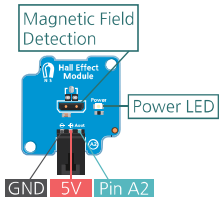
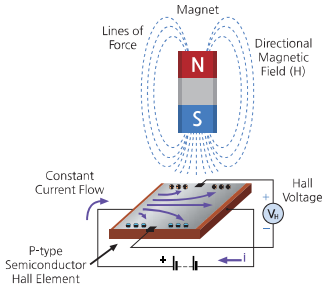
11 CHAPTER 2 Detecting Magnetics with a Hall Sensor

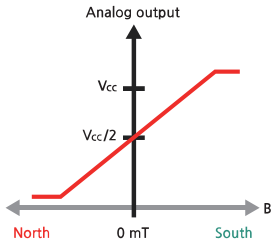
Magnetic Sensing Hall Sensor (Hall Effect Module)



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).





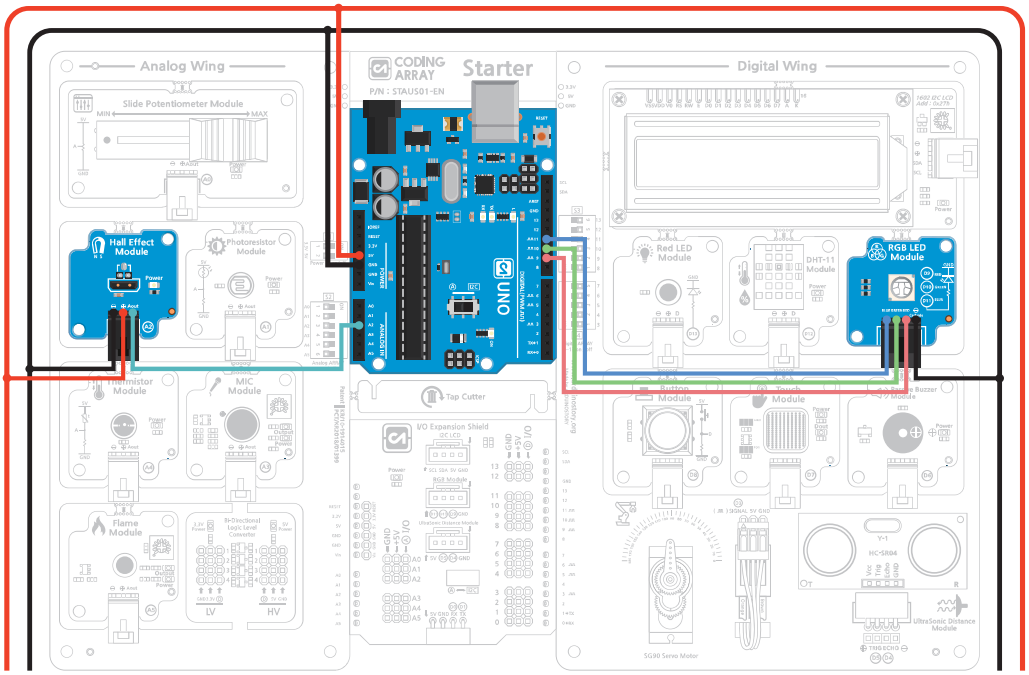
igital hall sensors can only detect whether or not a magnetic field exists, and analog hall sensors can detect the magnetic field poles as well as the strength of the magnetic field with linear hall effects. Analog hall sensor module is used in array kit. As the S-pole approaches the front of the hall 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

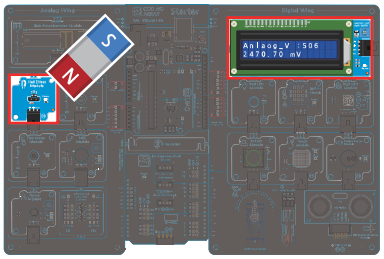
```
/* This sketch uses the Hall Effect module connected to Analog A2.
 * Read the value of the analogue sensor which varies with the surrounding magnetic field
 * Measure the strength of the magnetic field by converting it to voltage.
 * This module has a lower output voltage as the N pole approaches 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");
 delay(500);
}
```

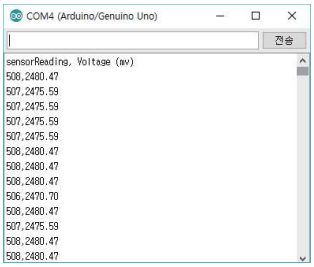
CODING ARRAY CIRCUIT



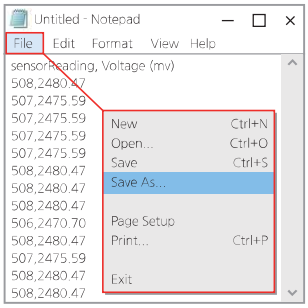
Analog magnetic sensing sensor module is connected to A2 pin. You can check the analog output value that changes as the N pole and S pole of the magnet approach the sensor. It also looks at moving serial output data to Excel.

View Results





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..

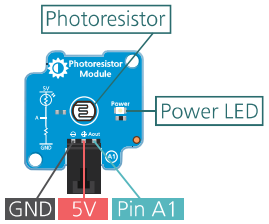
	A	B
1	sensorRea	Voltage (mv)
2	507	2475.59
3	506	2470.7
4	507	2475.59
5	507	2475.59
6	502	2451.17
7	476	2324.22
8	439	2143.55
9	385	1879.88
10	318	1552.73
11	268	1308.59
12	247	1206.05

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.

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

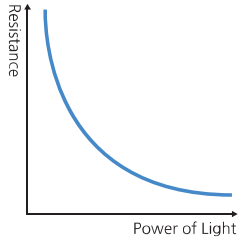
CHAPTER 2

Photoresistor

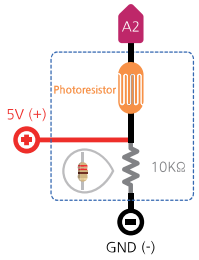


Light sensing resistance sensors are called by a variety of names such as light sensors, light sensing sensors, photoresistors, LDR (Light Dependent Resistors), Cadmium Sulfide (CdS), CdS Cells, CdS Photoresistors, Photometric Cells and Photocorel. As the intensity of light increases, the resistance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

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 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.

CODING ARRAY
STARTER KIT FOR ARDUINO

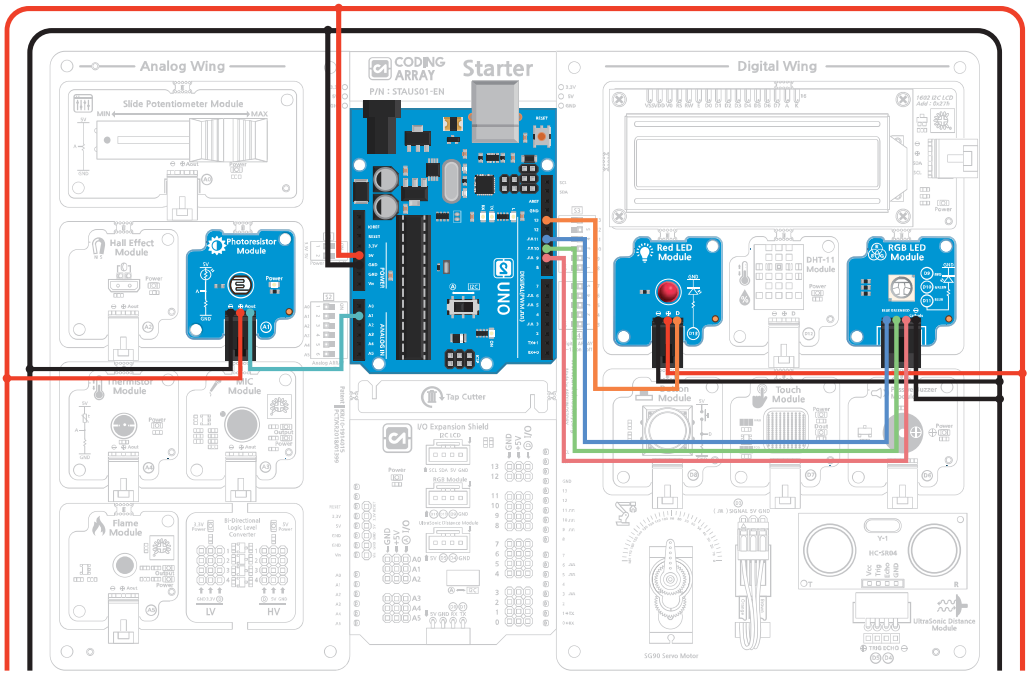
CAK Starter Code > 12_01_Photoresistor

```
/* 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
      break;

    case 1:                    // Put your hands close to the sensor and when the sensor value is 1,
      Serial.println("DIM");           // Dimprint and replace lines in 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("DIM");                // DIM output
      break;

    case 2:                    // Keep your hands away from the sensor and when the sensor value is 2
      Serial.println("MEDIUM");        // print medium on serial window and replace lines
      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("MEDIUM");             // MEDIUM Output
      break;

    case 3:                    // When the sensor value is 3 without touching the sensor nearby,
      Serial.println("BRIGHT");        // print the lightprint on the cereal window and replace the line
      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("BRIGHT");             // BRIGHT Output
      break;
  }
  delay(50);                   // 50 millisecond delay to reliably read values
}
```

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

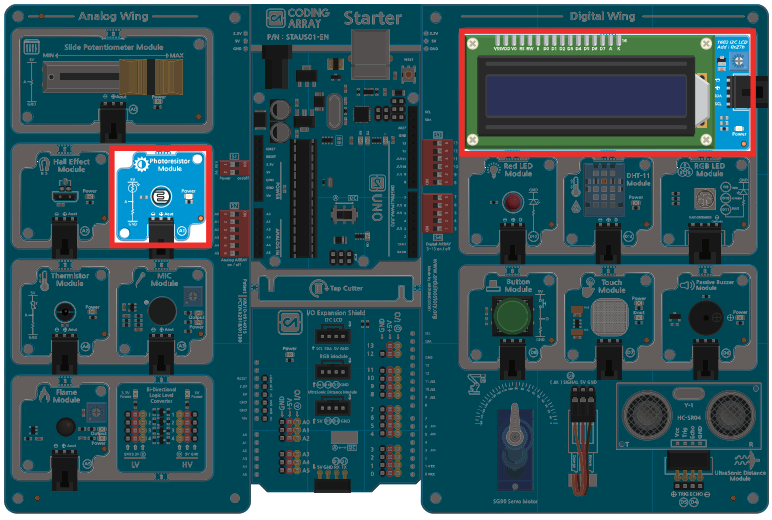
constrain(sensorValue, 0, 255);
Returns 0 value if sensorValue value is less than 0; returns 255 if sensorValue value is greater than or equal to 255.
Restrict the scope of 0 and 255.

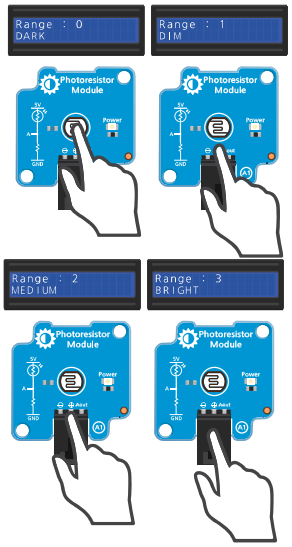
CODING ARRAY CIRCUIT



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



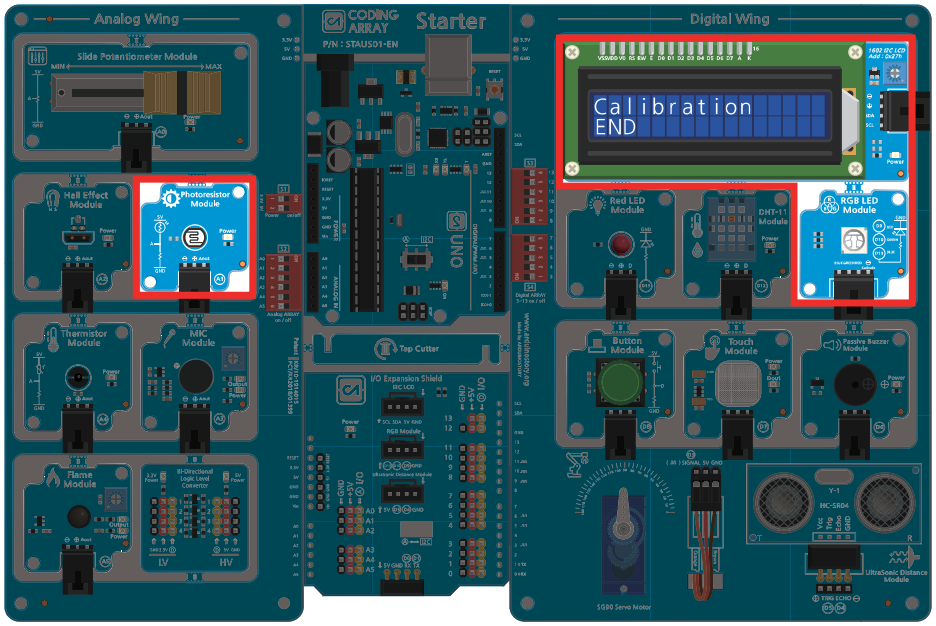


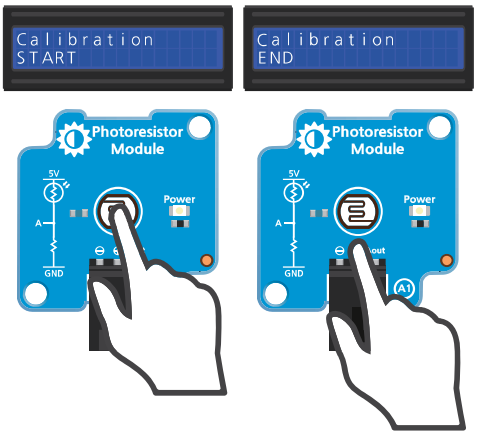
When the program is uploaded, the light sensor detects the light and divides the amount of light into four stages to display the results in an LCD window. Touch the light sensor and observe the results as you keep away from it..

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
    }
  }

// End calibration. Turn off the LED and output a message to the LCD screen
  digitalWrite(redLED,LOW);
  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("END");                 // output message
}

void loop() {
  sensorValue = analogRead (photoresistorPin);   // read an analogue sensor value and store it in a variable
  // read sensor value and convert to 0-255
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);

  // limit if sensor value is outside calibration range
  sensorValue = constrain(sensorValue, 0, 255);

  // adjust blue LED brightness with sensor value
  analogWrite(bluePin,sensorValue);
}
```

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..

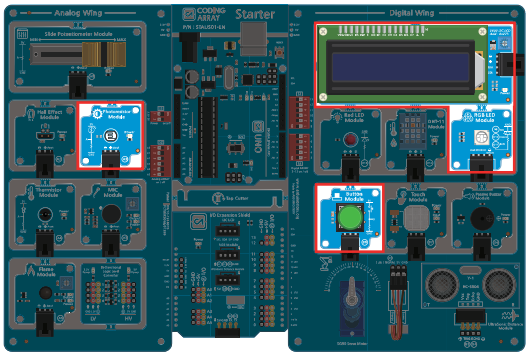
When calibration is finished, the red LED is turned off and the light sensor detects the amount of light from the analog input value between 0 and 1,023. Analog output should be expressed as a value between 0 and 255, so use the map function to convert 0 to 1023 to 0 to 255 to display results with blue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

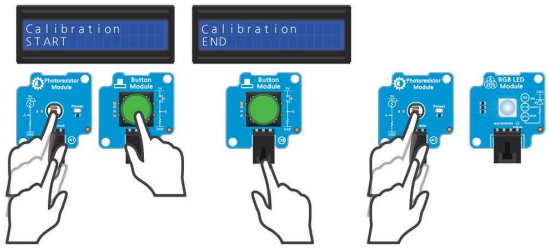
CAK Starter Code > 12_03_Photoresistor_CalibrationFunction

```
/ * This sketch runs while the button switch connected to the digital pin 8 is pressed.
 * Call up the calibration () function to find the maximum and maximum values of the analog A1 pin to which the light sensor is connected.
 * Return to the main loop if the button is not pressed.
 * This method can reset the maximum and maximum values of the light sensor when ambient brightness conditions change..
 */

// Set up for LCD use
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // set LCD I2C address. 16kans2joules LCD use

const int photoresistorPin = A1; // Connect the light sensor to the A1 pin
const int redLED = 13;        // connect the calibration notification LED to pin 13.
const int bluePin = 11;       // connect an LED for brightness display according to sensor value to No. 11.
const int Button = 8;       // Connect button switch to pin 2

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() {
  pinMode(redLED, OUTPUT);           // Set Calibration Notification LED Output
  pinMode(bluePin, OUTPUT);          // set LED output to indicate brightness
  pinMode(Button, INPUT);      // Enter buttons connected to pull-up resistance

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

void loop() {
  while (digitalRead(Button) == HIGH) {    // when button switch is pressed

    calibrate();                           // calibration function.

    digitalWrite(bluePin, LOW);            // Turn off the blue LED during calibration.
    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
  }

  digitalWrite(redLED, LOW);       // Turn off the red LED after calibration.
  lcd.setCursor(0, 0);             // First line first column
  lcd.print("Calibration");           // output messages
  lcd.setCursor(0, 1);             // First line first column
  lcd.print("END  ");                 // output message

  sensorValue = analogRead(photoresistorPin);          // store the light sensor value
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255); // calibrate sensor values to 0-255.
  sensorValue = constrain(sensorValue, 0, 255);        // limit if sensor value is outside calibration range
  analogWrite(bluePin, sensorValue);  // adjust the LED brightness with the calibrated value.
}

// calibrate() function setting: Reset the maximum and maximum values of the sensor according to the ambient brightness.
void calibrate() {
  digitalWrite(redLED, HIGH);          // Turn on the red LED for calibration notifications.
  sensorValue = analogRead(photoresistorPin);          // Read and save the value of the light sensor

  if (sensorValue > sensorMax) {           // the illumination sensor is greater than 1023.
    sensorMax = sensorValue;               // read the sensor value and save it to sensorMax
  }

  if (sensorValue < sensorMin) {           // the light sensor is less than zero
    sensorMin = sensorValue;               // read the sensor value and save it to sensorMin
  }
}
```

CODING ARRAY CIRCUIT

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

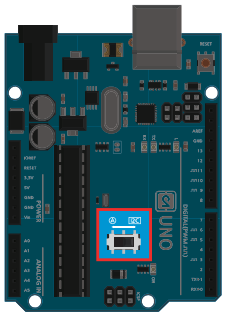
(parameter) : Put the factors to be used in the function. If there are multiple parameters, the order must also be observed. If the parameter is not required, leave ( ) blank.

{ 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..

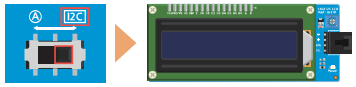
13 CHAPTER 2 Detect flame with flame detection sensor

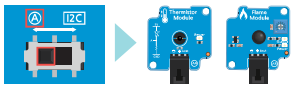


I2C (Inter-Integrated Circuit) is an NFC that can connect a 1 master (Arduino) : multiple slave (sensor modules) in one direction using a SCL (Serial Clock) pin and SDA (Serial Data) pin with full-up resistance connected. Arduino can use SDA, SCL pins as I2C communication pins or analog A4, A5 pins as 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

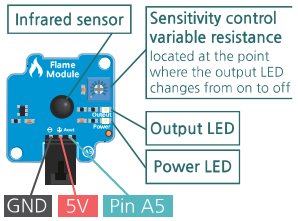
! CAUTION!

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..

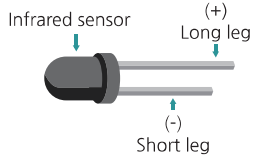




Flame Sensor



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.