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

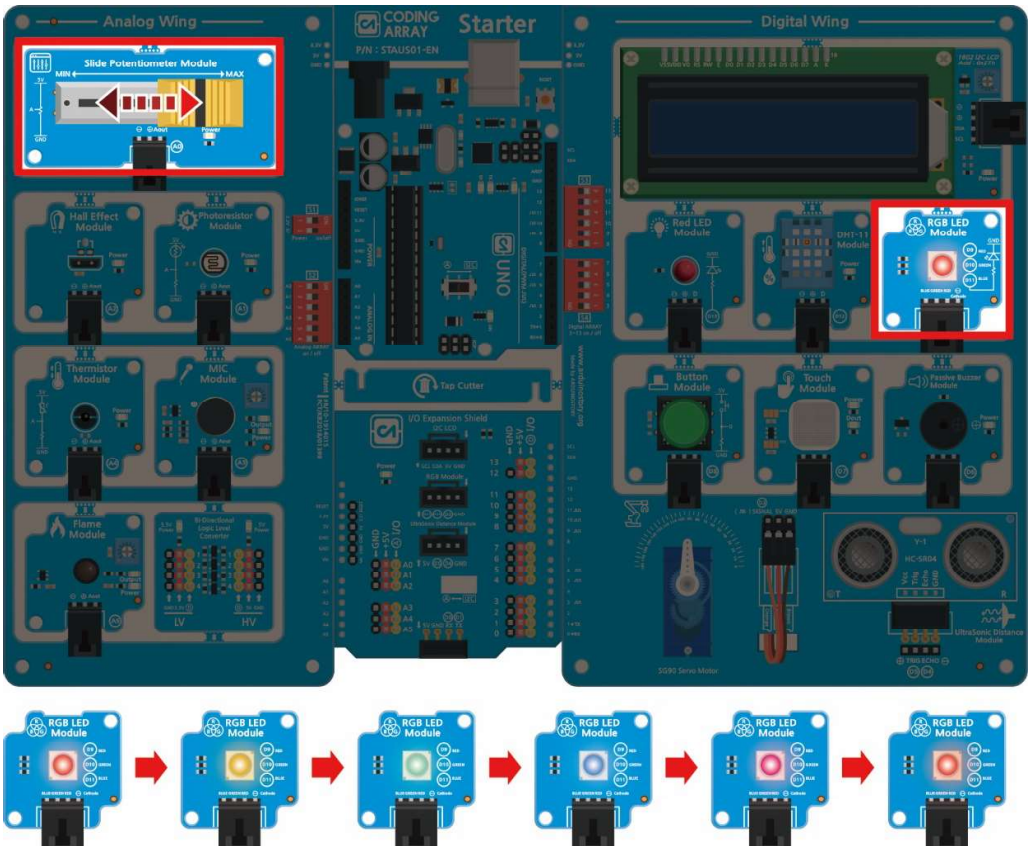


Figure 54

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

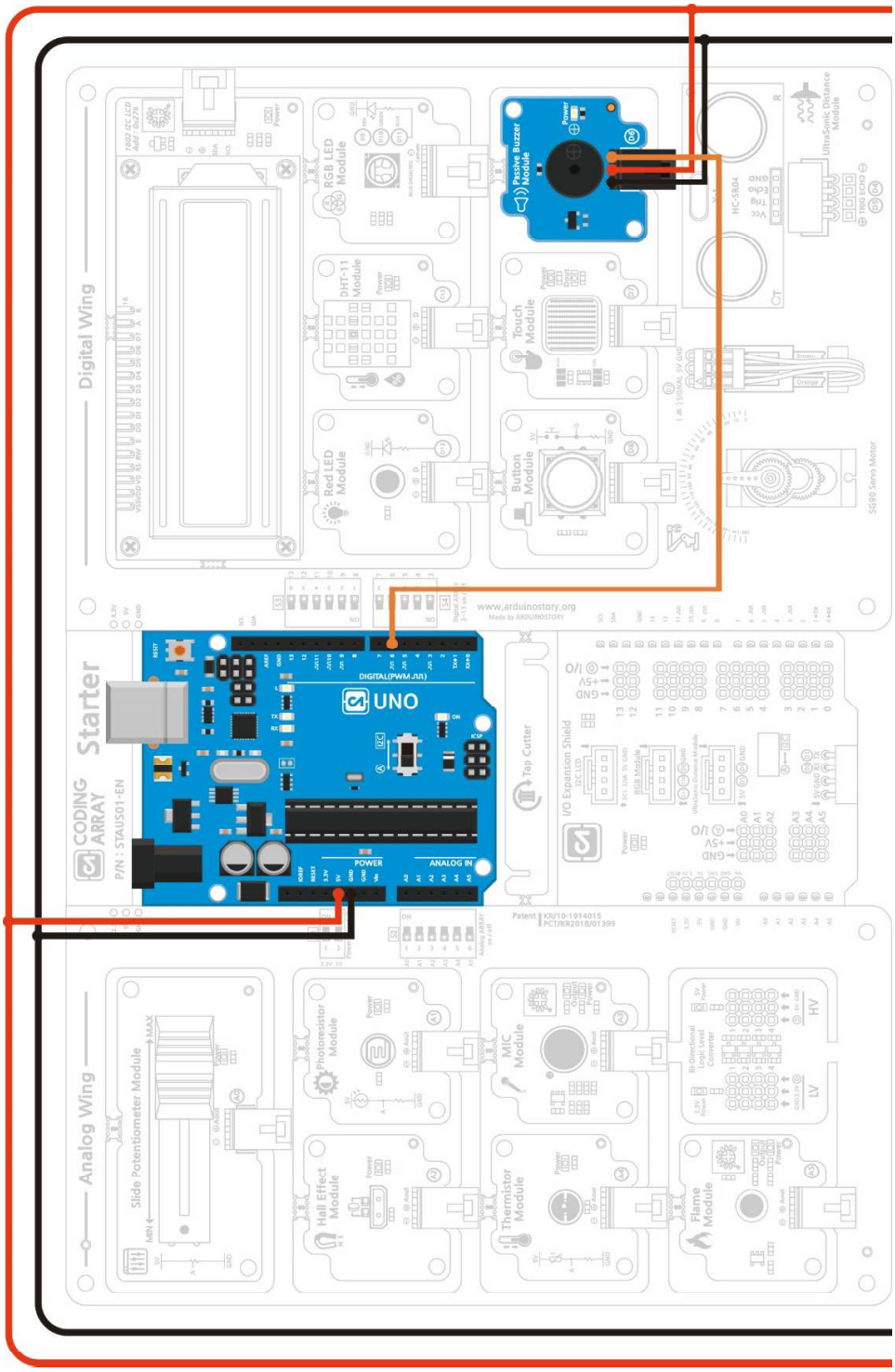
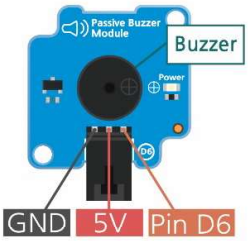


Figure 55

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

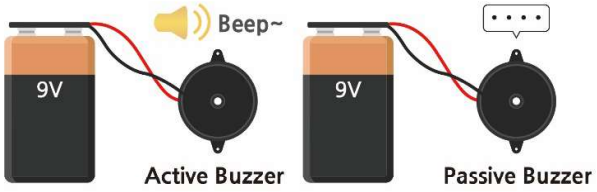
Passive Buzzer

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.

Figure 56



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..

Figure 57

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

Table 11

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..

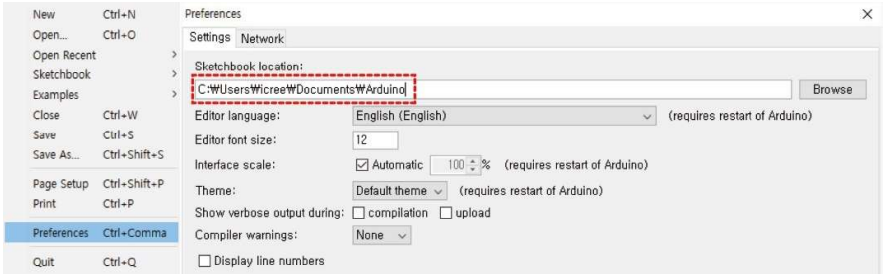


Figure 58

Create pitches folder under libraries folder.

Figure 59

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.

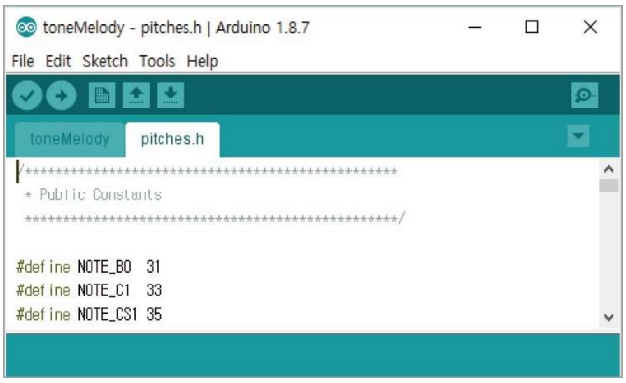


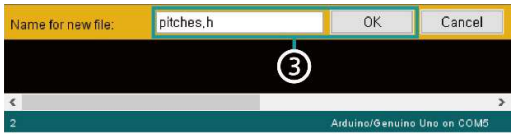
Figure 60

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..

Figure 61



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

Figure 62

Figure 63

When you open **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.

Figure 64

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.

Figure 65

View Results

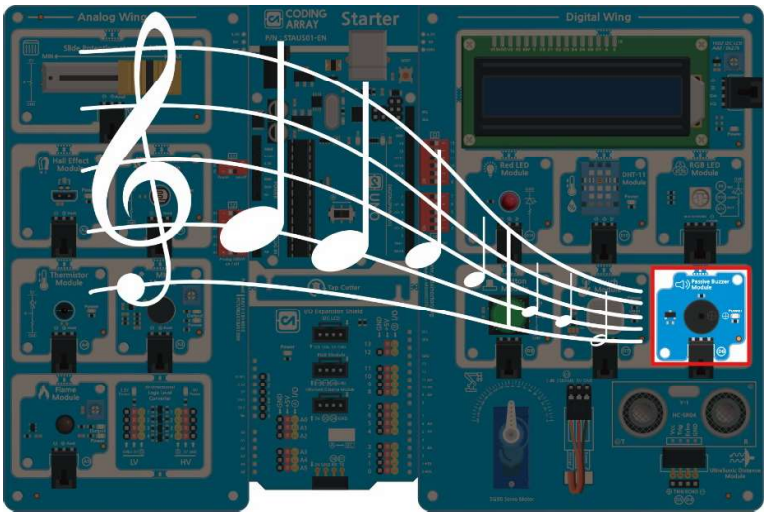


Figure 66

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 the layout..

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..

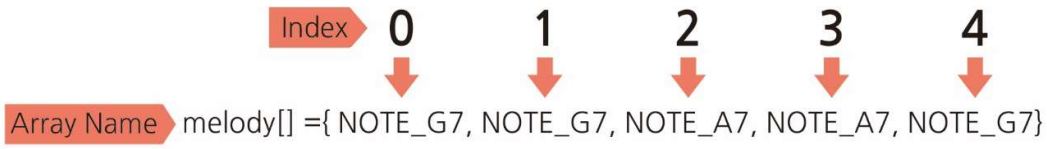


Figure 79

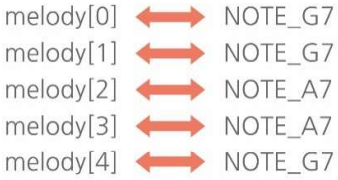


Figure 80

9. Texting on 1602 I2C LCD

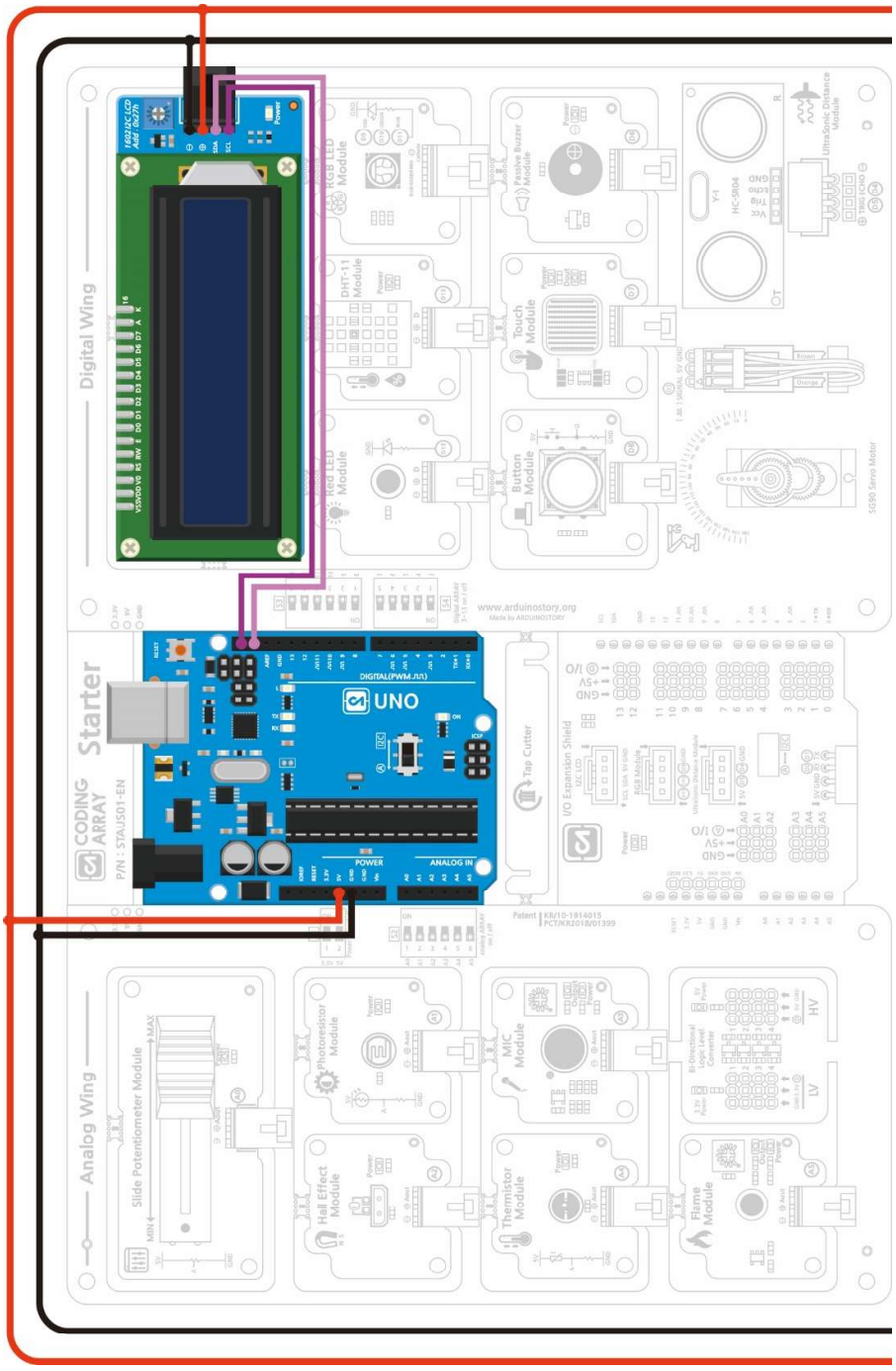


Figure 67

✧ 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..

LCD (Liquid Crystal Display)

LCD is a liquid crystal display device that is often used for wrist watches and calculator screens. Thin and light, low power consumption makes it good for portable devices. However, background lighting (backlight) is necessary for clear display, and the viewing angle is narrow, and Han-gul or special characters have the disadvantages of not being well expressed.

The 1602 LCD used in the starter kit is a liquid crystal display unit that outputs 16 characters (a total of 32 characters) on each of the two lines across which it is used to output messages or sensor data. The 1602 LCD consists of 16 pins as shown below for connection..

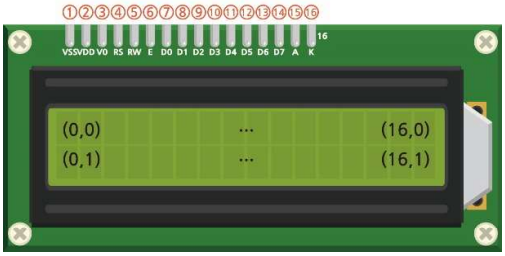


Figure 68

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	

Table 12

Inter-Integrated Circuit Interface

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

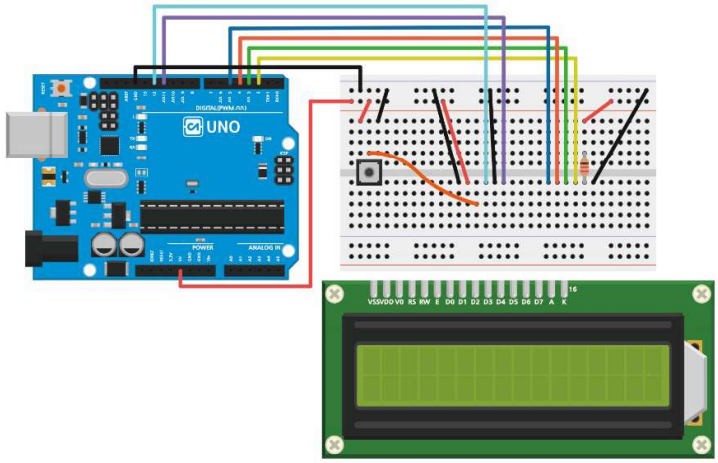


Figure 69

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..

Figure 70

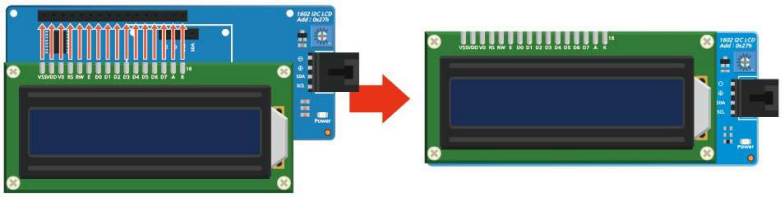


Figure 71

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..

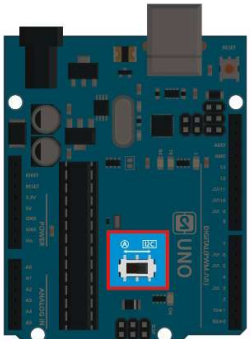


Figure 72

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..

Figure 73

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."**

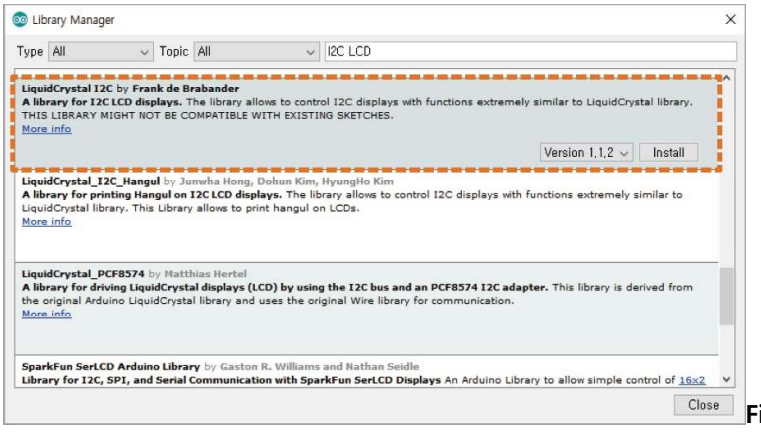


Figure 74

The default address is 0×27 (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.

CAK Starter Code > 09_01_1602LCD_AddressScanning

```
/* This sketch is designed to use 1602 LCDs using I2C communication.
 * Scans the address to show the results on the serial monitor.
 */

#include <Wire.h>              // Includes the Wire Library for I2C communication.

void setup() {
  Wire.begin();
  Serial.begin(9600);              // Starts serial communication at 9600 speeds
  while(!Serial);                  // Wait for the serial monitor.
  Serial.println("₩nI2C Scanner");
}

void loop() {
  byte error,address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices =0;

  for (address =1; address <127; address++)  {
    // The i2c_scanner has determined whether the device has approved the address.
    // Use the Write.endTransmission return value to know.
    Wire.beginTransmission(address);
    error =Wire.endTransmission();

    if (error ==0) {
      Serial.print("I2C device found at address 0x");
      if(address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");
      nDevices++;
    }
    else if(error==4) {
      Serial.print("Unknown error at address 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices ==0)
    Serial.println("No I2C devices found₩n");
  else
    Serial.println("done₩n");

```

```
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 "In" 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();	urn 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(데이터);	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.

Table 13

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

CAK Starter Code > 09_02_1602LCD_HelloCAK

```
/*  This sketch shows Hello! ^ _ ^ ! Coding Array Kit for 1602 LCDs using I2C communication
 *  Show the string, using scrollDisplayRight and scrollDisplayLeft methods
 *  Scrolls to the left and right.
 *  Flashes the screen using noDisplay and discaly method to indicate that one loop is complete.
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16-kan2 line LCD.
                                   // Put the scanned address instead of 0x27.
void setup(){
  lcd.init();
  lcd.backlight();                 // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
  lcd.setCursor(0,0);                  // first line first column
  lcd.print("Hello ! ^ _^ !");
  lcd.setCursor(0,1);                // 2nd line first column
  lcd.print("Coding Array Kit");
  delay(1000);
}

void loop(){
  // Scroll 16 length of string to the left
  for(int positionCounter = 0; positionCounter < 16; positionCounter++) {
    lcd.scrollDisplayLeft();                        // Scroll to the left by one position
    delay(200);                                     // for 200 milliseconds
  }

  // String length 16+ Rows 16 Scroll to the right at a total of 32 locations
  for(int positionCounter = 0; positionCounter < 32; positionCounter++) {
```

```
    lcd.scrollDisplayRight();                    // Scroll to the left by one position
    delay(200);                                  // for 200 milliseconds
  }

  // Scroll to the left 16 positions to center
  for(int positionCounter = 0; positionCounter < 16; positionCounter++) {
    lcd.scrollDisplayLeft();                     // Scroll to the left by one position
    // wait a bit:
    delay(200);                                  // for 200 milliseconds
  }

  // noDisplay and dispaly
  lcd.noDisplay();    // Turn off the screen
  delay(500);         // for 0.5 second
  lcd.display();      // Turn on the screen
  delay(500);         // for 0.5 second
}
```

CAK Starter Code > 09_03_1602LCD_Autoscroll

```
/*  This sketch shows that 1602 LCD uses I2C communication.
 *  Numbers from 0 to 9 are displayed on the screen.
 *  Use the autoscroll( ) and noAutoscroll( ) methods.
 *  Show how to move all strings left or right.
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16 Space 2-line LCD.

void setup(){
  lcd.init();
  lcd.backlight();              // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
}

void loop() {
  lcd.setCursor(0,0);                    // Position the cursor (0,0) in the upper left corner.

  for(int thisChar = 0; thisChar <10; thisChar++) {
    lcd.print(thisChar);                 // The numbers from 0 to 9 are shown on the LCD.
    delay(500);                          // For 0.5 Second
  }

  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  }
```

CAK Starter Code > 09_04_1602LCD_TextDirection

```
/*  This sketch shows that 1602 LCD uses I2C communication.
 *  Show one alphabet from a to Z.
 * 'a through l' is written from left to right.
 *  'm through r' is written from right to left
 *  's through z' causes text to appear from left to right again.
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16 Space 2-line LCD.

int thisChar ='a';                                  // thisChar Save

void setup(){
  lcd.init();
  lcd.backlight();              // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
  lcd.cursor();                 // Turn on the cursor.
}

void loop(){
  if(thisChar =='m') {                  // If thisChar is m, change direction.
    lcd.rightToLeft();                      // From the next letter, mark to the left.
  }

  if(thisChar =='s') {                  // If thisChar is s
    lcd.leftToRight();                      // From the next letter, mark to the left.
  }

  if(thisChar >'z') {                   // When thisChar is out of z,
    lcd.home();                             // Go to (0,0)
    thisChar ='a';                          // Restart from the beginning
  }

  lcd.write(thisChar);                  // Indicate thisChar value on LCD.
  delay(1000);                          // for a second
  thisChar++;                           // increase thisChar value one by one
}
```

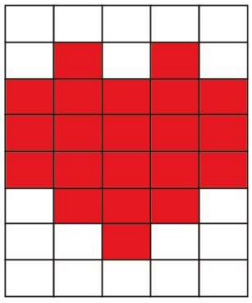
CAK Starter Code > 09_05_1602LCD_CustomCharacters

```
/*  This sketch shows that 1602 LCD uses I2C communication.
 *  Set up a special character or Figure to "I ♥ Array Kit !" and smile on the first line of the LCD.
 *  In the second row, the shape of a person who raises and lowers his or her arms is displayed.
 *  Learn how to indicate that the value of A0 variable resistance is constant.
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16 Space 2-line LCD.

// Create special characters
byte heart[8] ={
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000
};

byte smiley[8] ={
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};

byte frownie[8] ={
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b00000,
  0b01110,
  0b10001
};

byte armsDown[8] ={
  0b00100,
```

```
  0b01010,
  0b00100,
  0b00100,
  0b01110,
  0b10101,
  0b00100,
  0b01010
};

byte armsUp[8] ={
  0b00100,
  0b01010,
  0b00100,
  0b10101,
  0b01110,
  0b00100,
  0b00100,
  0b01010
};

void setup(){
  lcd.init();
  lcd.backlight();              // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)

  // Define New Characters
  lcd.createChar(0,heart);
  lcd.createChar(1,smiley);
  lcd.createChar(2,frownie);
  lcd.createChar(3,armsDown);
  lcd.createChar(4,armsUp);

  lcd.setCursor(0,0);           // first line first column
  // Write text to LCD
  lcd.print("I ");
  lcd.write(byte(0));                   // Use a heart that is stored at 0 bytes..
  lcd.print(" Array Kit! ");
  lcd.write((byte)1);                   // Use a smiley stored in one byte..
}

void loop(){
  int sensorReading = analogRead(A0);       // Read the variable resistance value of A0.
  int delayTime = map(sensorReading,0,1023,200,1000);  // Map resistance values from 200 to 1000
  lcd.setCursor(4,1);                   // Set the cursor to the bottom 5th position
  lcd.write(3);                         // Draw a person with his arm down on number 3.
  delay(delayTime);                     // Delay by variable resistance value
  lcd.setCursor(4,1);                   // Set the cursor to the bottom 5th position
  lcd.write(4);                         // Draw the person with the arm up stored in number 4.
  delay(delayTime);                     // Delay by variable resistance value
}
```

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

Figure 75

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.

View Results

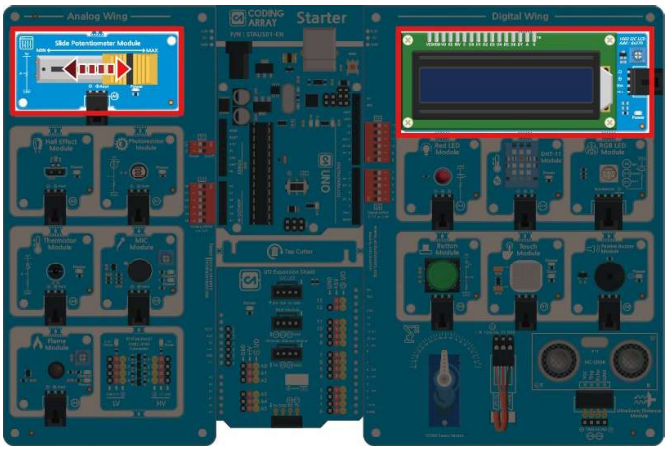


Figure 76

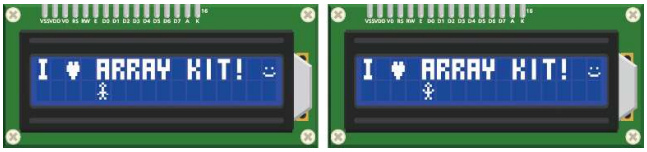


Figure 77

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

CAK Starter Code > 09_06_I2C_1602LCD_SerialDisplay

```
/*  This sketch shows that 1602 LCD uses I2C communication.
    Try printing the entered characters in the serial window.
*/

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address. Use 16 Space 2-line LCD.

void setup() {
  lcd.init();
  lcd.backlight();                  // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)

  Serial.begin(9600);                        // Starts serial communication at 9600 speed
}

void loop() {
  if (Serial.available()) {    // When the text arrives on the serial communication,
    delay(100);                  // Wait 0.1 seconds for the entire message to arrive.
    lcd.clear();                 // Clear the screen.
    while (Serial.available() > 0) { // while the text is coming in
      lcd.write(Serial.read());         // Write the characters you read on the LCD.
    }
  }
}
```

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.

View Results

Figure 78

Figure 79

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 whle statement, the multiplication ceremony is placed in the execution code..

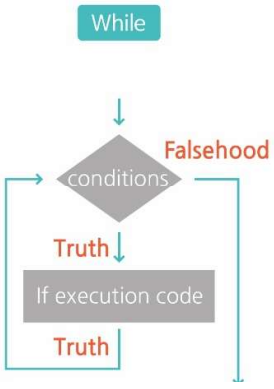


Figure 80

10. Distance measurement with ultrasonic sensor

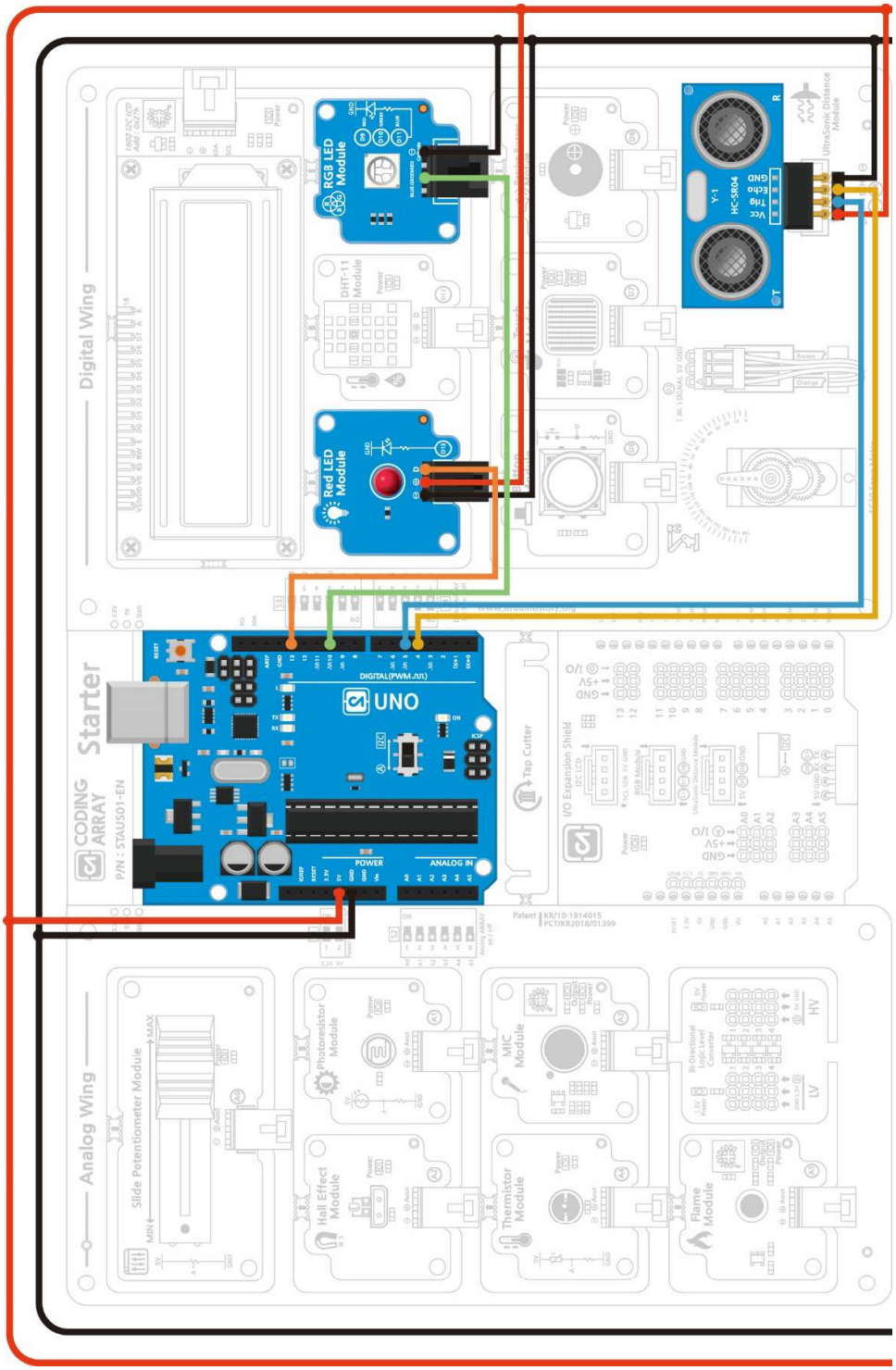


Figure 81

✧ 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..

Ultrasonic Distance Module

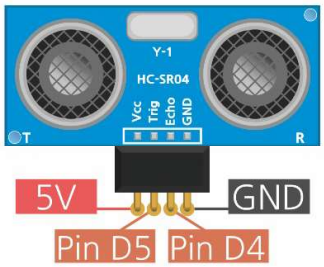


Figure 82

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.

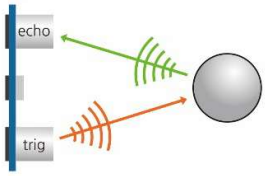


Figure 83

To detect the return of ultrasound, make sure that the reflector and the ultrasonic sensor are at right angles..

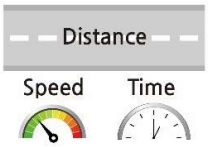


Figure 84

Distance, velocity and time form the following formular

$$\text{velocity} = \frac{\text{Distance}}{\text{time}} \quad , \quad \text{time} = \frac{\text{Distance}}{\text{velocity}} \quad , \quad \text{Distance} = \text{velocity} \times \text{time}$$

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..

$$Round-trip\ distance\ =\ Sound\ speed\ \times\ travel\ time$$

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

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

$$\therefore \ Distance\ to\ object = \frac{Sound\ speed\ \times\ the\ time\ it\ takes\ for\ a\ microwave\ to\ return(\mu s)}{2}$$
$$= \frac{1}{2} \times \frac{340\ m}{1\ s} \times \frac{100\ cm}{1\ m} \times the\ time\ it\ takes\ for\ a\ microwave\ to\ return\ \mu s\ \times \frac{1\ s}{1000000\ \mu s}$$
$$= 0.017 \times the\ time\ it\ takes\ for\ a\ microwave\ to\ return(cm)$$

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:.

$$\text{Sound speed} = 340 + 0.6 \times (\text{current temperature} - 15)$$

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..

Figure 85

CAK Starter Code > 10_Ultrasonic_Distance

```
/*  This sketch uses ultrasonic sensors to measure the distance.
 *  The trigger pin is connected to Arduino's number 5. The trigger pin produces ultrasonic waves.
 *  Echo pins are connected to Arduino's number 4. The echo pin detects reflected ultrasound.
 *  The measured distance should be shown in both the serial and LCD windows.
 *  If the measured distance exceeds the set value, the green LED,
 *  If the measured distance exceeds the set value, turn on the red LED.
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16 Space 2-line LCD

int redLED = 13;                  // Red LED No. 13
int greenPin=10;                  // Green LED No. 13
int threshold = 15;               // Set Distance Threshold Value

void setup() {
  pinMode(5,OUTPUT);      // Trigger pin connection No. 5
  pinMode(4,INPUT);       // Echopin to Pin 4

   pinMode(redLED,OUTPUT);       // Set pin 13 to output
   pinMode(greenPin, OUTPUT);    // Set pin 10 to output

   Serial.begin(9600);           // Starts serial communication at 9600 speeds

   // LCD Setting
   lcd.init();
   lcd.backlight();              // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
   lcd.clear();
}

void loop() {
//  Distance measurement with ultrasonic sensor
   float Duration, Distance;
   digitalWrite(5, HIGH);                          // Fire an ultrasound.
   delayMicroseconds(10);                          // for 10 microseconds
   digitalWrite(5, LOW);                           // Turn off the ultrasound.
   Duration =pulseIn(4, HIGH);                        // Save the time the ehofin is held in HIGH
   Distance =((float)(340 *Duration) /10000) /2;           // convert the distance to cm

// Measured Distance Output
   Serial.print(Distance);                  // Print distance in serial window without changing lines
   Serial.println("cm ");                   // unit output

   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.

View Results

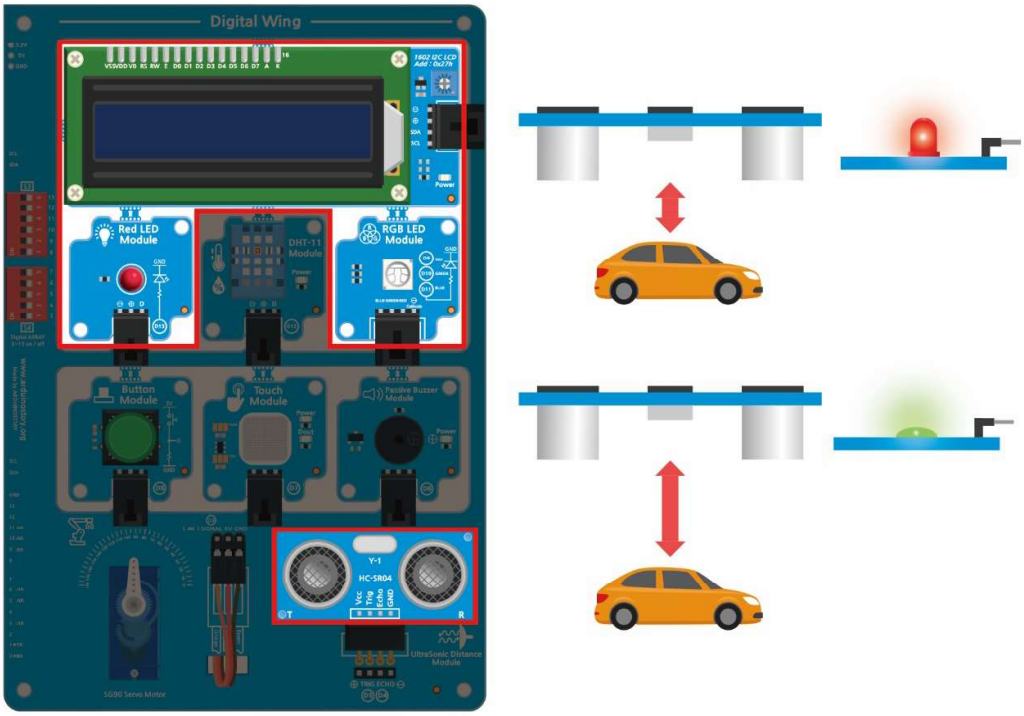


Figure 86

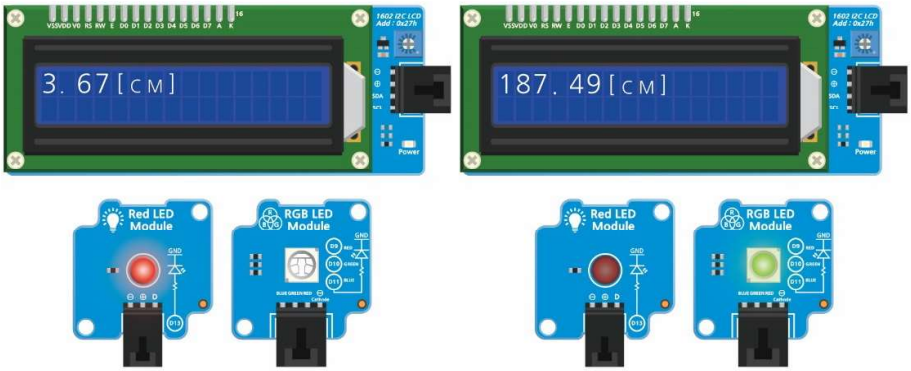


Figure 87

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.

11. Detecting Magnetics with a Hall Sensor

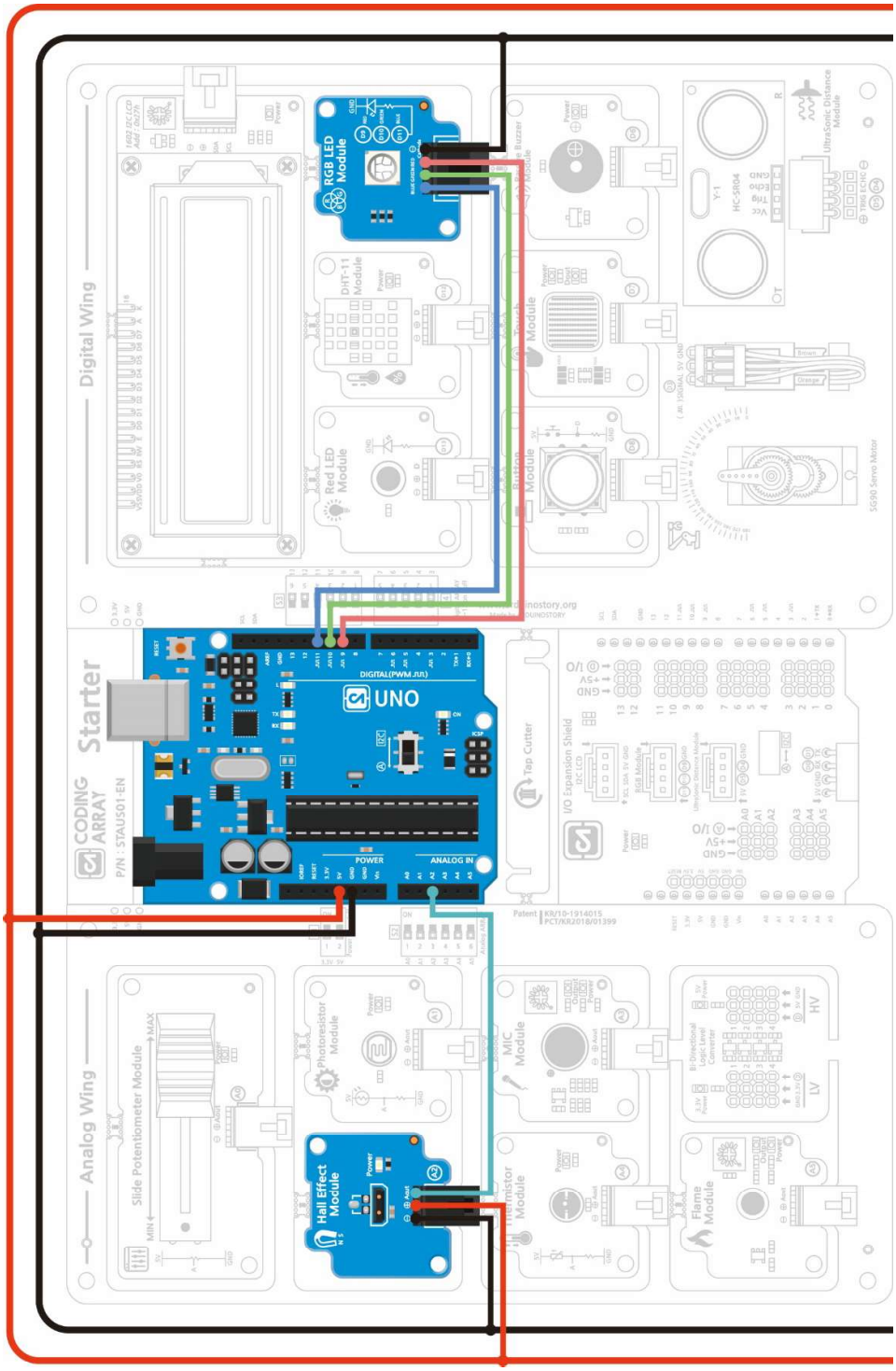


Figure 88

Analog magnetic sensing sensor 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.

Magnetic Sensing Hall Sensor (Hall Effect Module)

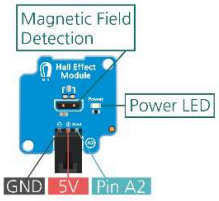
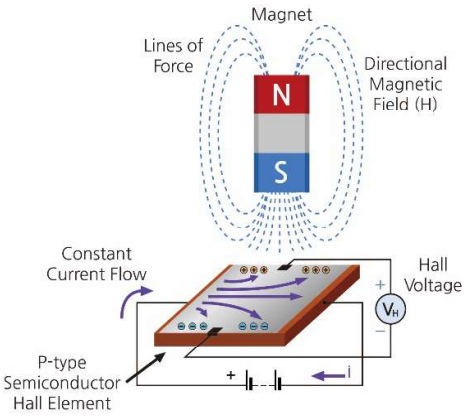


Figure 90

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.

Figure 89

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).



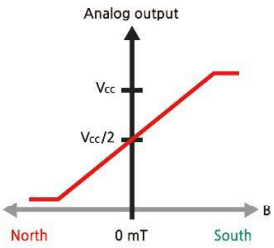


Figure 91

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 lock0 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

```
/*  This sketch uses the Hall Effect module connected to Analog A2.
 *  Read the value of the analogue sensor which varies with the surrounding magnetic field
 *  Measure the strength of the magnetic field by converting it to voltage.
 *  This module has a lower output voltage as the N pole approaches and a higher output voltage as the S pole approaches.
 *  Outputs a comma-separated serial message for easy transfer of result values to Excel.
 *  Copy the message from the serial window and paste it into the memo pad.Save as CSV" extension
 *  You can create a file that can be retrieved from Excel.
*/

// set up for LCD use
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // Set the LCD I2C address. Use 16 Space 2-line LCD

const int hallPin = A2; //connect hall sensor to A2 pin
int sensorReading;                    // Store analog sensor values
int voltage;                          // Store the converted value to voltage

void setup() {
  Serial.begin(9600);       // Set communication speed to 9600
  Serial.println("sensorReading, Voltage (mv)");       // Output message for csv file in serial window

  // LCD initialization
  lcd.init();
  lcd.backlight();                // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
  delay(1000);
}

void loop() {
  sensorReading = analogRead(hallPin);          // Read and save the analog value of the sensor
  voltage = sensorReading * (5.0 / 1024.0) * 1000; // Convert Analog Values from Voltage0 to 5000

  // csv (when you want to receive data in comma-separated text format)
  Serial.print(sensorReading);
  Serial.print(",");
  Serial.println(voltage);

  // Output a message in an LCD window
    lcd.clear();
    lcd.setCursor(0, 0);              // first line first column
    lcd.print("Anlaog_V :");
    lcd.print(sensorReading);
    lcd.setCursor(0, 1);              // 2nd line first column
    lcd.print(voltage);
    lcd.print(" mV");
```

```
46    delay(500);
47  }
```

View Results

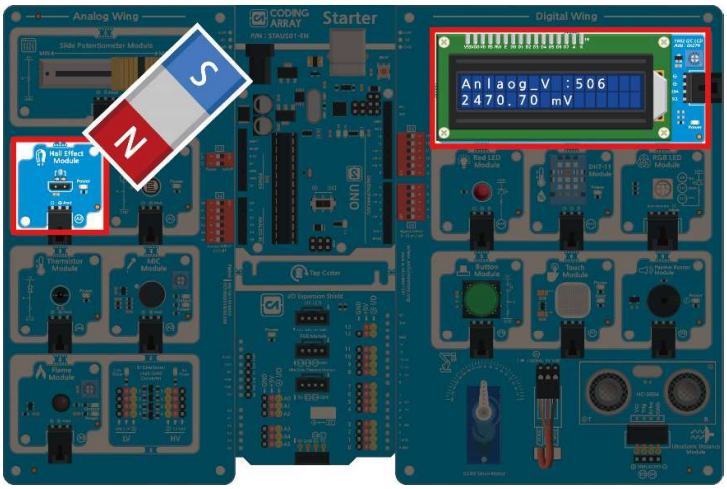
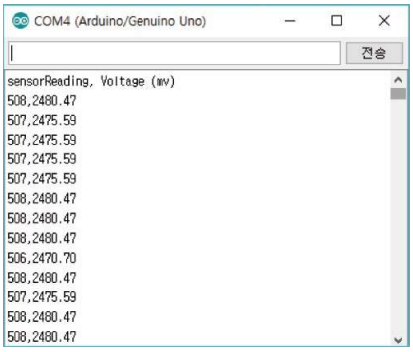
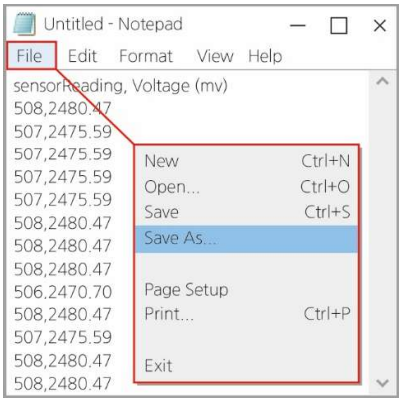


Figure 92



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..

Figure 93



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..

Figure 94

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.

	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

Figure 95

12.Learn how to use light sensing sensors to detect light intensity and calibrate sensor values.

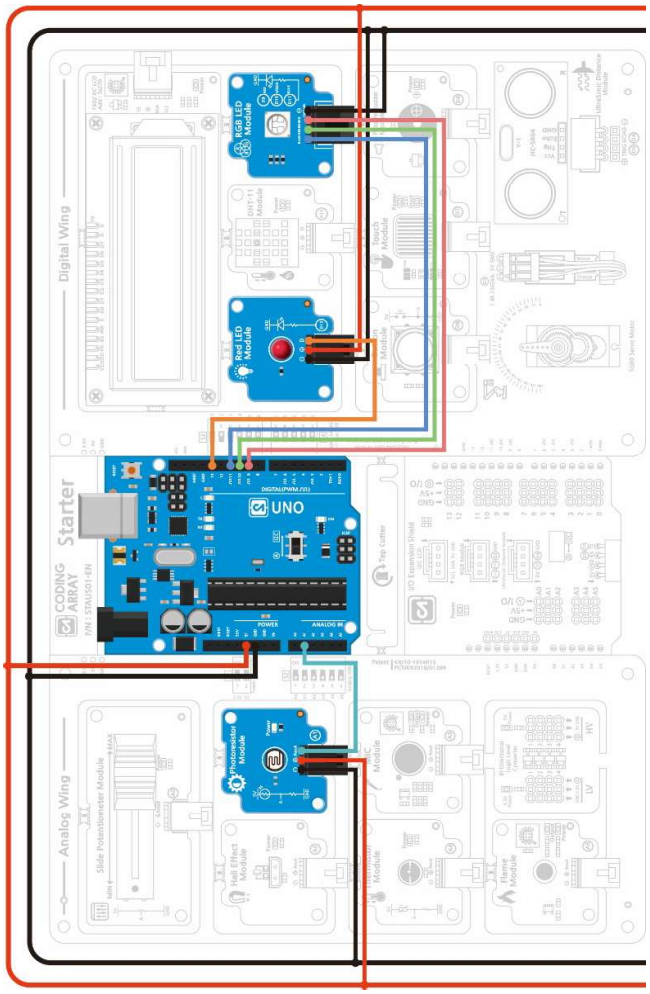
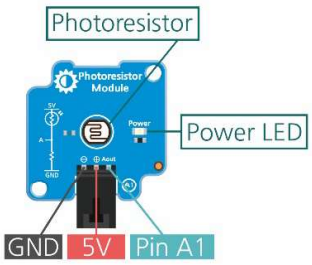


Figure 96

✧ 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.

Photoresistor

Figure 97



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..

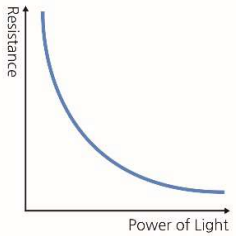


Figure 98

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..

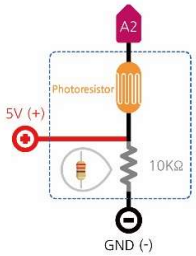


Figure 99

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 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.

CAK Starter Code > 12_01_Photoresistor

```
/ *  This sketch measures the brightness of light using a light sensing sensor.
 *  A1 pin connected to the light sensing sensor enters an analog input value between 0 and 1023, depending on the brightness of the light.
 *  Analog input values are divided into 0 - 3 4 steps through map function and used for switch-case.
 */

// Settings for LCD use
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // Set the LCD I2C address. Use 16 space 2 line LCD.

const int photoresistorPin=A1;  // connect the light sensor to the A1 pin
const int sensorMin =0;     // Minimum sensor value found by experiment, can be modified.
const int sensorMax =700;    // Maximum sensor value found by experiment, can be modified.

void setup() {
  lcd.init();                    // LCD initialization
  lcd.backlight();              // Turn on the backlight. (lcd.noBacklight() turns off the backlight.)
  lcd.setCursor(0,0);             // first line first column
  lcd.print("Range : ");          // Message Output

  Serial.begin(9600);          // Starts serial communication at 9600 speeds
}

void loop() {
// Read sensor values to map ranges
  int sensorReading = analogRead(photoresistorPin);   // Read the light sensor value from the A2 pin
  Serial.println(sensorReading);
  int range = map (sensorReading,sensorMin,sensorMax,0,3);  // Map the sensor values from 0 to 3.

// output messages according to sensor range
  switch(range) {              // according to range 0-3
    case 0:                   // touch the sensor and when the sensor value is zero,
      Serial.println("DARK");          // Darkprint and replace lines in the serial window
      lcd.setCursor(9,0);             // the ninth column of the first line
      lcd.print(range);               // indicate brightness phase on LCD window
      lcd.setCursor(0,1);              // the first column of the second line
      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); x to a specific range.

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.

View Results

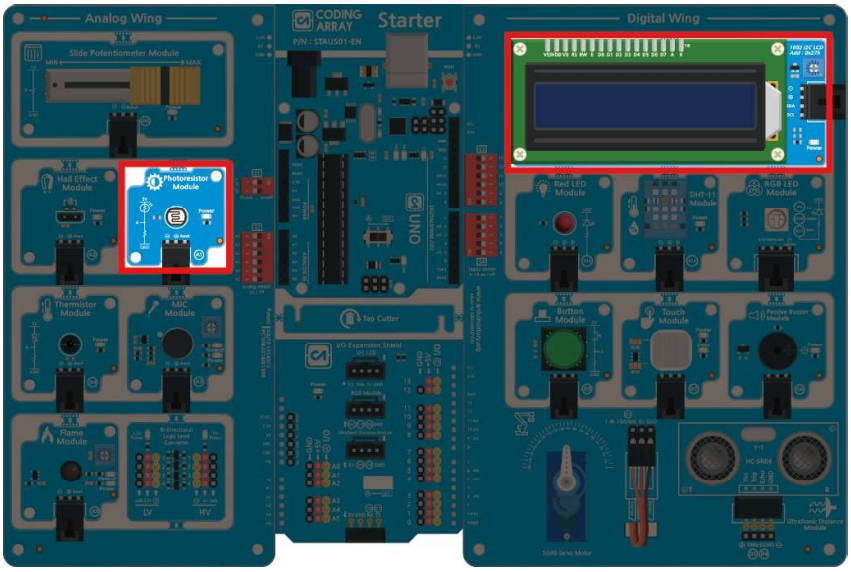


Figure 100

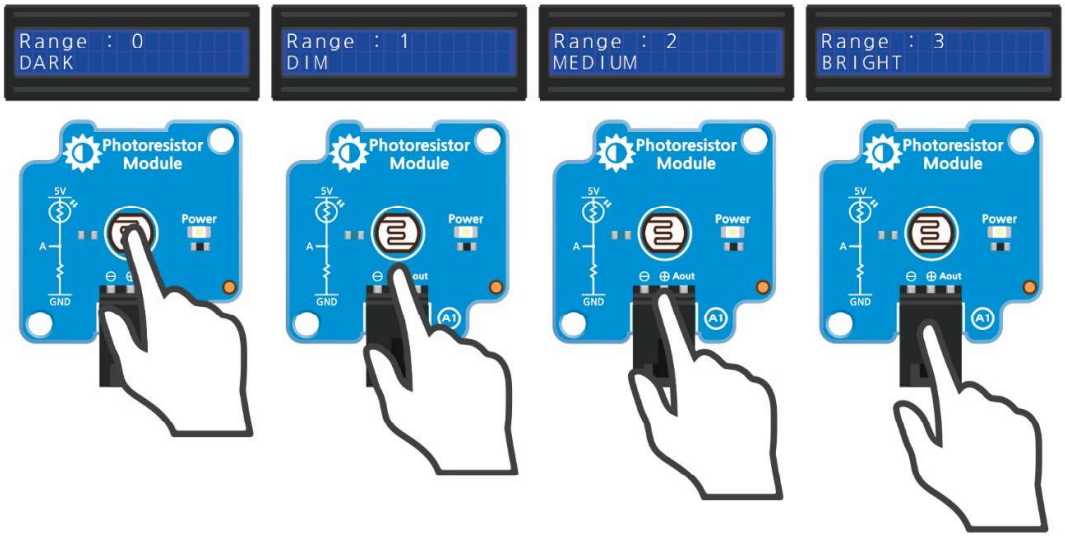


Figure 101

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..

CAK Starter Code > 12_02_Photoresistor_Calibration

```
/ *  This sketch will learn how to calibrate the light sense sensor input values according to the surroundings.
 *  Read the value of the analog A1 pin to which the light sensor is connected for 5 seconds to store the maximum and maximum values.
 *  The maximum value stored is 0 and the maximum value is 255, which is converted through the map function.
 *  After uploading the code, cover the light sensor with your hands and press the reset button.
 *  Allow the maximum and maximum values to be stored while keeping hands away from the light sensor for 5 seconds.
 *  Then, move your hands around the light sensor to check the change in the blue brightness of the RGB LED.
 */

// set up for LCD use
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD I2C address. 16 space 2 lines LCD use
// Variable setting
const int photoresistorPin=A1;  // Connect light sensor to A1 pin
const int redLED=13;            // connect the calibration notification LED to pin 13.
const int bluePin=11;           // connect LEDs for brightness display according to sensor value to number 11.
int sensorValue =0;         // store the light sensor value
int sensorMin =1023;        // set sensor max to 1023.
int sensorMax =0;           // set the sensor maximum value to 0

void setup() {
  // Calibration Notification LED Output Settings
  pinMode(redLED, OUTPUT);
  digitalWrite(redLED, HIGH);

  // LCD setting
  lcd.init();
  lcd.backlight();              // turn on the backlight (lcd.noBacklight() turns off the backlight).
  lcd.clear();
  lcd.setCursor(0,0);               // First line first column
  lcd.print("Calibration");            // output messages
  lcd.setCursor(0,1);               // First line first column
  lcd.print("START");             // output messages

 // Sensor value compensation
  while(millis() < 5000) {                  // for 5 seconds
    sensorValue = analogRead(photoresistorPin);         // save sensor values
    if(sensorValue > sensorMax) {               // sensor value is greater than maximum
      sensorMax = sensorValue;                      // reset to maximum value
    }

    if(sensorValue < sensorMin) {               // sensor value is less than the maximum value
      sensorMin = sensorValue;                      // reset to maximum value
    }
```