you must add a decimal point. (Example: 5.0 /1023.0) Otherwise (e.g. 5 /1023= 0) treated as an integer int.



! CAUTION!

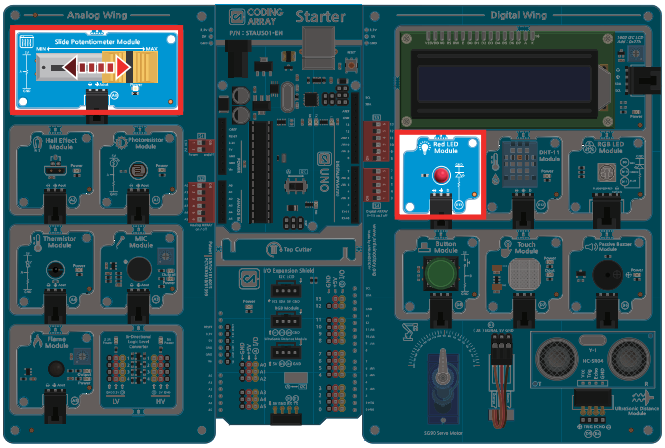
- The analog input/output does not have to be set in pinMode separately.
- Click Menu Bar > Tools > Serial Plotter (Ctrl+Shift+L) to open the serial plotter. During the output of the serial plotter, the serial monitor window does not open simultaneously..

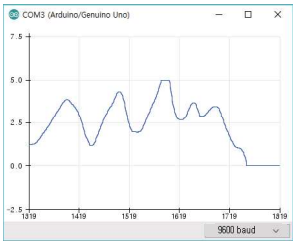
CODING ARRAY CIRCUIT

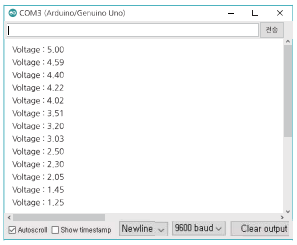


Touch sensors connected to pin 7 digital can be used like button switches as sensors that return HIGH (1) digital input values when the body touches them and LOW (0) digital input values if the body does not touch them. Continue to return the HIGH input value while the body touches it, but by adjusting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

```
/*  As the variable resistance value increases, the color of the RGB LED changes to red->green->blue... */

int potPin = A0;        // Output of variable resistance connected to analogue pin A0
int potVal = 0;         // Variables that store analog (0-1023) values read from variable resistance

const int redPin = 9;        // Red LED No. 9
const int greenPin = 10;     // Green LED No. 10
const int bluePin = 11 ;     // Blue LED No. 11

int redV = 0;                // Set red LED analog value (0-255)
int greenV = 0;              // Set green LED analog value (0-255)
int blueV = 0;               // Set blue LED analog value (0-255)

void setup() {
  pinMode(redPin, OUTPUT);   // Set pin 9 to output
  pinMode(greenPin, OUTPUT); // Set pin 10 to output
  pinMode(bluePin, OUTPUT);  // Set pin 11 to output
}

void loop() {
  potVal = analogRead(potPin);  // The analog output value of variable resistance is read from the A0 pin (0-1023).
  int ledLevel = map(potVal, 0, 1023, 0, 255); // Converts the analog input value to the analog output value (0-255)

  if (potVal < 341){  // When the value of variable resistance is 1 divided into three stages (0-340)
    redV = 255 - ledLevel;     // The red is getting lighter
    greenV = ledLevel;         // The green is getting darker
    blueV = 1;                 // Blue has no effect
  }

  else if (potVal < 682) { // When the value of variable resistance is 2 divided into three stages (341-681)
    redV = 1;                      // Red has no effect
    greenV = 255 - ledLevel;       // The green is getting lighter
    blueV = ledLevel;              // The blue is getting darker
  }

  else { // When the value of variable resistance is 3 divided into three stages (682-1023)
    redV = ledLevel;               // The Red is getting darker
    greenV = 1;                    // Green has no effect
    blueV = 255 - ledLevel;        // The blue is getting lighter
  }

  analogWrite(redPin, redV);       // Write values to red pins
  analogWrite(greenPin, greenV);   // Write values to green pins
  analogWrite(bluePin, blueV);     // Write values to blue pins
}
```

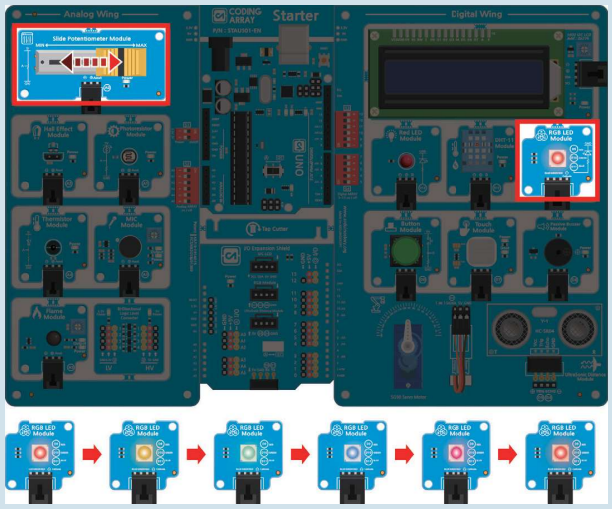
map (number to convert, current maximum value, current maximum value, current maximum value to be converted, maximum value to be converted)

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

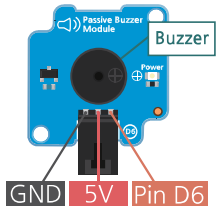


After uploading the sketch, pushing the slider of variable resistance left and right changes the analog input value.
As the resistance value increases, you can see that the color of the RGB LED changes to red – 〉 green–〉 blue..

8 Melody with a Passive Buzzer

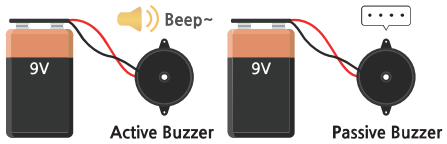
CHAPTER 2

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..

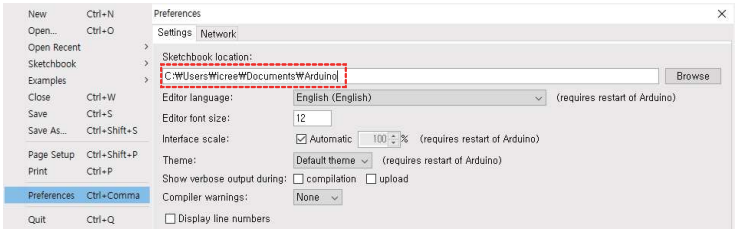
Each sound meter has its own frequency of shaking and should be predefined so that the frequency (in units Herz HZ) value of the sound meter. Write the frequency required to play the melody by defining it as the #define constant name value (e.g. #define NOTE_C1 33) before { }. #define is a function that 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
```

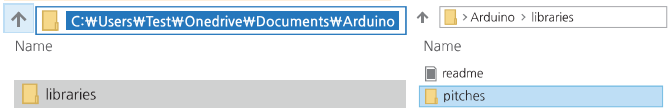
Let's learn how to save the pitches.h header file..

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.".

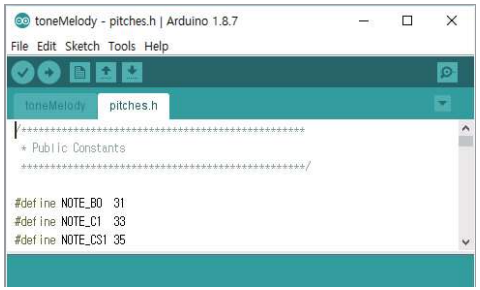
Way 1. File > Preferences > Open the folder in the Sketchbook location..



Create pitches folder under libraries folder.

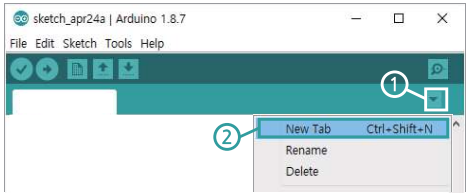


When you open the file > Example > 02. Digital > ToneMelody in the pitches folder, copy the pitches.h on the right tab and save it as a file and make it a library.

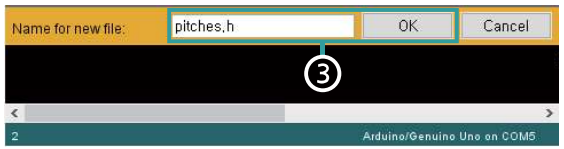


You can also edit and use it as needed. Bring up header file to #include "pitches.h" before void setup { }.

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 { }.

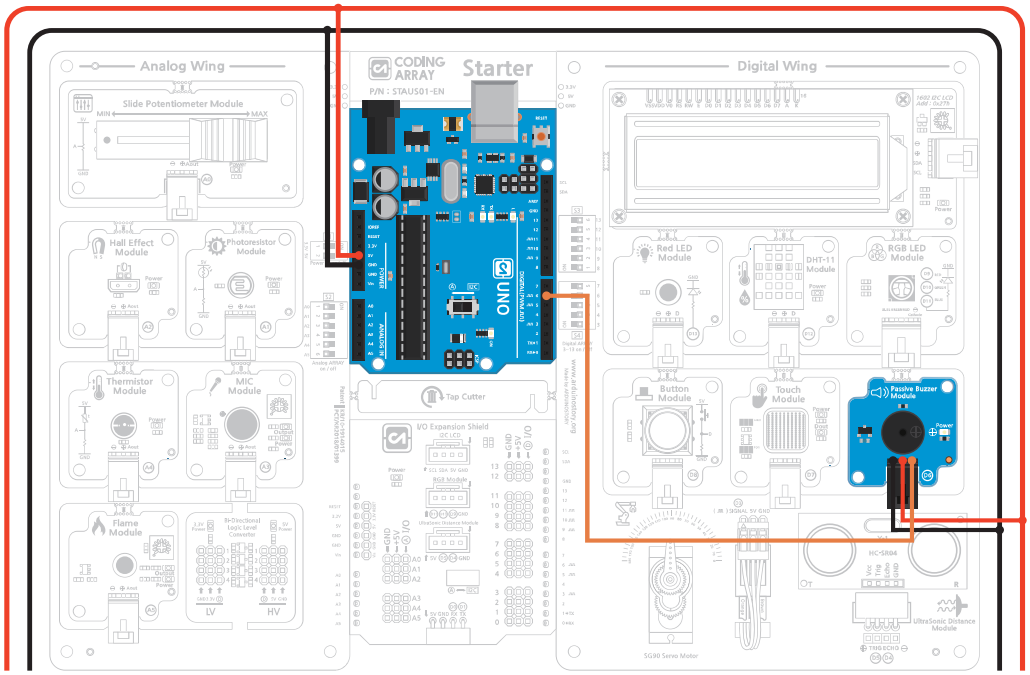
CAK Starter Code > 08_PassiveBuzzer

```
/* This sketch plays a note through a buzzer connected to pin 6.
 * The * tone() function calls up the pitches.h header file to give a given frequency sound.
 * Play a familiar 'school bell' song to us.
 * In this sketch, the code is inserted in the setup part and played only once.
 * If the loop part is coded, the performance may be repeated.
 */

#include "pitches.h"

int buzzer=6;          // Connect the Piezo buzzer to No. 6.
                       // the pitch of playing
int melody[] ={NOTE_G7,NOTE_G7,NOTE_A7,NOTE_A7,NOTE_G7,
               NOTE_G7,NOTE_E7,NOTE_G7,NOTE_G7,NOTE_E7,
               NOTE_E7,NOTE_D7,0,NOTE_G7,NOTE_G7,NOTE_A7,
               NOTE_A7,NOTE_G7,NOTE_G7,NOTE_E7,NOTE_G7,
               NOTE_E7,NOTE_D7,NOTE_E7,NOTE_C7,0
              };
                       // Sound length, 4 = crotchet, 2 = minim
int noteDurations[] ={4,4,4,4,4,4,2,4,4,4,4,3,1,4,4,4,4,4,4,2,4,4,4,4,3,1};

void setup() {
 for(int thisNote =0; thisNote <26; thisNote++)
 {
   int noteDuration =1000 /noteDurations[thisNote];
   tone(buzzer, melody[thisNote], noteDuration);      // Connect the Piezo buzzer to No. 6
   int pauseBetweenNotes =noteDuration *1.30;         // phonetic delimiting
   delay(pauseBetweenNotes);                          //delay
   noTone(buzzer);                                    // Stop playing music
 }
}

void loop() {
}
```

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.

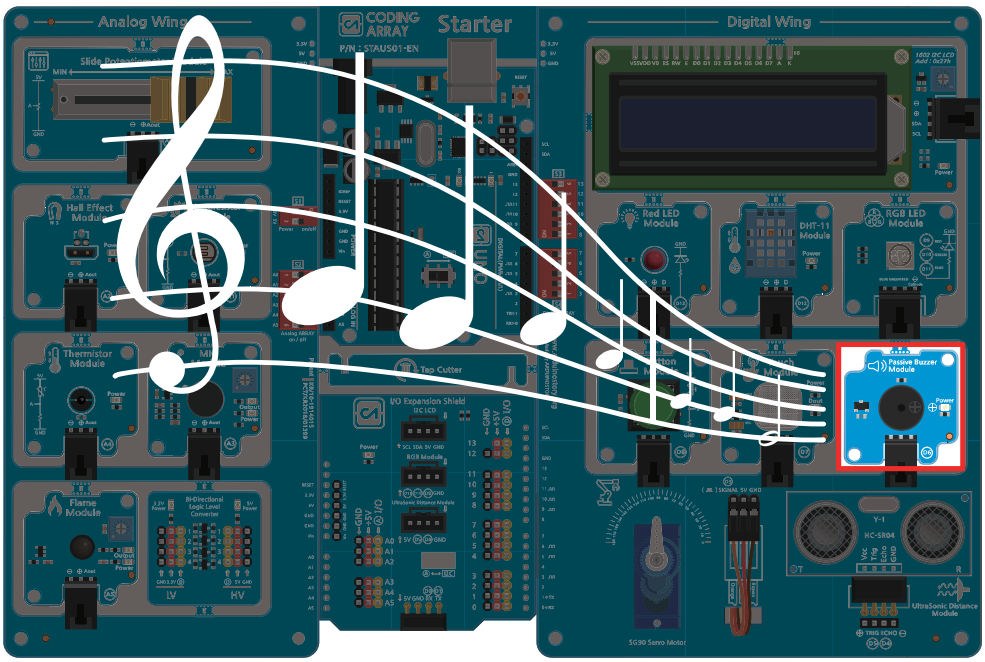
noTone (Pin Number); Stop the waveform generated by tone( ).
In order to play different scales on different pins, you must call noTone before calling the next pin.

CODING ARRAY CIRCUIT



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

View Results



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.

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

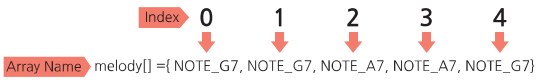
A number in [ ] is called an index, and the index value begins at zero..

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..



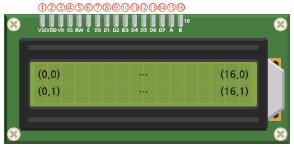
melody[0] ⟷ NOTE_G7
melody[1] ⟷ NOTE_G7
melody[2] ⟷ NOTE_A7
melody[3] ⟷ NOTE_A7
melody[4] ⟷ NOTE_G7

CODING ARRAY
STARTER KIT FOR ARDUINO

9 Texting on 1602 I2C LCD

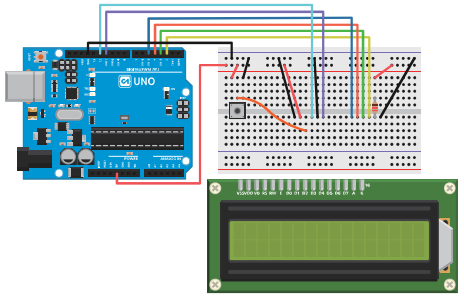
LCD (Liquid Crystal Display)

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.

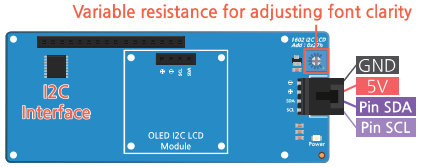


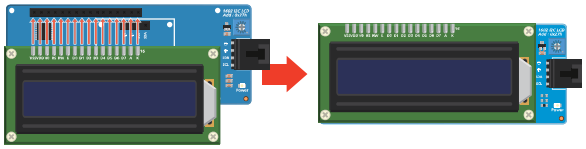
1	V_{SS}	GND connection	power
2	V_{dd}	5V connection	
3	V_0	the darkening of letters	Setting
4	RS	Select space to store LCD values	
5	RW	Select a value for the LCD in read/write mode	
6	E	To write a value to the LCD	
7	DB0	Data input/output pin Used when LCD and Arduino exchange prices.	data
8	DB1		
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	

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..