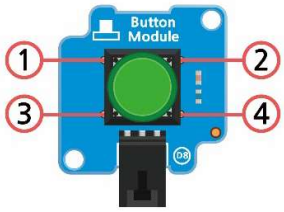
✧ Use a button switch connected to pin 8 using a pull-down resistance inside the module. Use the digitalRead function to receive the digital input value of the button switch and to issue a digital output command to the LED. Let's also find out how button switches divaunhings..

Push Button



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

Figure 21

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

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

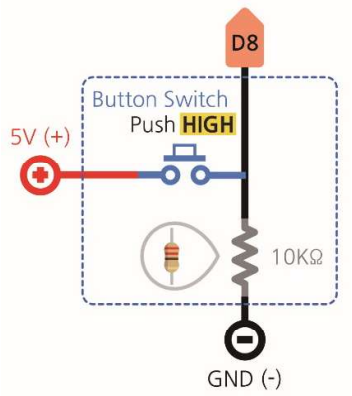


Figure 22

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

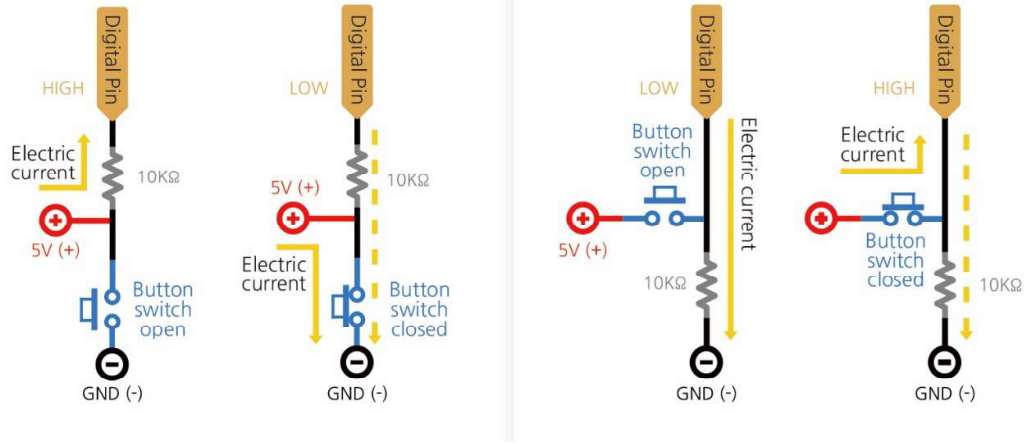


Figure 23

CAK Starter Code > 04_01_Button

```
/ *   If condition statement is used
 *   Press button switch to connect circuit and return HIGH (1) value to light LED
 *   If the button switch is not pressed, the circuit will open and the LOW (0) value will be returned to turn off the LED.
 *
 */

int redLED = 13;       //   Set red LED pin to 13.
int Button = 8;        // Button switch pin set to 8

void setup() {
  pinMode(Button, INPUT);                 // Set button pin to input
  pinMode(redLED, OUTPUT);                //   Set redLED pin to output

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

}

void loop() {

  int sensorVal = digitalRead(Button);   // Receive button input value for sensorVal variable.
                                         // Variables inserted in void function are regional variables


  Serial.println(sensorVal);           // Mark the value of the button one line in the serial window.
// If the pull-down resistance is connected, give LOW (0) when the button is open and HIGH (1) when pressed.
// give HIGH (1) value when the button is open and LOW (0) when pressed.

  if (sensorVal == LOW) {          // If the button is open,
    digitalWrite(redLED, LOW);           // RedLED OFF
  }

  else {                         // If the button is pressed,
    digitalWrite(redLED, HIGH);        // Turn on the redLED.
  }
  delay(10);
}
```

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

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

View Results

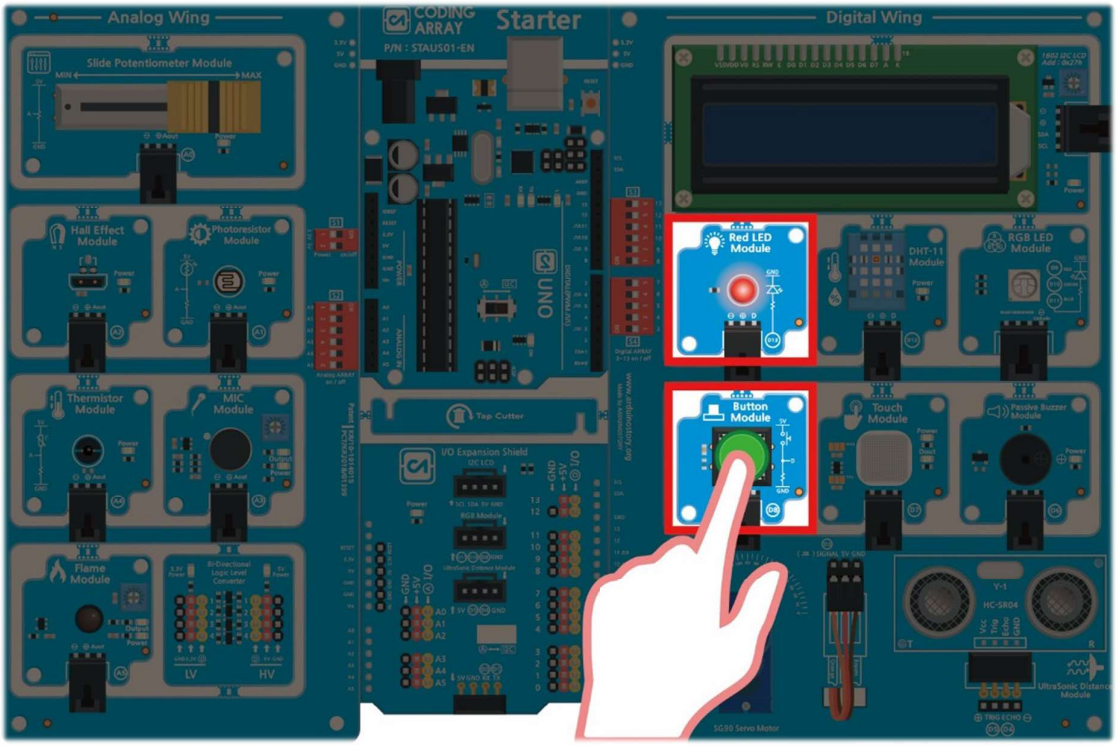


Figure 24

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

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

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

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

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

CAK Starter CODE > 04_02_Button_Debounce

```
/* When a button switch is opened and closed, it often generates incorrect signals due to mechanical and physical problems.
 *  Avoid this situation by pressing and reading multiple times in a very short time that can fool a program
 *  This process is called divauning.
 */

const int Button =8;                //  Set button switch pin to 8
const int redLED = 13;              //  Set LED pin to 13

int ledState =HIGH;                 //  Set output pin to HIGH
int buttonState;                    //  Variables that read and store the current button switch status
int lastButtonState =HIGH;          //  Read and save the previous button switch status, reset to LOW

unsigned long lastDebounceTime =0;    //  Save the last time the output pin was switched.
unsigned long debounceDelay =50;      //  Time to wait for steady state (milliseconds)

void setup() {
  pinMode(redLED, OUTPUT);                    //  Set red LED pin to output
  pinMode(Button, INPUT);
  digitalWrite(redLED,ledState);                    //     Turn the LED on and off according to the ledState.
}

void loop() {
  int reading = digitalRead(Button);   //  Read button status and save to reading variable
  if(reading != lastButtonState) {     //  If the status of the button changes to Noise or Press,
    lastDebounceTime =millis();          //  Reset the debounging timer,
  }

  //  Whatever value you've read, if it's longer than the debounce delay,
  if((millis() -lastDebounceTime) > debounceDelay) {
    if(reading != buttonState) {              //  If the status of the button changes,
      buttonState =reading;                   //  Save the status of the button.
      if(buttonState ==LOW) {                     //  If the new button status is HIGH
        ledState =!ledState;                      // Change the status of the LED.
      }
    }
  }
  digitalWrite(redLED,ledState);          //  LEDs are on/off with values stored in ledState
  //  Store the value of the reading variable in lastButtonState (used in the next loop)
  lastButtonState =reading;
}
```

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

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

Relational operator

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

Table 9

Arithmetic operator

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

Table 8

If / if ~else / Multiple if condition statements

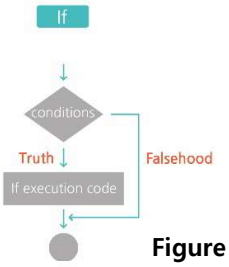


Figure 33

If (conditions) { If execution code }

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

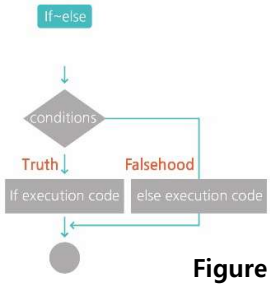


Figure 25

If (conditions) {If execution code;}

else { else execution code; }

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

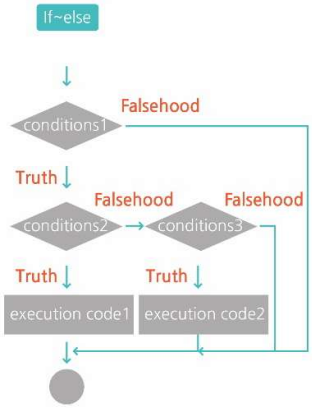


Figure 35

If (conditions1) {

If (conditions2) {execution code1 }

If (conditions3) {execution code2 }

}

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

5. Change RGB LED Color Using Digital Output and PWM Features

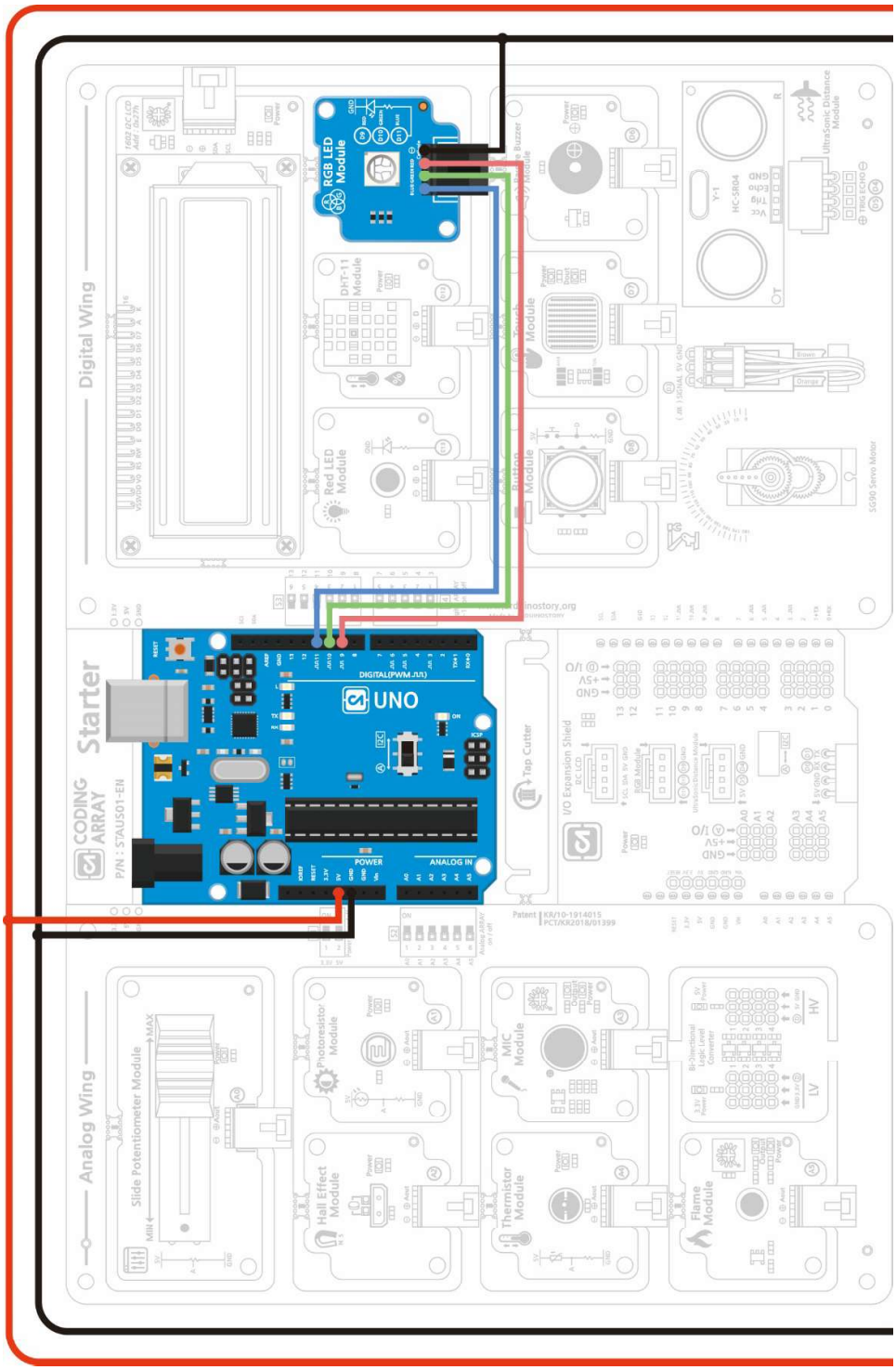
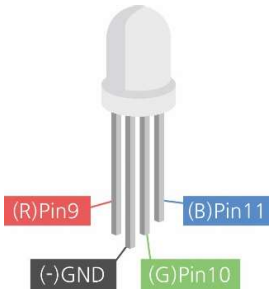


Figure 26

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

RGB LED



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

Figure 27

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

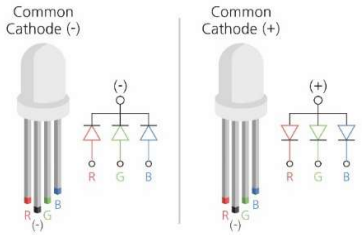


Figure 28

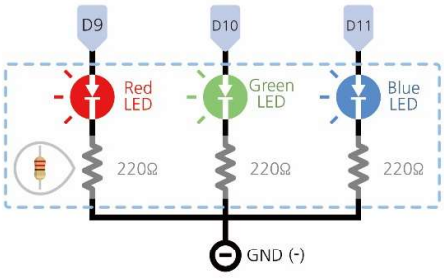


Figure 29

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

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

CAK Starter Code > 05_01_RGB_DigitalMixing

```
/*  RGB LEDs combine the three primary colors of red, green and blue to release a variety of colors.
 *  Pin 910 and 11 are connected to pins that control red, green and blue LEDs respectively.
 *  In this example, we will use a common cathode RGB LED to find out the tri-circular mixture of light from the digital output.
 */

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

void setup() {
    pinMode(redPin, OUTPUT);    // Set pin 9 to output
    pinMode(greenPin, OUTPUT); // Set pin 10 to output
    pinMode(bluePin, OUTPUT);   //Set pin 11 to output
   Serial.begin(9600);          // 9600-speed serial communication start
}

void loop() {
    Serial.println("RED on");          // Red LED illuminated
     digitalWrite(redPin,HIGH);
     digitalWrite(greenPin,LOW);
     digitalWrite(bluePin,LOW);
     delay(1000);                       // for a second

     Serial.println("GREEN on");        // Green LED illuminated
     digitalWrite(redPin,LOW);
     digitalWrite(greenPin,HIGH);
     digitalWrite(bluePin,LOW);
     delay(1000);                       // for a second

     Serial.println("BLUE on");         // Blue LED illuminated
     digitalWrite(redPin,LOW);
     digitalWrite(greenPin,LOW);
     digitalWrite(bluePin,HIGH);
     delay(1000);                      // for a second

     Serial.println("Yellow on");       // Yellow LED illuminated
     digitalWrite(redPin,HIGH);
     digitalWrite(greenPin,HIGH);
     digitalWrite(bluePin,LOW);
     delay(1000);                       // for a second

     Serial.println("Magenta on");      // Magenta illuminated
     digitalWrite(redPin,HIGH);
     digitalWrite(greenPin,LOW);
     digitalWrite(bluePin,HIGH);
```

```
46      delay(1000);                    // for a second
47
48    Serial.println("Cyan on");         // Cyan LED illuminated
49    digitalWrite(redPin,LOW);
50    digitalWrite(greenPin,HIGH);
51    digitalWrite(bluePin,HIGH);
52    delay(1000);                      // for a second
53
54    Serial.println("White on");        // White LED illuminated
55    digitalWrite(redPin,HIGH);
56    digitalWrite(greenPin,HIGH);
57    digitalWrite(bluePin,HIGH);
58    delay(1000);                      // for a second
59  }
```

View Results

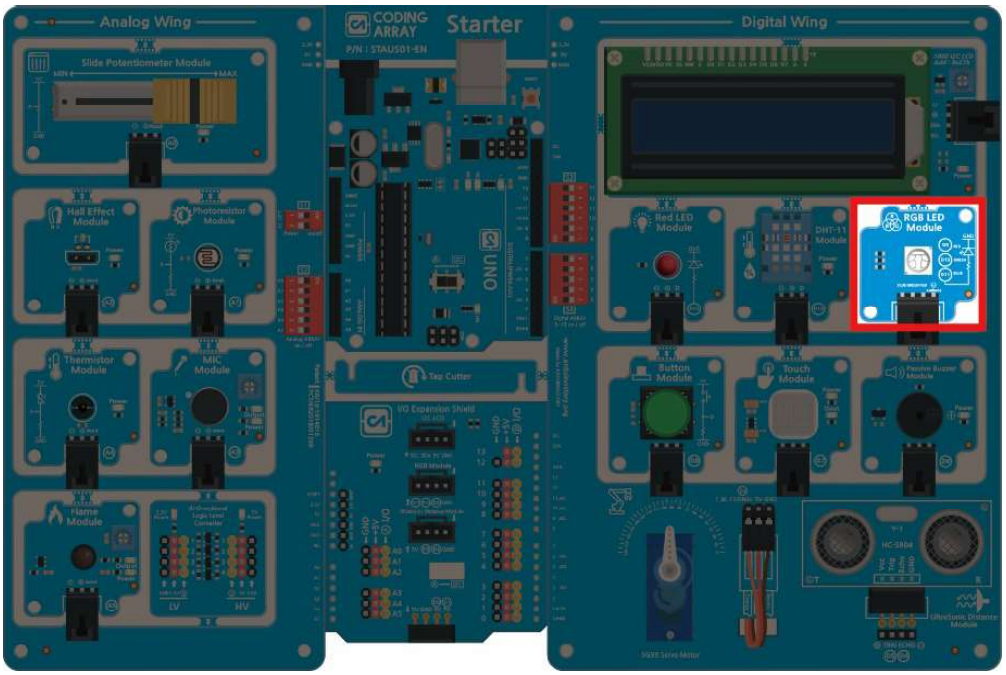
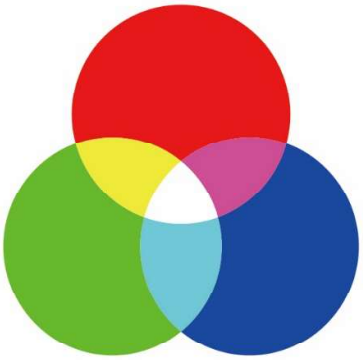


Figure 30

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

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



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

Figure 31

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

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

Figure 32

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

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

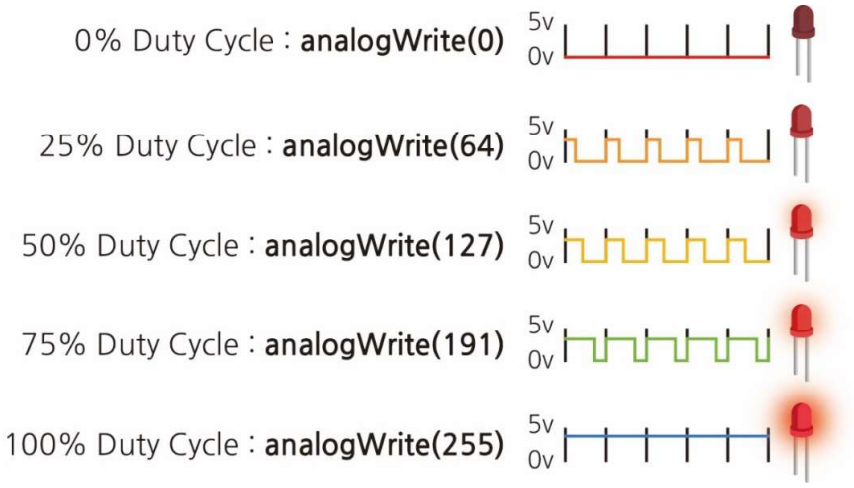


Figure 33

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

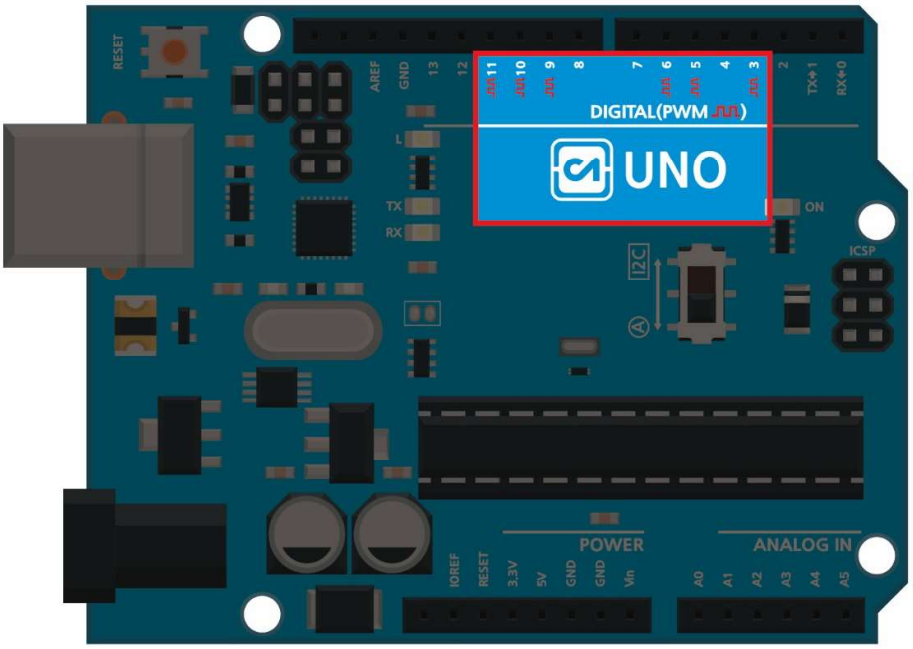

RESET
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX→1
RX←0
DIGITAL(PWM~)
UNO
L
TX
RX
ON
ICSP
I2C
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

Figure 34

CAK Starter Code > 05_02_RGB_LED_Fading

```
/*  In this example, we use a common cathode RGB LED
 *  Pin 910 and 11 are connected to pins that control red, green and blue LEDs respectively.
 *  Use the PWM (Pulse-Width Modulation) function to adjust the brightness of the LEDs.
 *  The 'radom' command adjusts the brightness of the three colors of RGB to enable a variety of color blends.
 */

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

int delayTime=30;             // Delay time setting

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

int fadeAmount =5;            // fade storage variable

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

void loop() {
  // Red LED brightness adjustment
  greenV=0;
  blueV=0;
  for(redV =0; redV <=255;redV +=5) {
                                        // Increase the value by 5 times from 0 to 255.
    analogWrite(redPin,redV);                  // Turn on the LED more and more and more.
    analogWrite(greenPin,greenV);
    analogWrite(bluePin,blueV);
    delay(delayTime);                          // 30-millisecond wait
  }

  for(int redV=255 ;redV >=0; redV -=5) {
                                        // Reducing the value by 5 times from 255 to 0.
    analogWrite(redPin,redV);                  // Turn on the darker LEDs.
    analogWrite(greenPin,greenV);
    analogWrite(bluePin,blueV);
    delay(delayTime);                          // 30-millisecond wait
  }

  // Green LED brightness adjustment
   redV=0;
   blueV=0;
```

```
48    for(greenV=0  ; greenV <=255; greenV +=5) {
49                                        // Increase the value by 5 times from 0 to 255.
50      analogWrite(redPin,redV);                  // Turn on the LED more and more and more.
51      analogWrite(greenPin,greenV);
52      analogWrite(bluePin,blueV);
53      delay(delayTime);                          // 30-millisecond wait
54    }
55
56    for(int greenV =255 ; greenV >=0; greenV -=5) {
57                                        // Reducing the value by 5 times from 255 to 0.
58      analogWrite(redPin,redV);                  // Turn on the darker LEDs.
59      analogWrite(greenPin,greenV);
60      analogWrite(bluePin,blueV);
61      delay(delayTime);                          // 30-millisecond wait
62    }
63
64    // Blue LED brightness adjustment
65    redV=0;
66    greenV=0;
67    for(blueV=0 ; blueV <=255; blueV +=5) {
68                                        // Increase the value by 5 times from 0 to 255.
69      analogWrite(redPin,redV);                  // Turn on the LED more and more and more.
70      analogWrite(greenPin,greenV);
71      analogWrite(bluePin,blueV);
72
73      delay(delayTime);                              // 30-millisecond wait
74    }
75
76    for(int blueV=255 ;blueV >=0; blueV -=5) {
77                                        // Reducing the value by 5 times from 255 to 0.
78      analogWrite(redPin,redV);                  // Turn on the darker LEDs.
79      analogWrite(greenPin,greenV);
80      analogWrite(bluePin,blueV);
81      delay(delayTime);                          // 30-millisecond wait
82    }
83
84  // 20 Random colors
85    for (int i=0; i<20; i++){
86      analogWrite(redPin,random(0,255));
87      analogWrite(greenPin,random(0,255));
88      analogWrite(bluePin,random(0,255));
89      delay(1000);
90    }
91  }
```

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

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

View Results

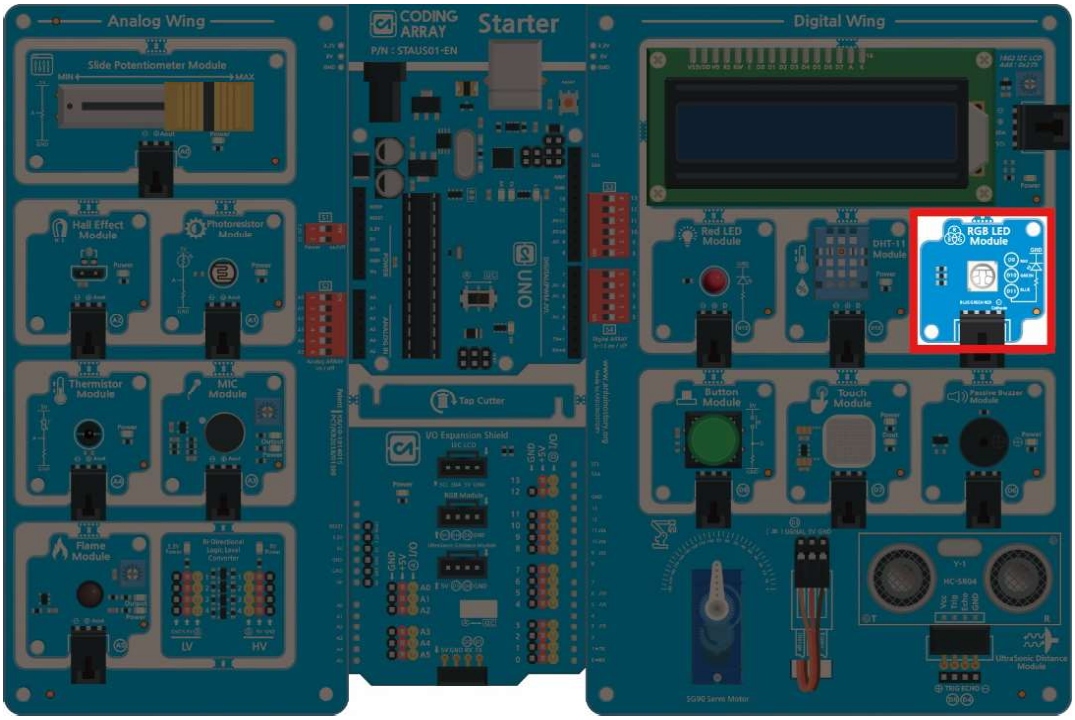


Figure 45

Figure 46

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

6. Implementing moods, etc. with capacitive touch sensors

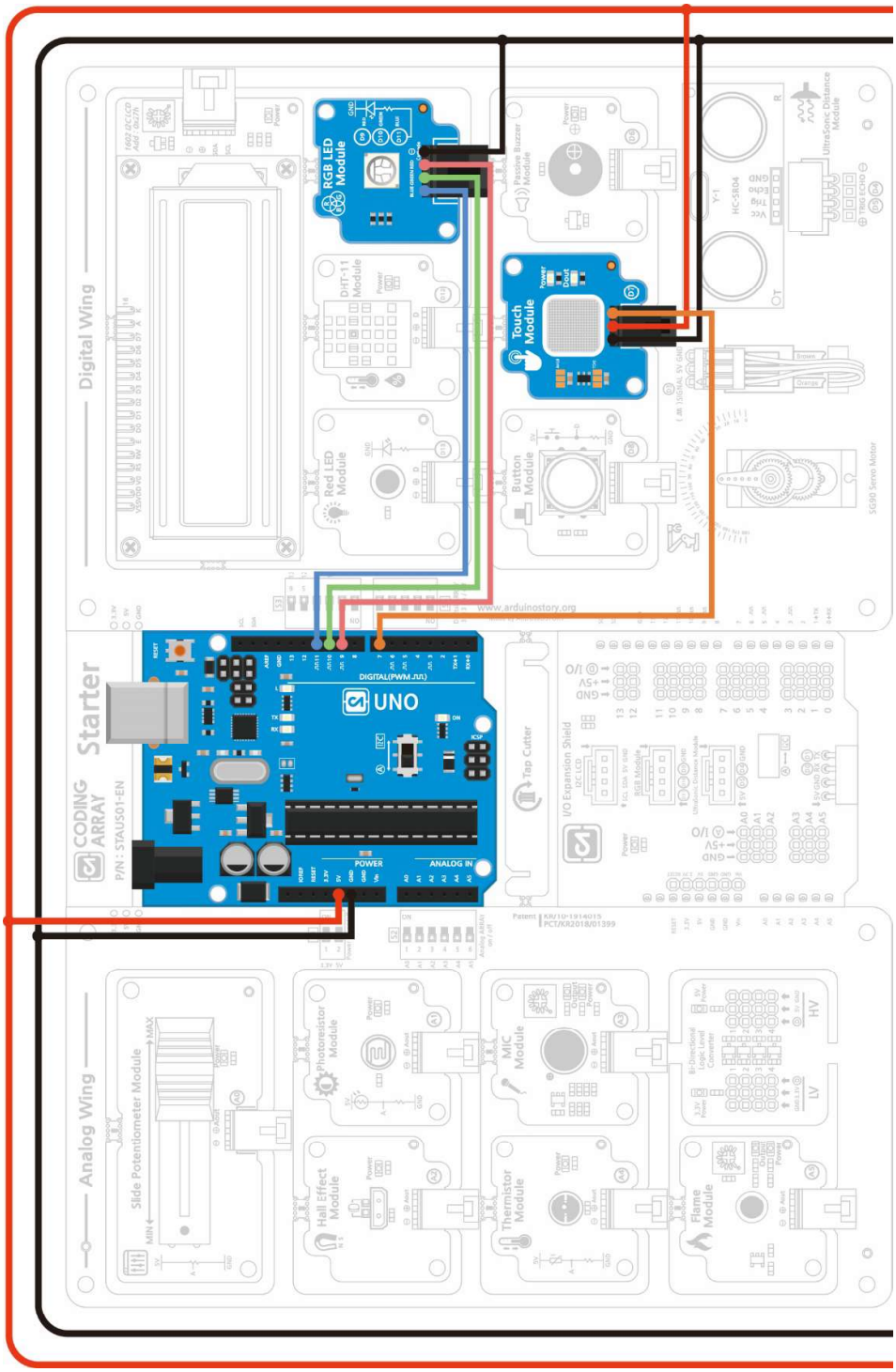


Figure 35

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

Capacitive Touch Sensor

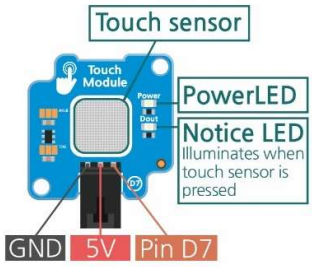


Figure 36

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

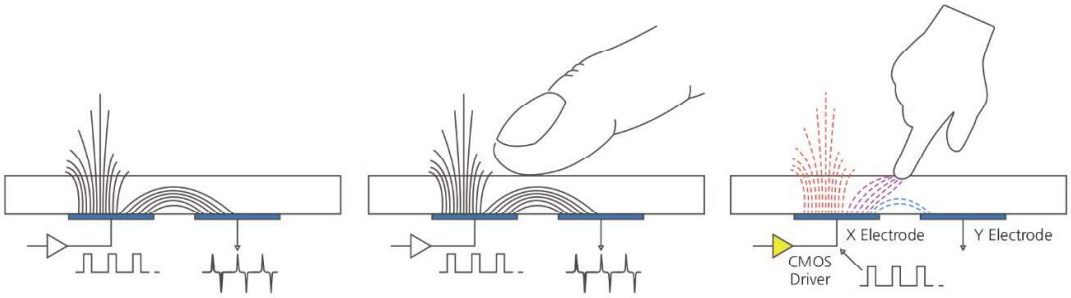


Figure 37

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

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

CAK Starter Code > 06_01_TouchSensor

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

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

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

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

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

global variable

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

View Results

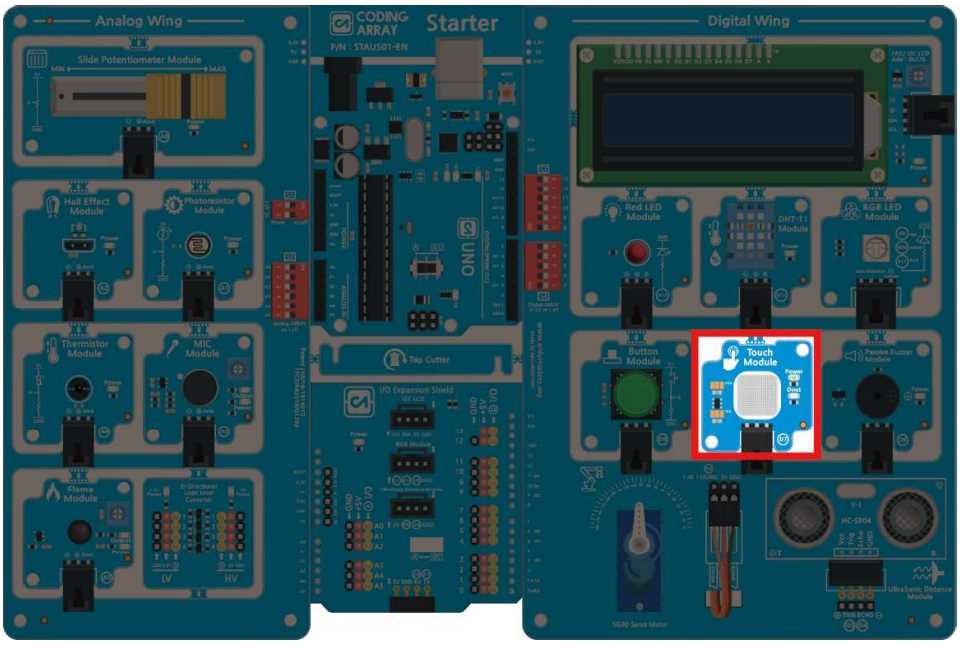


Figure 38

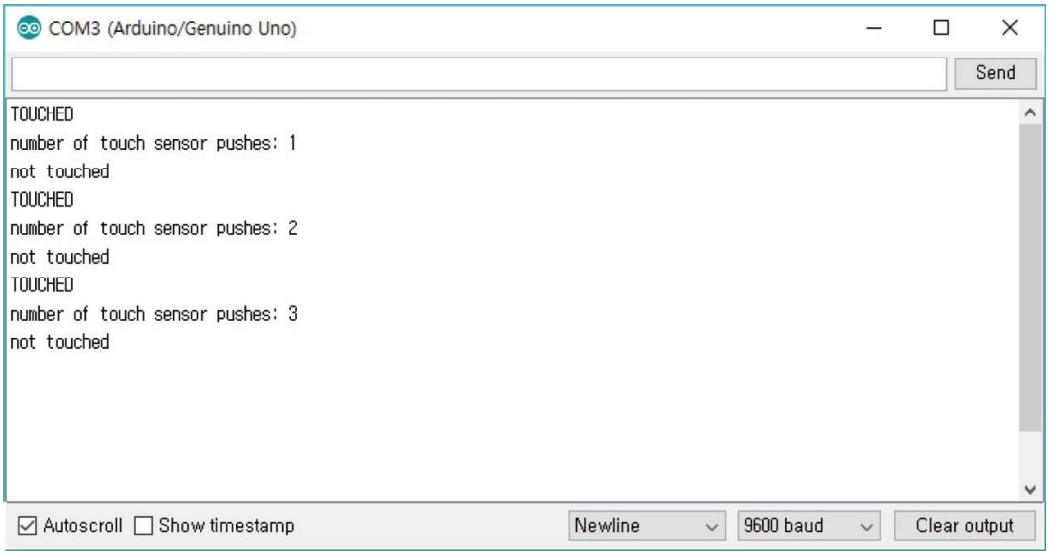


Figure 39

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

CAK Starter Code > 06_02_TouchStateChange

```
/*   This sketch shows a touch sensor
     Touch 1, 2 and 3 to light up the green LED.
     Touch four times to show the LED is off.
     This allows users to set the desired brightness mood using the touch sensor.
*/

#define Touch 7              // Connect the capacitive touch sensor to   7.

const int greenPin = 10 ;        // Green LED to No.10

int touchCounter = 0;          // Variables that store the number of times a touch sensor is pressed
int lastTouchState = 0;          // Read and save the previous Touch Sensor status
int alanlogValue;               //   Set analog value of LED (0-255)

void setup() {
  pinMode(Touch, INPUT);      // Set the touch sensor connected to pin 7 to input
  pinMode(greenPin, OUTPUT);   // Set pin 11 to output
  Serial.begin(9600);        // Starts serial communication at 9600 speeds
}

void loop() {
  int touchState = digitalRead(Touch); // Read touch sensor switch values and store them in touchState

  if (touchState != lastTouchState) { // Touch sensor status has changed
    if (touchState == HIGH) {              //
      touchCounter ++;                                  // Increase the number of touch sensors pressed
      Serial.println("TOUCHED");                        // Write "TOUCHED" in the serial window and replace lines
      Serial.print("number of touch sensor pushes: ");     // "~:" to the cereal window
      Serial.println(touchCounter);                    // Connect and press and replace touch sensor

      // Use current touchState as lastTouchState in the next loop
      lastTouchState = touchState;

      if (touchCounter % 4 == 0) {    // Touch sensor presses 0
        alanlogValue = 0;
        Serial.println("OFF");
      }

      if (touchCounter % 4 == 1) {    // Touch sensor presses 1
        alanlogValue = 85;
        Serial.println("First level");
      }

      if (touchCounter % 4 == 2) {    // Touch sensor presses 2
        alanlogValue = 170;
        Serial.println("Second level");
      }
```

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

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

View Results

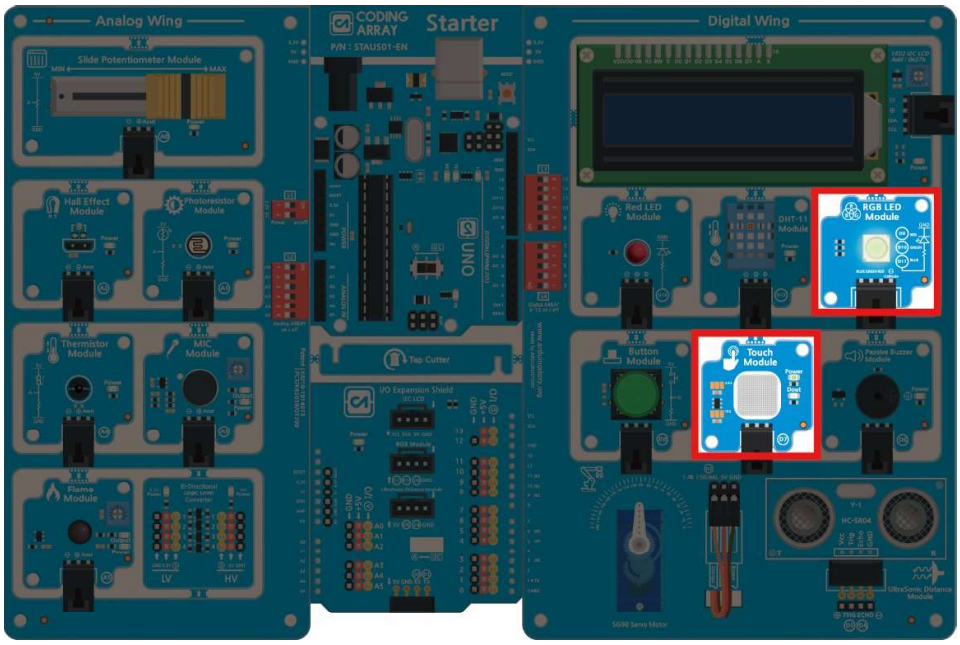


Figure 40

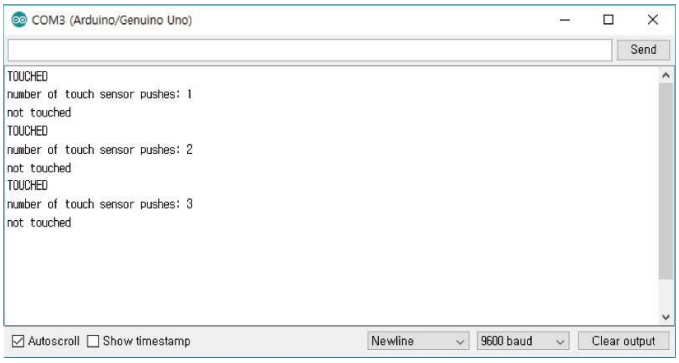


Figure 42

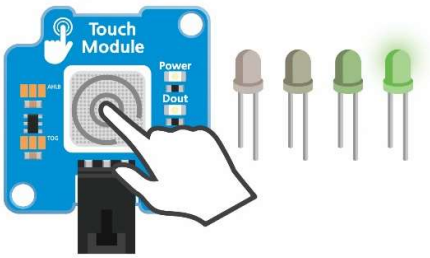


Figure 41

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

7. Read slide variable resistance value with analog input

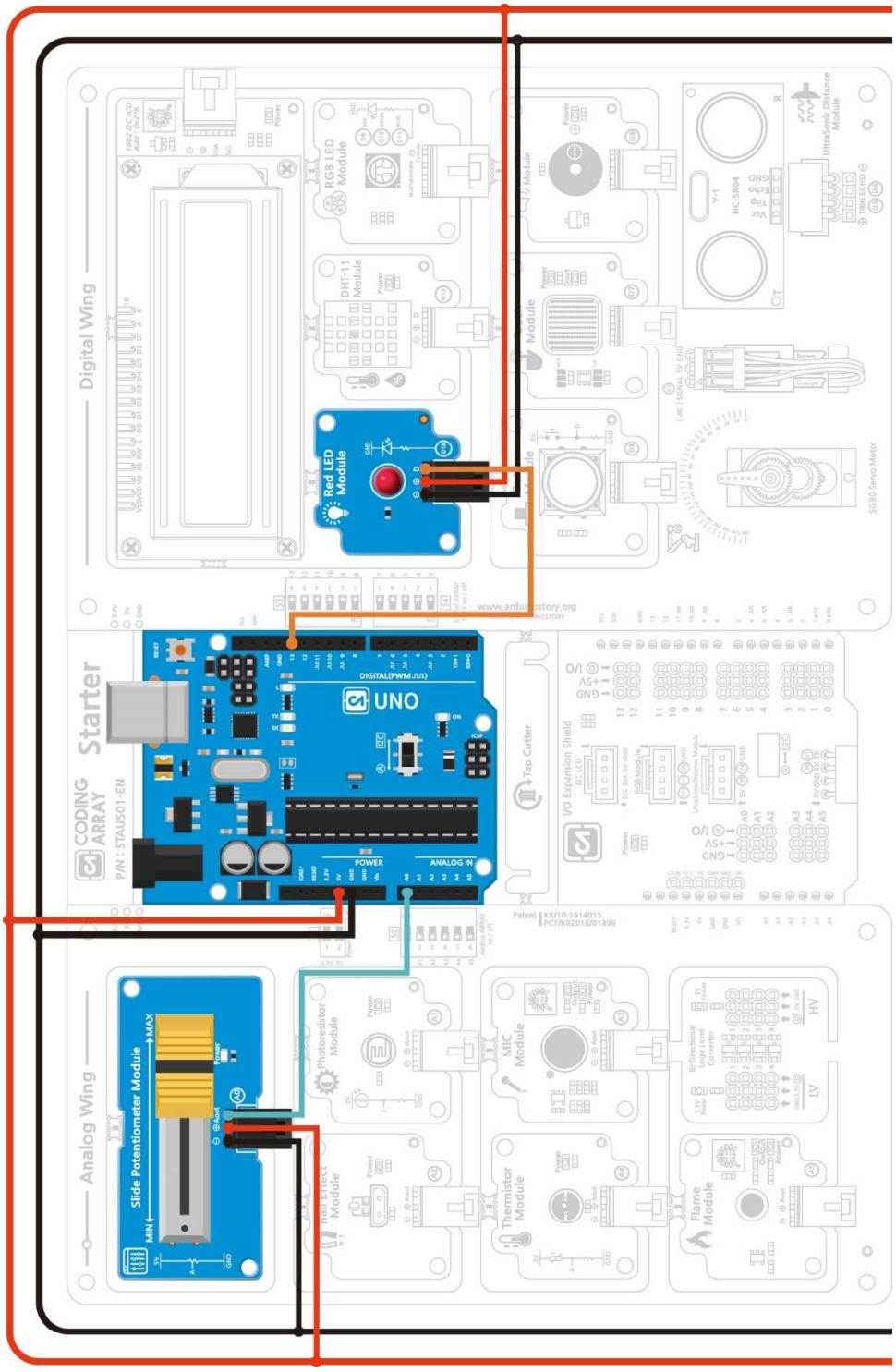


Figure 43

✧ Using the variable resistance connected to the analog A0 pin, receive the 0-1023 analog input value and convert it to voltage. In addition to viewing numerical data in the serial window, use the serial plotter function to graph the data..

Slide Potentiometer

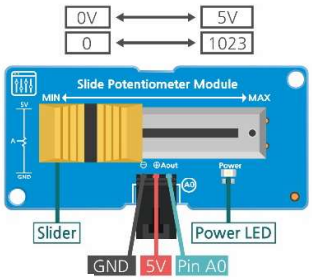


Figure 44

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

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

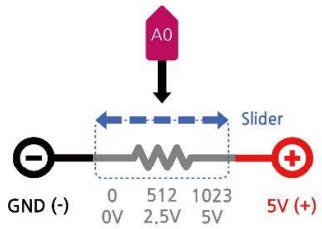


Figure 45

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

Let's learn about analog input..

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

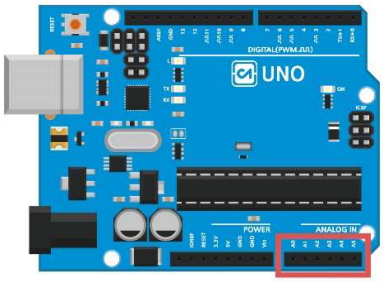


Figure 46

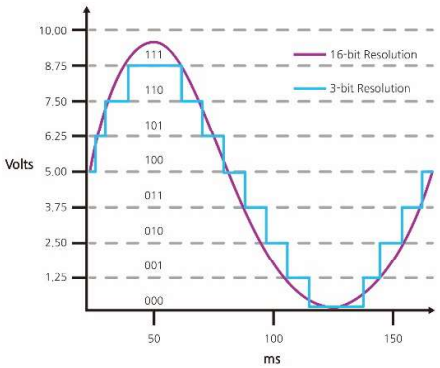
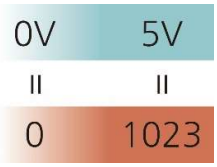


Figure 47

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

Figure 48

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



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

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

Table 10

Let's find out about voltage distribution..

Calculate the Voltage Divider as follows:.

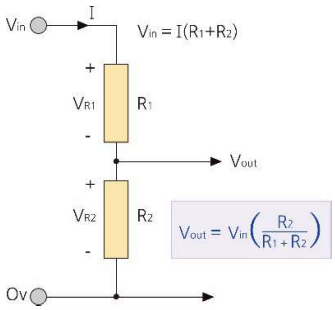


Figure 49

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

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

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

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

According to Ohm's Law V=IR,

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

$\mathrm{I} = I_{R_2} = \frac{\mathrm{V}_{in}}{\mathrm{R}_1 + R_2}$

$V_{R_2} = \frac{\mathrm{V}_{in}}{\mathrm{R}_1 + R_2} \times R_2$

Then, $V_{R_2} = \frac{R_2}{\mathrm{R}_1 + R_2} \times \mathrm{V}_{in}$

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

CAK Starter Code > 07_01_SlidePotentiometer

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

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

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

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

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

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

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

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

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

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

Caution)The analog input/output does not have to be set in pinMode separately.

Caution) **Click Menu Bar > Tools > Serial Plotter (Ctrl+Shift+L)** to open the **serial plotter**. During the output of the serial plotter, the serial monitor window does not open simultaneously..

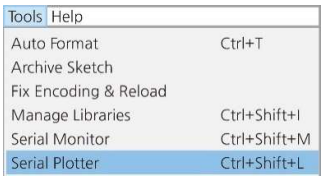


Figure 50

View Results

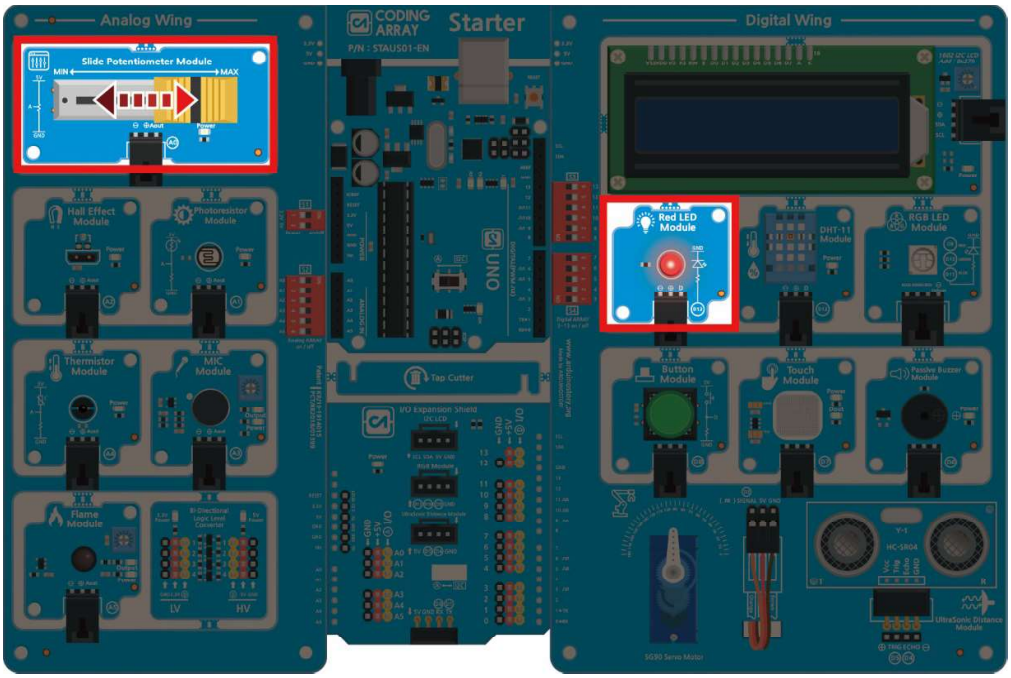
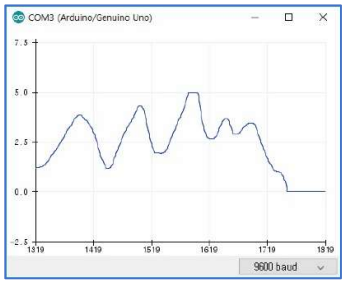


Figure 51

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



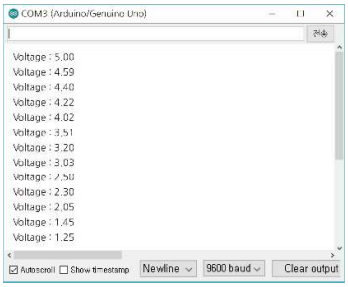


Figure 52

Figure 53

CAK Starter Code > 07_02_SlidePotentiometer2

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

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

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

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

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

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

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

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

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

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