```
45    }
46
47  // End calibration. Turn off the LED and output a message to the LCD screen
48    digitalWrite(redLED,LOW);
49    lcd.clear();
50    lcd.setCursor(0,0);               // First line first column
51    lcd.print("Calibration");           // output messages
52    lcd.setCursor(0,1);               // First line first column
53    lcd.print("END");                  // output message
54  }
55
56  void loop() {
57    sensorValue = analogRead (photoresistorPin);   // read an analogue sensor value and store it in a variable
58    // read sensor value and convert to 0-255
59    sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
60
61    // limit if sensor value is outside calibration range
62    sensorValue = constrain(sensorValue, 0, 255);
63
64    // adjust blue LED brightness with sensor value
65    analogWrite(bluePin,sensorValue);
66  }
```

View Result

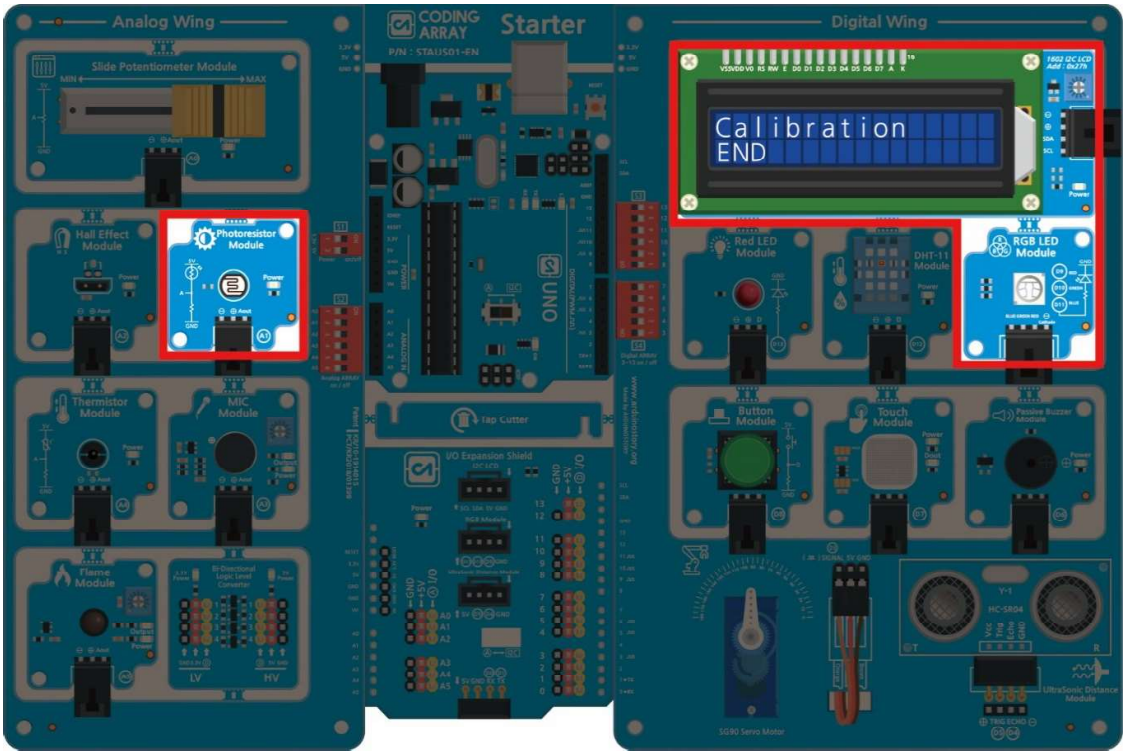


Figure 102

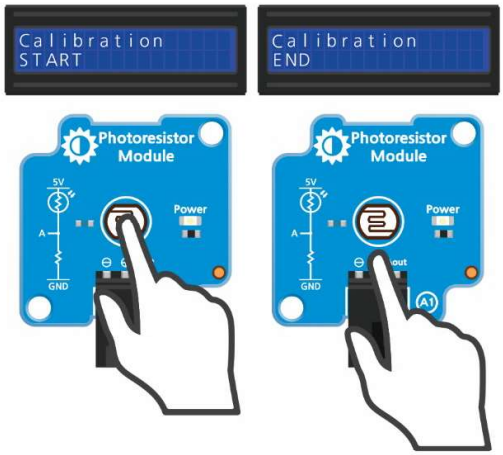


Figure 103

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
```

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

Veiw Results

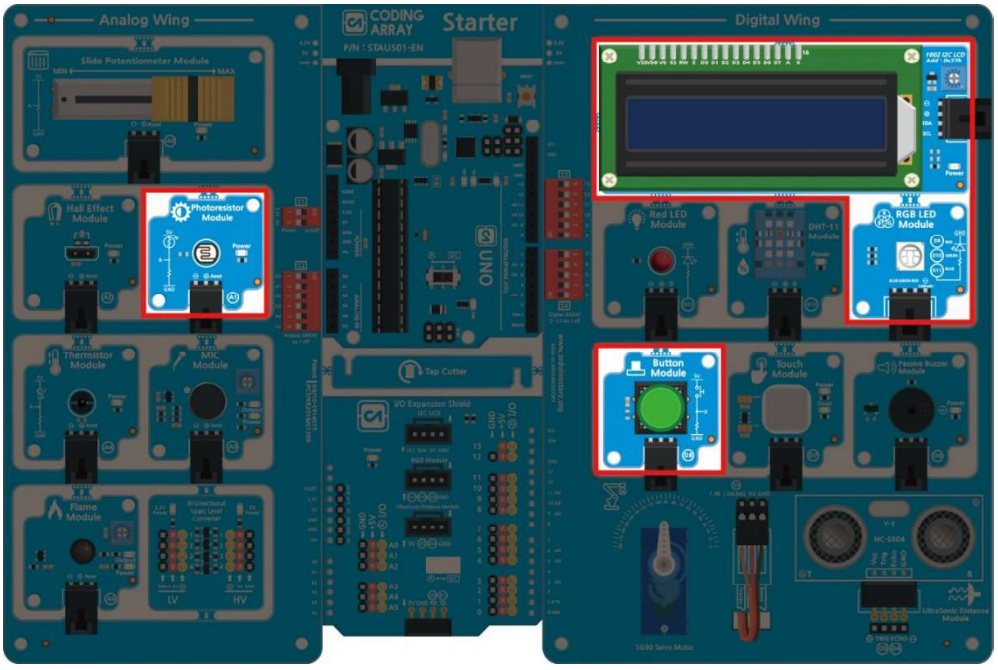


Figure 104

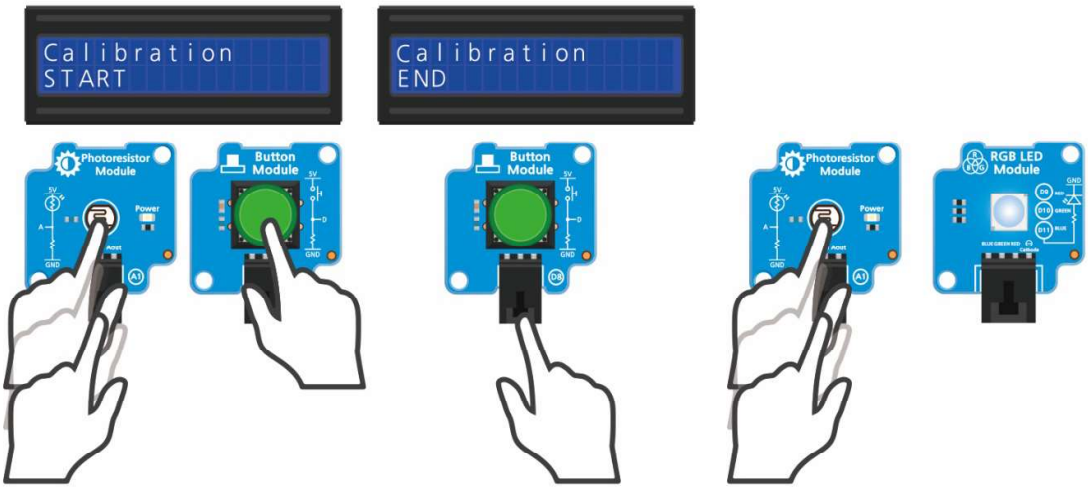


Figure 119

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); ⋮ }

Table 14

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. Detect flame with flame detection sensor

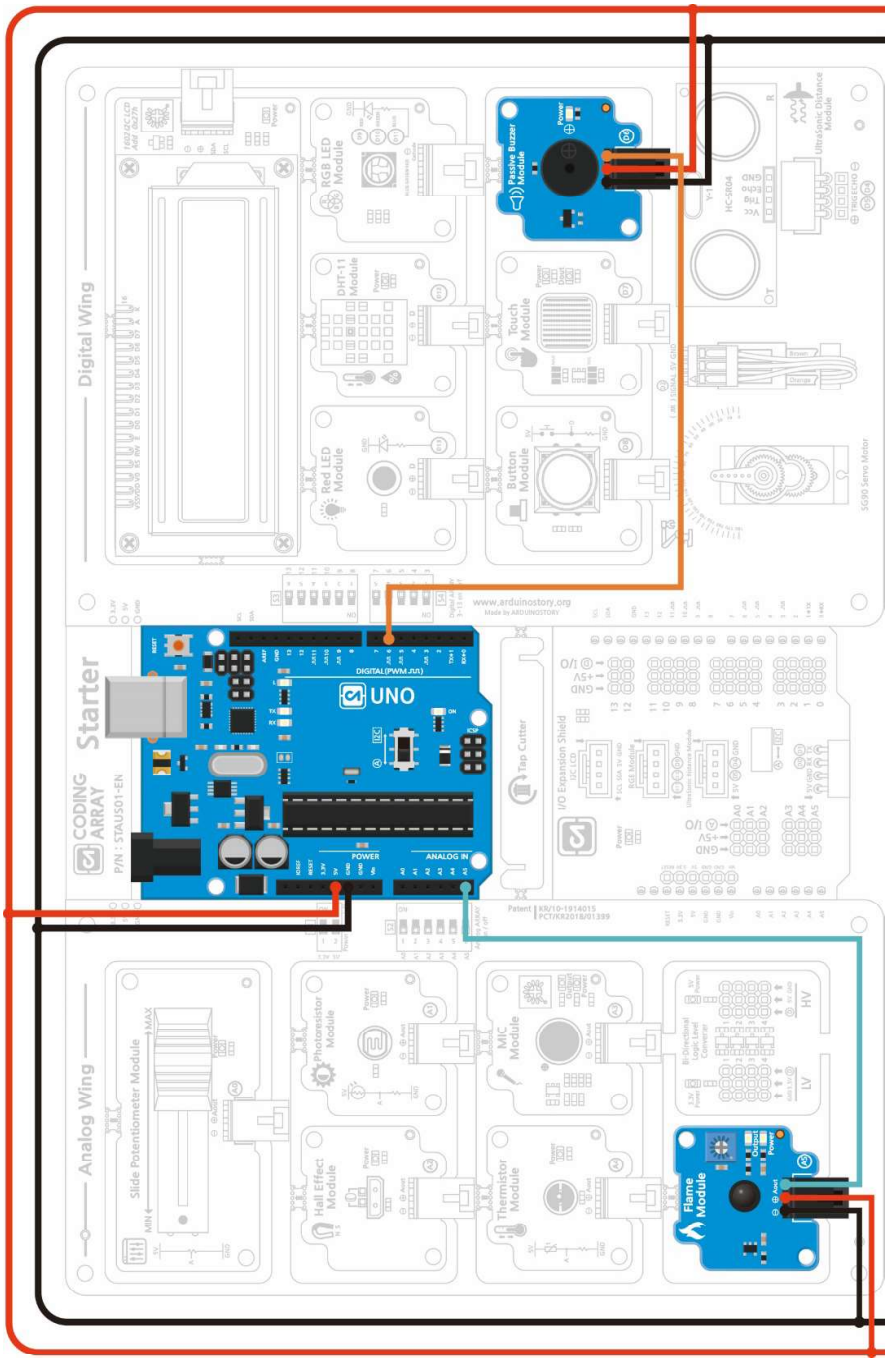


Figure 120

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

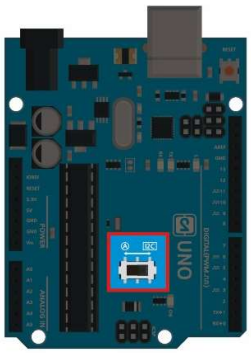


Figure 121

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

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

Figure 1052

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.

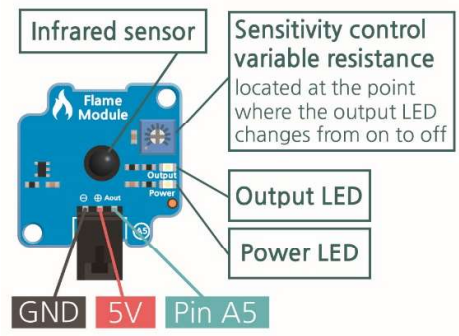


Figure 106

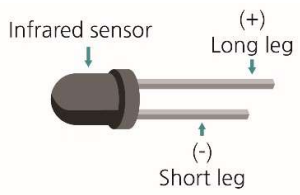


Figure 124

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.

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.

CAUTION) 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.

```
/*  This sketch uses the flame sensor module connected to analogue A5.
*  Detects flame strength around (prepare lighter and bring flame near module)
*  If the threshold value set is exceeded, an audible alarm will be
*/

#define PI 3.141592       // Set circumference PI value

int flameSensor =A5;              // Connect flame detection sensor to analogue pin A5
int Buzzer =6;                    // connect the piezo buzzer to pin 6.
int sensorReading =0;             // Variables for storing sensor output values

void setup() {
  pinMode(Buzzer, OUTPUT);          // Output settings for piezo buzzer pins
  pinMode(flameSensor, INPUT);      // set flame sensor pin to input
  Serial.begin(9600);               // initiate serial communication at 9600 speed
}

void loop() {
  sensorReading = analogRead(flameSensor);  // save an analogue value of the flame detection sensor
  Serial.println(sensorReading);            // Print the value of the flame detection sensor to the serial window
  if(sensorReading <=1000) {                // flame detection sensor value greater than 1000
    Serial.println("Fire !!");                    // Fire! Outputs on screen
    playTone();                             // Alarm negative
  } else {                                  // If flame detection sensor value is less
    noTone(Buzzer);                         // off Peugeot Booger
  }
  delay(500);                               // 0.5 second interval
}

// set alarm negative function
void playTone() {
  float sinVal;         // save the sine wave value
  int toneVal;          // store value for alarm sound generation

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

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

sin(rad) Calculate the Sin value of the radian angle. -1≤ sin(rad)≤ 1 range.

Rad :Radian angle, actual type

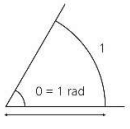


Figure 125

Loudness Method: To display angles using arc length $1 \text{ radian} = \frac{180^\circ}{\pi}$ $1^\circ = \frac{\pi}{180}\ radian$

Veiw Results

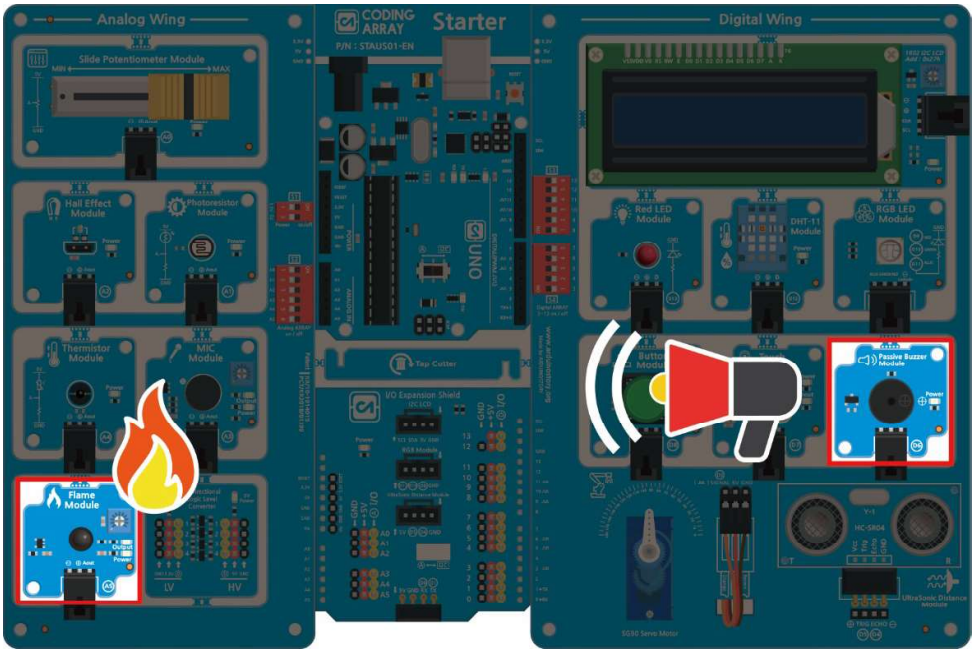


Figure 126

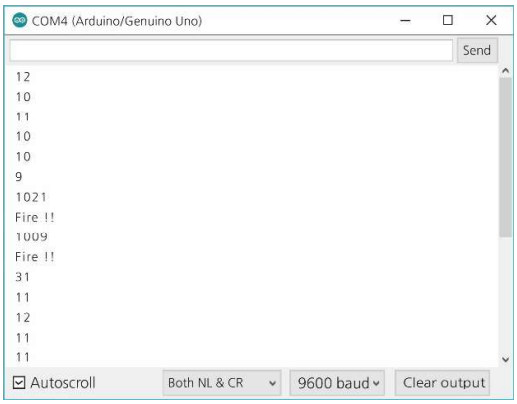


Figure 127

After uploading the sketch, bring the lighter flame near the flame sensor (if the lighter is not available, replace it with an infrared remote control at home). If a flame is detected within about 10cm, it indicates an analog value of less than 1000 and the manual buzzer produces a sin-wave alert.

After module removal, digital pins can be connected to D and used as digital sensors. If the flame is not detected, it will return "0" or "1" if the flame is detected. The sensitivity of the sensor is adjusted by rotating the variable resistance on the board.

14. Temperature measurement with NTC thermistor

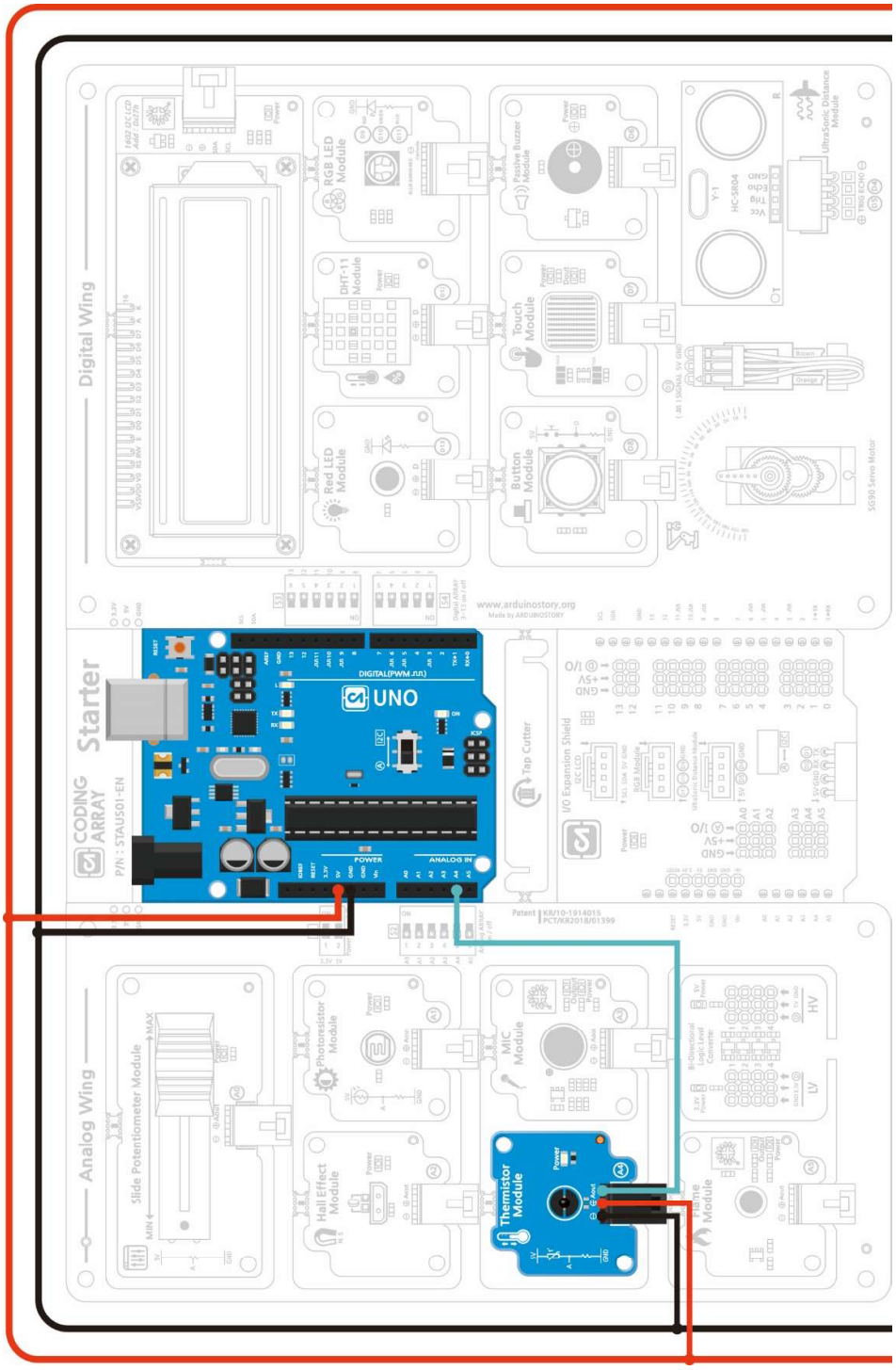
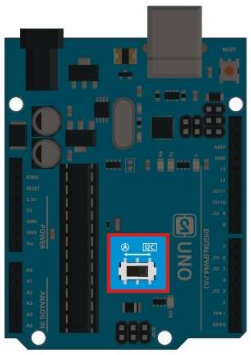


Figure 128

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



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.

Figure 129

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 130

NTC Thermistor (Negative Temperature Coefficient Thermistor)

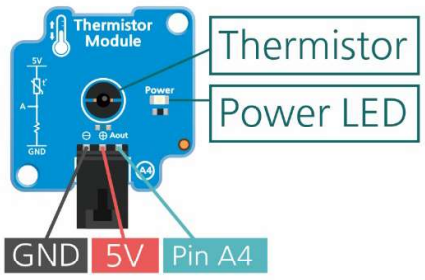
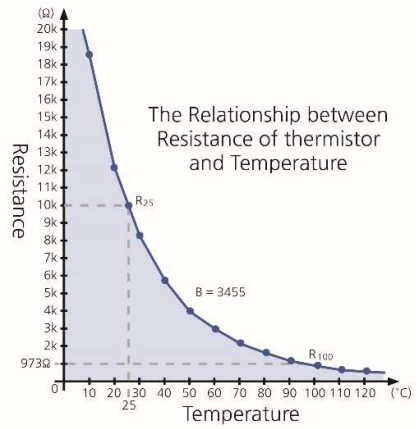


Figure 131

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



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

Step1) On the analog input A4 pin to which the thermistor is connected, measure the voltage using the voltage distribution method.

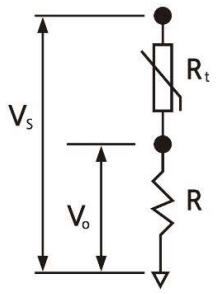
Step2) Convert voltage to resistance.

Step3) Convert resistance to temperature.

Figure 1072

Step1) Voltage distribution

Thermistors indicate temperature due to changes in electrical resistance. However, the analog input pins of the Arduino board measure voltages other than resistance. Therefore, the resistance of the thermistor should be converted to voltage using the voltage distribution method. Series connection between Ohm's law and resistance (current flowing to each resistor is the same. The following expressions can be derived using the addition of two resistors). Use known resistance R=10KΩ, Rt= thermistor to use voltage divider circuits. V0 is measured at A4..



Vs: Full Voltage

Vo: Voltage to 10KΩ resistance

I: Total Current

R: 10KΩ resistance

Rt: thermistor resistance

Figure 133

$$V_s = I(R+R_t) \qquad I = \frac{V_s}{R+R_t} \qquad V_0 = IR$$

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

Step2) Convert voltage to resistance.

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

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

Step3) Convert resistance to temperature.

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

The first <Shenhart-Hart>

$$\frac{1}{T} = A + B\ln(R) + C\ln(R)^3$$

T: Kelvin temperature (absolute temperature)

R: Resistance value at temperature T

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

Second <B or β parameter >

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

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

CAK Starter Code > 14_01_Thermistor_SH

```
/*    This sketch converts the voltage distributed to the thermistor connected to the A4 into a resistor.
    One of the ways to change the resistance value to temperature
    Use the Steinhart-Hart formula to calculate the temperature.
    The A4 pin and I2C LCD module cannot be used together and must be selected as a jumper.
*/

#include <math.h>

const int thermistorPin = A4;                          // connect thermistor to A4 pin

// parameters can vary the value depending on the module.
double ParamA = 0.001129148;
double ParamB = 0.000234125;
double ParamC = 0.0000000876741;

void setup() {
  Serial.begin(9600);                                  // initiate serial communication at 9600 speeds
}
void loop() {
  int readVal = analogRead(thermistorPin);
  double temp = Thermistor(readVal);                   // recall temperature measurement function
  double tempC = temp - 273.15;                        // Convert Absolute Temperature to Celsius
  double tempF = (tempC * 9.0) / 5.0 + 32.0;           // Convert temperature to Fahrenheit

  // Output Serial Monitor
  //Serial.println(readVal);
  Serial.print(tempC);                                 // display temperature
  Serial.println(" C");
  delay (500);
}
// set Steinhart-Hart temperature measurement function
double Thermistor(int RawADC) {
  double Temp;
  Temp = log (10000.0 * ((1024.0 / RawADC - 1)));
  Temp = 1 / (ParamA + ( ParamB + (ParamC * Temp * Temp )) * Temp );
  return Temp;
  }
  return Temp;
}

```

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

View Results

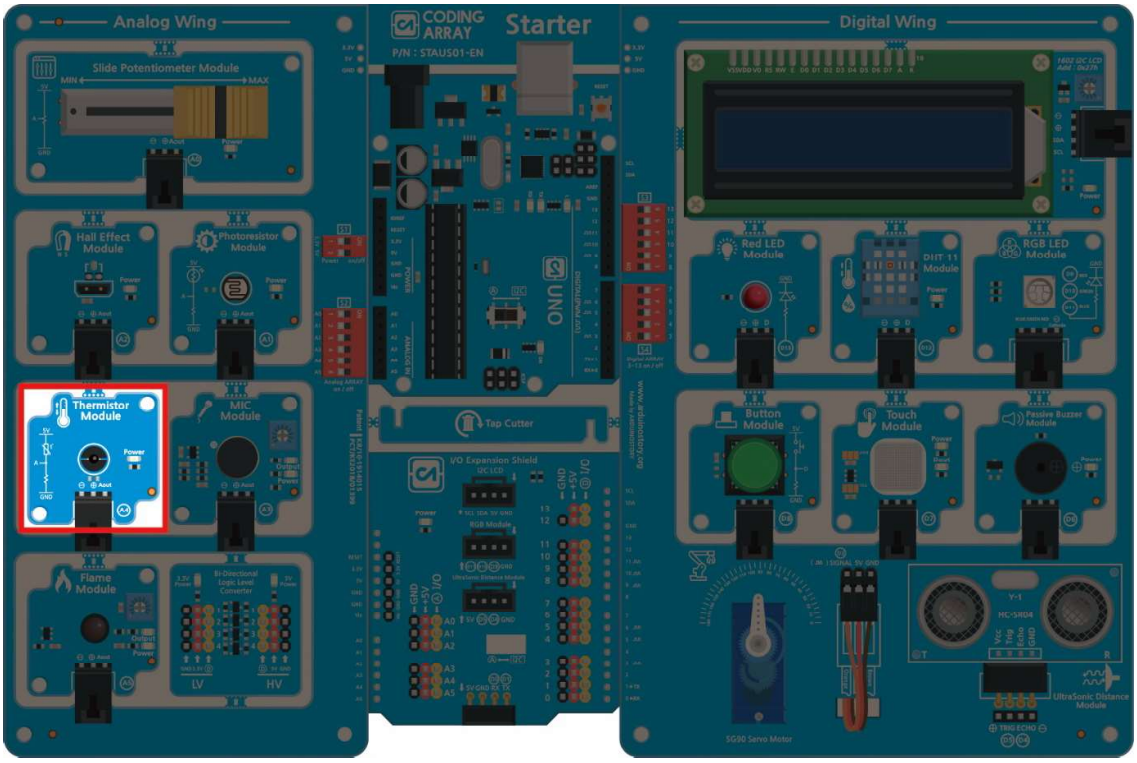


Figure 134

Figure 1085

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

CAK Starter Code > 14_02_Thermistor_BParameter

```
/*  This sketch converts the voltage distributed to the thermistor connected to the A4 into a resistor.
 *  One of the ways to change the resistance value to temperature
 *  Use the B parameter expression to calculate the temperature.
 */

#include <math.h>

const int thermistorPin=A4;      // connect thermistor to A4 pin

//B parameter
float ParmB=3435.0;

void setup() {
  Serial.begin(9600);              // initiate serial communication at 9600 speeds
}

void loop() {

  float readVal =analogRead(thermistorPin);

  // calculate temperature
  float resistor =(1023.0*10000)/readVal-10000;
  float tempC =(ParmB/(log(resistor/10000)+(ParmB/(273.15+25)))) -273.15;

// Output Serial Monitor
 // Serial.println(readVal);       // output analog values read from Serial.println(readVal); A4
  Serial.print(tempC);             // temperature output
  Serial.println(" C");            // output units

  delay(1000);                     //1 second delay
}
```

View Results

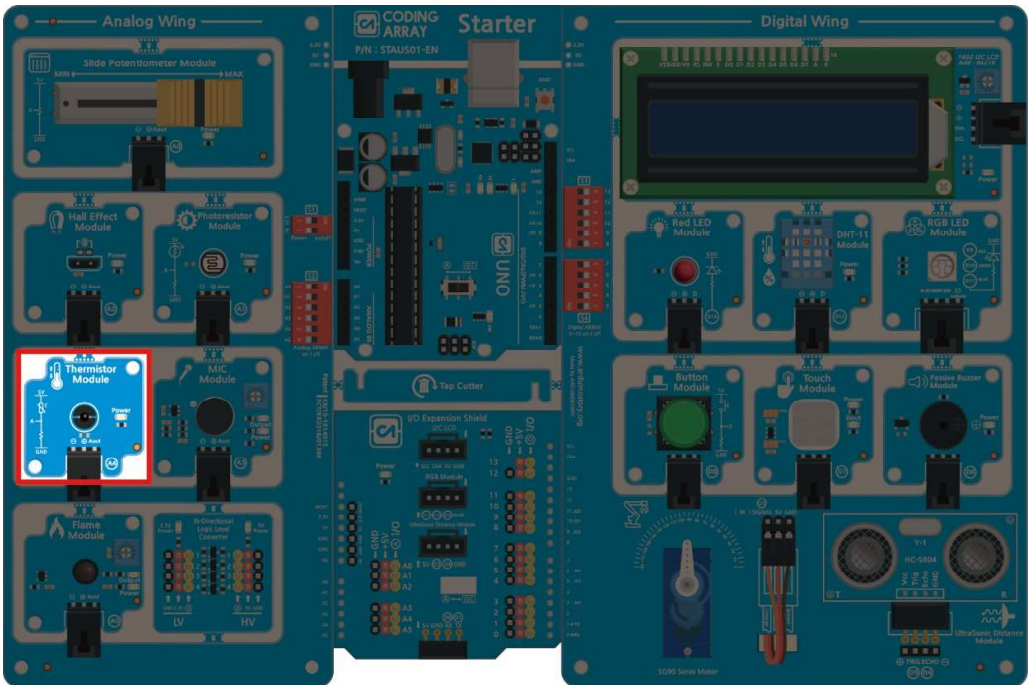


Figure 1096

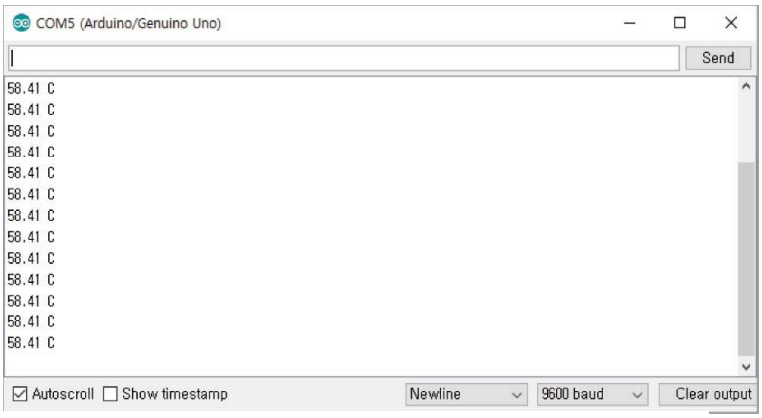


Figure 137

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

15. Detect sound with microphone sensor

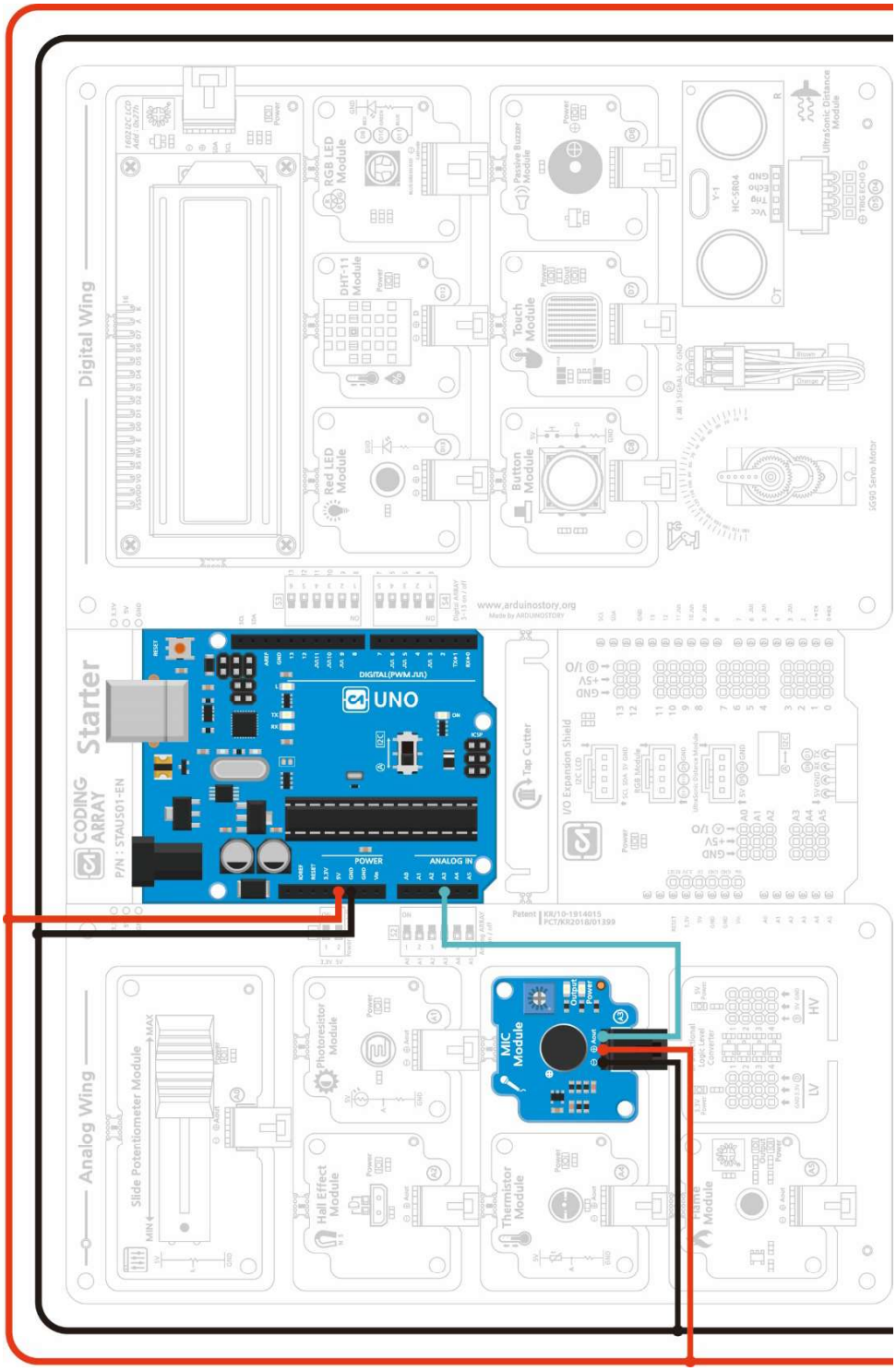


Figure 138

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

MIC Sensor

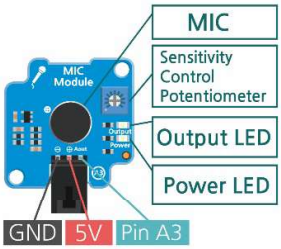


Figure 1109

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

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

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

Figure 140

CAK Starter Code > 15_MIC_Clap_ONOFF

```
/*    This sketch turns on the LED when you clap twice.
*     Shows the LED illuminates when hit twice again.
*/

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

LiquidCrystal_I2C lcd(0x27, 16, 2); // set LCD I2C address. 16kans2joules LCD use

const int sampleWindow = 125;   // sample period milliseconds (125 mS = 8 Hz)
int ledPin = 13;                // Red LED connection to pin 13

int soundValue = 0;          // store the value of the sound sensor
int clapCounter = 0;         // Save clap count
double threshold = 2.0;             // set clapping sound threshold voltage value

void setup() {

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

  lcd.init();                       // Initialize LCD
  lcd.backlight();                  // turn on the backlight (lcd.noBacklight() turns off the backlight).
  lcd.clear();                      // clear LCD screen
  lcd.setCursor(0, 0);              // First line first column
  lcd.print("LED OFF");                 // message output
  lcd.setCursor(0, 1);              // Line 1st column
  lcd.print("CLAP TWICE~");                // output messages

  pinMode(ledPin, OUTPUT);       // set red LED to output

}



void loop() {

  unsigned long start= millis();       // start sampling
  unsigned int peakToPeak = 0;      // Amplitude Value Variables
  unsigned int signalMax = 0;
  unsigned int signalMin = 1024;

  // collect data for 125 milliseconds.
  while (millis() - start < sampleWindow)
  {
    soundValue = analogRead(3);    // specify analogue pin number 3.
    if (soundValue < 1024)             // Read data up to the ADC maximum (1024=10bit).
    {
      if (soundValue > signalMax)
```

```
47          {
48            signalMax = soundValue;   // Store the maximum sound value in the signalMax variable.
49          }
50          else if (soundValue < signalMin)
51          {
52            signalMin = soundValue;   // store the sound minimum in the variable (signalMin).
53          }
54        }
55    }
56    peakToPeak = signalMax - signalMin;   // calculate peak-to-peak amplitude
57    double volts = (peakToPeak * 5) / 1024; // convert the ADC value to a voltage value. Reference Voltage 5V
58
59    Serial.println(volts);
60    if (volts >=threshold)
61    {
62      clapCounter ++;
63      Serial.println(soundValue);         // Output soundValue value to serial monitor
64      Serial.println(clapCounter);        // Output number of applause to serial monitor
65      delay(50);
66      // turn on the LED in two claps
67      if(clapCounter == 2)
68      {
69        digitalWrite(ledPin, HIGH);                 // turn on the red LED.
70        Serial.println("LED ON" );                  // Output a message to the serial monitor
71        // LCD output
72        lcd.clear();
73        lcd.setCursor(0, 0);                        // First line first column
74        lcd.print("LED ON");                         // message output
75        lcd.setCursor(0, 1);                        // Line 1st column
76        lcd.print("CLAP TWICE   ");                  // output message
77      }
78      // turn off the LED in four claps
79      if (clapCounter == 4)
80      {
81        digitalWrite(ledPin, LOW);                    // Turn off the red LED.
82        Serial.println("LED OFF");                 // output messages to serial monitors
83        // LCD output
84        lcd.clear();
85        lcd.setCursor(0, 0);                       // First line first column
86        lcd.print("LED OFF");                       // message output
87        lcd.setCursor(0, 1);                       // Line 1st column
88        lcd.print("CLAP TWICE   ");                  // output message
89        clapCounter = clapCounter % 2;              // Save remaining 0 (initialize clap count)
90      }
91    }
92  }
```

View Results

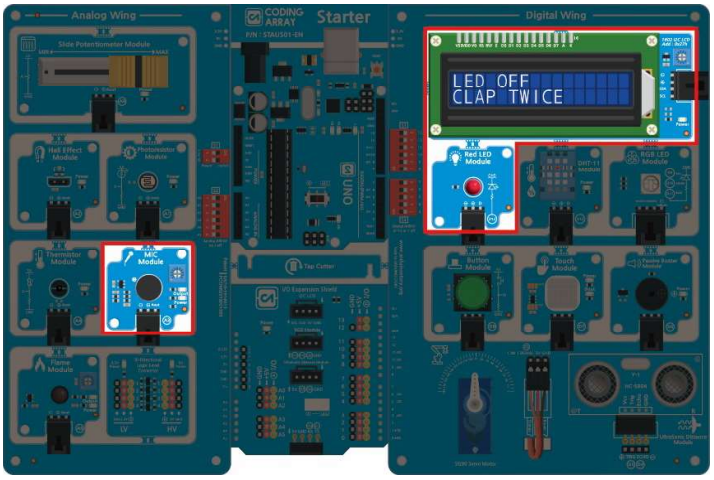


Figure 141

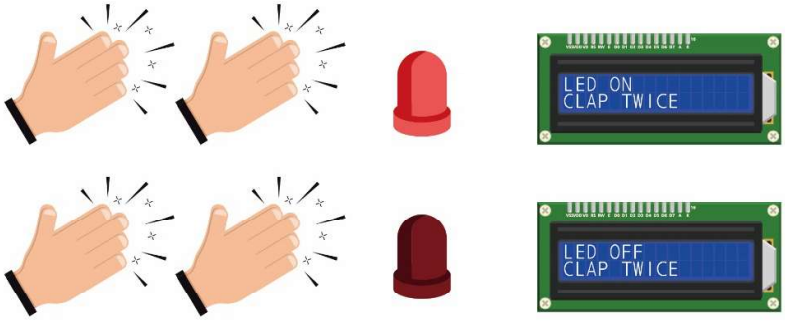


Figure 1112

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

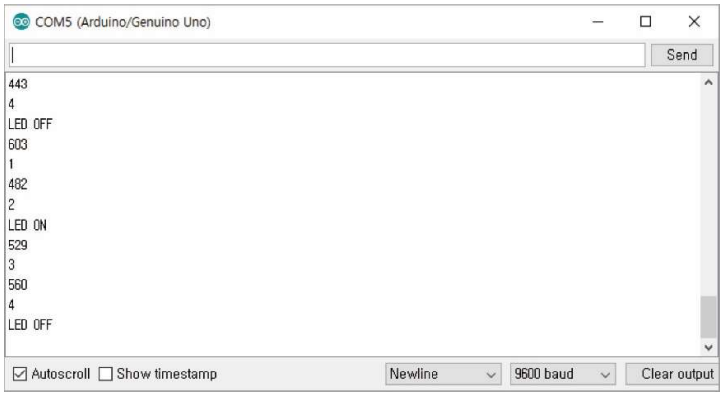


Figure 1123

16. Measure the temperature and humidity with the sensor

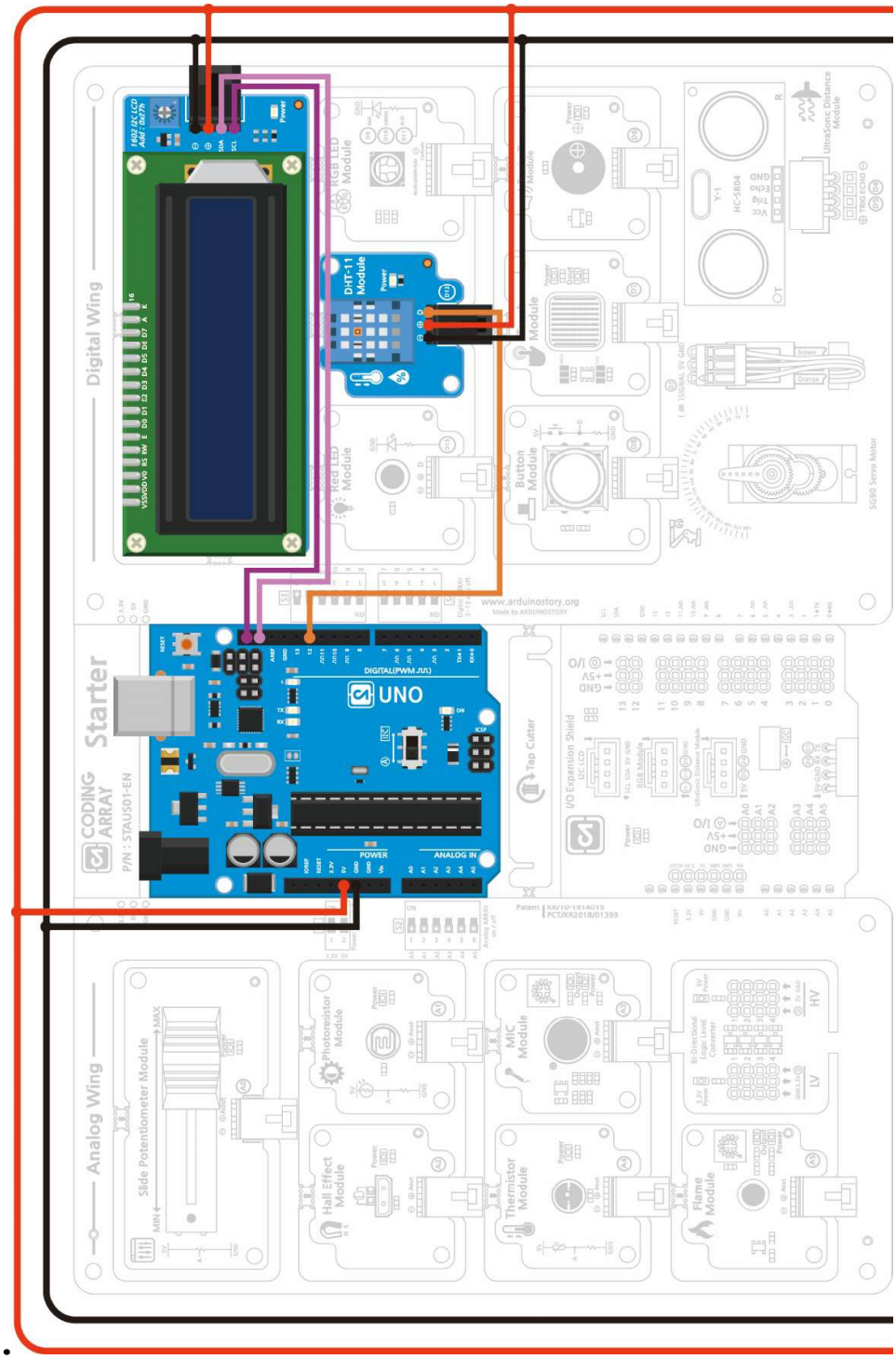
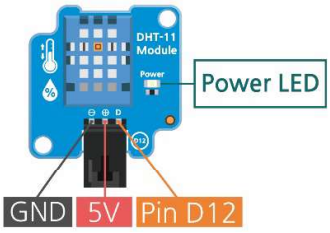


Figure 1134

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

Humidity & Temperature Sensor

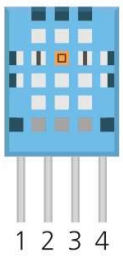
Figure



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

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

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



DHT11 Pins	
1	VCC
2	DATA
3	NC
4	GND

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

Figure 1156

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

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

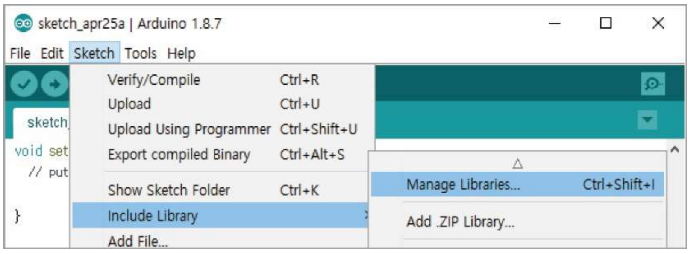


Figure 1167

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

Figure 1178

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

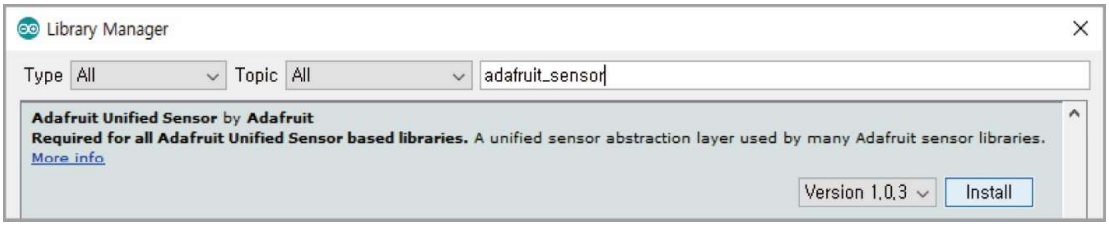


Figure 1189

- Adafruit Unified Sensor Library: *https://github.com/adafruit/Adafruit_Sensor*

- DHT Sensor Library: *https://github.com/adafruit/DHT-sensor-library*

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

CAK Starter Code > 16_DHT11

```
/*   This sketch uses the temperature sensor DHT11 module connected to digital pin 12.
*    Measure the ambient temperature and humidity
*    Outputs result values to I2C LCD.
*/

// Library for DHT11 Module Use
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

// library for I2C LCD use
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// set the temperature sensor
#define DHTPIN            12         // DHT sensor to pin 12.
#define DHTTYPE           DHT11      // DHT 11.

DHT_Unified dht(DHTPIN,DHTTYPE);     // form a dht object.

uint32_t delayMS;

// LCD settings
LiquidCrystal_I2C lcd(0x27,16,2);     // set the LCD I2C address. 16kans2joules LCD use

void setup() {
  Serial.begin(9600);                 // initiate serial communication at 9600 speed

  // Initialize LCD
  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);               // the first column of the second line
  lcd.print("DHT Sensor Start");
  delay(1000);
   lcd.clear();
 // start DHT sensor
  dht.begin();
  Serial.println("DHT11 Unified Sensor Example");

  // print temperature sensor information
  sensor_t sensor;

  dht.temperature().getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.println("Temperature");
```

```
48    Serial.print ("Sensor:     "); Serial.println(sensor.name);
49    Serial.print ("Driver Ver: "); Serial.println(sensor.version);
50    Serial.print ("Unique ID: "); Serial.println(sensor.sensor_id);
51    Serial.print ("Max Value: "); Serial.print(sensor.max_value); Serial.println(" *C");
52    Serial.print ("Min Value: "); Serial.print(sensor.min_value); Serial.println(" *C");
53    Serial.print ("Resolution:"); Serial.print(sensor.resolution); Serial.println(" *C");
54    Serial.println("------------------------------------");
55
56    // Print the Humidity Sensor Information
57    dht.humidity().getSensor(&sensor);
58    Serial.println("------------------------------------");
59    Serial.println("Humidity");
60    Serial.print ("Sensor:      "); Serial.println(sensor.name);
61    Serial.print ("Driver Ver: "); Serial.println(sensor.version);
62    Serial.print ("Unique ID:   "); Serial.println(sensor.sensor_id);
63    Serial.print ("Max Value:   "); Serial.print(sensor.max_value); Serial.println("%");
64    Serial.print ("Min Value:   "); Serial.print(sensor.min_value); Serial.println("%");
65    Serial.print ("Resolution: "); Serial.print(sensor.resolution); Serial.println("%");
66    Serial.println("------------------------------------");
67
68    // Time to read the sensor
69    delayMS =sensor.min_delay /1000;
70  }
71
72  void loop() {
73    // Delay between measurements.
74    delay(delayMS);
75
76    sensors_event_t event;
77    dht.temperature().getEvent(&event);   // Get temperature event and print its value.
78    if(isnan(event.temperature)) {
79      Serial.println("Error reading temperature!");
80    }
81
82    else{
83      Serial.print("Temperature: ");
84      Serial.print(event.temperature);
85      Serial.println(" *C");
86
87      lcd.setCursor(0,0);                    // First line first column
88      lcd.print("Temp     : " );             // Output message
89      lcd.print(event.temperature,0);          // Output without decimal point of measurement temperature
90      lcd.print(" [C]");                     // Output in units
91    }
92
93    dht.humidity().getEvent(&event);    // Get humidity event and print its value.
94    if(isnan(event.relative_humidity)) {
95      Serial.println("Error reading humidity!");
96    }
```

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

View Results

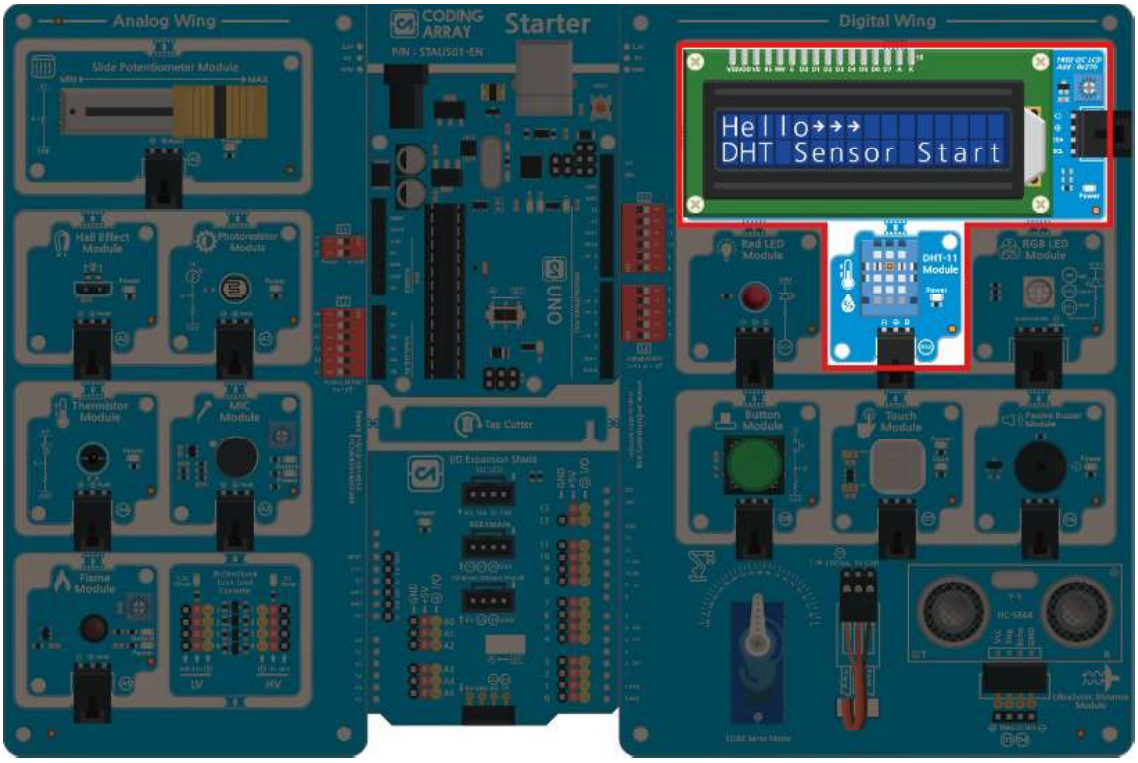


Figure 11950

Figure 1201

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

Figure 1212

Let's learn how to add a library..

- Using the Arduino Library

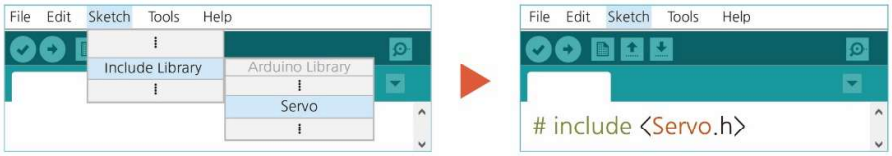


Figure 1223

- Search and install libraries in 'Library Manager'

Figure 1234

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

- Installing the Extended Library

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

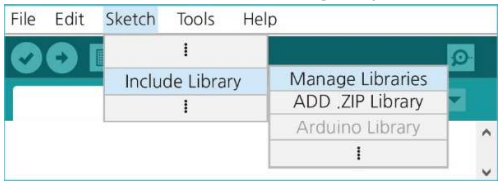


Figure 1245

17. Controlling servo motors

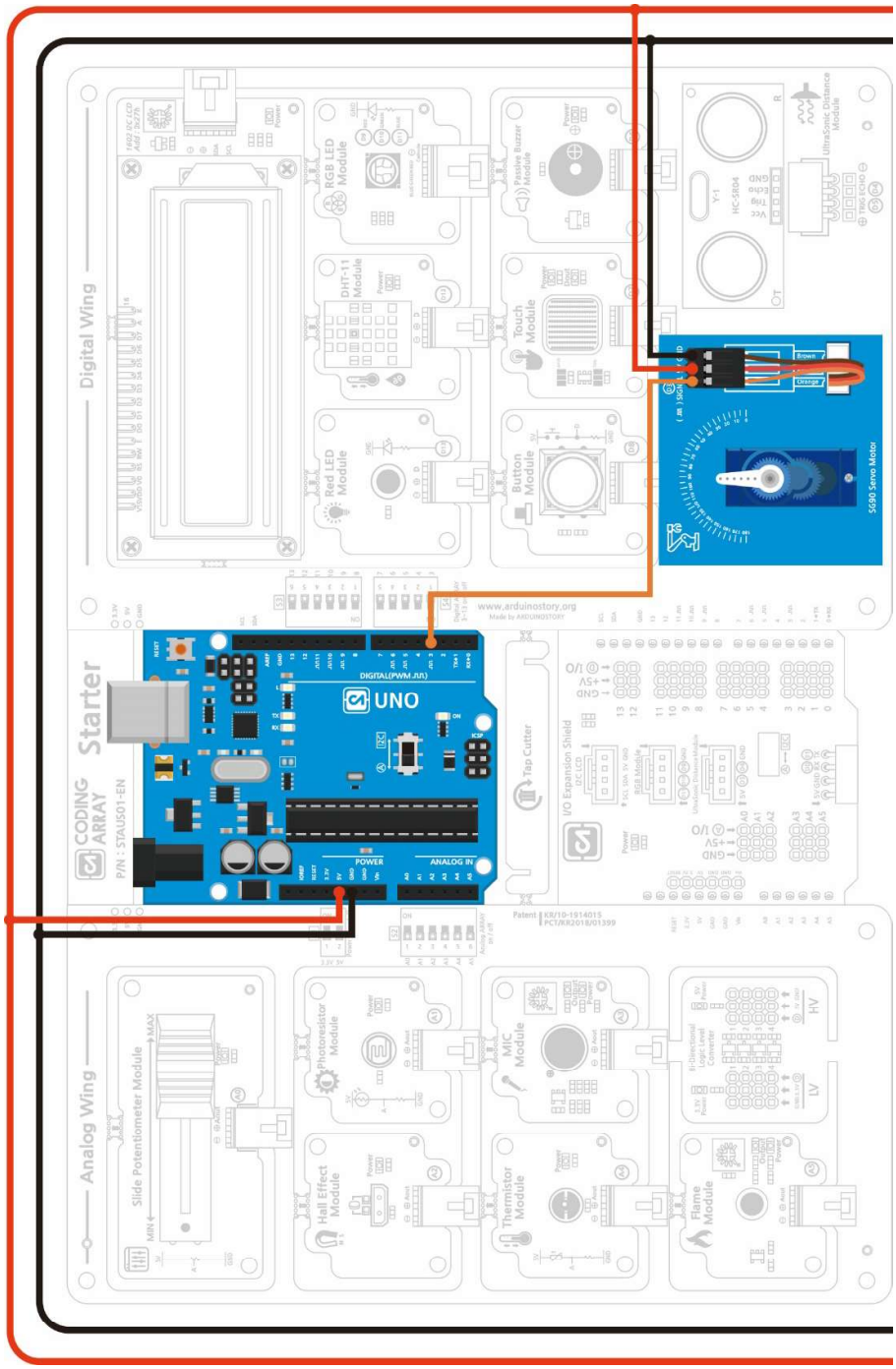
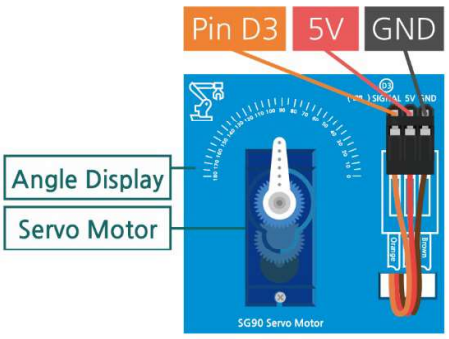


Figure 1256

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

Servo Motor

Figure 1267



The servo motor is a motor that can rotate the axis at the desired angle and was used for small RC toys. It is also the first motor connected to Arduino. The array kit used a 9g servo motor showing the rotation angle between 0 and 180 °. The servo motor has a line that can connect three pins, brown to GND, red to 5 V and orange to Arduino's PWM pin. Servo motors consume significant power and, if more than one movement is required, must be supplied with external power rather than 5V of Arduino..

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

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

CAK Starter Code > 17_01_Servo_Sweep

```
/*  This sketch uses servo live
 *  Move the servo motor by 0 - > 180 degrees
 *  Move back to 180 - >0 degrees.
 *  Note that servo motors cannot be rotated 360 degrees.
 */

#include <Servo.h>          // Include servo library

Servo myservo;              // create object myservo to control servo

int position =0;           // store the servo's position. Initial value 0

void setup() {
  myservo.attach(3);       // attach servo motor to pin 3
}

void loop() {
  myservo.write(90);         // position in the center of the servo motor shaft (90 degrees)
  delay (1000);

  myservo.write(0);         // position 0 degrees on servo motor shaft
  delay (1000);

  myservo.write(180);    // position 180 degrees on servo motor shaft
  delay (1000);

  myservo.write(90);    // located in the center of the servo motor shaft
  delay (1000);

  for(position = 0; position <= 180; position += 1) {      // increase by 1 degree to 0 to 180 degrees.
    myservo.write(position);                                // move to a specified angle
    delay(30);                                             // wait until servo to arrive.
  }

  for(position = 180; position >= 0; position -= 1) {      // decrease by 1 degree to 180 degrees.
    myservo.write(position);                                // move to a specified angle
    delay(30);                                             // wait until servo to arrive.
    }
}
```

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

recognises servo motor connected to PWM pin.

myservo : servo object

Pin number : Pin number with servo attached

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

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

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

myservo : servo object

Angle : Value from 0 to 180 for servo to move

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

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

View Results

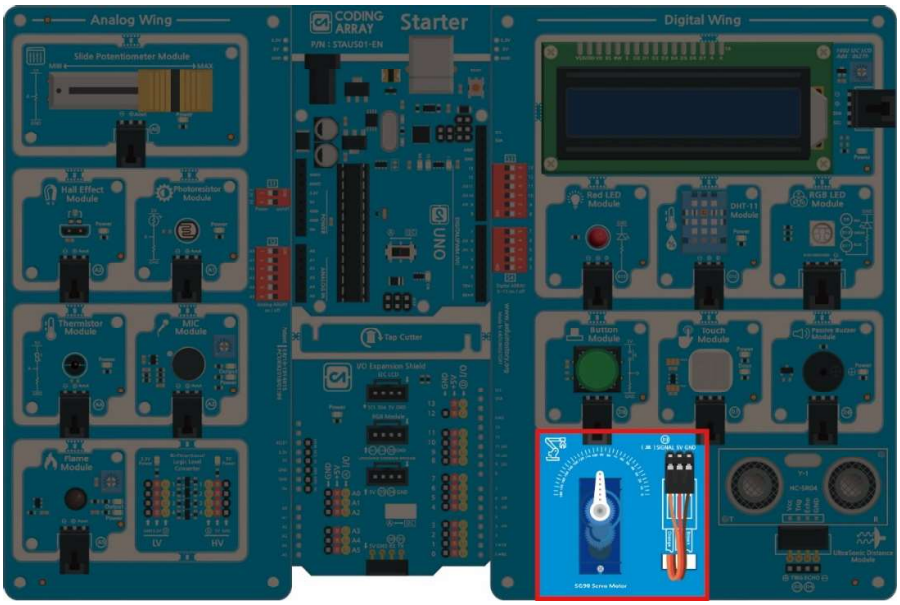


Figure 158

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

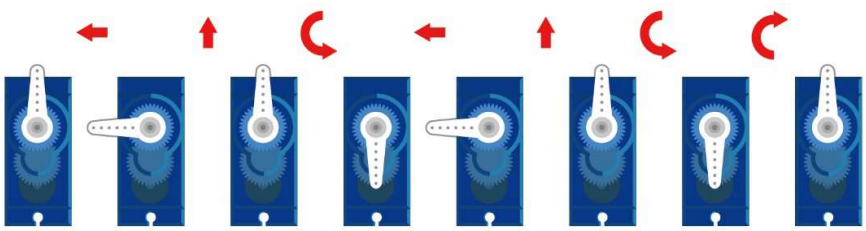


Figure 12759

CAK Starter Code > 17_02_Servo_Knob

```
/*  This sketch uses servo live
 *  The servo motor is between 0 and 180 degrees depending on the variable resistance value
 *  Move the axis.
 */

#include <Servo.h>          // Include servo library

Servo myservo;              // create object myservo to control servo; up to 12 can be created

int position =0;            // store the servo's position. Initial value 0
int potPin=A0;              // Connect variable resistance to A1 pin

void setup() {
  myservo.attach(3);        // attach servo motor to pin 3
}

void loop() {

  int val =analogRead(potPin);            // Read variable resistance value (0-1023)
  int servoVal =map(val,0,1023,0,180); // map variable resistance values to 0-180 degrees servo motor rotation angle
  myservo.write(servoVal);                // position of mapped servo
  delay(15);                              // wait for the servo to arrive.

  int pitchVal=map(val, 0, 1023,120,1500);  // map variable resistance values to 120 to 1500 Hz sound frequencies
  tone(6, pitchVal, 10);      // output mapped sound value to manual buzzer connected to pin 6 for 10 milliseconds

  delay (1);                              // delay of 1 millisecond for safety

}
```

View Results

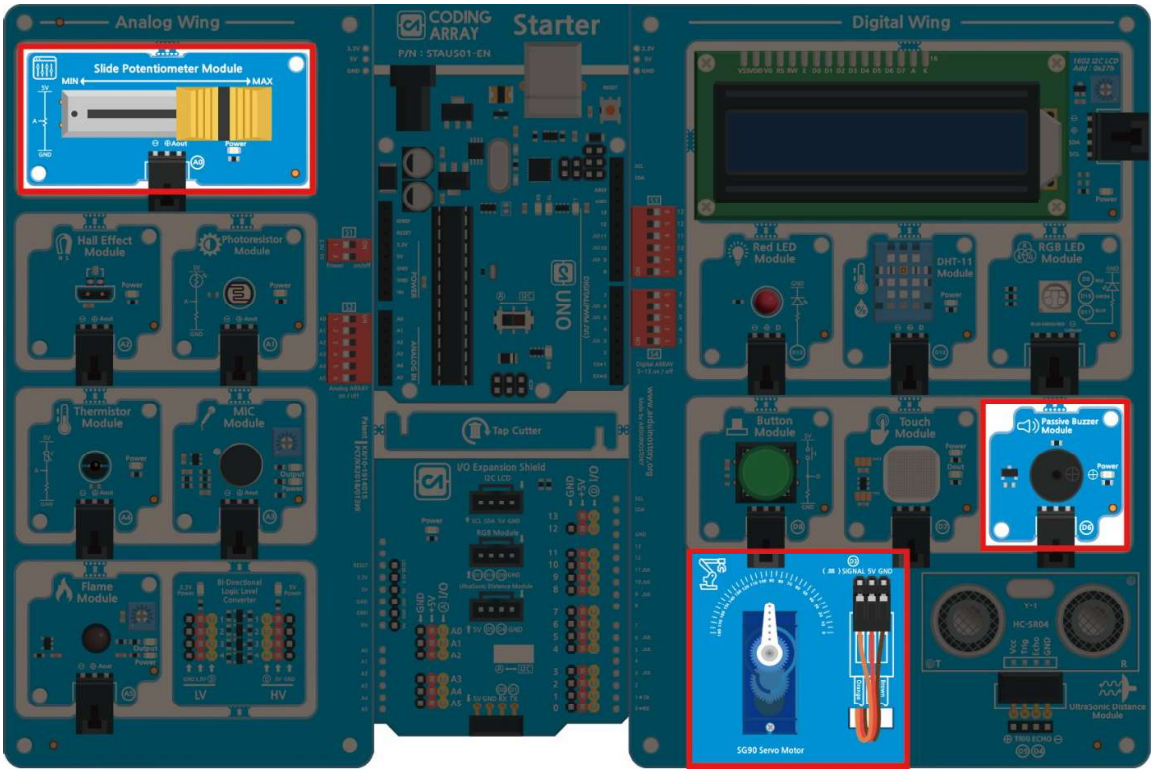


Figure 12860

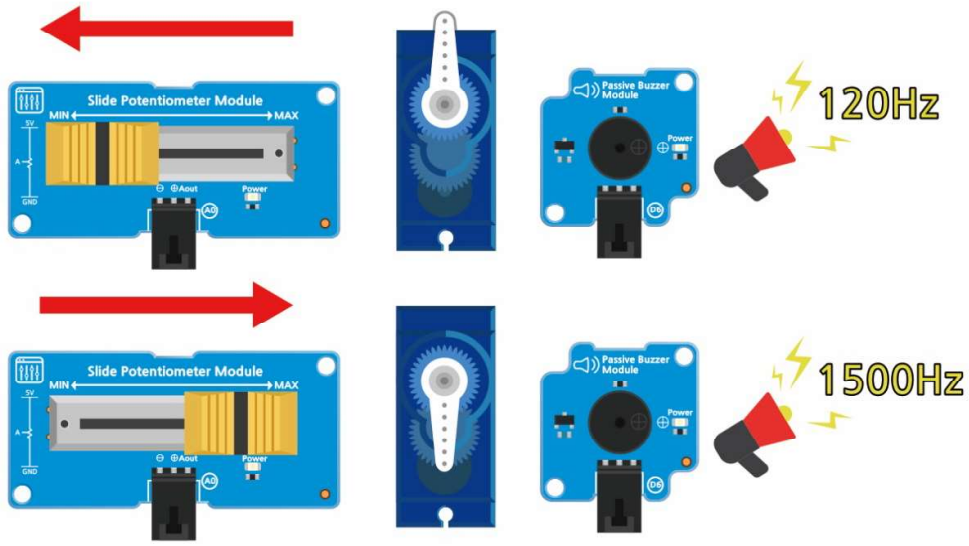


Figure 12961

Let's learn about the methods available in the Servo library.

Servo object name; form objects of the Servo class

Form an object called Servo myservo; myservo to activate the servo site.

myservo.attach (pin number, max value, max value); connecting servo motor to input/output pin

myservo : servo object

Pin number : Pin number with servo attached

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

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

myservo.write (angle); set the angle to move the servo motor

myservo : servo object

Angle : The value from 0 to 180 for the servo to move, the wrong angle is treated with microseconds.

CAUTION) When rotating using the letter(), wait for the time to rotate.

A short setting of this time will not allow rotation to the desired angle.

myservo.writeMicrosconds (microseconds); sets the pulse width of the servo motor in microseconds.

1000 : Fully rotating counterclockwise

2000 : Fully rotating clockwise

1500 : Center

myservo.read (angle); set the angle to move the servo motor

myservo.attached ( ); returns true if servo motor is connected or false.

myservo.detach ( ); disconnect the servo motor from the pin. When the servo variable is disconnected, PWM output can be used for pins 9 and 10 with analogWrite( ).

FAQ and Solution

Question	Solution
The touch sensor module turns on a blue light when it is not pressed.	1. 1. Switch on due to current consumption rather than failure, remove and plug in the USB cable. 2. 2. Connect the 7V to 12V adaptors separately and use them. - This is mainly caused by high current consumption during servo motor operation.