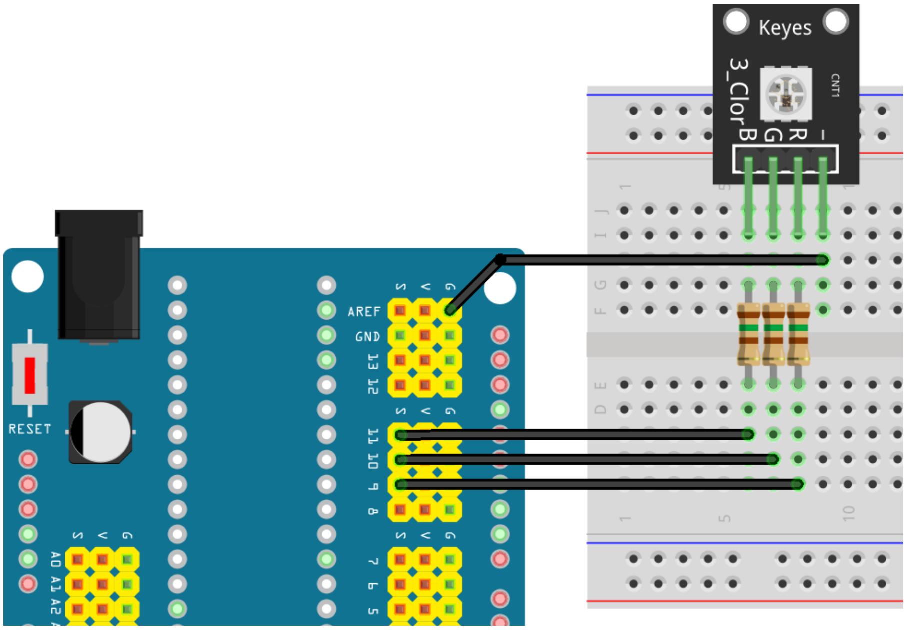
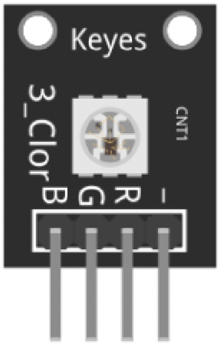
Task.1. Connect the circuit, as shown in the picture.



You need to use 120R resistors to prevent LED burnout.



Arduino board	KY-009 board	
11	B	blue component
10	G	green component
9	R	red component
Gnd	-	cathode

Use the following code example. Observe results.

```
#define pinR  9
#define pinG  10
#define pinB  11
#define BAUDRATE  115200

void setup() {
  pinMode(pinR,OUTPUT);
  pinMode(pinG,OUTPUT);
  pinMode(pinG,OUTPUT);

  digitalWrite(pinR,LOW);
  digitalWrite(pinG,LOW);
  digitalWrite(pinB,LOW); }

void loop() {
  digitalWrite(pinR,HIGH);
  digitalWrite(pinG,LOW);
  digitalWrite(pinB,LOW);
  delay(2000);
  digitalWrite(pinR,LOW);
  digitalWrite(pinG,HIGH);
  digitalWrite(pinB,LOW);
  delay(2000);
  digitalWrite(pinR,LOW);
  digitalWrite(pinG,LOW);
  digitalWrite(pinB,HIGH);
  delay(2000); }
```

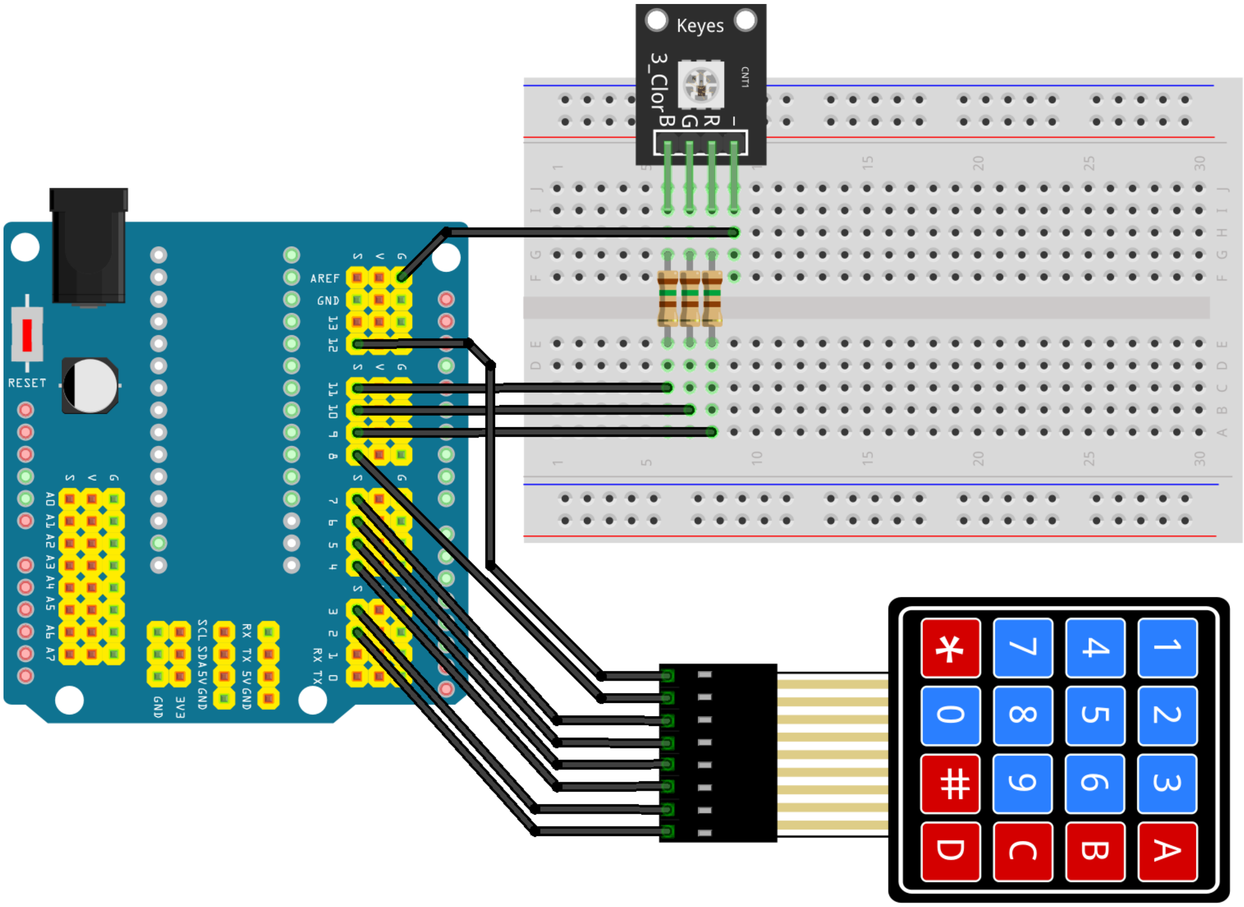
KY-009 is RGB full-color LED Module and is capable of emitting a range of colors by mixing red, green and blue. Red and green mixed together gives us yellow:

```
digitalWrite(pinR,HIGH);
digitalWrite(pinG,HIGH);
digitalWrite(pinB,LOW);
```

Additive mixing of red, green, and blue can produce white light:

```
digitalWrite(pinR,HIGH);
digitalWrite(pinG,HIGH);
digitalWrite(pinB,HIGH);
```

Task.2. Use the matrix keyboard to control the KY-009 RGB LED module.



Matrix keyboard	row 1 - A	row 4 - B	row 7 - C	row * - D	col. 1 - *	col. 2 - 0	col. 3 - #	col. A - D
Arduino board	12	8	7	6	5	4	3	2

Buttons assignment:

- button ‘1’ - RED On
- button ‘4’ - RED Off
- button ‘2’ - GREEN On
- button ‘5’ - GREEN Off
- button ‘3’ - BLUE On
- button ‘6’ - BLUE Off
- button ‘A’ - all On
- button ‘B’ - all Off

```
#include <Keypad.h>
#define pinR  9
#define pinG  10
#define pinB  11
#define ROWS 4
#define COLS 4
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'D','0','E','F'} };
byte rowPins[ROWS] = {11,8,7,6};
byte colPins[COLS] = {5,4,3,2};
Keypad keyb = Keypad(makeKeymap(keys),rowPins,colPins,ROWS,COLS);

void setup() {
 pinMode(pinR,OUTPUT);
 pinMode(pinG,OUTPUT);
 pinMode(pinG,OUTPUT);
 digitalWrite(pinR,LOW);
 digitalWrite(pinG,LOW);
 digitalWrite(pinB,LOW); }

void loop() {
 char key = keyb.getKey();
 if(key) {
  switch(key) {
      case '1':
       digitalWrite(pinR,HIGH);
      break;

      case '4':
       digitalWrite(pinR,LOW);
      break;

     default:
      break; }
 }
}
```

Task.2.1. Implement the rest of the buttons' functionalities by Yourself.

Task.3. Using the Task 2 circuit add dimming functionality for every light primary color. Proposed buttons assignment:

- button ‘7’ - R brightness +
- button ‘*’ - R brightness -
- button ‘8’ - G brightness +
- button ‘0’ - G brightness -
- button ‘9’ - B brightness +
- button ‘#’ - B brightness -
- button ‘C’ - RGB brightness +
- button ‘D’ - RGB brightness -

What’s new:

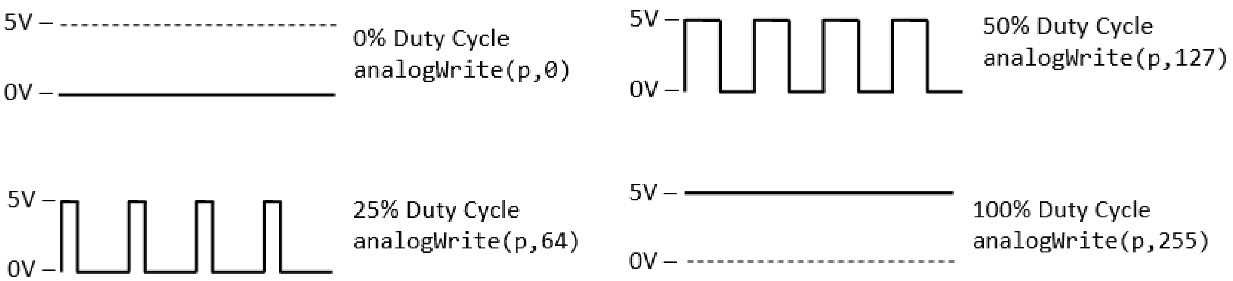
***analogWrite(**`p, value`**)**;

`p` - number of PWM pin - from 0 to 13

`value` - duty cycle - between 0(always off) and 255(always on)

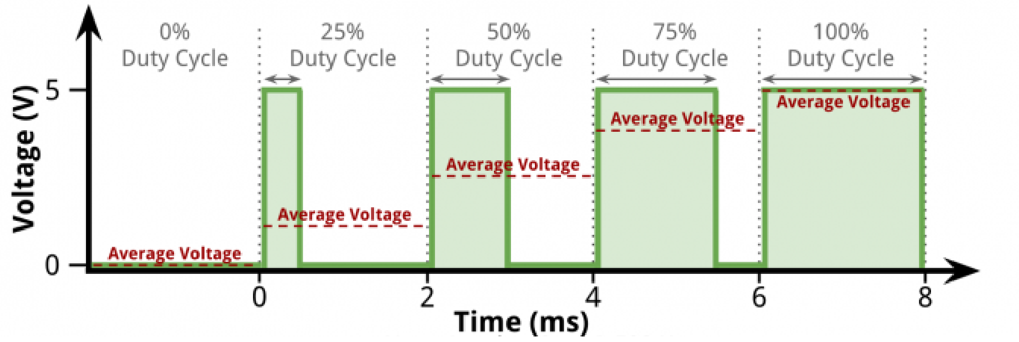
Reference:

www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/



**www.ntu.edu.sg*

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on(HIGH = 5V) and off(LOW = 0V). This on-off pattern can simulate voltages by changing the portion of the time the signal spends on versus the time that the signal spends off. Duty cycle (time_high/(time_high + time_low) * 100%) is proportional to the average voltage on the selected PWM pin.



**robotic-controls.com*

Replace `loop()` function from the Task 2 with the following code:

```
void loop() {
char key = keyb.getKey();
if(key) {
  switch(key) {
    case '1':
    analogWrite(pinR,brightnessR);
    break;

    case '4':
    analogWrite(pinR,0);
    break;

    case '7':
    if(brightnessR < 100)
      analogWrite(pinR,brightnessR += 10);
    while(keyb.getState() == HOLD);
    break;

    case 'D':
    if(brightnessR >= 10)
      analogWrite(pinR,brightnessR -= 10);
    while(keyb.getState() == HOLD);
    break;

    default:
    break;
    }
  }
}
```

Add global variable: `int brightnessR = 10;`

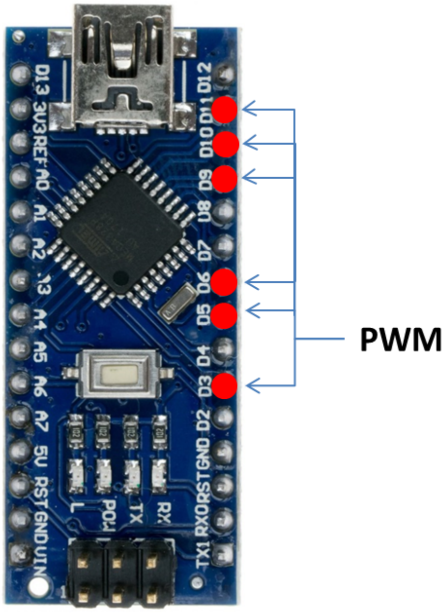
Reminder:

`KeyState getState()`

`keyb.getState()` returns the current state of any of the keys. The four states are *`IDLE`*, *`PRESSED`*, *`RELEASED`*, and *`HOLD`*.

Reference:

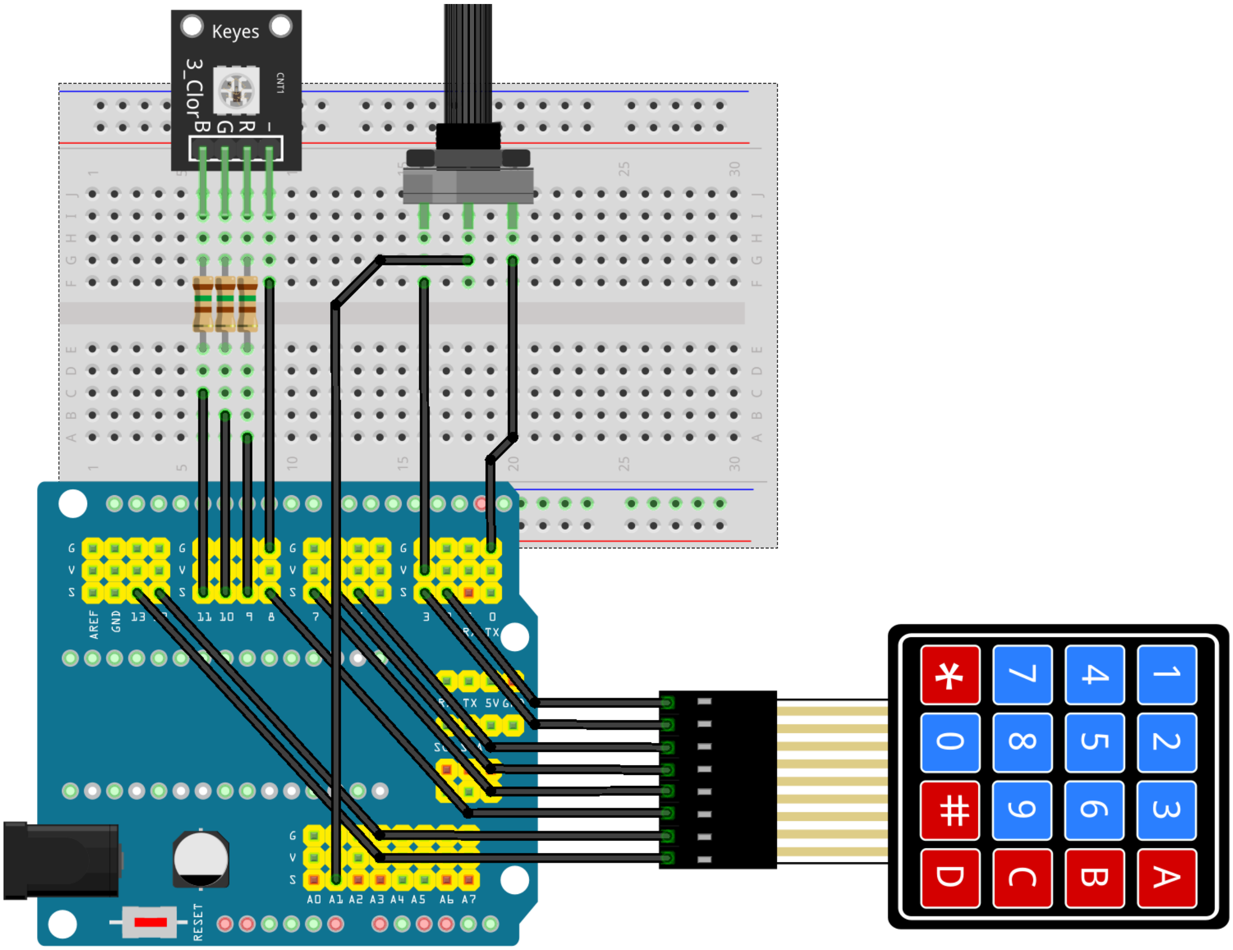
https://playground.arduino.cc/code/keypad



On Arduino Nano, there are a total of 6 PWM pins available. These pins are numbered 3, 5, 6, 9, 10, and 11. The default PWM frequency for all pins is 490 Hz, except pins 4 and 13 whose default frequency is 980Hz.

Task.3.1. Implement the rest of the buttons' functionalities by Yourself.

Task.4. Connect the circuit as shown in the picture:



Prepare a program that allows controlling the brightness of all primary colors using a variable resistor.

Task.5. Control the color and brightness of the LED using a Node-Red-based interface.

Task.6. Use a joystick to control the brightness of all primary colors.

For those interested:

1. Secrets of Arduino PWM:

   docs.arduino.cc/tutorials/generic/secrets-of-arduino-pwm

2. How to use a RGB LED with Arduino:

   howtomechatronics.com/tutorials/arduino/how-to-use-a-rgb-led-with-arduino/