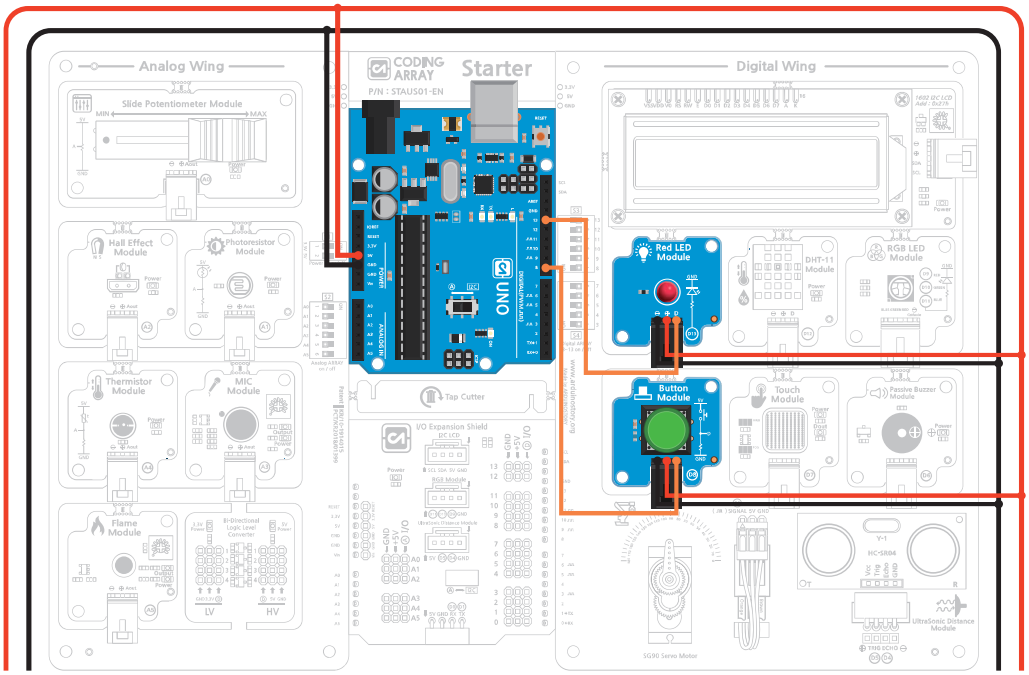
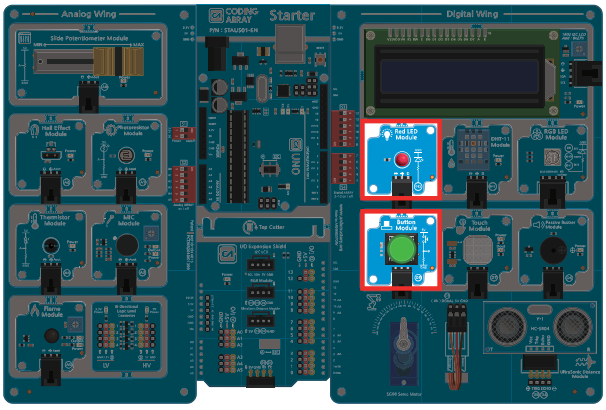
CODING ARRAY CIRCUIT



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..

View Results



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))

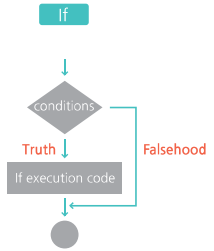
RELATIONAL OPERATOR

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

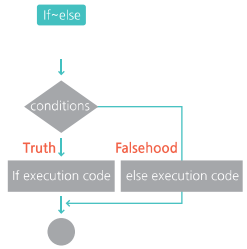
ARITHMETIC OPERATOR

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

If / if ~else / Multiple if condition statements

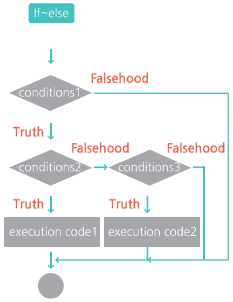


If (conditions) { If execution code }
If the conditional statement is true, perform the if execution code and if not move on to the next execution statement.



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.

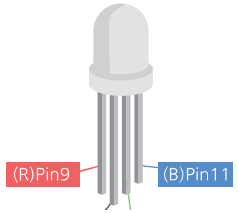


If (conditions1) {
If (conditions2) {execution code1 }
If (conditions3) {execution code2 }
}
If condition 1 is satisfied and condition 2 is satisfied at the same time, process execution code 1 and execute code 2 if condition 1 is satisfied and condition 3 is satisfied at the same time. You can also put another if statement inside the if statement. If a statement is executed inside, then indentation is required.

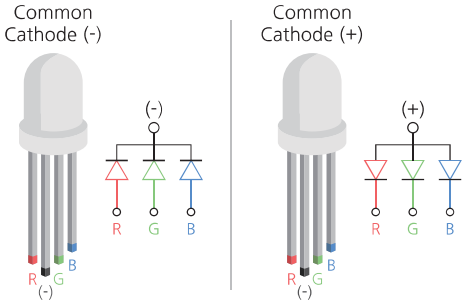
5 Change RGB LED Color Using Digital Output and PWM Featuresinput

CHAPTER 2

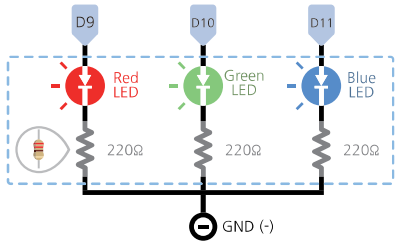
RGB LED



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..



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.



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.

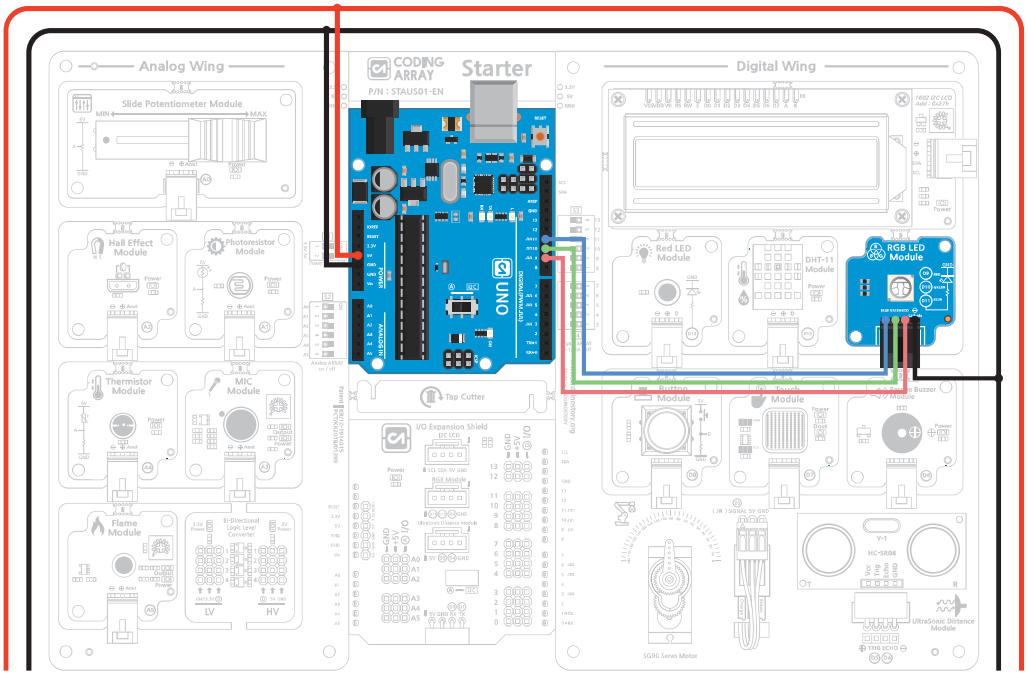
Using RGB LEDs, a variety of lighting effects can be obtained by producing different colors from a single LED. It is often used to decorate the computer's main case with colorful lights or change the color of billboards..

CODING ARRAY
STARTER KIT FOR ARDUINO

CAK Starter Code > 05_01_RGB_DigitalMixing

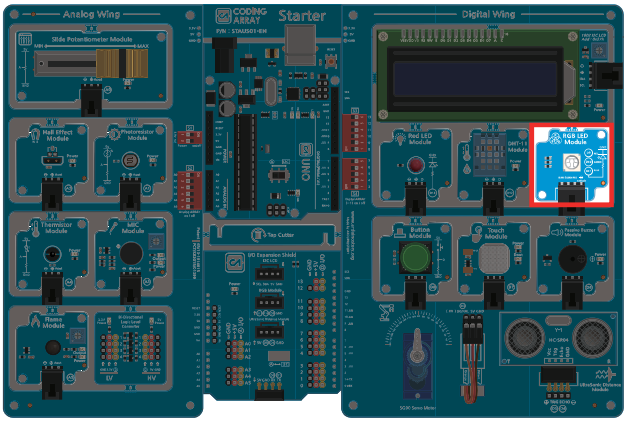
```
/*  RGB LEDs combine the three primary colors of red, green and blue to release a variety of colors.
 *  Pin 910 and 11 are connected to pins that control red, green and blue LEDs respectively.
 *  In this example, we will use a common cathode RGB LED to find out the tri-circular mixture of light 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);
    delay(1000);                  // for a second

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

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

CODING ARRAY CIRCUIT



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.

View Results



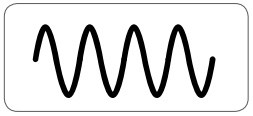
When a program is uploaded, it uses digital outputs to implement a three-way color mixture of light

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

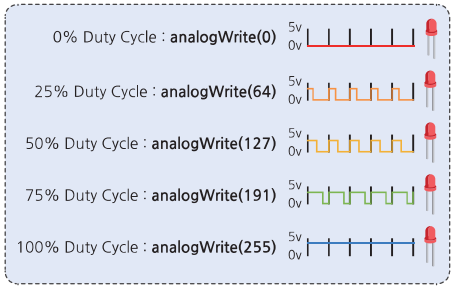
Let's learn about analog output / pulse width modulation (PWM).

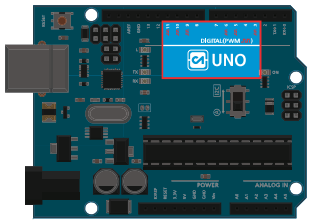
Unlike turning on and off LEDs, opening and closing circuits, many values such as light intensity, temperature, distance, sound size adjustment, and light intensity are made up of continuous analog signals..



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( )**..





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.**

CAK Starter Code > 05_02_RGB_LED_Fading

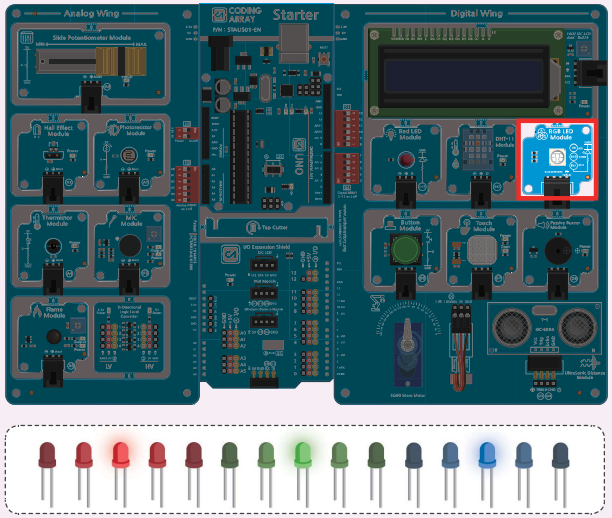
```
/* In this example, we use a common cathode RGB LED
 * Pin 910 and 11 are connected to pins that control red, green and blue LEDs respectively.
 * Use the PWM (Pulse-Width Modulation) function to adjust the brightness of the LEDs.
 * The 'radom' command adjusts the brightness of the three colors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

const int red Pin =9;          // Red LED No. 9
const int green Pin = 10;      // Green LED No. 10
const int blue Pin=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;
  for(greenV=0  ; greenV <=255; greenV +=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 greenV =255 ; greenV >=0; greenV -=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
  }

  // Blue LED brightness adjustment
  redV=0;
  greenV=0;
  for(blueV=0 ; blueV <=255; blueV +=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 blueV=255 ;blueV >=0; blueV -=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
  }

 // 20 Random colors
  for (int i=0; i<20; i++){
    analogWrite(redPin,random(0,255));
    analogWrite(greenPin,random(0,255));
    analogWrite(bluePin,random(0,255));
    delay(1000);
  }
}
```

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

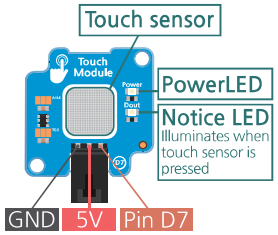


Red LEDs connected to pin 13 are not only lit/off outputs, but three-color LEDs connected to pins with PWM function can also be brightness controlled as well as on and off. Once the RGB_LED_Fading program is uploaded, you can see that it is getting brighter and darker in the order of red, green and blue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

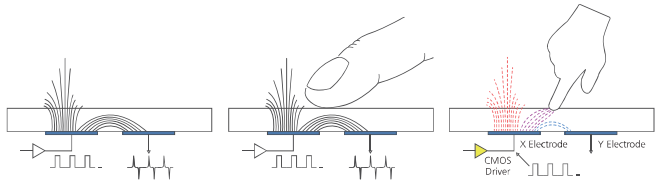
6 Implementing moods, etc. with capacitive touch sensors

CHAPTER 2

Capacitive Touch Sensor



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.



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 }
```

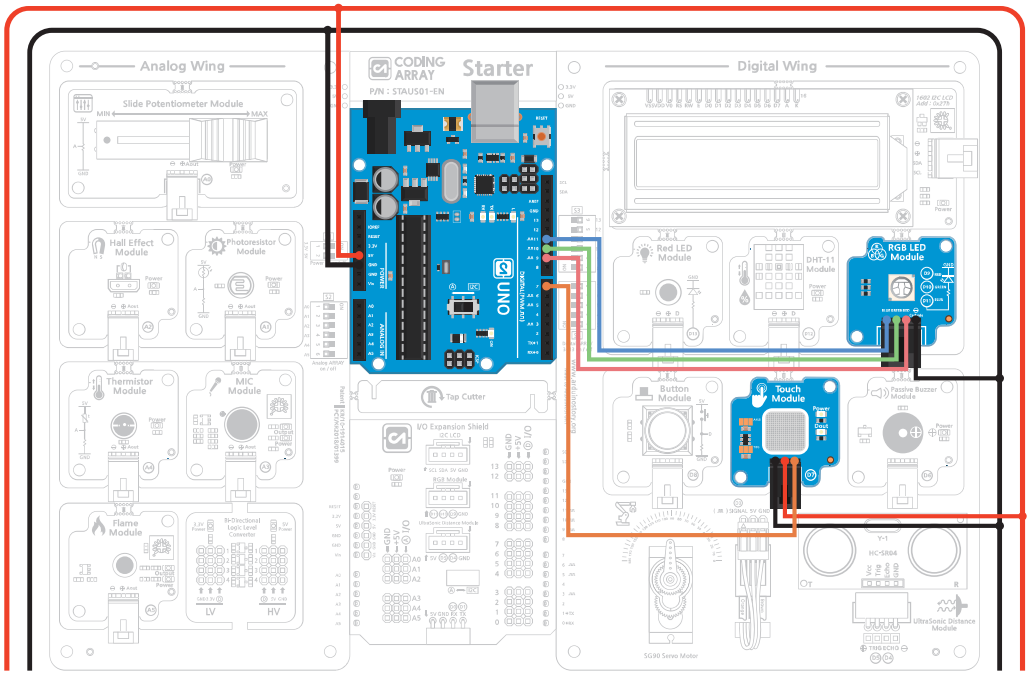
#define Constant name value : One of the pre-processing statements processed before program compilation is named constant value (you cannot change the data value while the program is running). Constants created in Define are compiled with all the constants of the source code replaced with values, so 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);
```

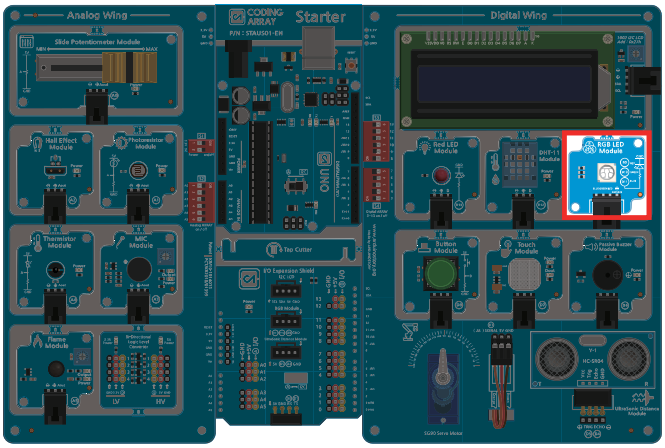
Caution) If a variable is declared before setting the void, use the variable in all parts of the program. : global variable

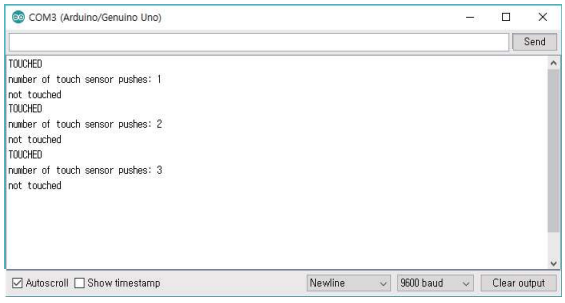
touchCounter ++; Increase the value of the touchCounter variable by 1.



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

View Results





After 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");
    }

    if (touchCounter % 4 == 3) {   // Touch sensor presses3
     alanlogValue = 255;
     Serial.println("Third level");
    }

    // Press the touch sensor to turn on the changed analog value.
    analogWrite(greenPin, alanlogValue);

  } else {                                // If the touch sensor has changed from TOUCHED to not touched
    Serial.println("not touched");               // Write "not touched" in the serial window and replace lines
  }
  delay(100);
 }

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

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..