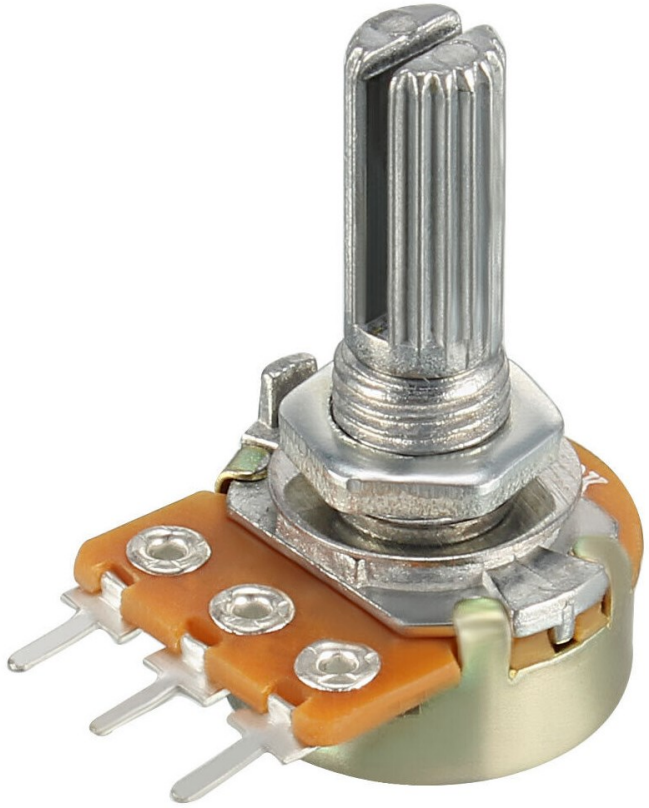
Embedded Systems

Analog Control System

Embedded Team, BFCAI

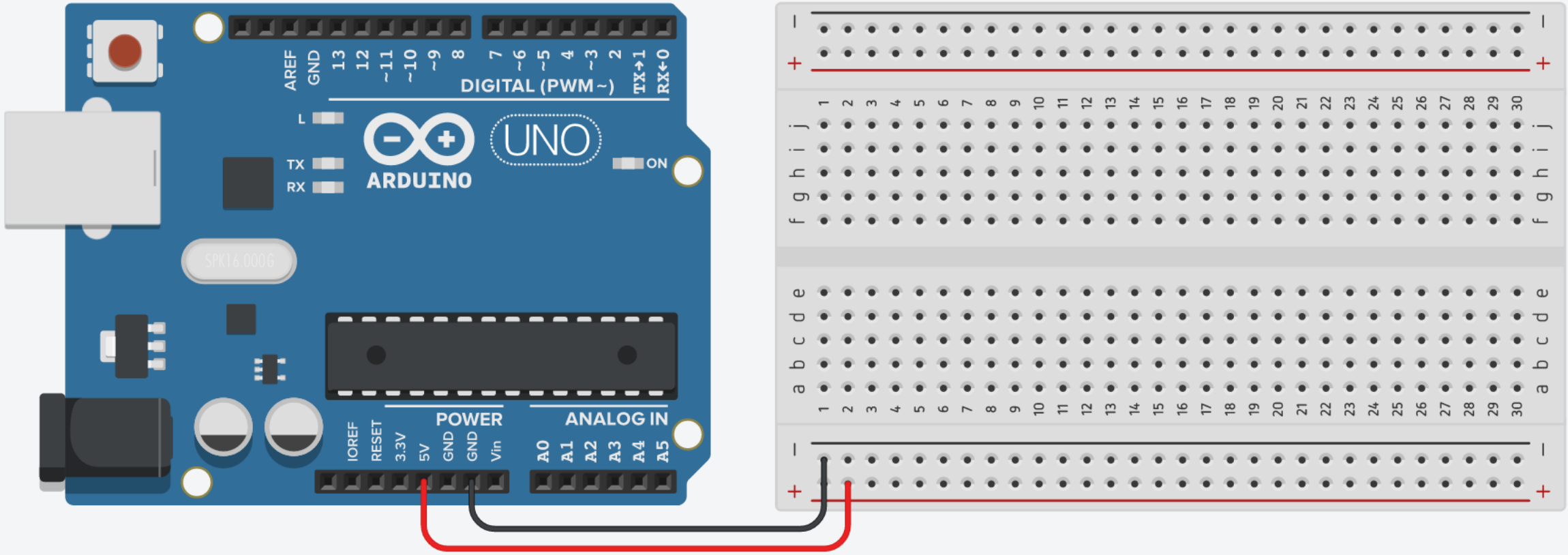
Potentiometer

- A potentiometer is a three-terminal resistor with a rotating contact that forms an adjustable voltage divider.



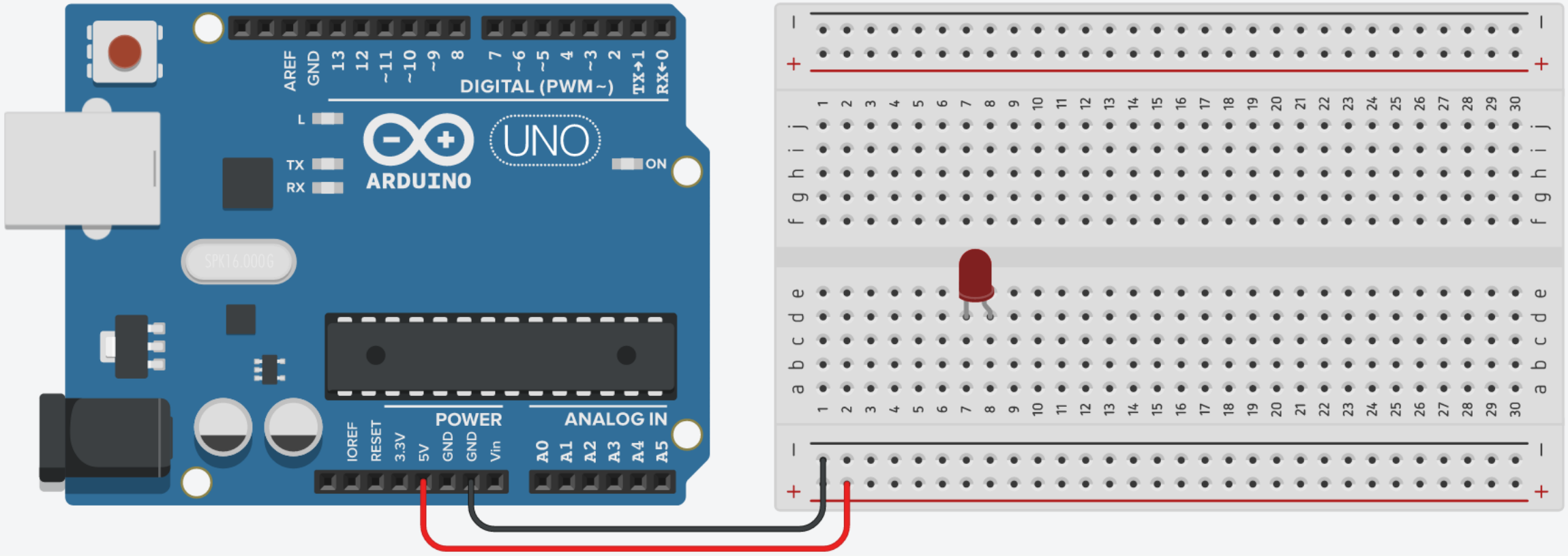
Potentiometer: Steps

1. Connect breadboard power (+) and ground (-) rails to Arduino 5V and ground (GND), respectively.



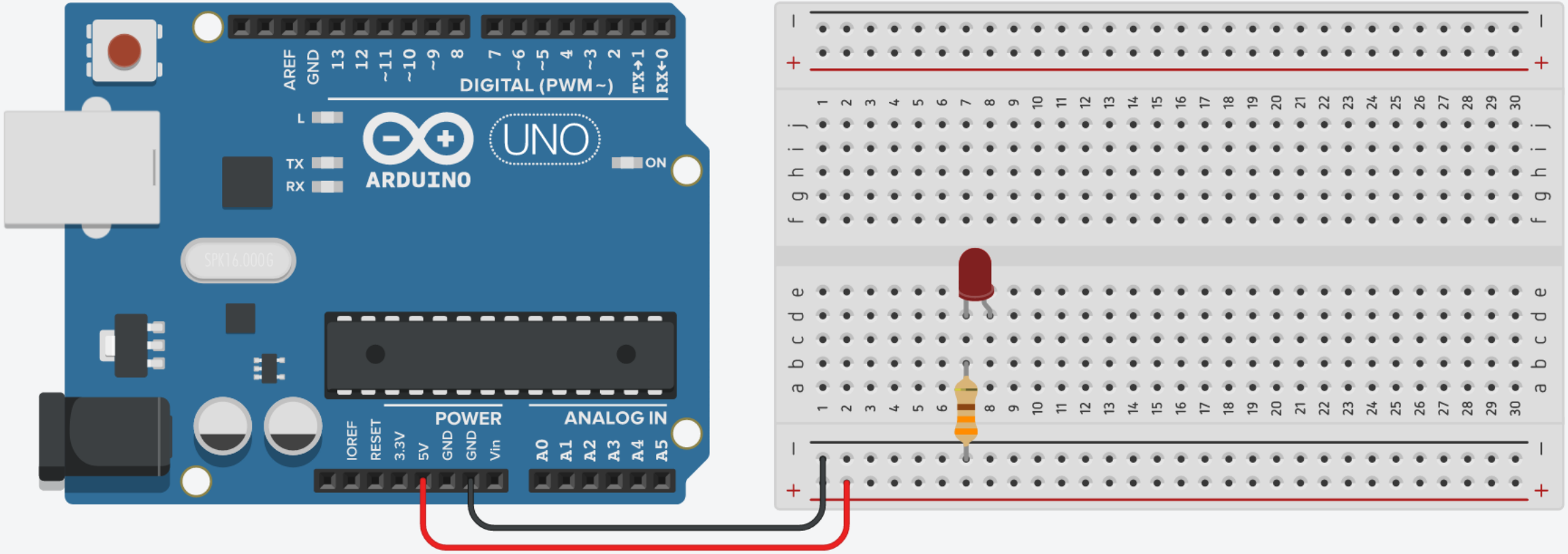
Potentiometer: Steps

2. Plug the LED into two different breadboard rows.



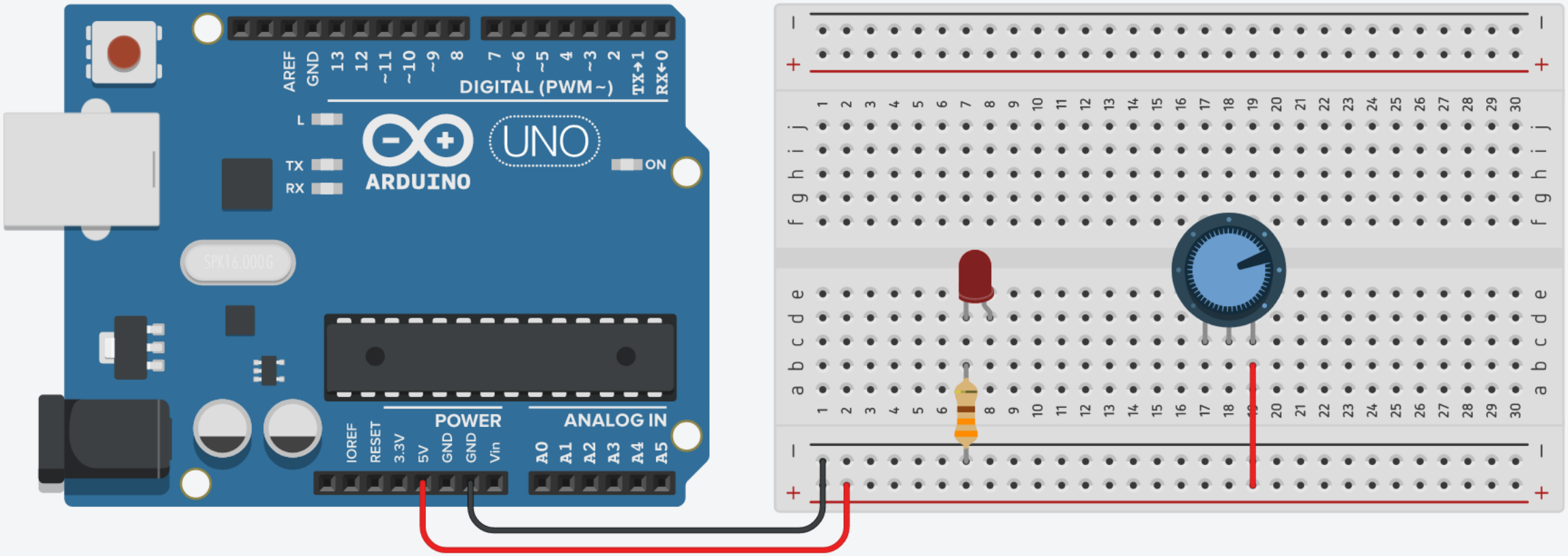
Potentiometer: Steps

3. The cathode (shorter leg) connects to one leg of a resistor of 330Ω, and the other resistor leg to the ground.



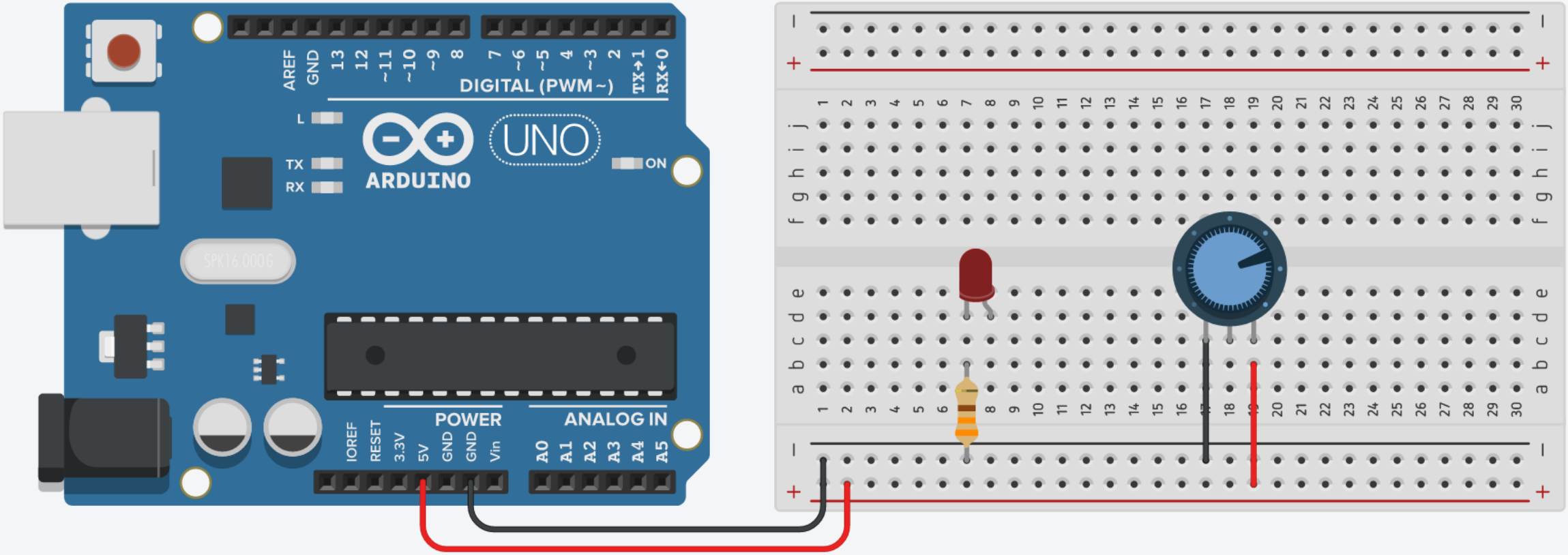
Potentiometer: Steps

4. Connect one terminal of the potentiometer to the 5V on Arduino.



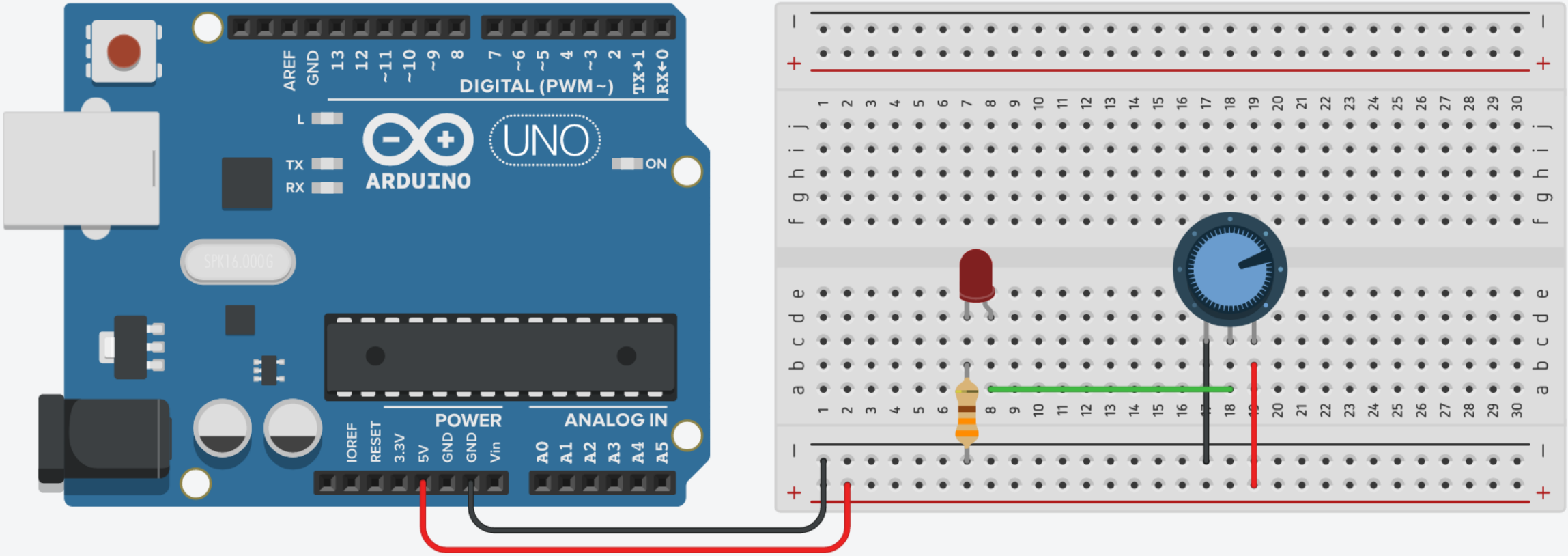
Potentiometer: Steps

5. Connect the other terminal of the potentiometer to the ground.

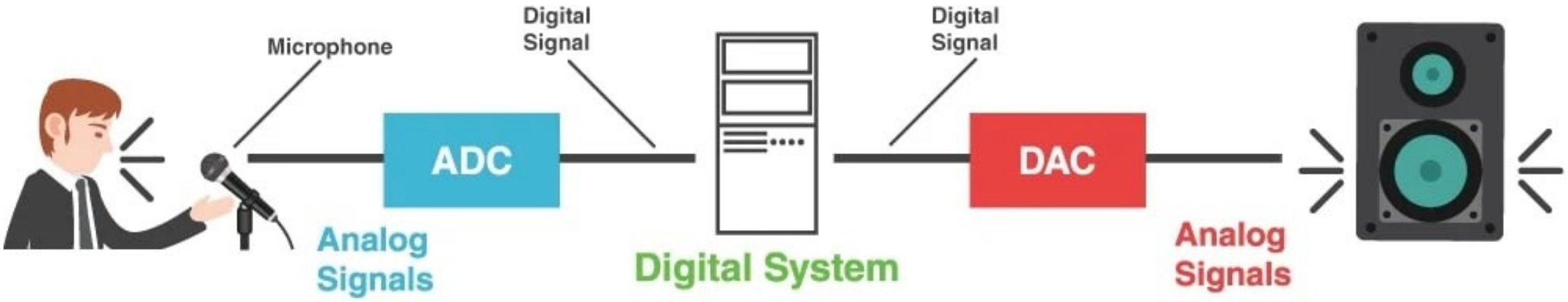


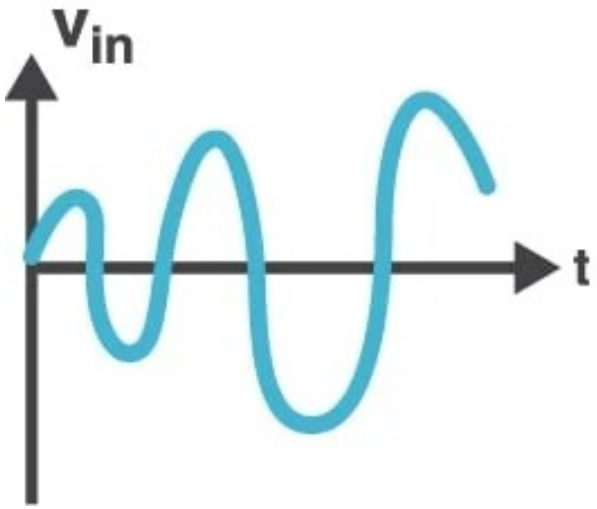
Potentiometer: Steps

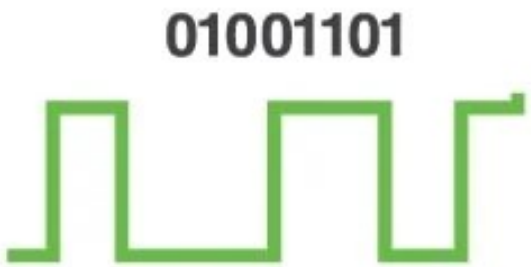
6. Connect the wiper of the potentiometer to anode of the LED.

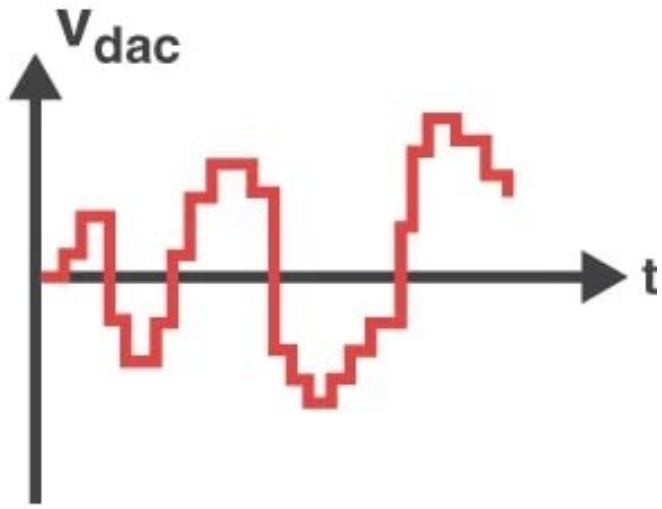


ADC vs. DAC

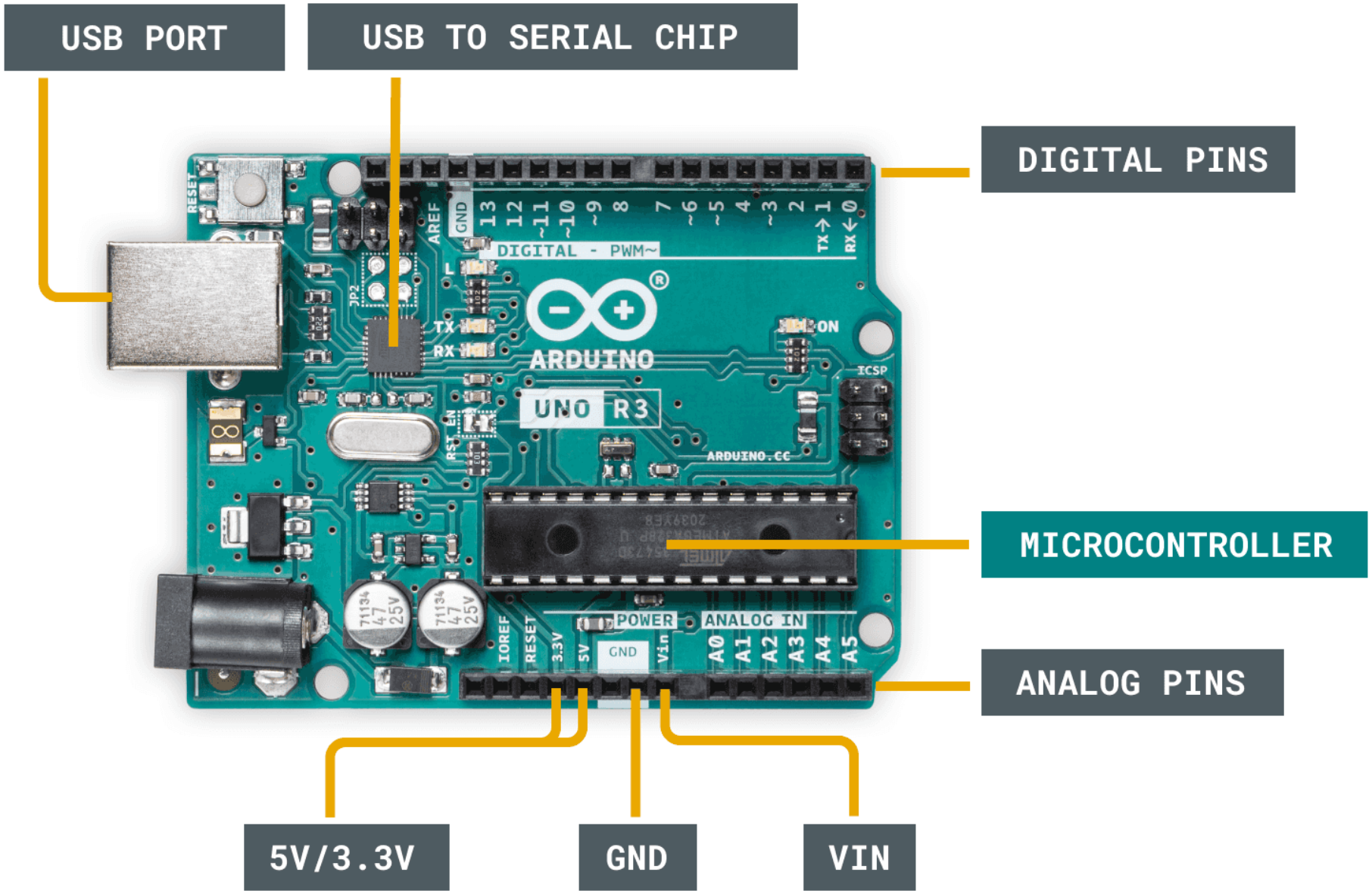

Microphone
Digital Signal
Digital Signal
ADC
DAC
Analog Signals
Digital System
Analog Signals


Vin
t


01001101


Vdac
t

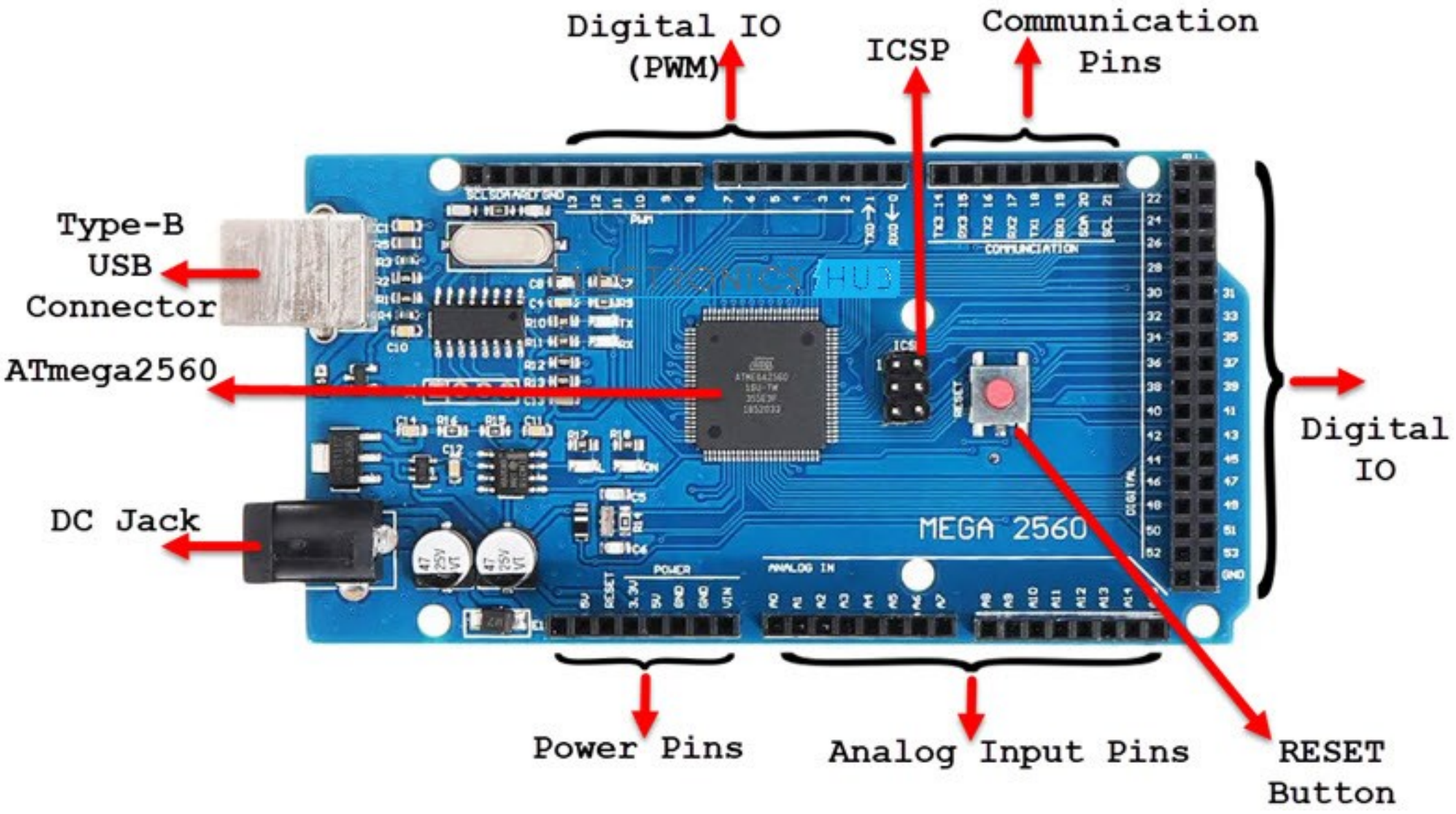
Arduino Uno Board



Arduino Uno Board

Microcontroller	ATmega328P
Clock Speed	16 MHz
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Digital I/O Pins	14 (of which 6 PWM)
PWM Digital I/O Pins	6
Analog Input Pins	6
Operating Voltage	5V
DC Current per I/O Pin	20 mA

Arduino Mega Board

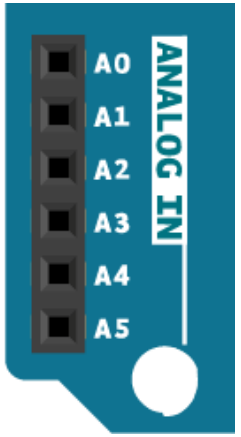


Arduino Mega Board

Microcontroller	ATmega2560
Clock Speed	16 MHz
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Digital I/O Pins	54 (of which 15 PWM)
PWM Digital I/O Pins	15
Analog Input Pins	16
Operating Voltage	5V
DC Current per I/O Pin	20 mA

Analog Inputs

- When we interface sensors to the microcontroller, the output of the sensor many of the times is analog in nature.
- But microcontroller processes digital signals.
- Hence, we use ADC in between sensor and microcontroller.
- The ADC converts an analog signal into digital and gives it to the microcontroller.



Read Analog Voltage

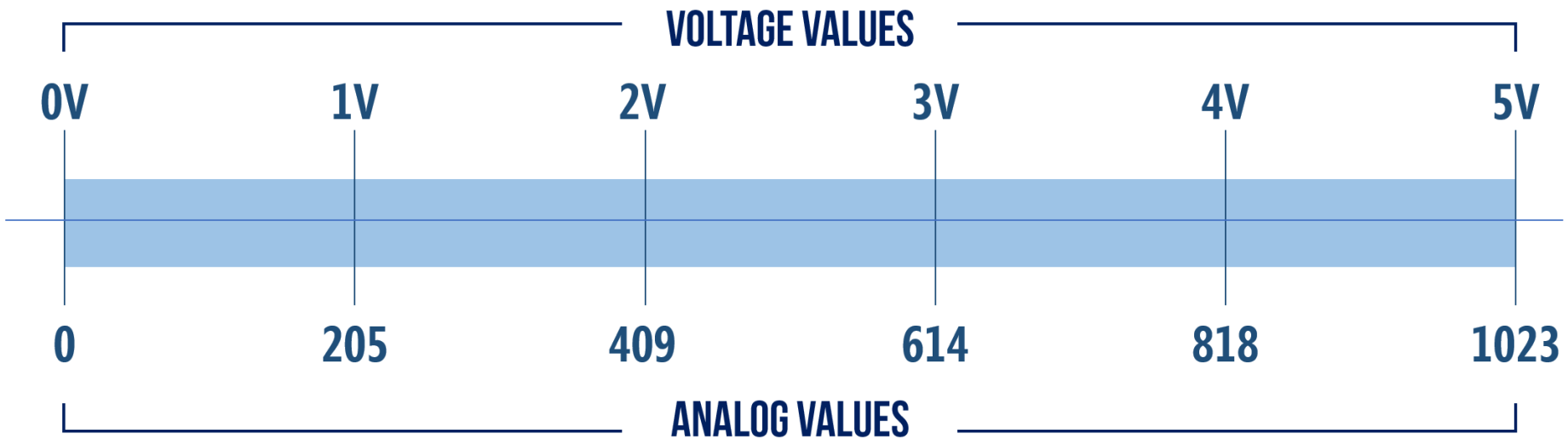
- The microcontroller of the board has a circuit inside called an analog-to-digital converter (ADC) that reads this changing voltage and converts it to a number between 0 and 1023.
- The ADC in Arduino is 10-bit.

0	0	0	0	0	0	0	0	0	0	0V
1	1	1	1	1	1	1	1	1	1	5V

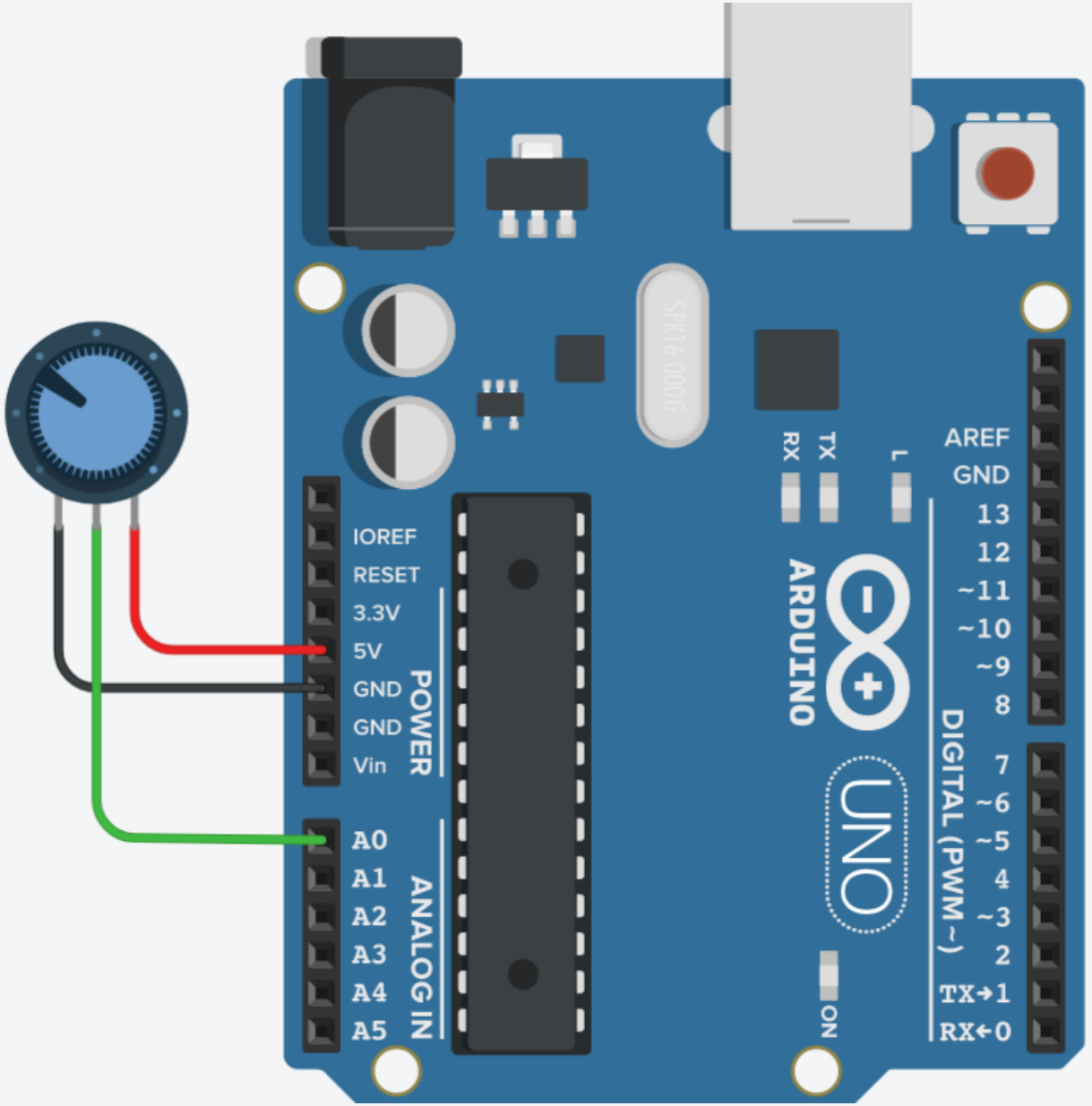
Read Analog Voltage

- The `analogRead()` returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.
- To scale the numbers between 0 and 5, divide 5 by 1023 and multiply that by `sensorValue` :

```
voltage = sensorValue * (5.0 / 1023.0);
```



Potentiometer as Analog Input: Circuit

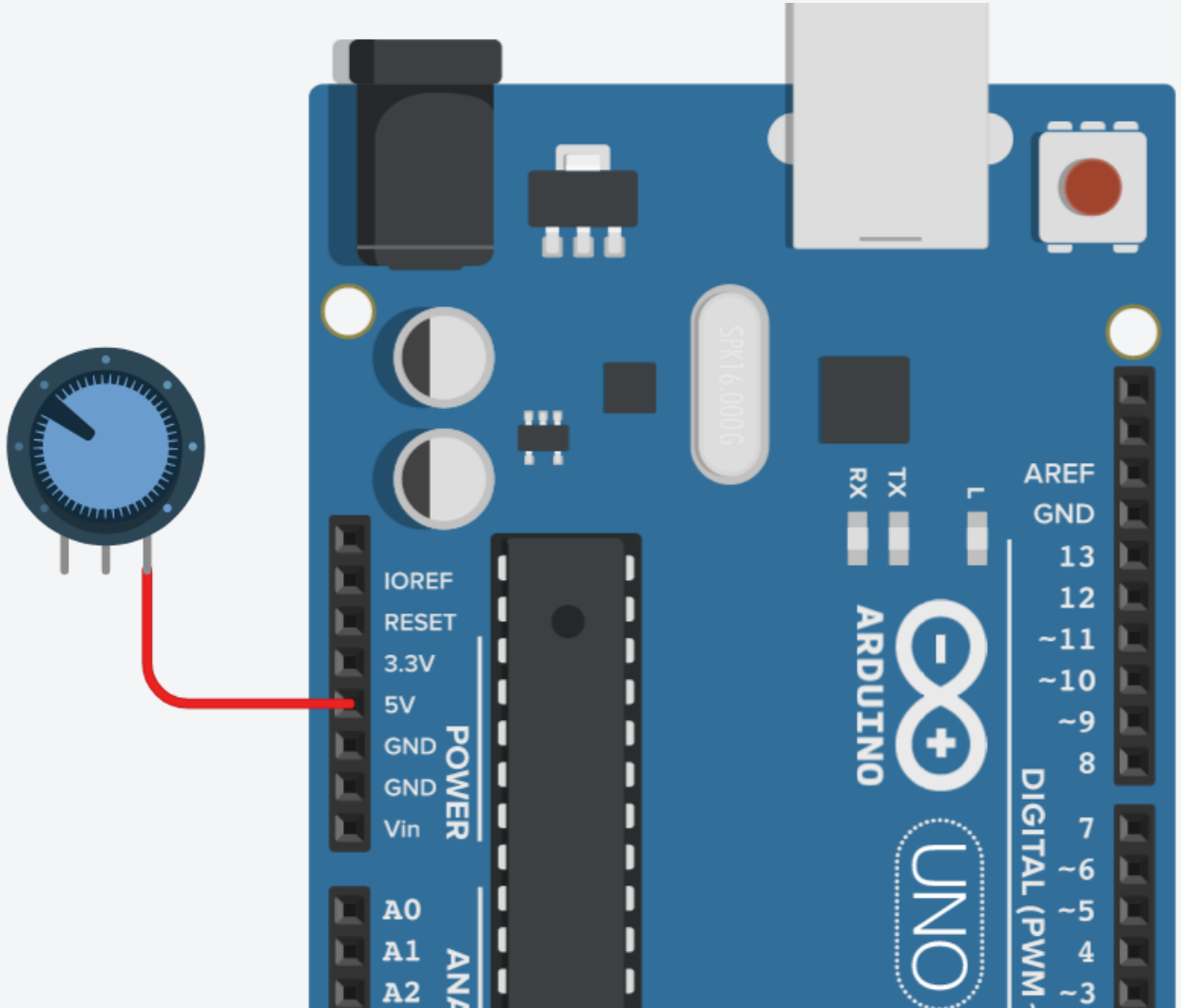


Potentiometer as Analog Input: Components

- We need
  - Arduino
  - Potentiometer
  - Jumpers

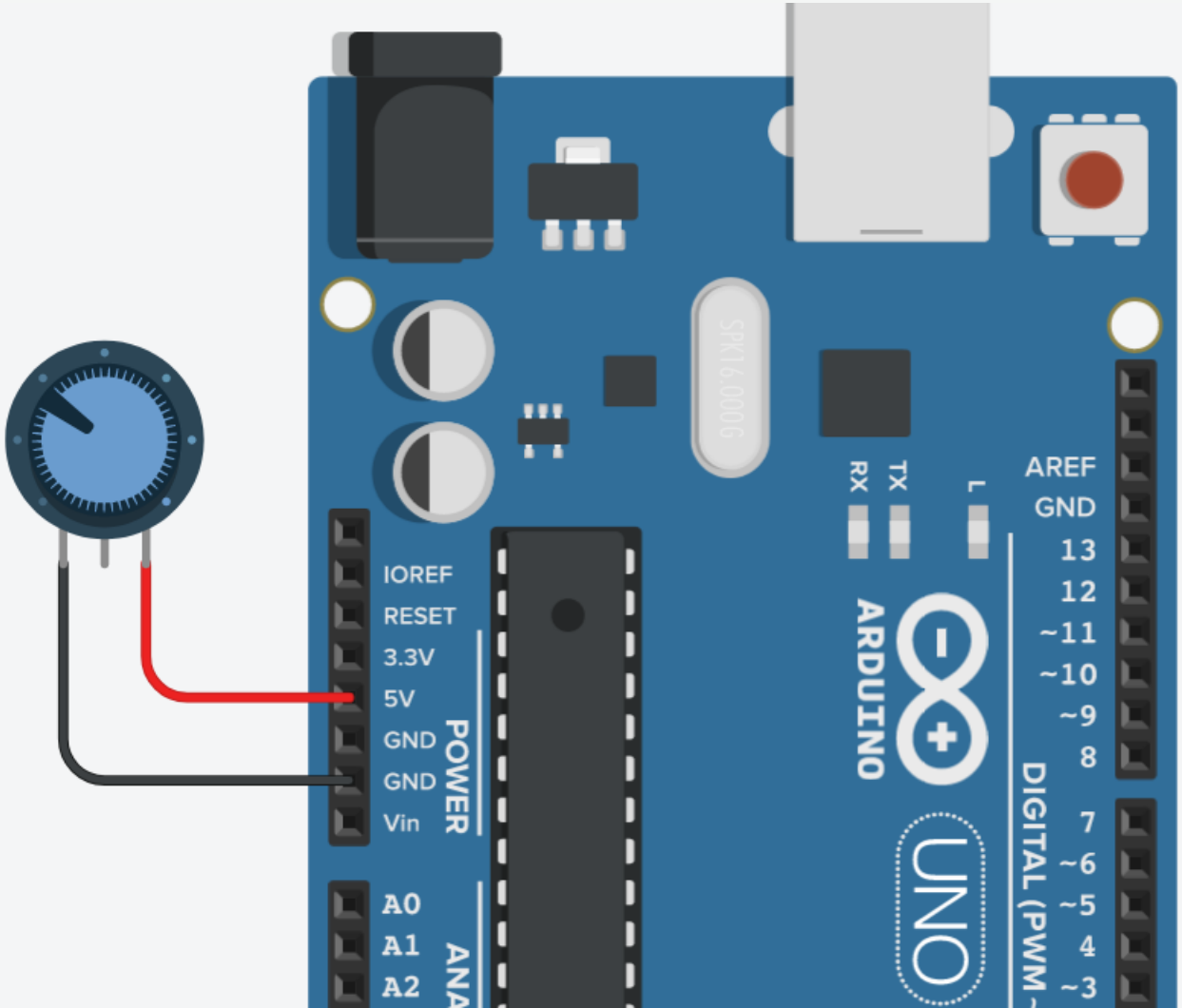
Potentiometer as Analog Input: Steps

1. Connect one terminal of the potentiometer to the 5V on Arduino.



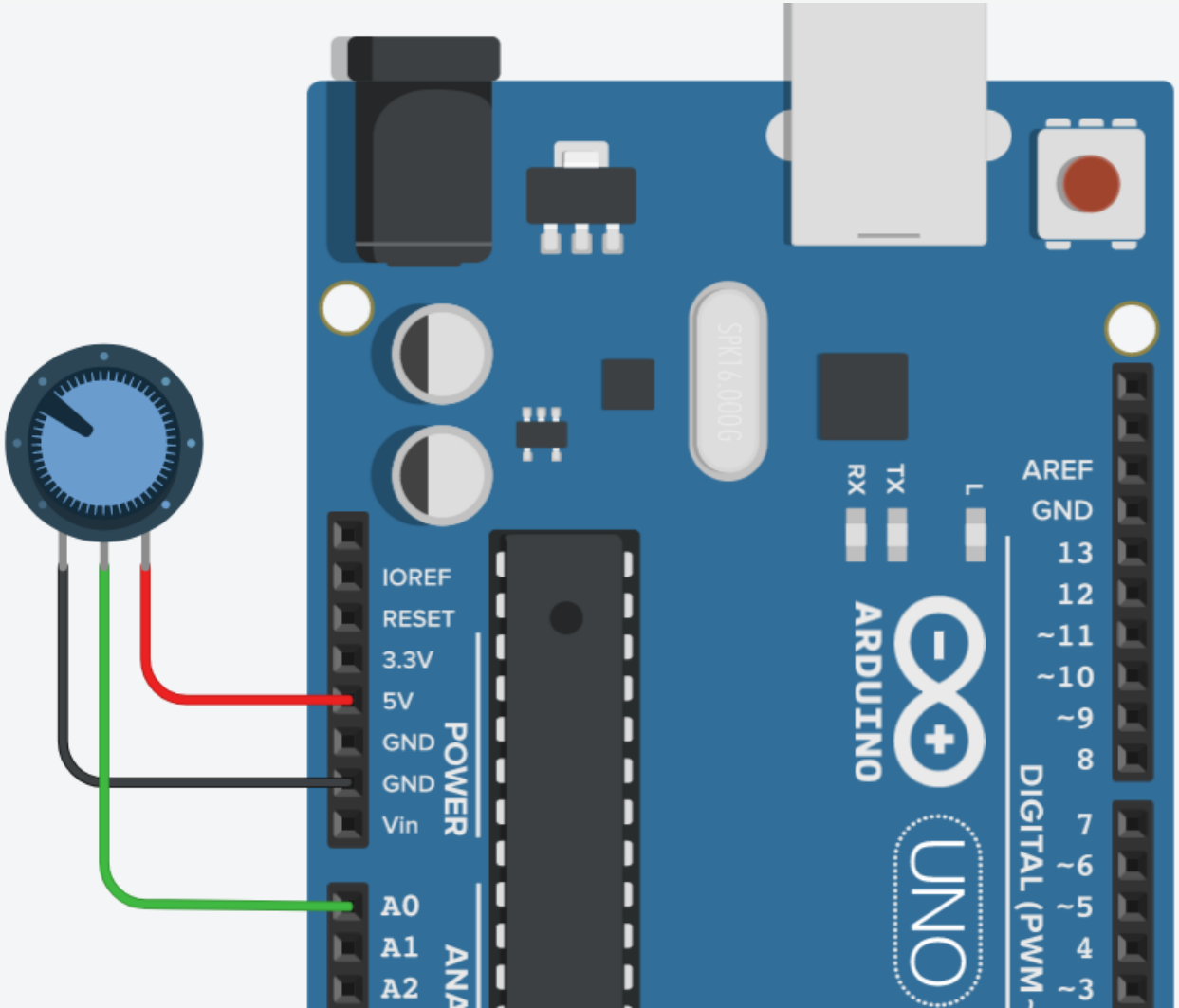
Potentiometer as Analog Input: Steps

2. Connect the other terminal of the potentiometer to the ground.



Potentiometer as Analog Input: Steps

3. Connect the wiper of the potentiometer to the analog input A0.



Potentiometer as Analog Input: Code

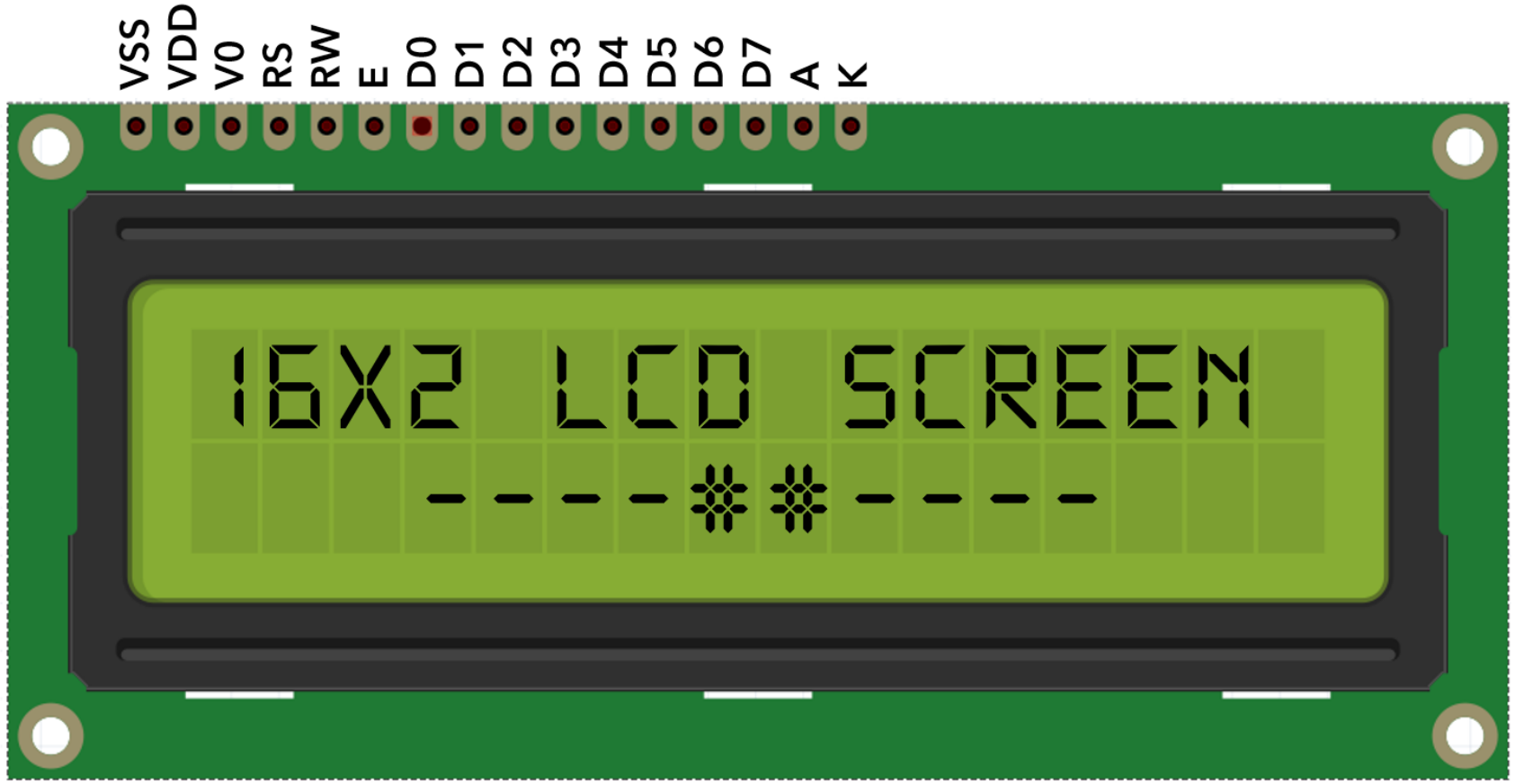
```
int value;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  value = analogRead(A0);
  Serial.println(value);

  delay(100);
}
```

Potentiometer as Analog Input: Code

```
int value;
float volt;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  value = analogRead(A0);
  volt = value * (5.0 / 1023.0);
  Serial.println(volt);

  delay(100);
}
```

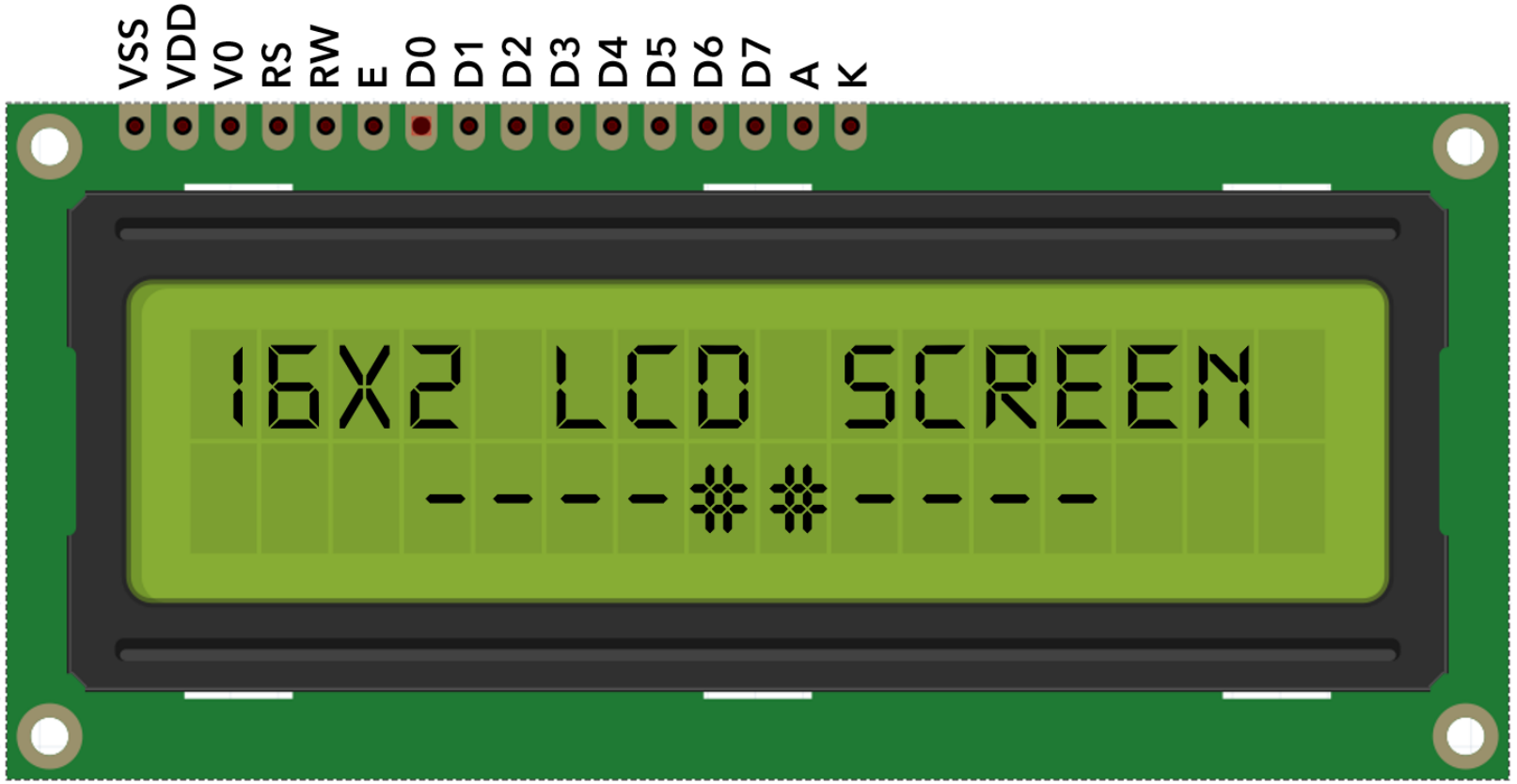
16×2 LCD

- Why is it called 16×2 ?
- You can write 16 characters in column- wise and two in row-wise.



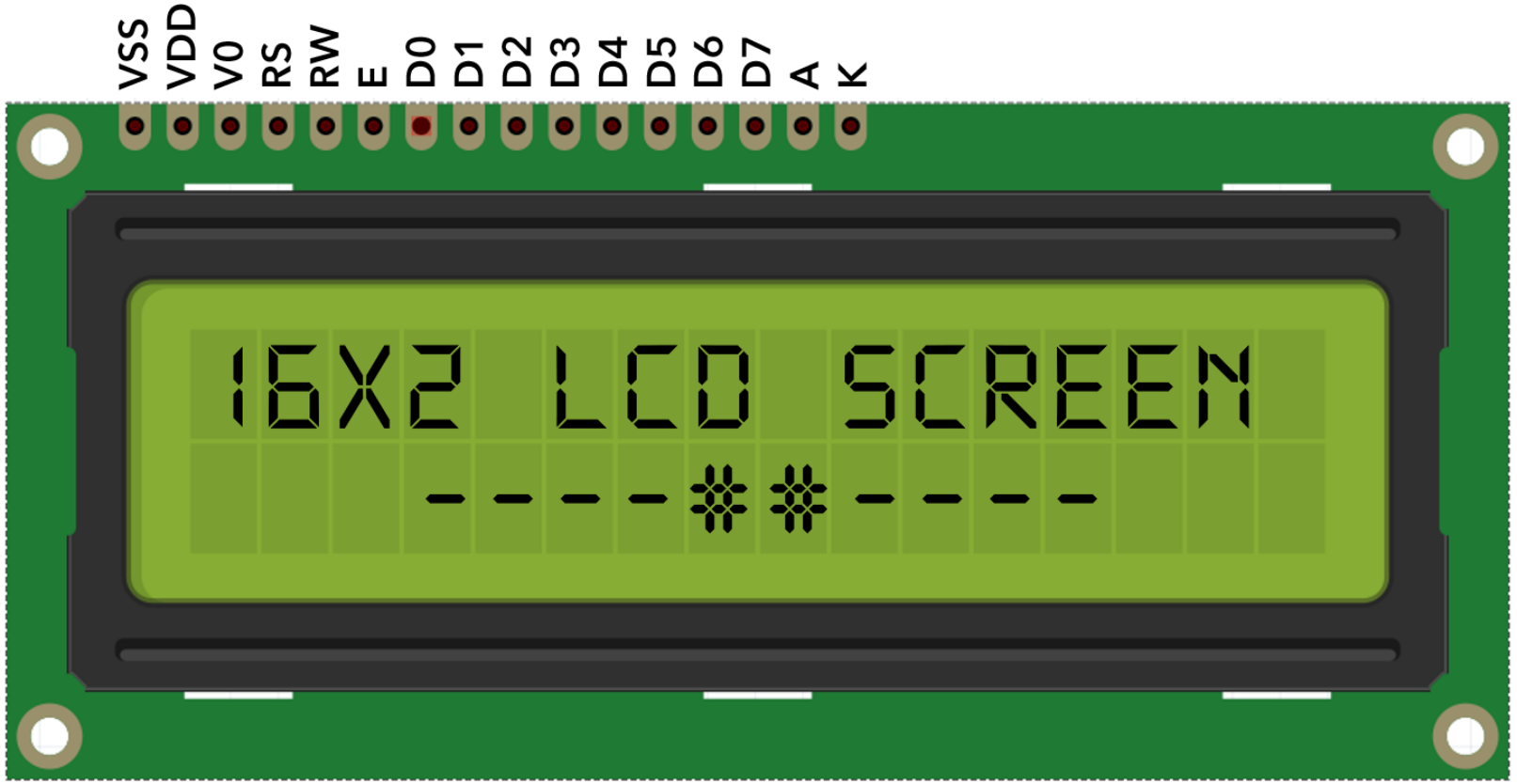
16×2 LCD: VSS Pin

- Connect the VSS (GND) pin to the ground of the power supply.



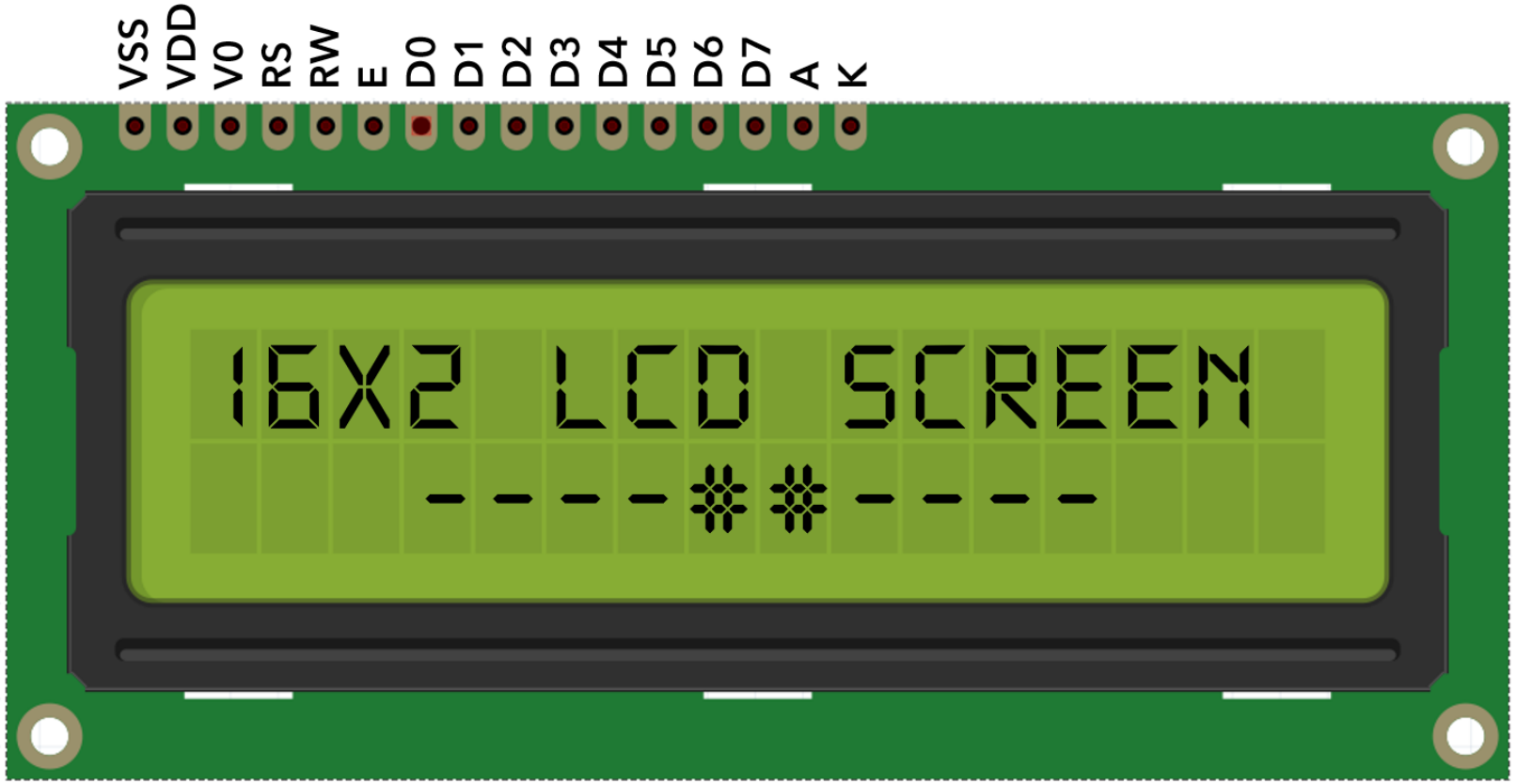
16×2 LCD: VDD Pin

- Connect the VDD pin to the 5V of the power supply.



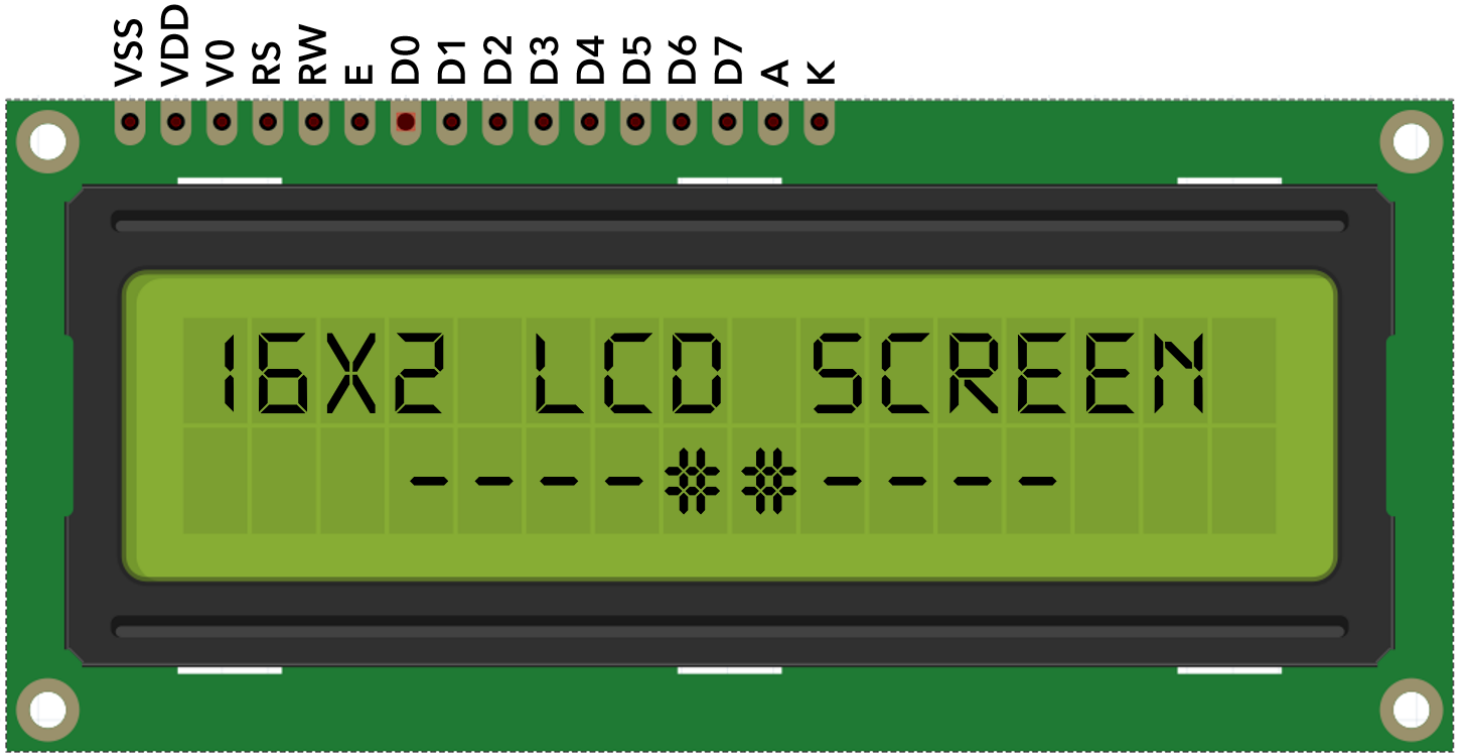
16×2 LCD: VO (Contrast) Pin

- This pin is used to adjust the contrast of the Display.
- Connect the wiper of the POT to this pin.



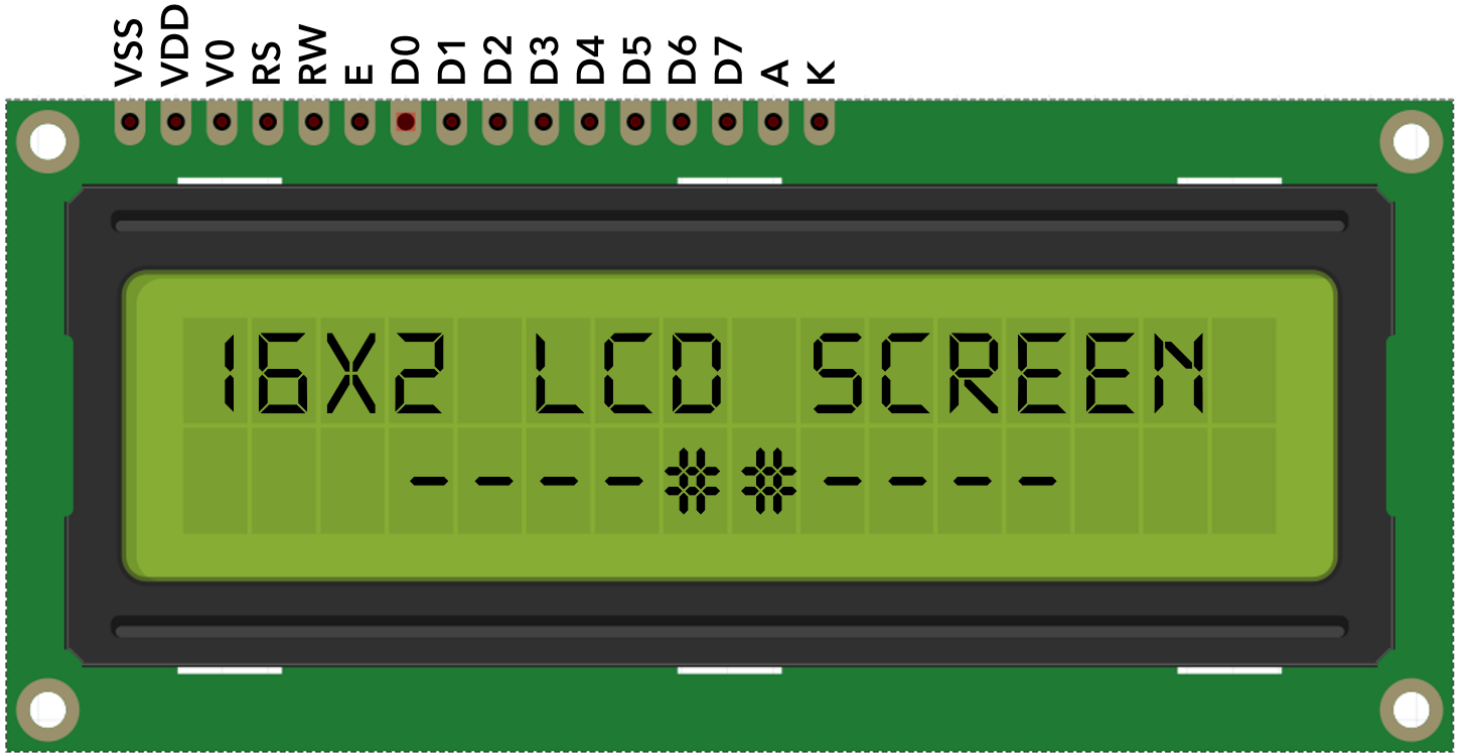
16×2 LCD: RS Pin

- The RS pin is the Register select pin.
- Selects command register when the pin is LOW.
- Selects data register when this pin is HIGH.



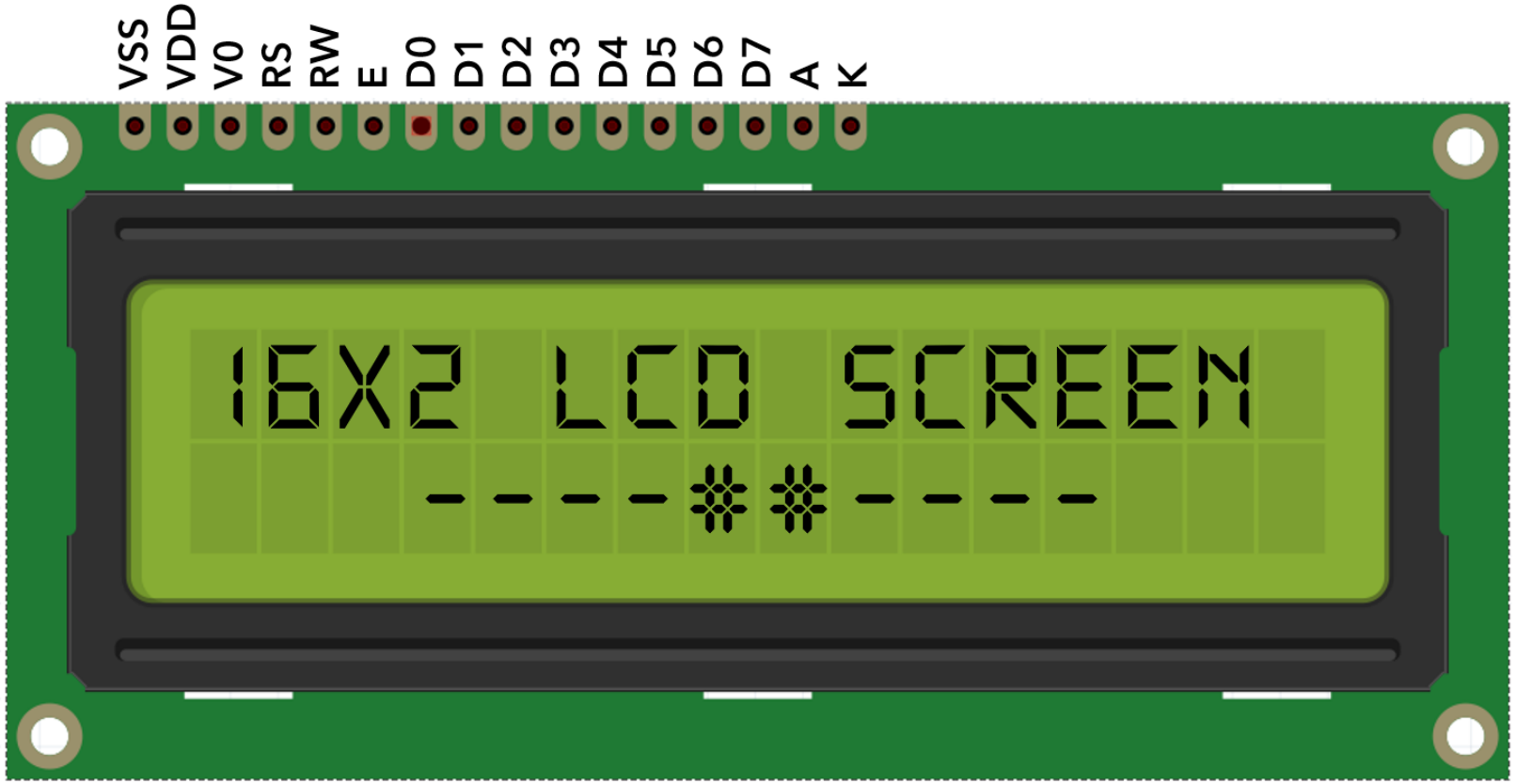
16×2 LCD: RW Pin

- It represents the Read/Write pin.
- When this pin is LOW, the MCU writes to register.
- When the pin is HIGH, MCU reads from the register.



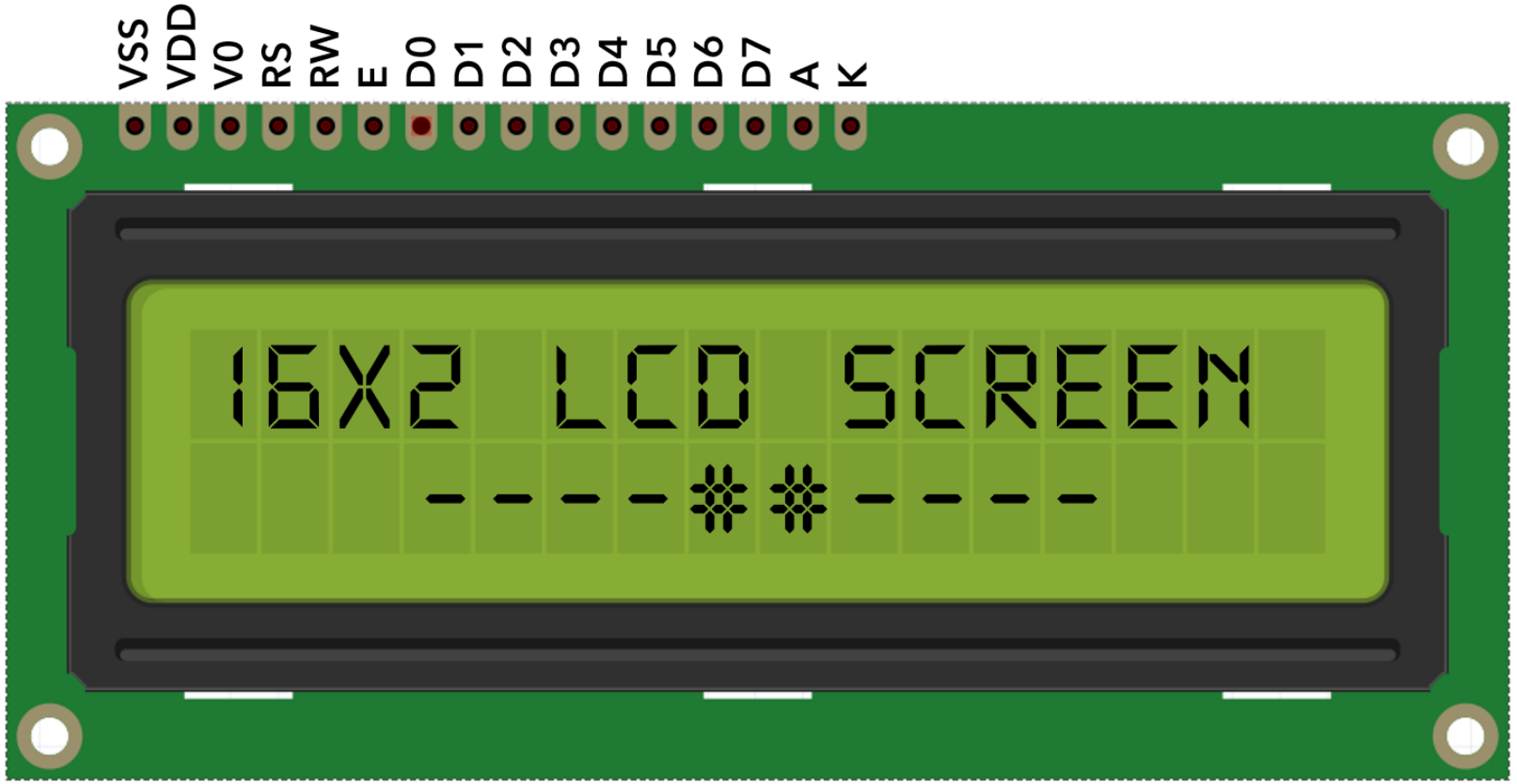
16×2 LCD:EN Pin

- The EN pin means the Enable pin.
- Send data to data pins when a HIGH to LOW pulse is given.



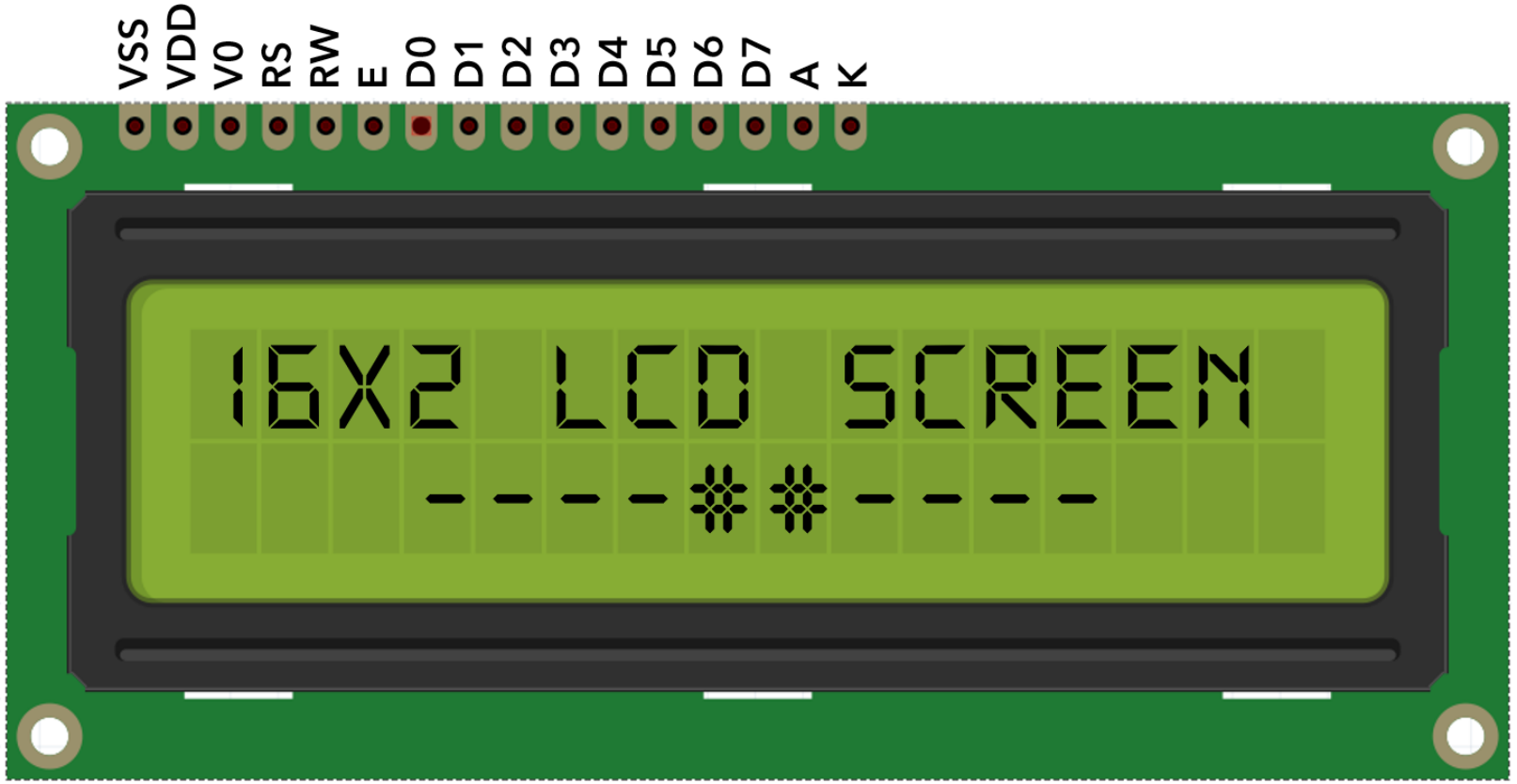
16×2 LCD: D0-D7 Pins

- These D0-D7 pins are 8 data pins.



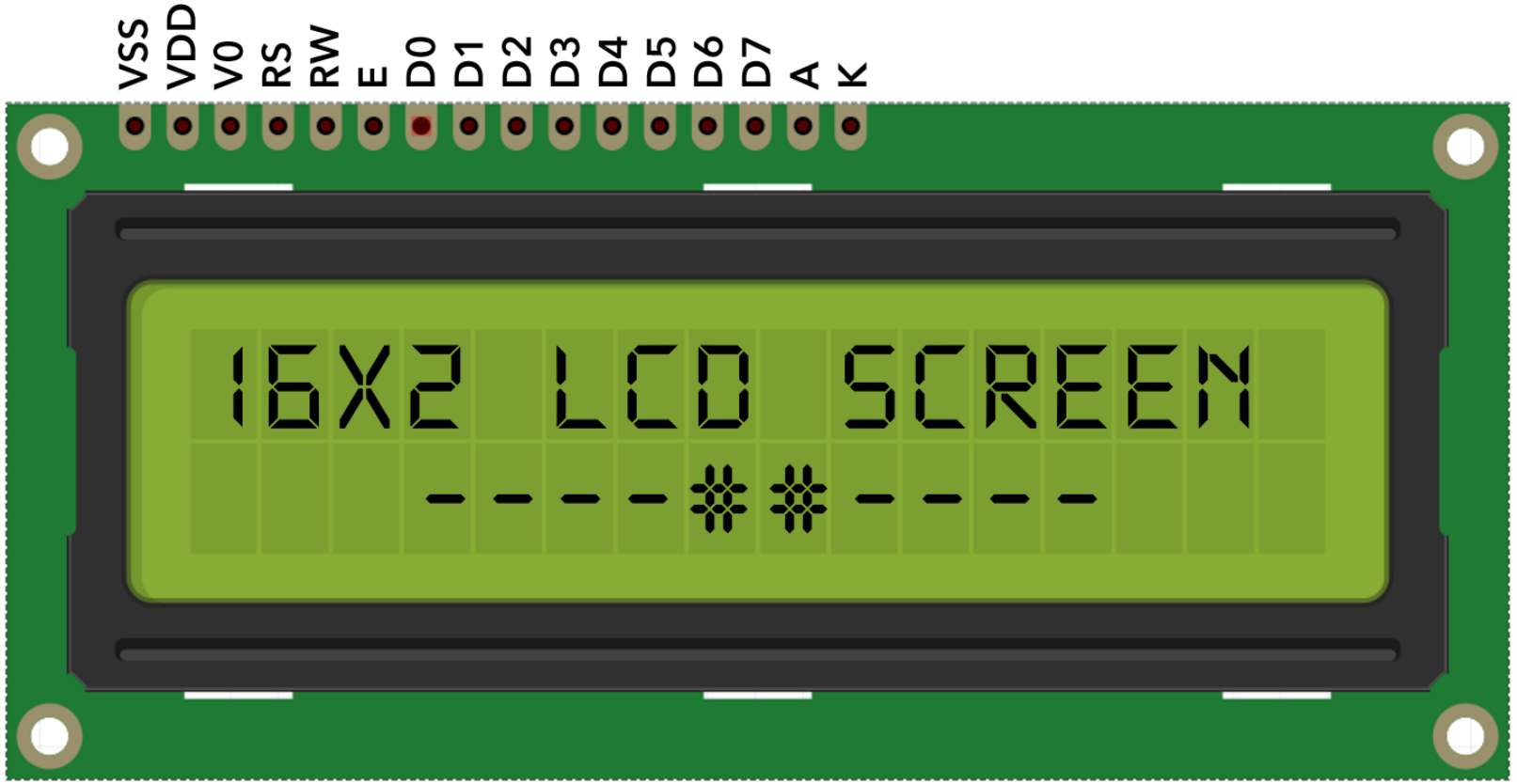
16×2 LCD: A Pin

- The A (Anode) pin connects to the 5V.
- This is the anode pin of the backlight of the display.

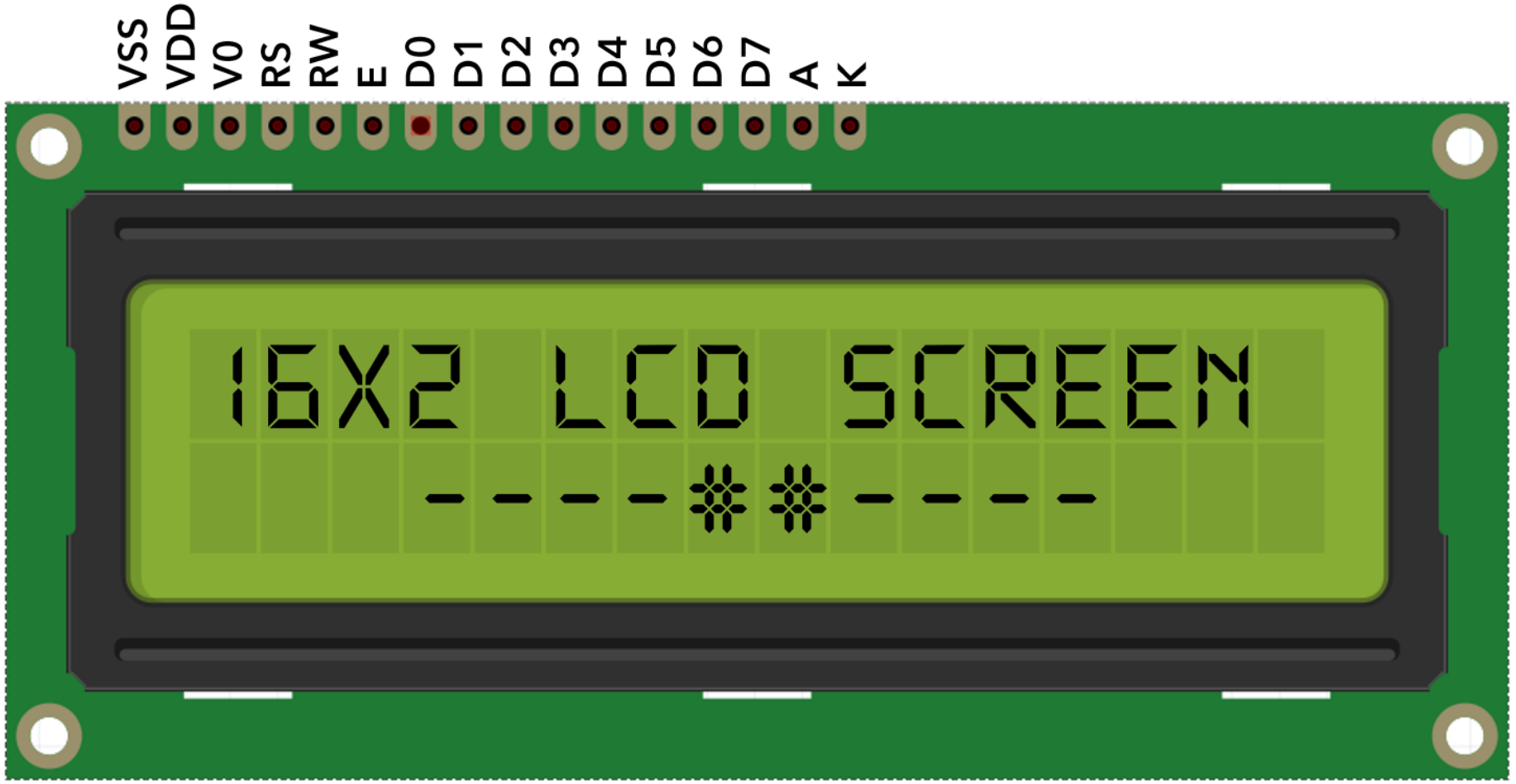


16×2 LCD: K Pin

- The K (Cathode) pin connects to the ground.
- This is the cathode pin of the backlight of the display.



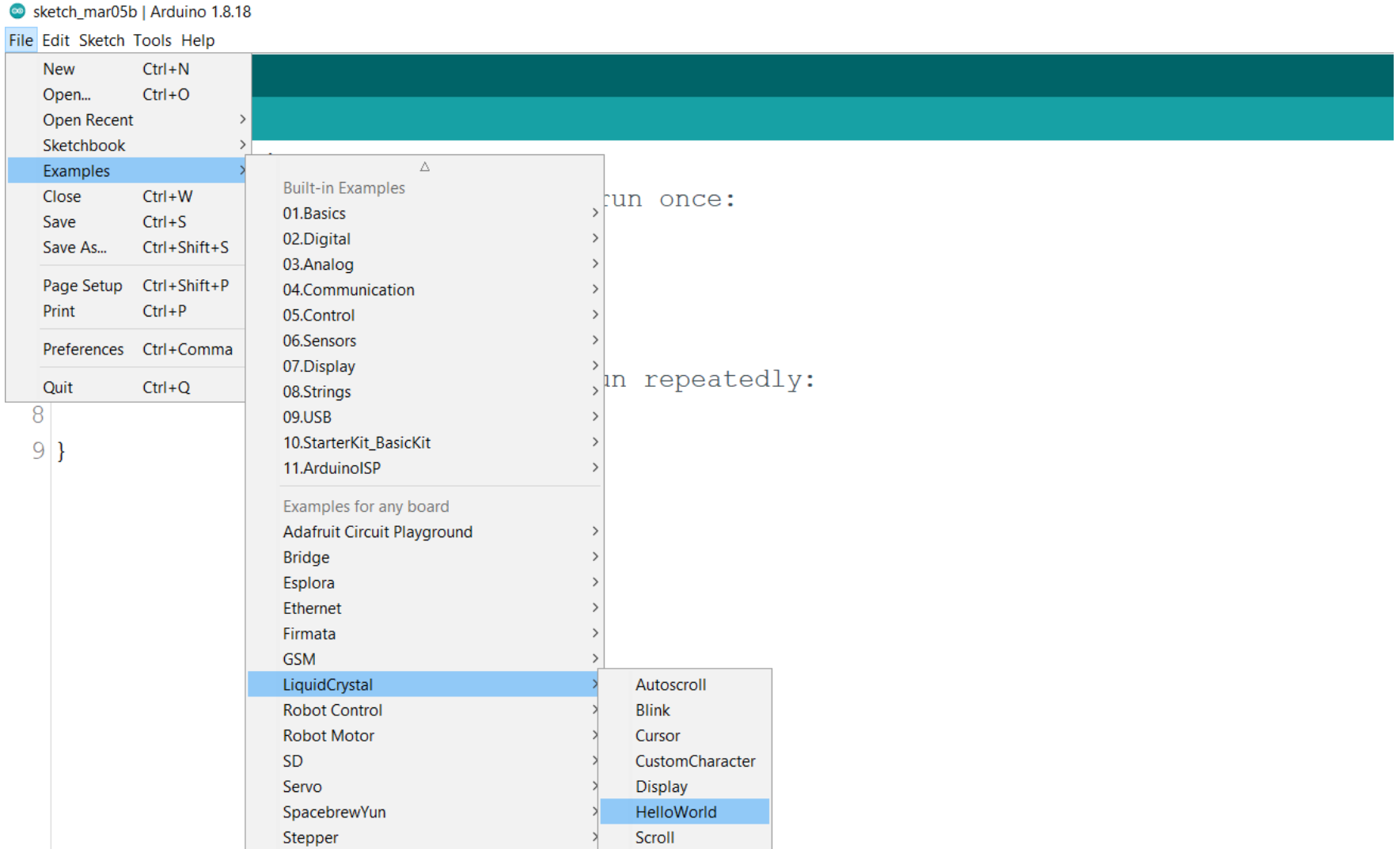
16×2 LCD: Pins Summary



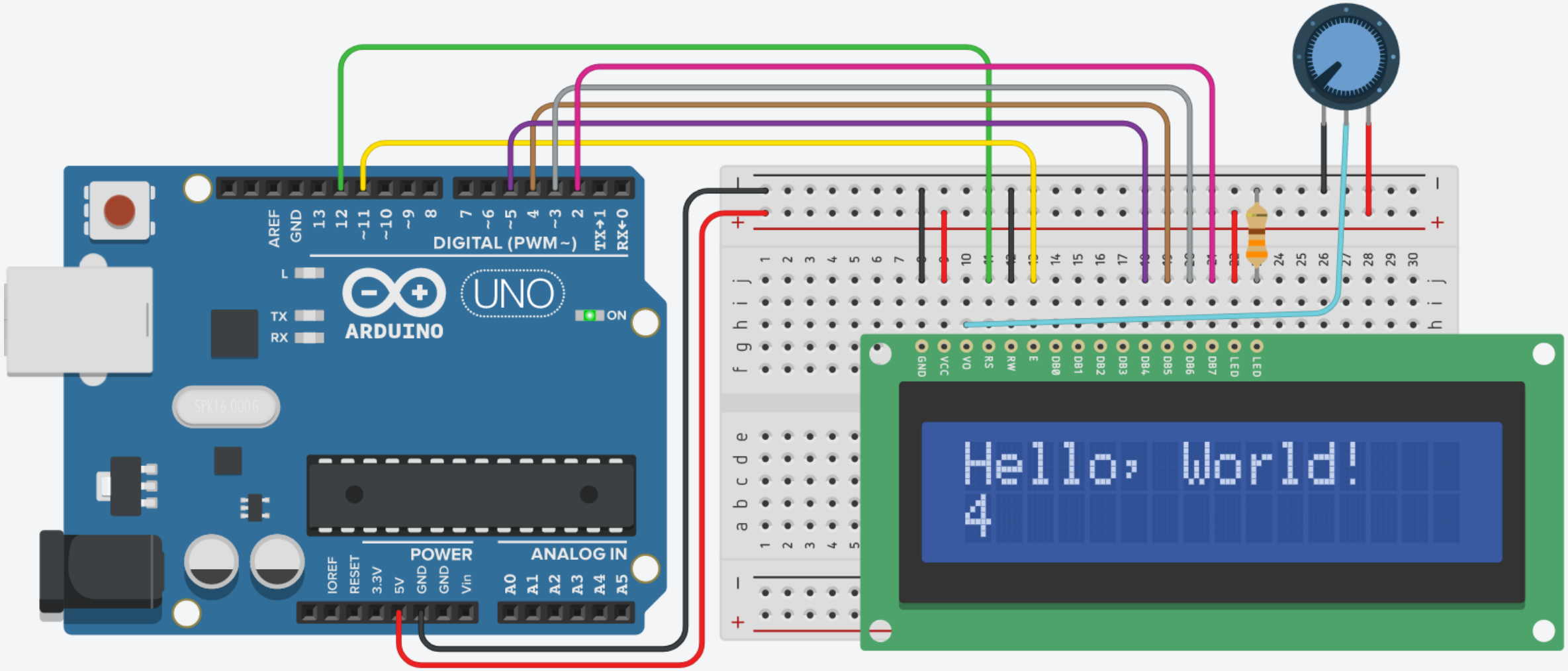
No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

16×2 LCD: Liquid Crystal Library

- Go to File → Examples → LiquidCrystal → HelloWorld

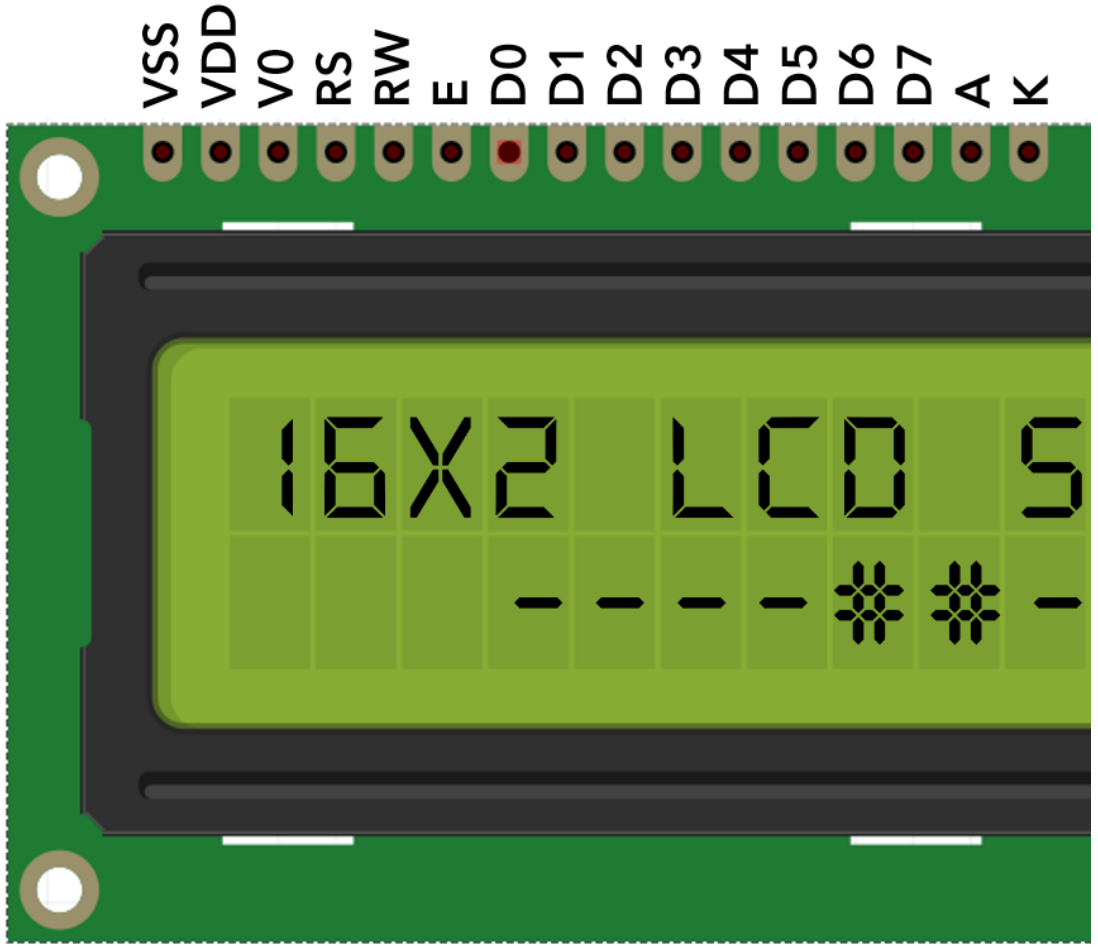


16×2 LCD: Circuit



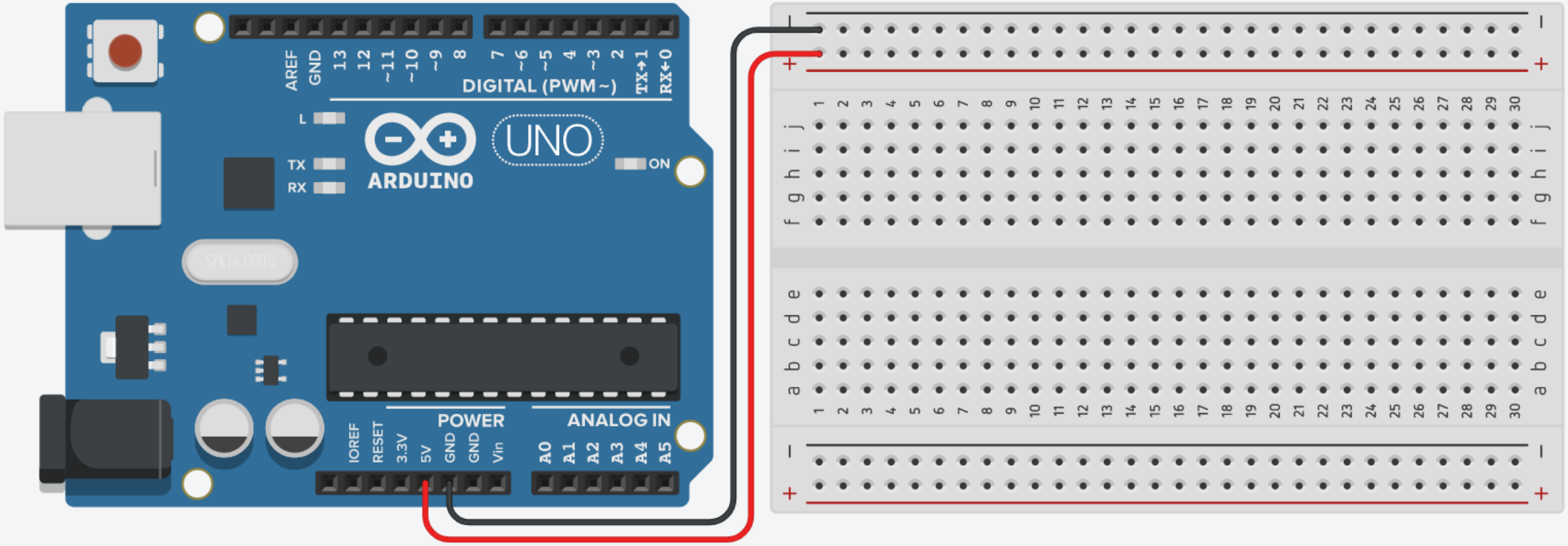
16×2 LCD: Connections

- LCD VSS pin to ground
- LCD VCC pin to 5V
- LCD VO pin to POT wiper
- LCD RS pin to digital pin 12
- LCD R/W pin to ground (write mode)
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2
- LCD A pin to 5V
- LCD K pin to ground



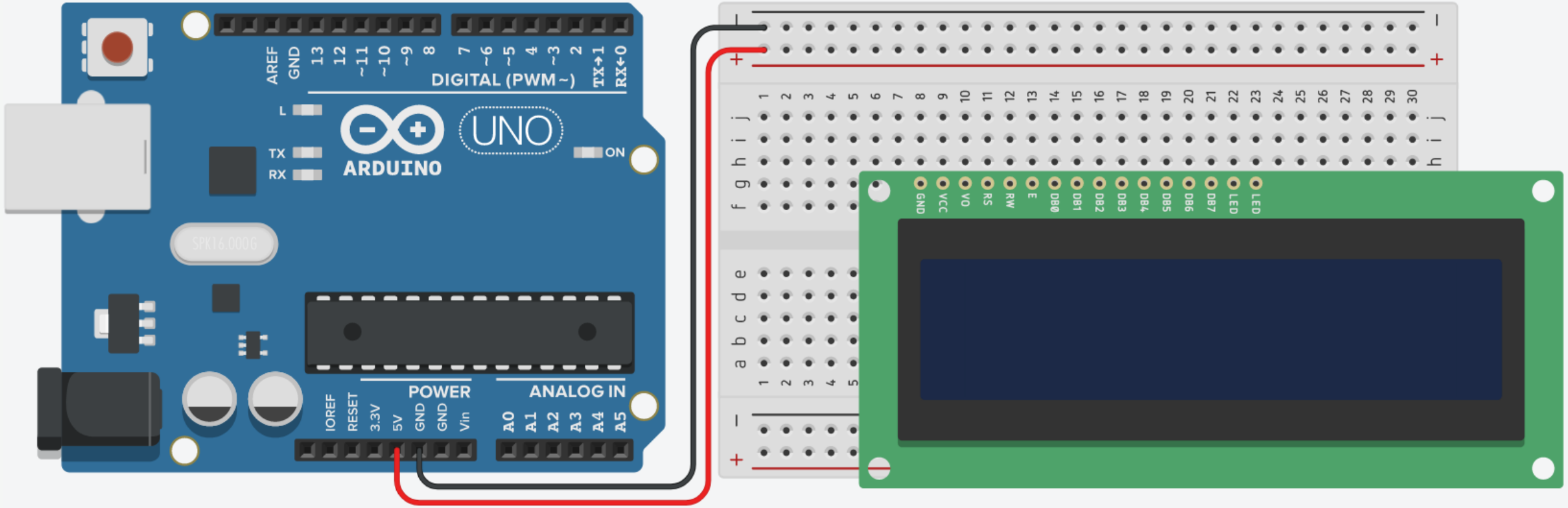
16×2 LCD: Steps

1. Connect breadboard power (+) and ground (-) rails to Arduino 5V and ground (GND), respectively.



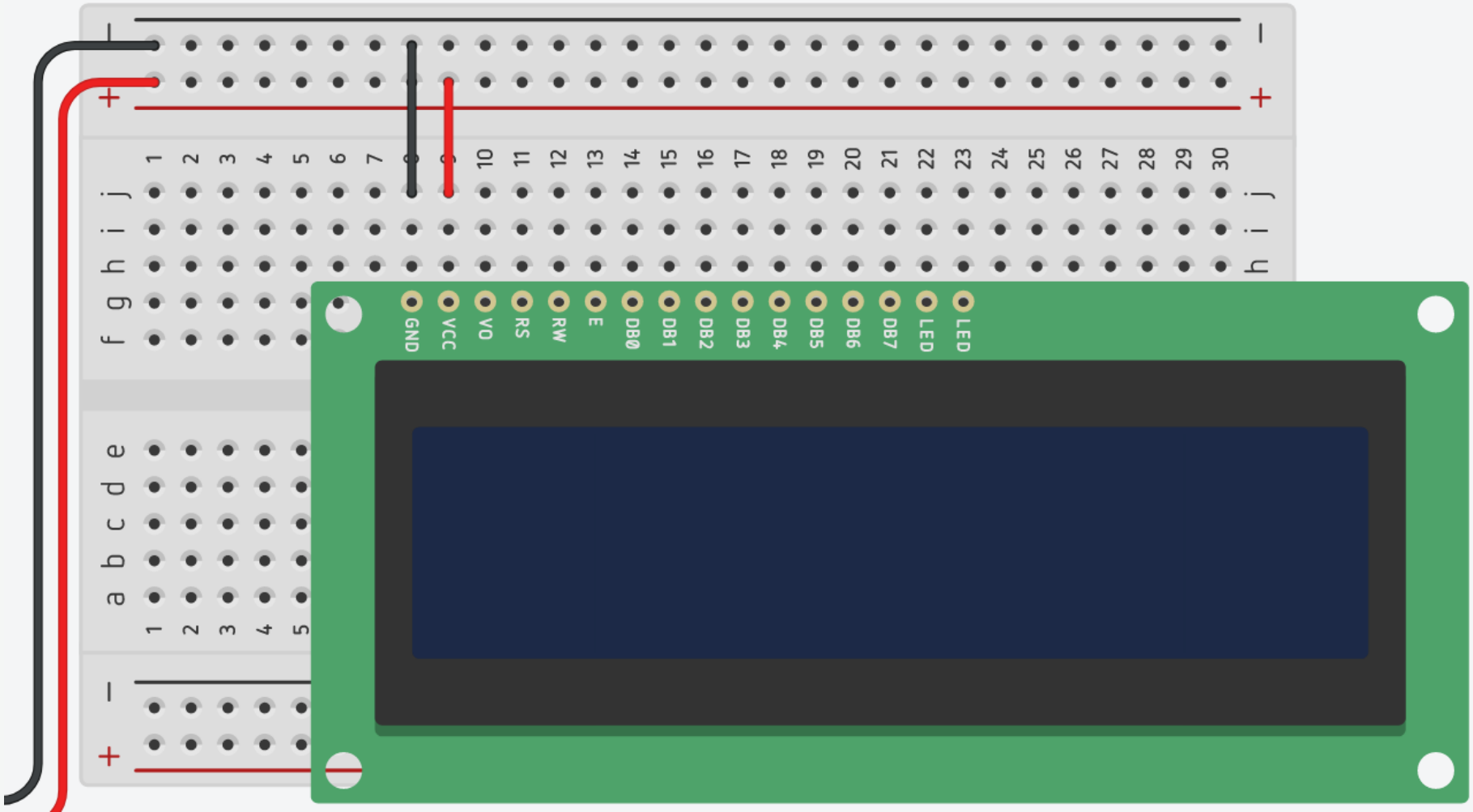
16×2 LCD: Steps

2. Plug the LCD into the breadboard.



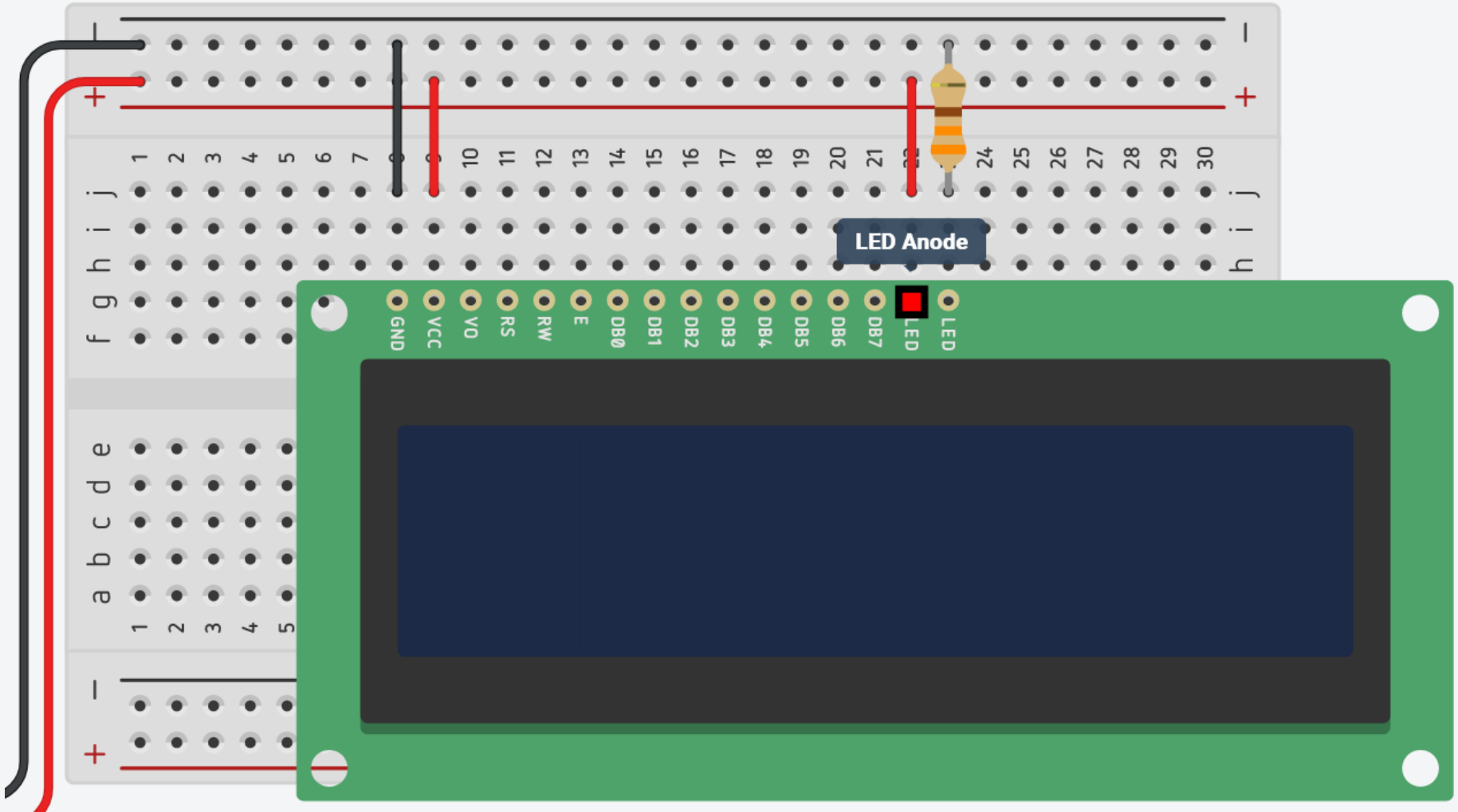
16×2 LCD: Steps

3. Connect the LCD VSS pin to ground, and LCD VCC pin to 5V.



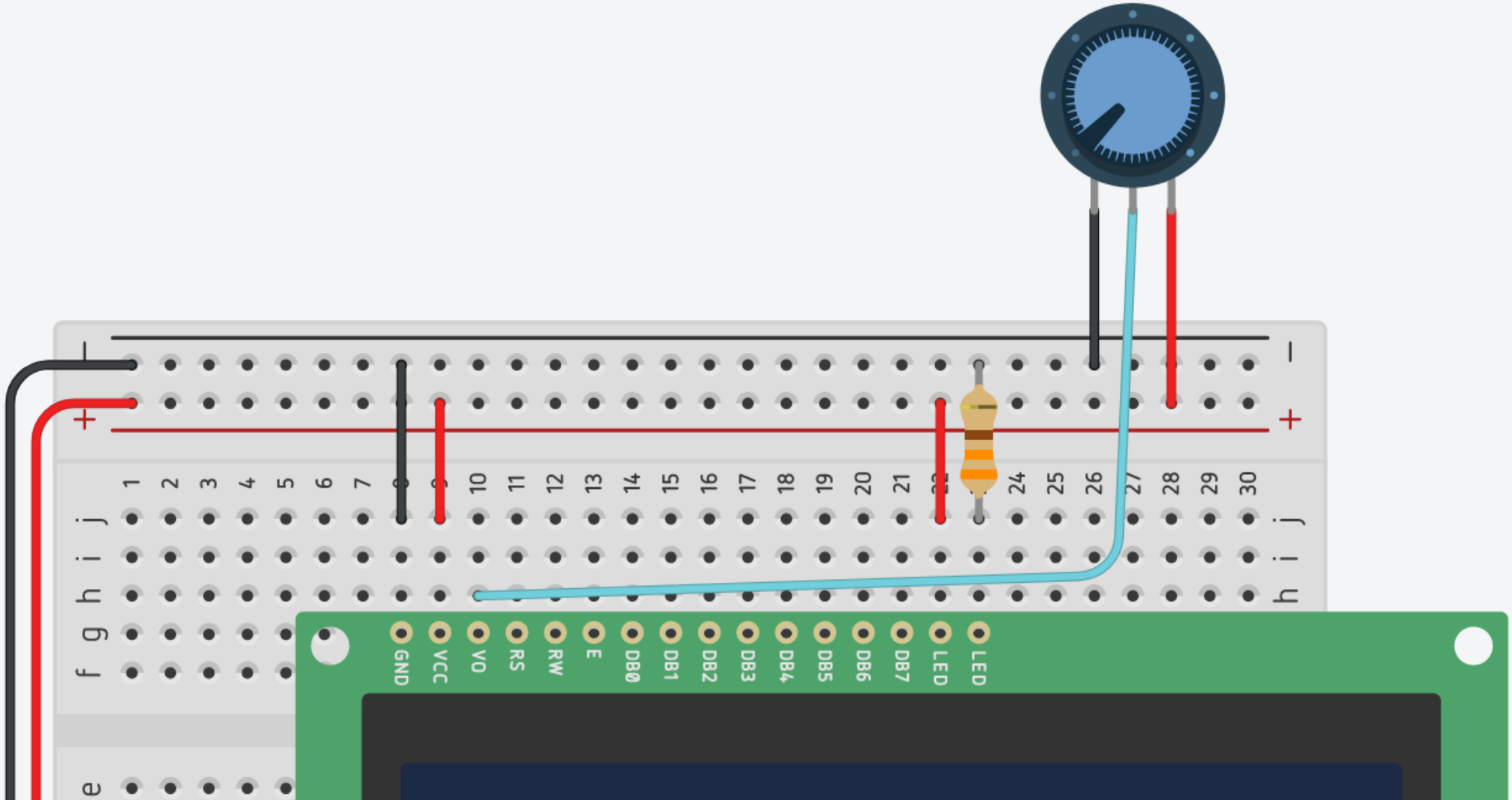
16×2 LCD: Steps

4. LCD A pin to 5V, and LCD K pin to ground using a 330Ω resistor.



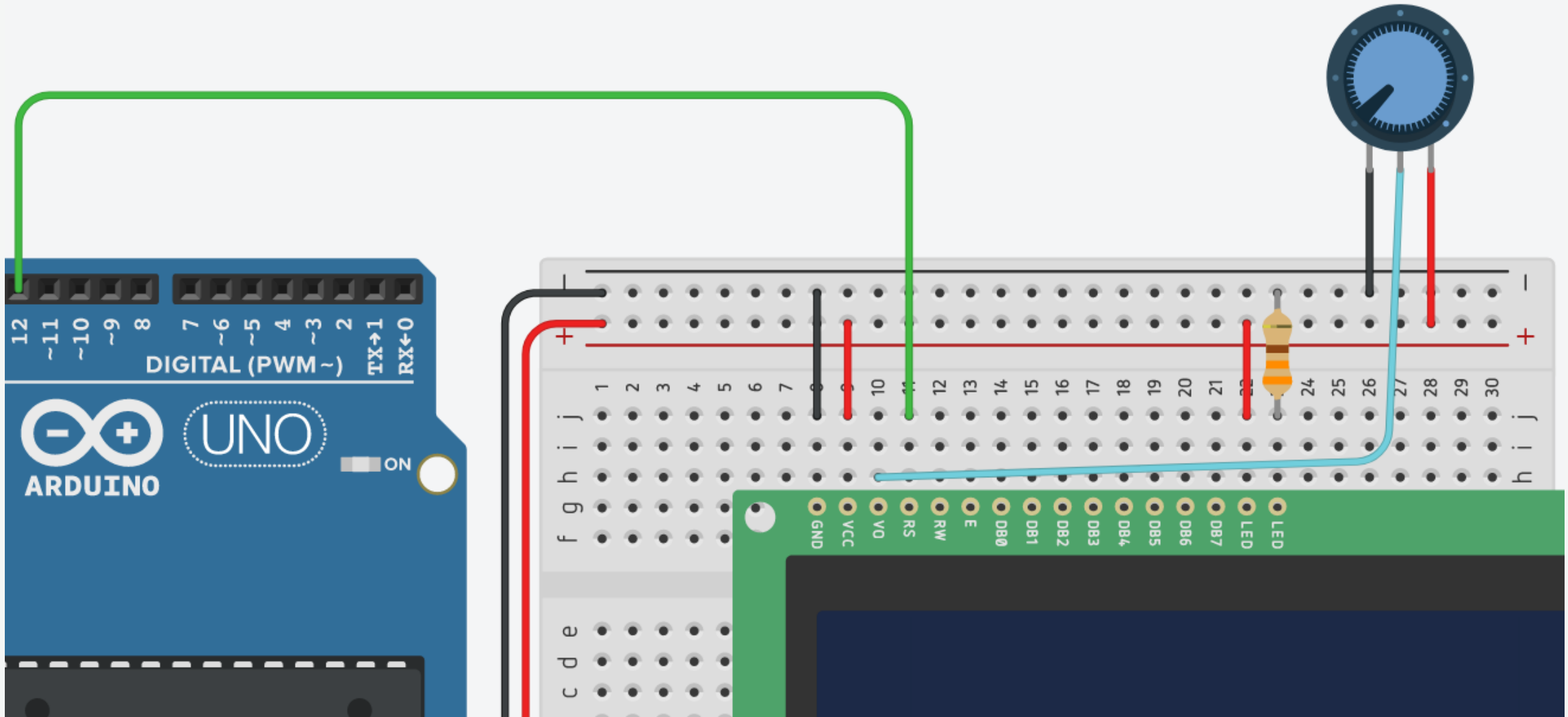
16×2 LCD: Steps

5. Connect the LCD VO pin to the POT wiper.



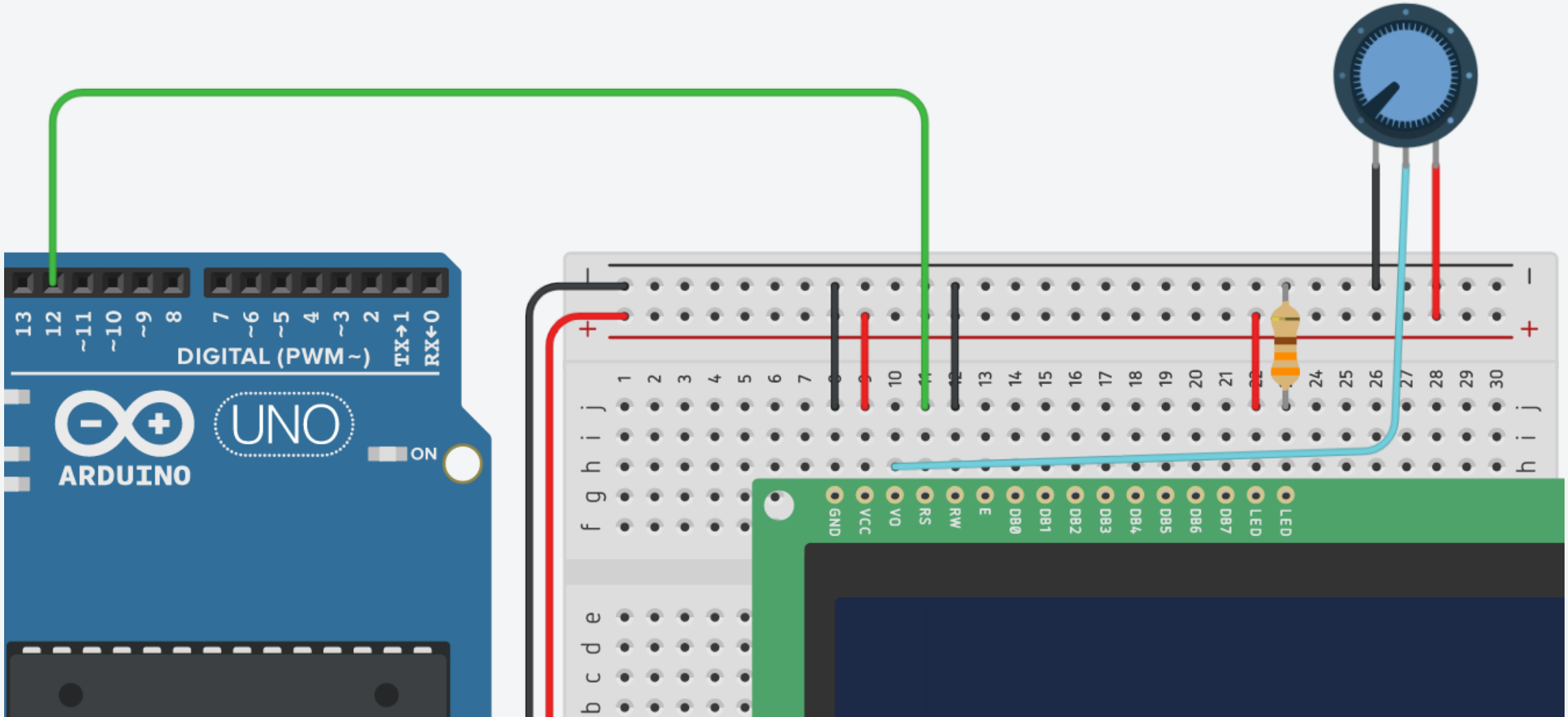
16×2 LCD: Steps

6. Connect the LCD RS pin to digital pin 12.



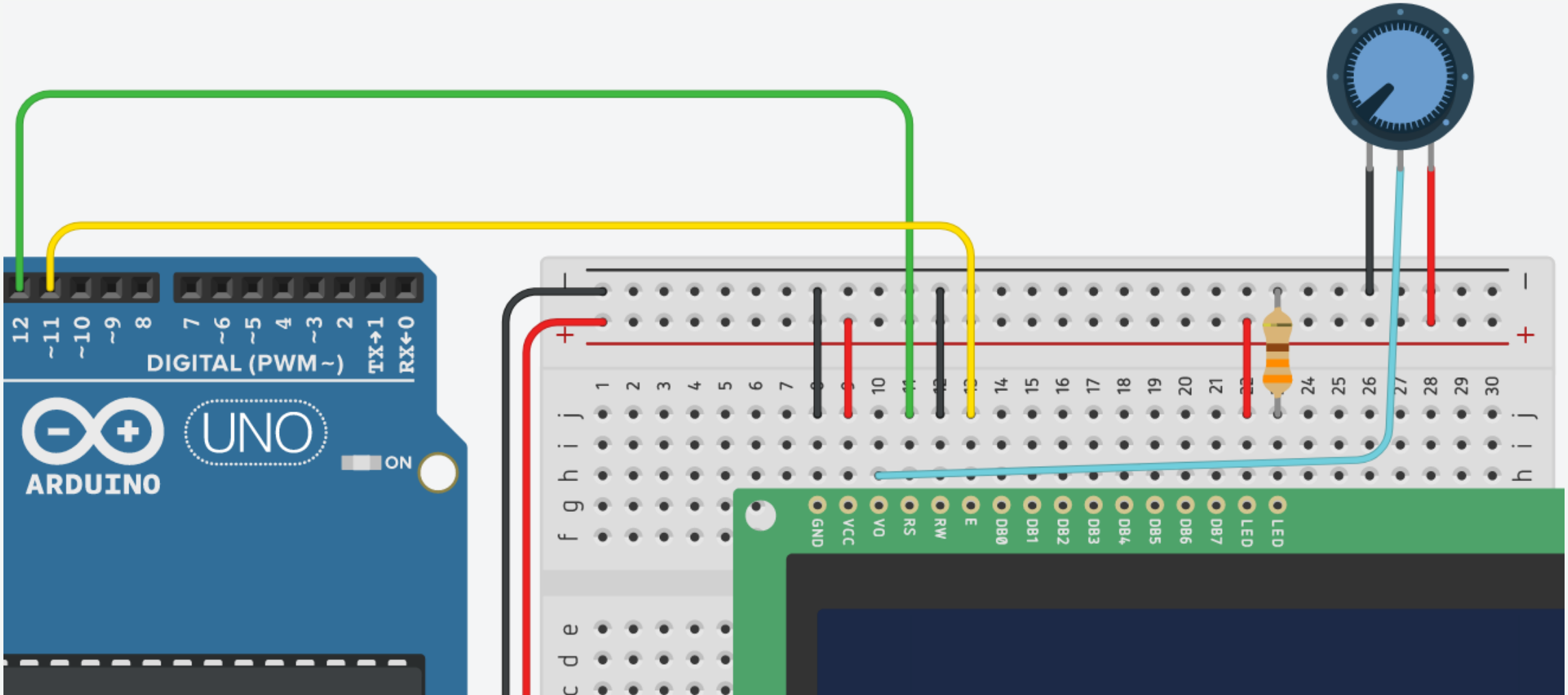
16×2 LCD: Steps

7. Connect the LCD R/W pin to ground (write mode).



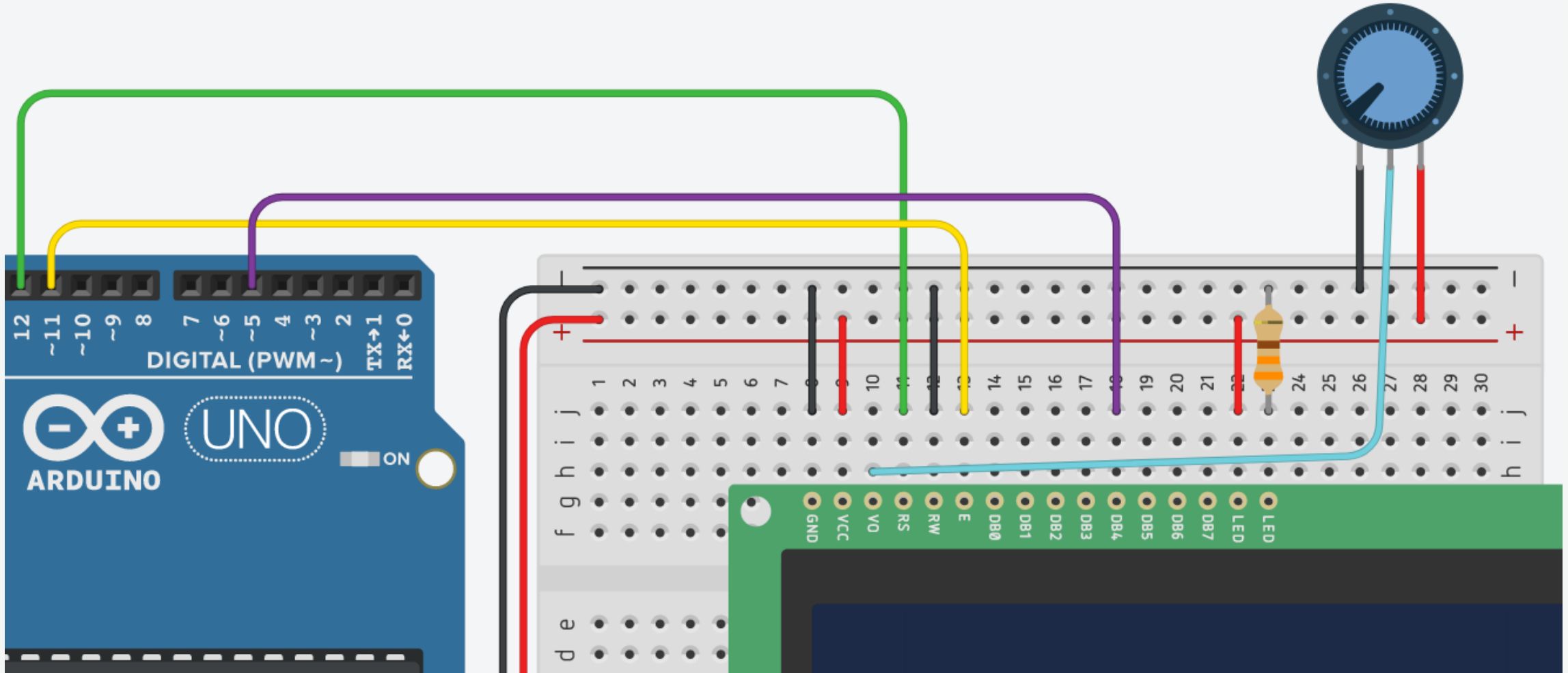
16×2 LCD: Steps

8. Connect the LCD Enable pin to digital pin 11.



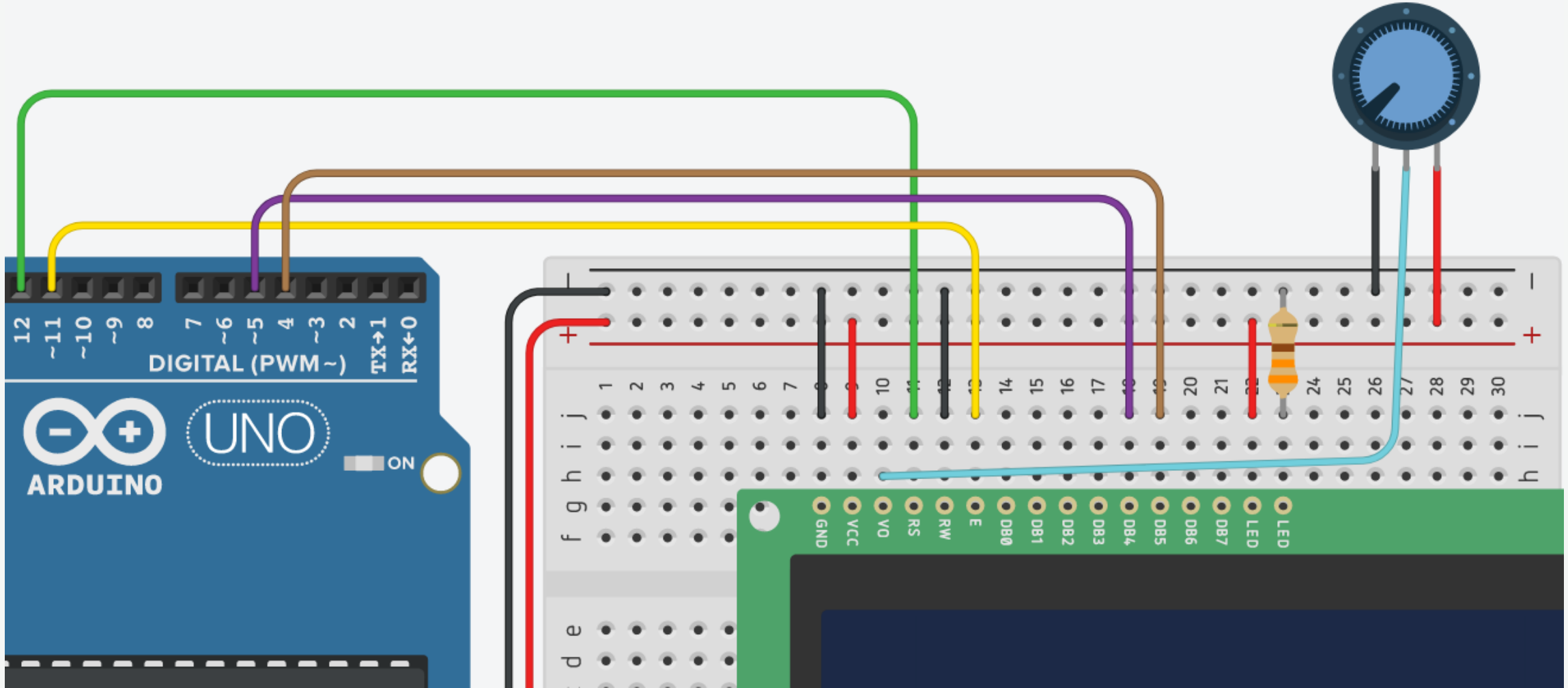
16×2 LCD: Steps

9. Connect the LCD D4 pin to digital pin 5.



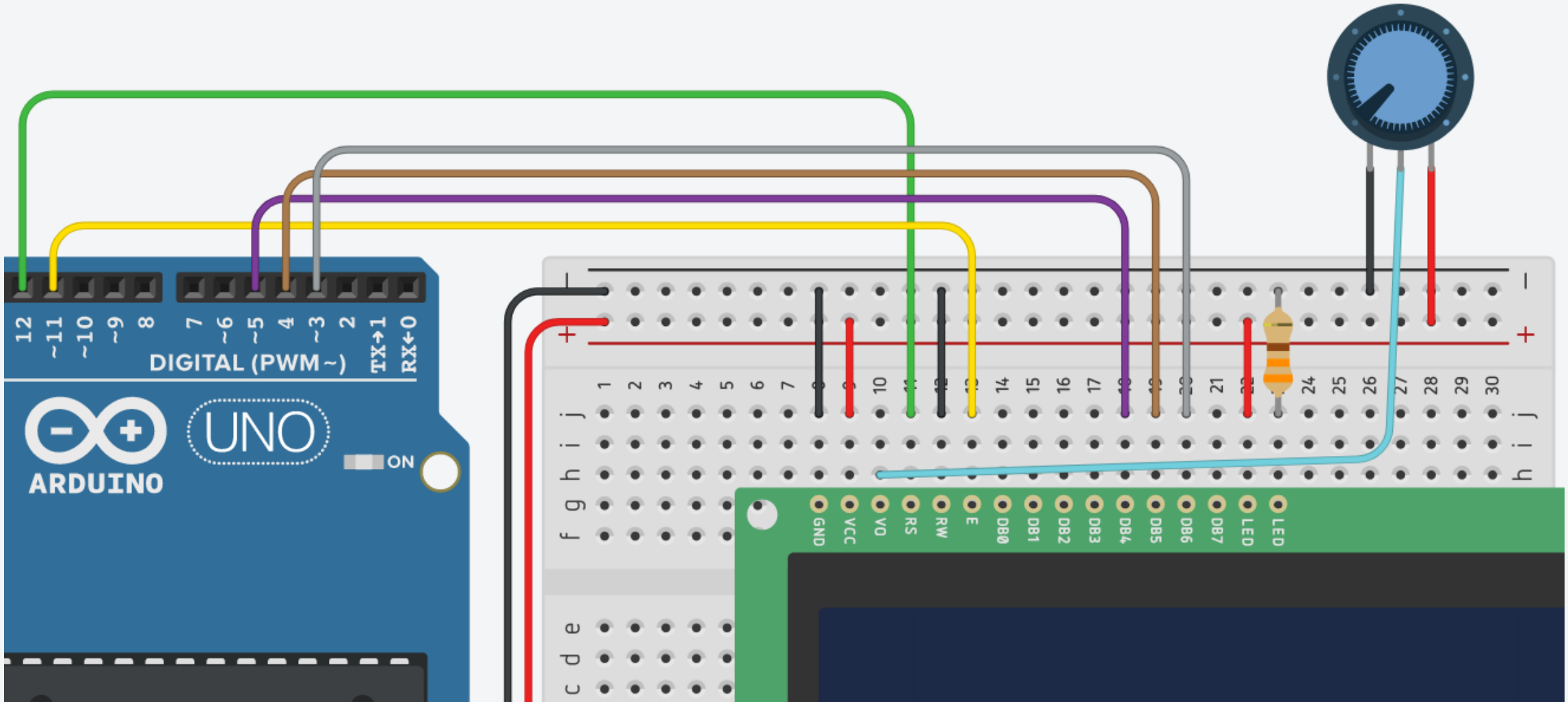
16×2 LCD: Steps

10. Connect the LCD D5 pin to digital pin 4.



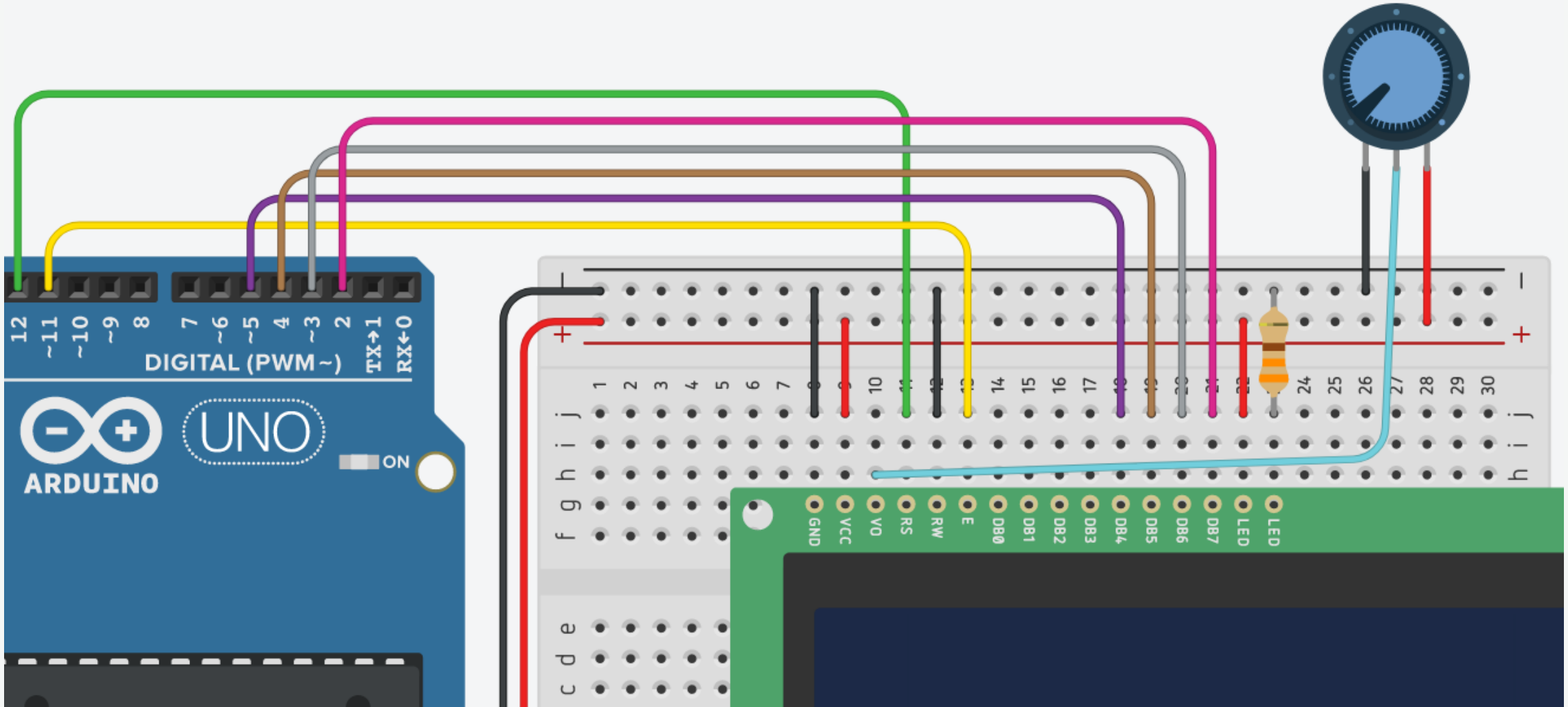
16×2 LCD: Steps

11. Connect the LCD D6 pin to digital pin 3.



16×2 LCD: Steps

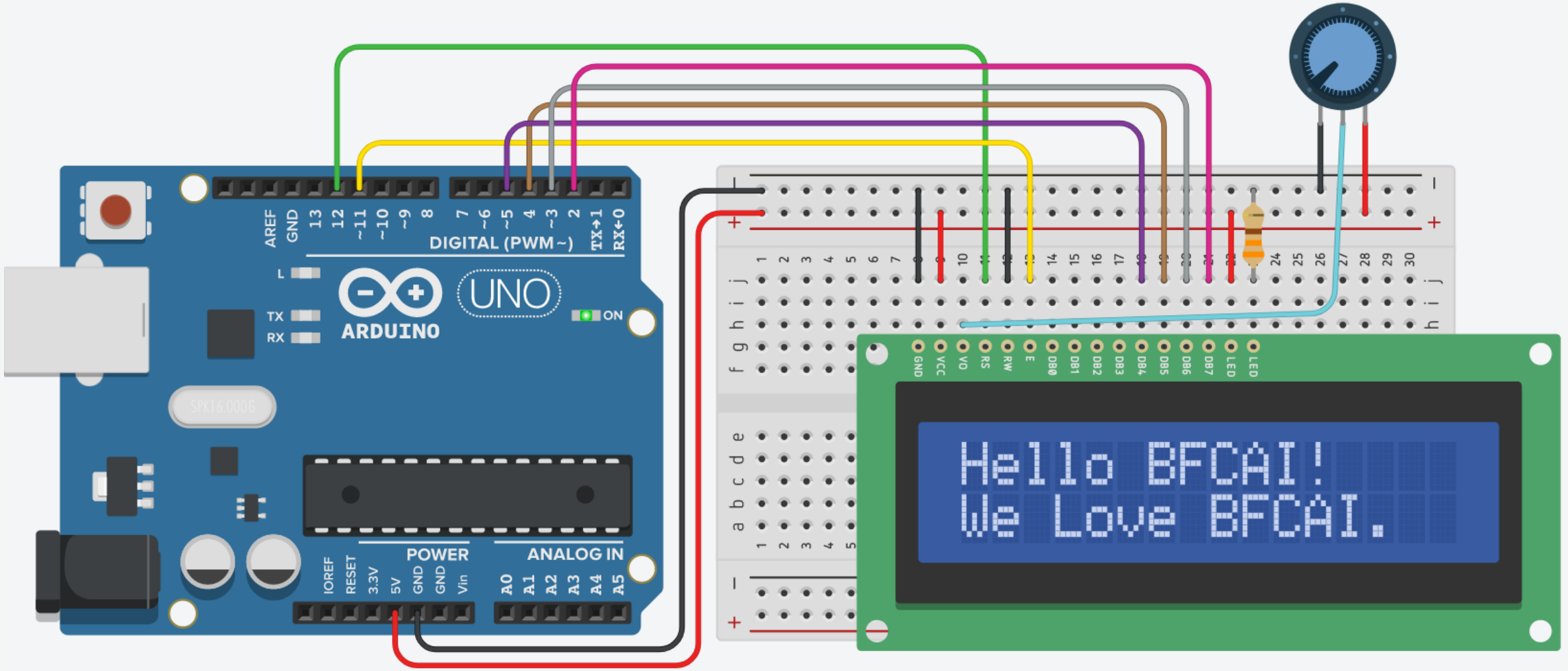
12. Connect the LCD D7 pin to digital pin 2.



16×2 LCD: Code

```
#include <LiquidCrystal.h>

// Initialize the library
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // Print a message to the LCD.
  lcd.print("Hello, World!");
}

void loop() {
  // Set the cursor to column 0, line 1
  lcd.setCursor(0, 1);

  // Print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

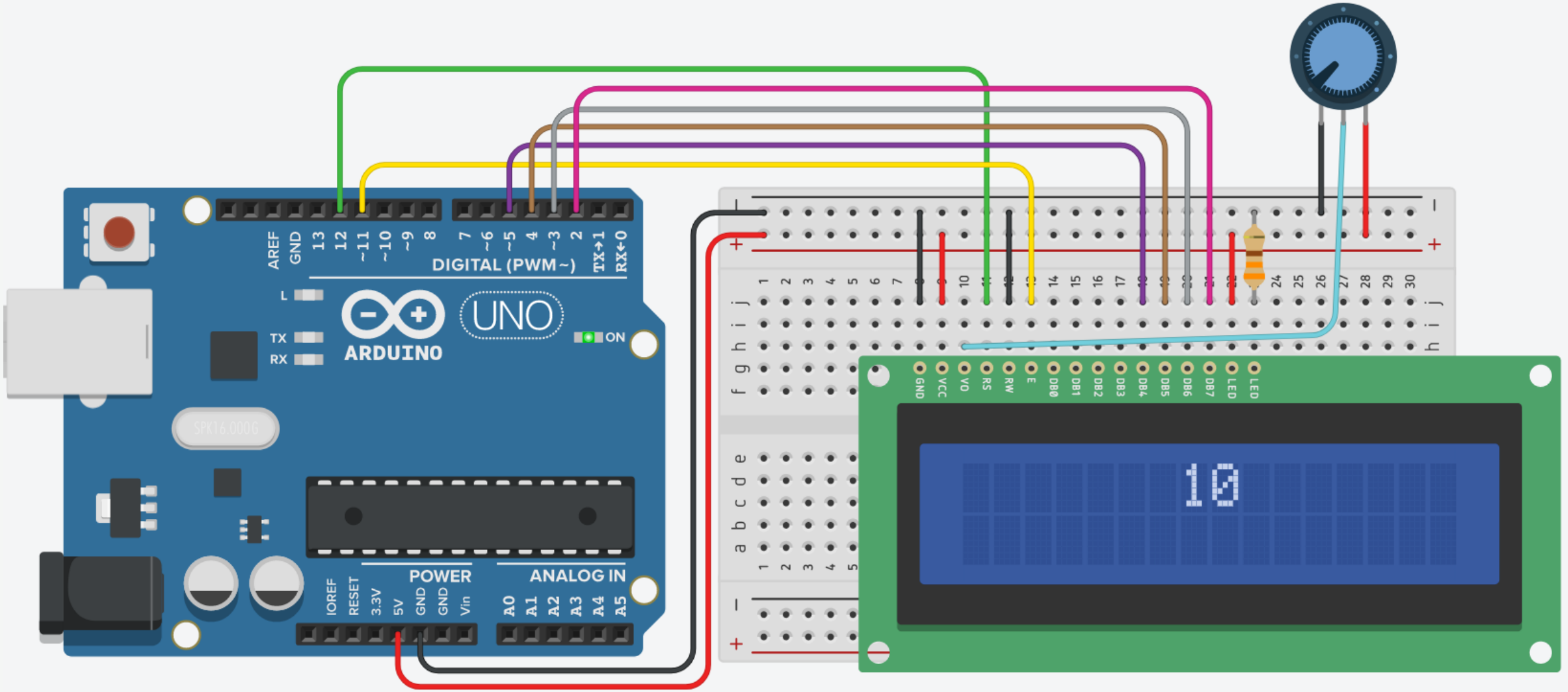
16×2 LCD HelloBFCAI: Circuit



16×2 LCD HelloBFCAI: Code

```
#include <LiquidCrystal.h>

// Initialize the library
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // Print a message on the first row
  lcd.print("Hello BFCAI!");

  // Print a message on the second row
  lcd.setCursor(0, 1);
  lcd.print("We Love BFCAI.");
}

void loop() {
}
```

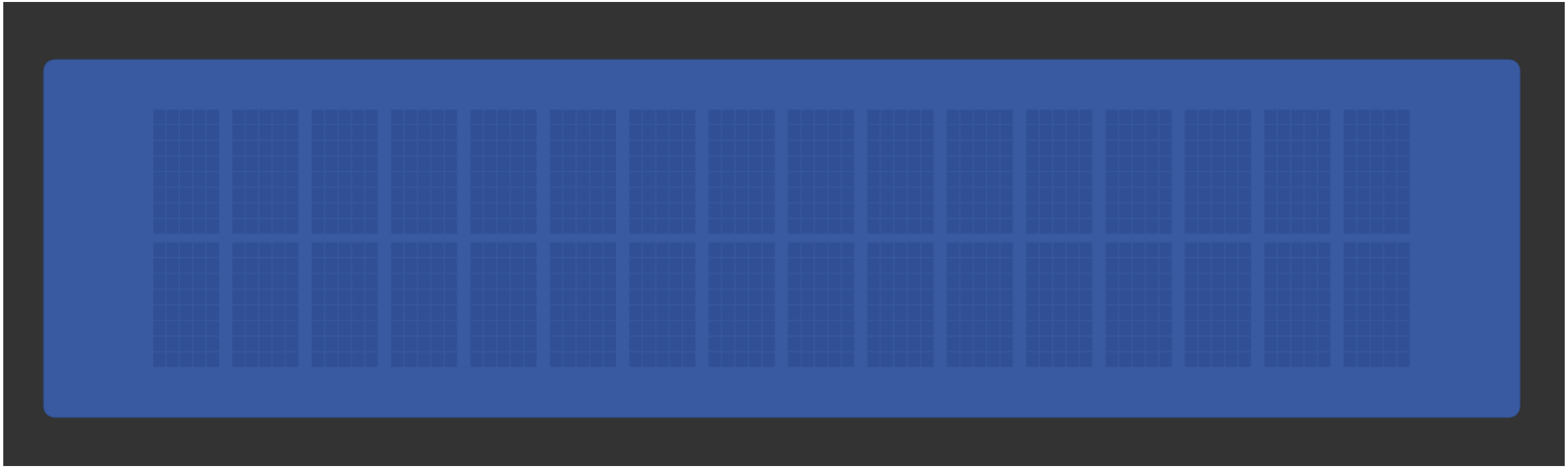
16×2 LCD Counter: Circuit



16×2 LCD Counter: Code

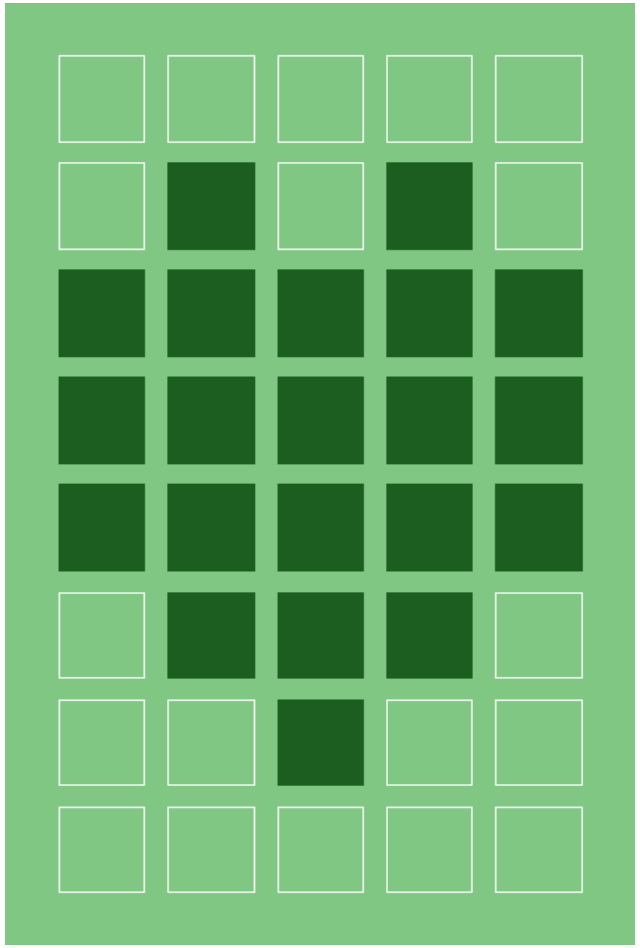
```
#include <LiquidCrystal.h>

// Initialize the library
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
}

void loop() {
  for(int i = 1; i <= 100; i++){
    lcd.clear();                    // Clear the LCD
    lcd.setCursor(7, 0);            // Center text
    lcd.print(i);                   // Print the number on the LCD

    delay(500);                     // Short delay
  }
}
```

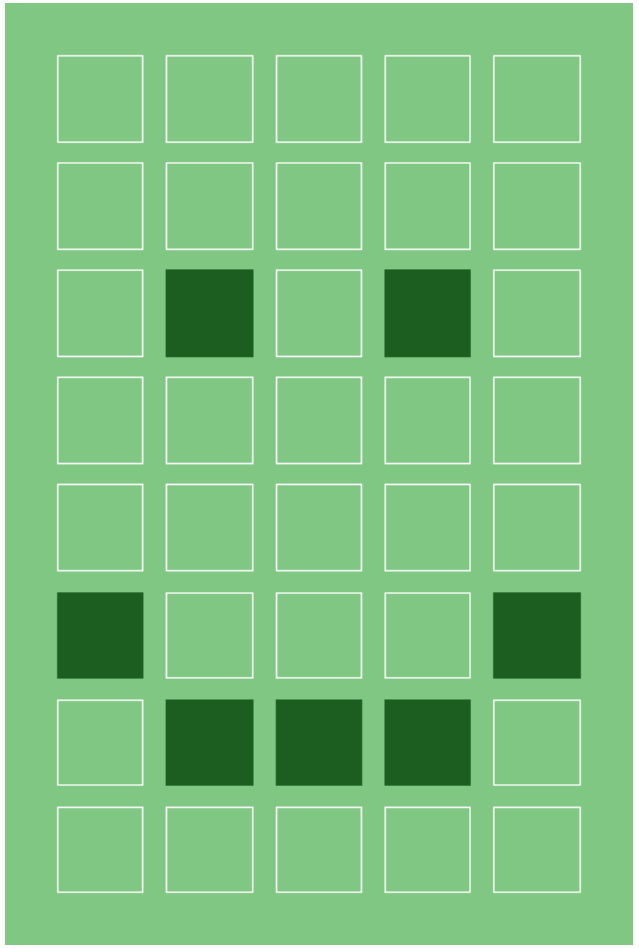
16×2 LCD - Custom Characters

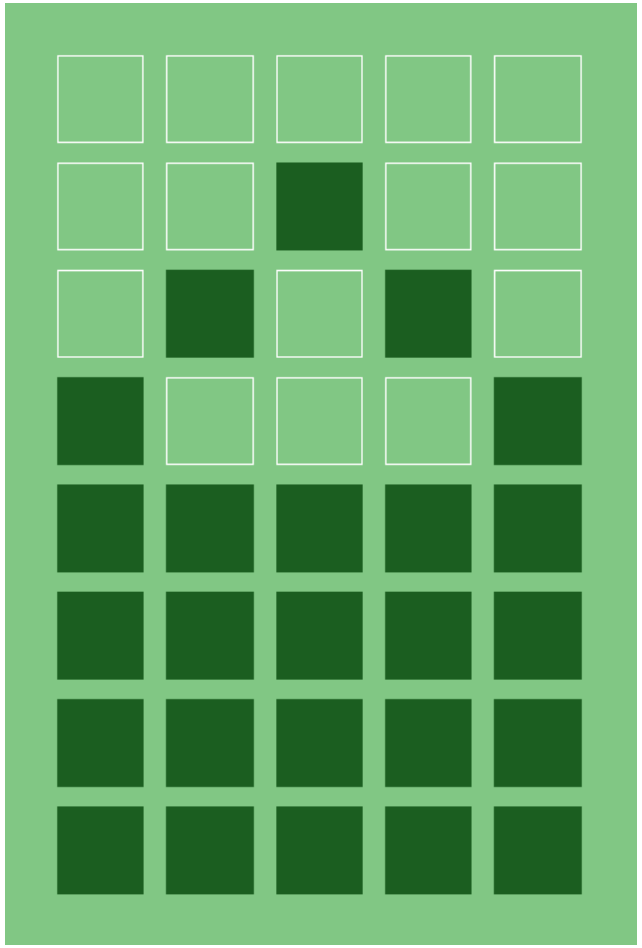


16×2 LCD - Custom Characters

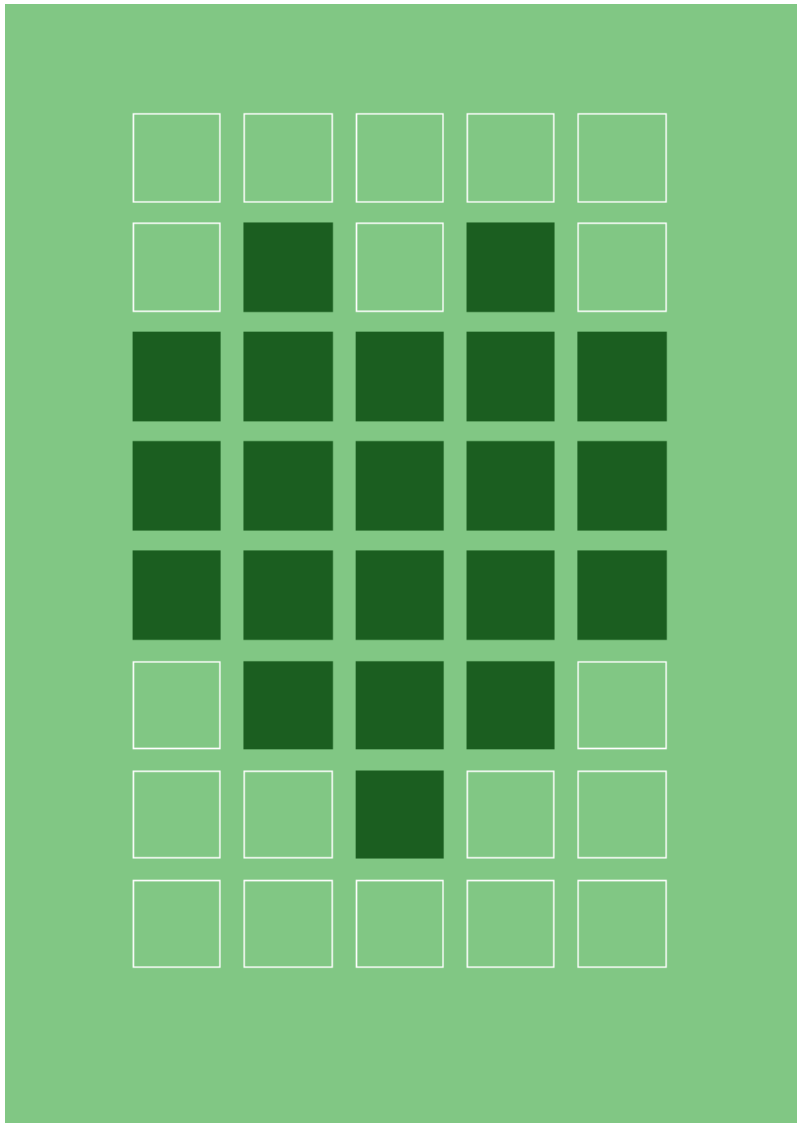
- The 16×2 LCD has an internal CG-RAM (Character Generated Ram) in which we can generate or place our custom character.





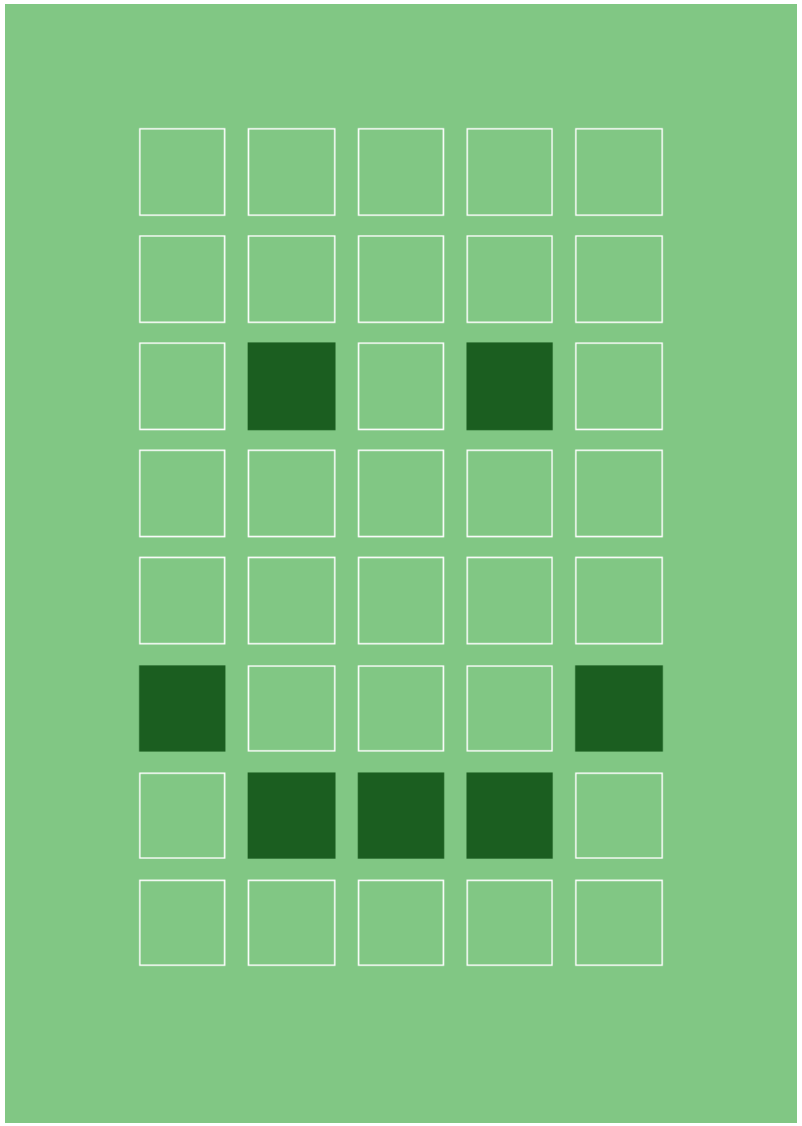


16×2 LCD - Custom Characters



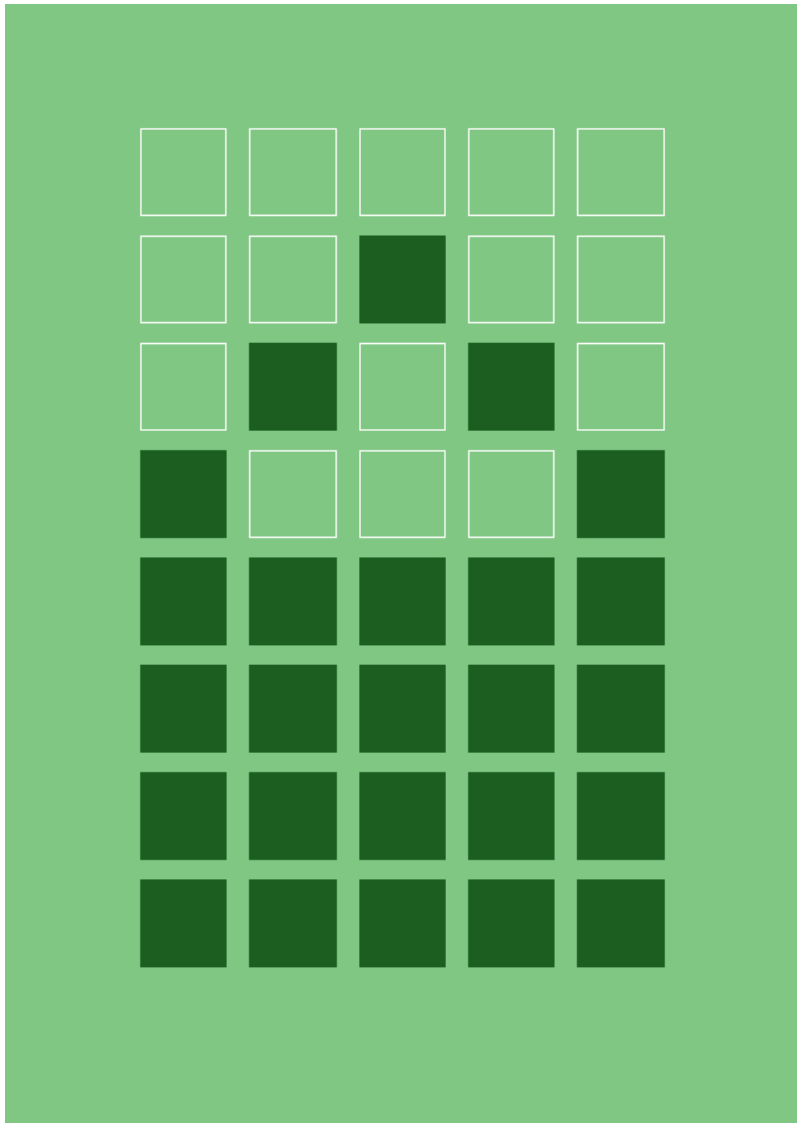
```
byte heart[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000
};
```

16×2 LCD - Custom Characters



```
byte smiley[8] = {
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};
```

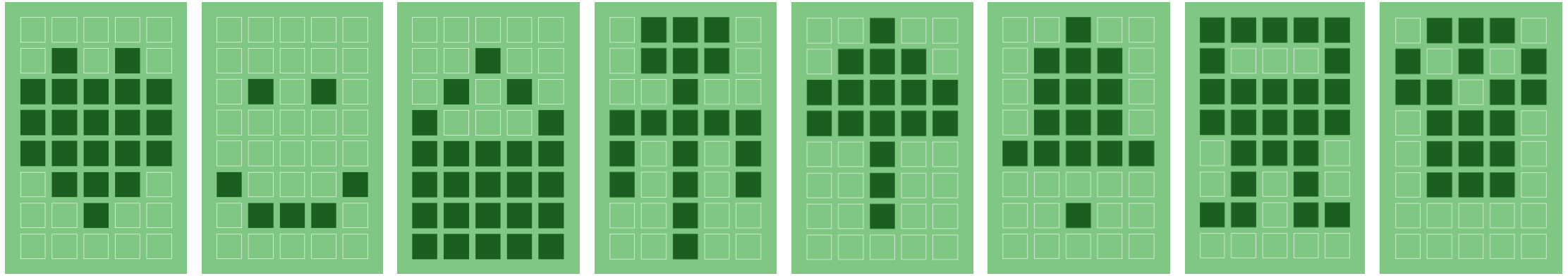
16×2 LCD - Custom Characters



```
byte home[8] = {
  0b00000,
  0b00100,
  0b01010,
  0b10001,
  0b11111,
  0b11111,
  0b11111,
  0b11111
};
```

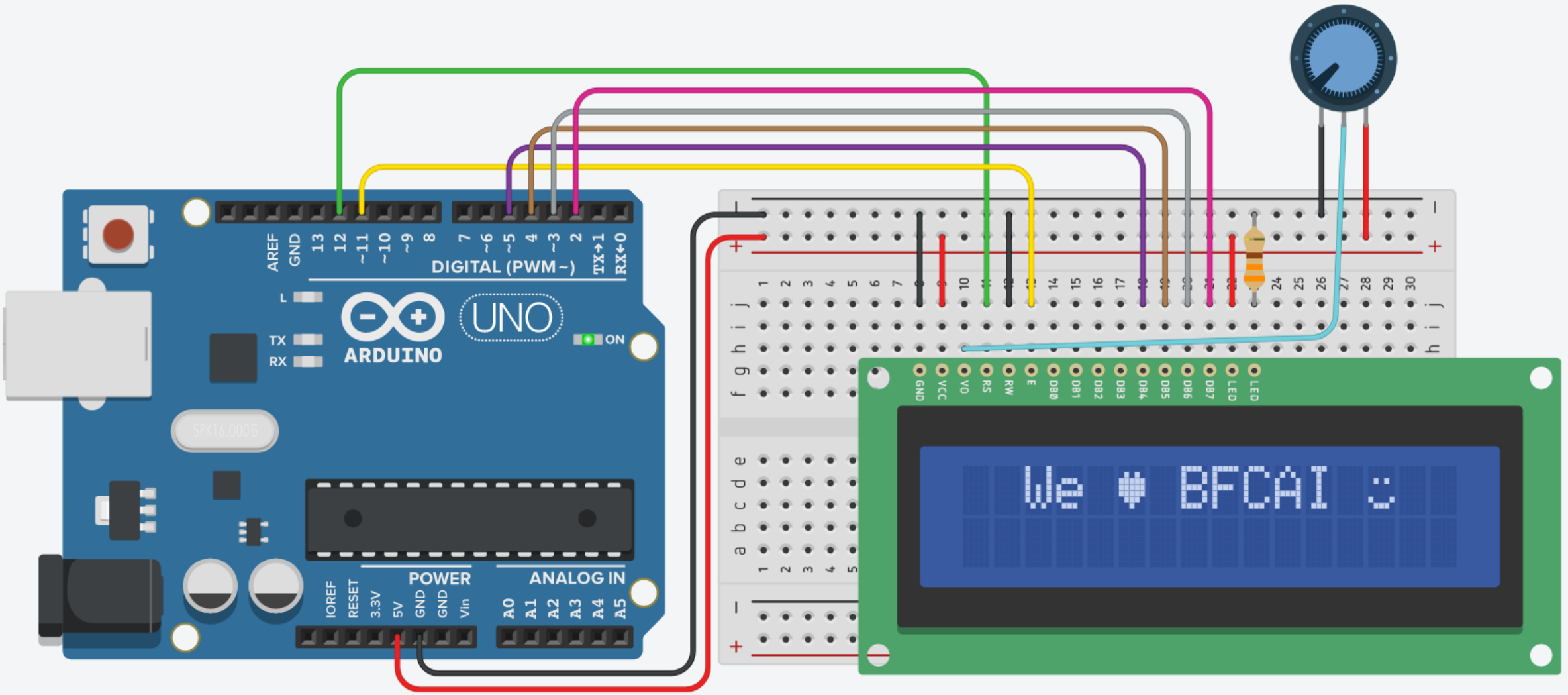
16×2 LCD - Custom Characters

- The 16×2 LCD has an internal CG-RAM (Character Generated Ram) in which we can generate or place our custom character.
- We can place 8 characters of size 5×8 at a time in CG-RAM.



https://maxpromer.github.io/LCD-Character-Creator/

16×2 LCD - Custom Characters: Circuit



16×2 LCD - Custom Characters: Code

```
// Include the library code:
#include <LiquidCrystal.h>

// Initialize the library
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Make a custom character
byte heart[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000
};
```

16×2 LCD - Custom Characters: Code

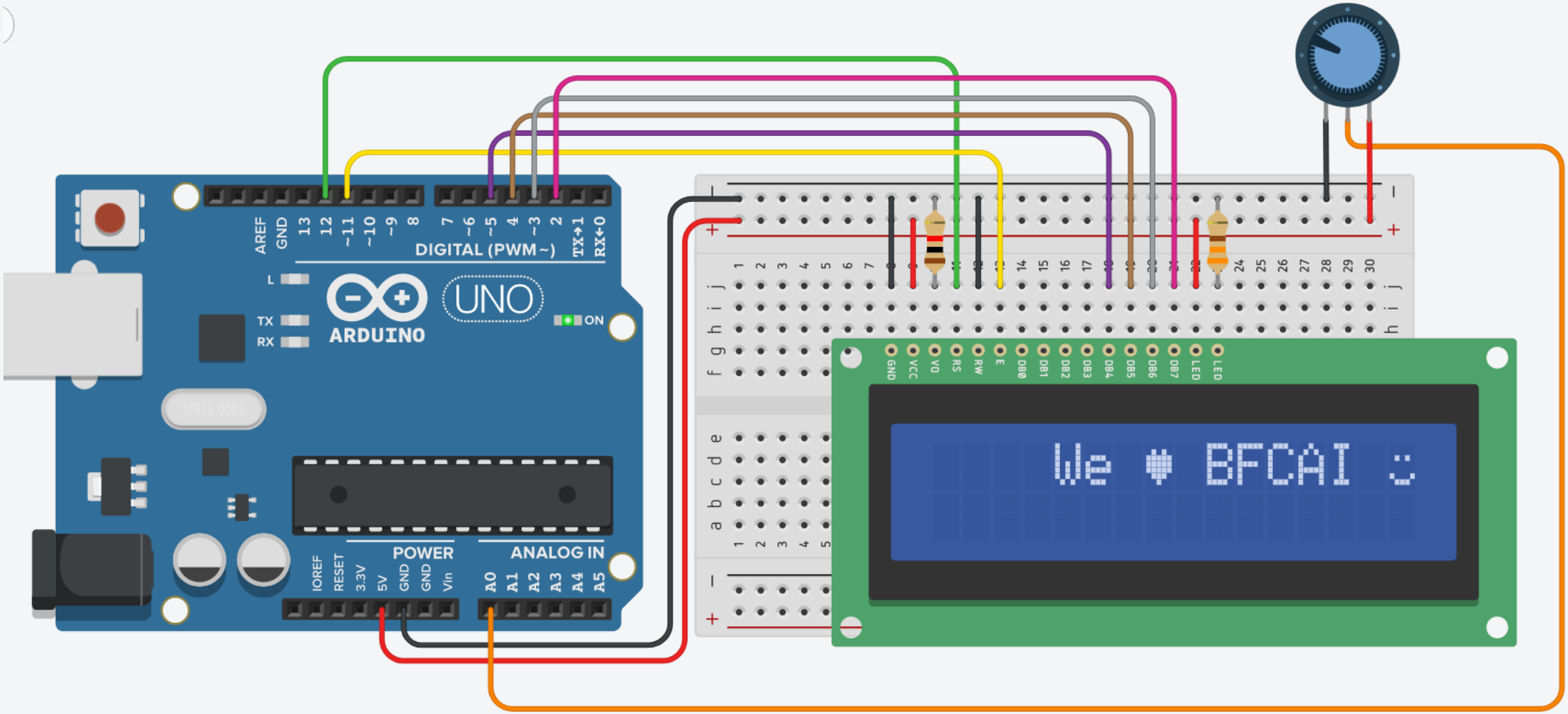
```
// Make a custom character
byte smiley[8] = {
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};
```

16×2 LCD - Custom Characters: Code

```
void setup() {
  // Set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // Create custom characters
  lcd.createChar(0, heart);
  lcd.createChar(1, smiley);

  // Print a message
  lcd.setCursor(2, 0);
  lcd.print("We ");
  lcd.write(byte(0));
  lcd.print(" BFCAI ");
  lcd.write(byte(1));
}

void loop() {
}
```

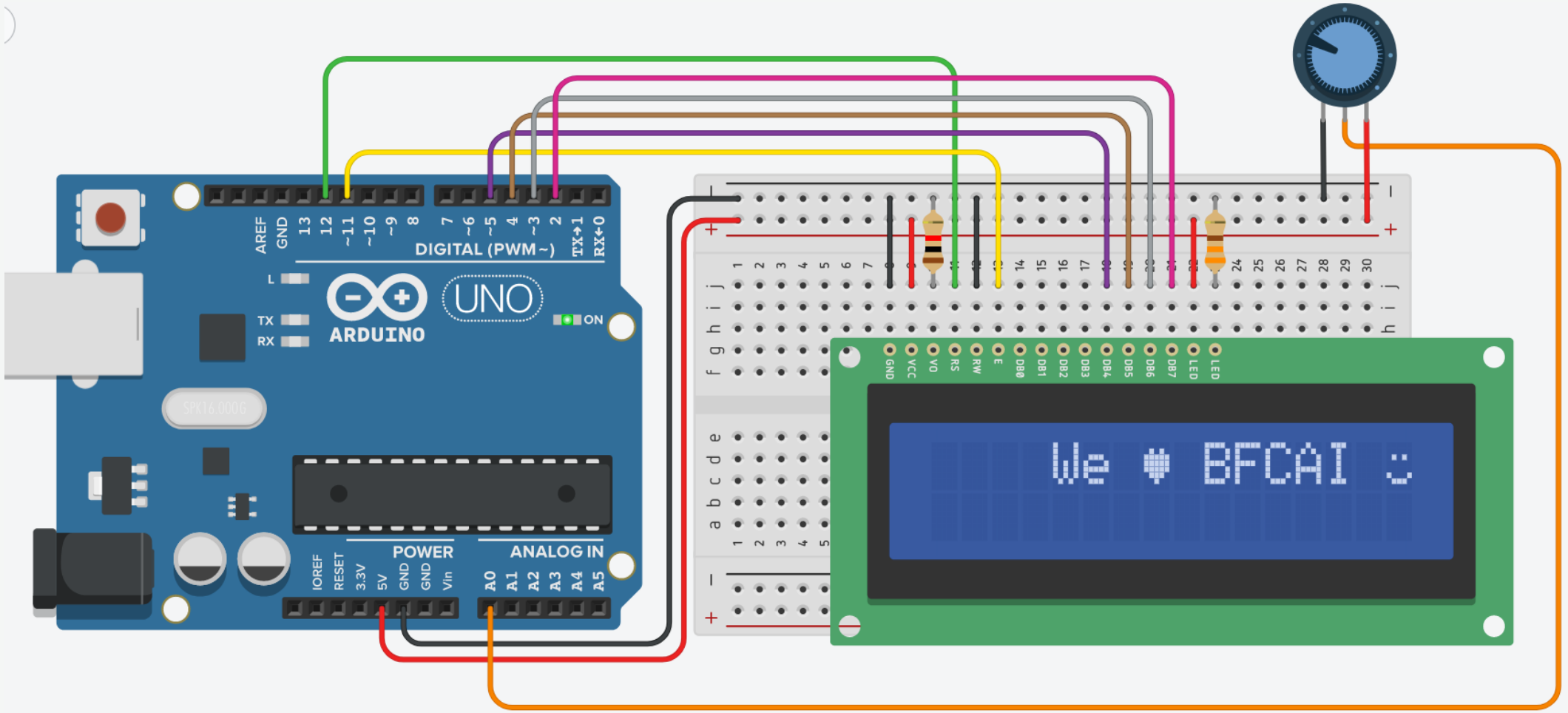
Analog Control System Prototype

- Change the position of the cursor based on the value of the potentiometer.



Analog Control System Prototype: Connections

- We will use the same connections, except connecting the wiper of the POT to the analog input A0, and connecting VO to GND with a fixed resistance.



Analog Control System Prototype: Code

```
// Include the library code:
#include <LiquidCrystal.h>

// Initialize the library
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Make a custom character
byte heart[8] = {
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000
};
```

Analog Control System Prototype: Code

```
// Make a custom character
byte smiley[8] = {
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};

int pot = 0;                        // Potentiometer value
int cur;                            // Cursor position

void setup() {
  lcd.begin(16, 2);                 // LCD's number of columns and rows
  lcd.createChar(0, heart);         // Create a custom character
  lcd.createChar(1, smiley);        // Create a custom character
}
```

Analog Control System Prototype: Code

```
void loop() {
  pot = analogRead(A0);       // Read the value of the potentiometer

  // Select the cursor position based on the potentiometer value
  if(pot >= 0 && pot <= 63)             cur = 0;
  else if(pot >= 64 && pot <= 127)      cur = 1;
  else if(pot >= 128 && pot <= 191)     cur = 2;
  else if(pot >= 192 && pot <= 255)     cur = 3;
  else if(pot >= 256 && pot <= 319)     cur = 4;
  else if(pot >= 320 && pot <= 383)     cur = 5;
  else if(pot >= 384 && pot <= 447)     cur = 6;
  else if(pot >= 448 && pot <= 511)     cur = 7;
  else if(pot >= 512 && pot <= 575)     cur = 8;
  else if(pot >= 576 && pot <= 639)     cur = 9;
  else if(pot >= 640 && pot <= 703)     cur = 10;
  else if(pot >= 704 && pot <= 767)     cur = 11;
  else if(pot >= 768 && pot <= 831)     cur = 12;
  else if(pot >= 832 && pot <= 895)     cur = 13;
  else if(pot >= 896 && pot <= 959)     cur = 14;
  else if(pot >= 960 && pot <= 1023)    cur = 15;

  lcd.clear();                // Clear the LCD
  lcd.setCursor(cur, 0);      // Set cursor
  lcd.print("We ");
  lcd.write(byte(0));
  lcd.print(" BFCAI ");
  lcd.write(byte(1));
  delay(350);                 // Short delay
}
```

Liquid Crystal Library: Summary

Function	Description
`lcd.setCursor(col, row)`	Position the LCD cursor; that is, set the location at which subsequent text written to the LCD.
`lcd.print(data)`	Prints text to the LCD.
`lcd.clear()`	Clears the LCD screen and positions the cursor in the upper-left corner.
`lcd.createChar(num, data)`	Create a custom character for use on the LCD.
`lcd.write(data)`	Write a character to the LCD.
`lcd.blink()`	Display the blinking LCD cursor.
`lcd.noBlink()`	Turns off the blinking LCD cursor.

Arduino Reference

- Go to https://www.arduino.cc/reference/en/ to learn Arduino basics.