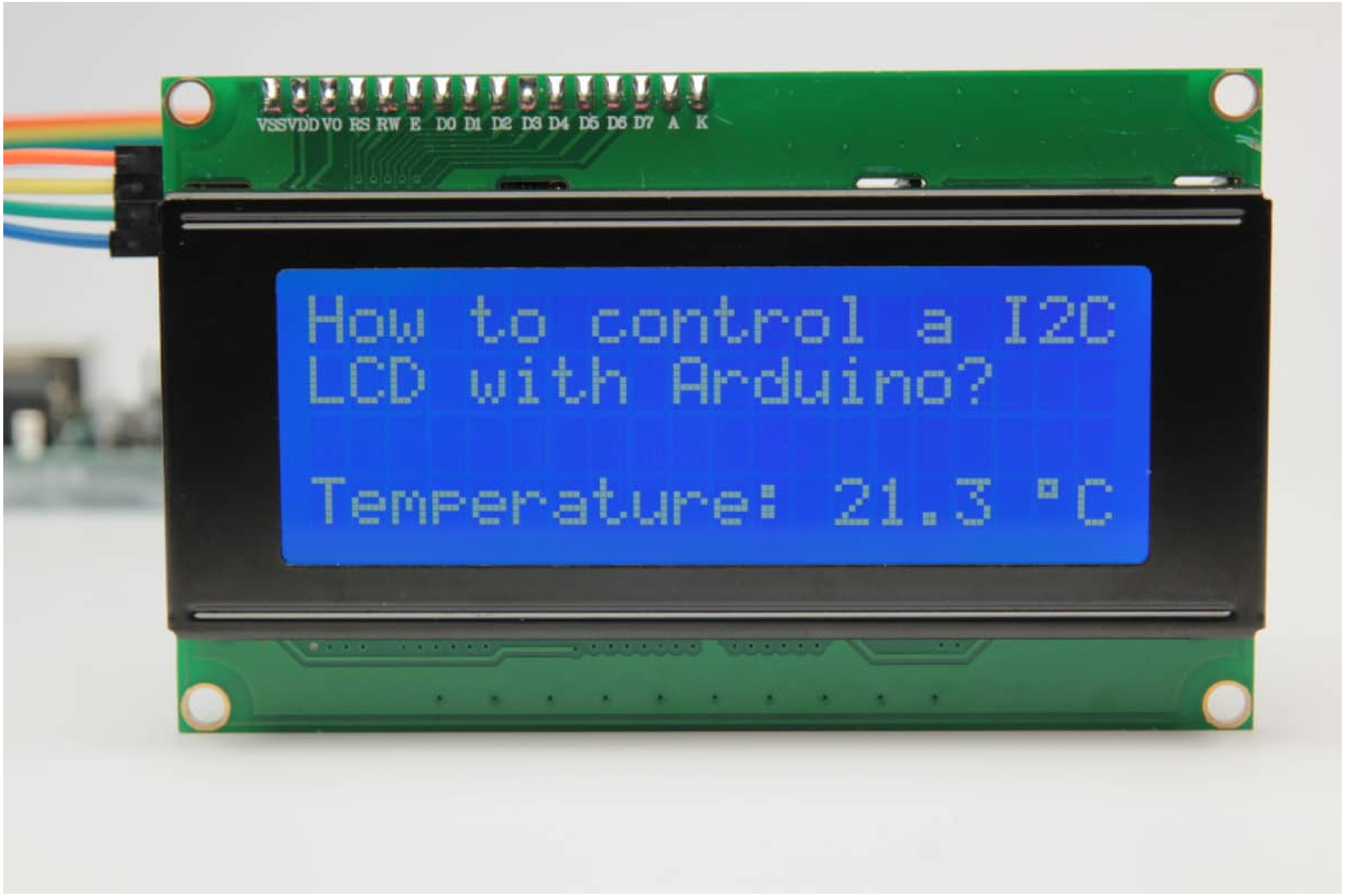
How to control a character I2C LCD with Arduino

Written by Benne de Bakker


VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
How to control a I2C
LCD with Arduino?
Temperature: 21.3 °C

This article includes everything you need to know about using a character I2C LCD with Arduino. I have included a wiring diagram and many example codes.

The first part of this article covers the basics of displaying text and numbers. In the second half I will go into more detail on how to display custom characters and how you can use the other functions of the LiquidCrystal_I2C library.

Once you know how to display text and numbers on the LCD, I suggest you take a look at the articles below. In these tutorials, you will learn how to measure and display sensor data on the LCD.

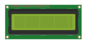
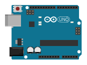
Recommended articles

- How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino
- How to use DHT11 and DHT22 Sensors with Arduino

If you have any questions, please leave a comment below.

Supplies

Hardware components

	16×2 character I2C LCD	× 1	☐
	20×4 character I2C LCD(alternative)	× 1	☐
	Arduino Uno Rev3	× 1	☐
	Jumper wires (male to female)	× 4	☐
	USB cable type A/B	× 1	☐

Tools

Small screwdriver ☐

Software

Arduino IDE ☐

I2C LCD Basics

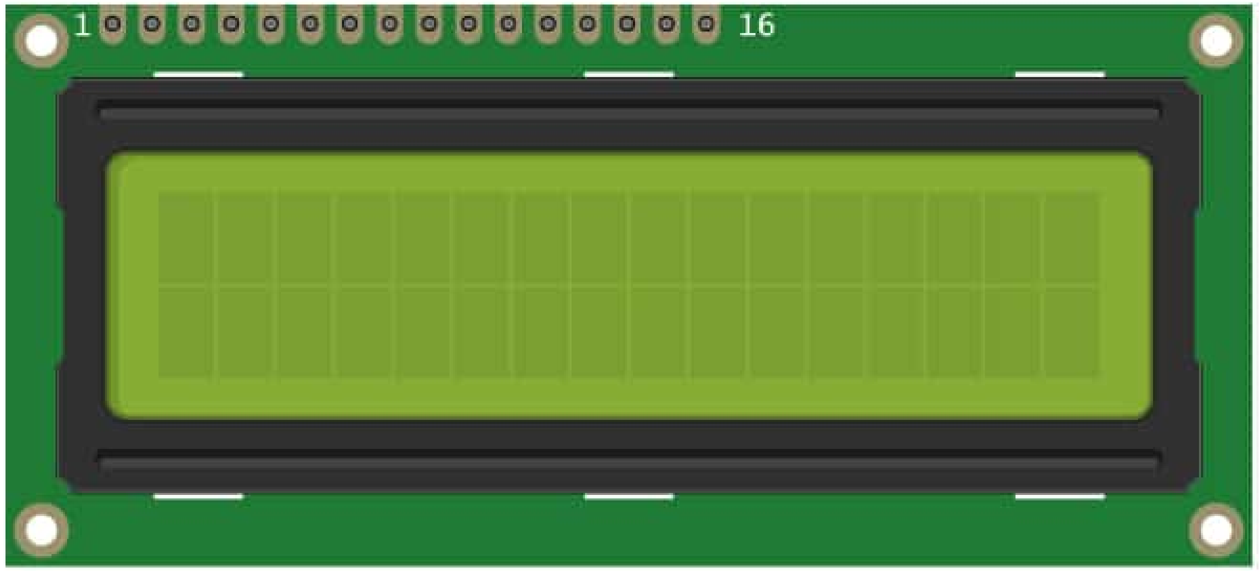
This type of LCD is ideal for displaying text and numbers, hence the name 'Character LCD'. The I2C LCDs that we are using in this tutorial come with a small add-on circuit mounted on the back of the module. This module features a PCF8574 chip (for I2C communication) and a potentiometer to adjust the LED backlight. The advantage of an I2C LCD is that the wiring is very simple. You only need two data pins to control the LCD.

Standard LCDs typically require around 12 connections, which can be a problem if you do not have many GPIO pins available. Luckily, you can also buy the I2C add-on circuit separately on Amazon, so you can easily upgrade a standard LCD as well.

For a tutorial and wiring diagram for standard character LCDs, see the following article:

- How to use a 16×2 character LCD with Arduino

If you look closely at the LCD, you can see the small rectangles that form the individual characters of the LCD. Each rectangle is made up of a grid of 5×8 pixels. Later in this tutorial, I will show you how you can control the individual pixels to display custom characters on the LCD.



Specifications

The specifications of the 16×2, 20×4 and other sized LCDs are mostly the same. They all use the same HD44780 Hitachi LCD controller, so you can easily swap them. You will only need to change the size specifications in your Arduino code.

The specifications of a typical 16×2 I2C display can be found in the table below.

16×2 I2C LCD Specifications

Operating voltage	5 V
Controller	Hitachi HD44780 LCD controller
Default address	0x27
Screen resolution	2-lines x 16 characters
Character resolution	5 x 8 pixels
Module dimensions	80 x 36 x 12 mm
Viewing area dimensions	64.5 x 16.4 mm
Cost	Check price

For more information, you can check out the datasheets below. The 16×2 and 20×4 datasheets include the dimensions of the LCD and in the HD44780 datasheet you can find more information about the Hitachi LCD driver. The PCF8574 chip is used in the I2C module on the back of the LCD.

16×2 LCD Datasheet

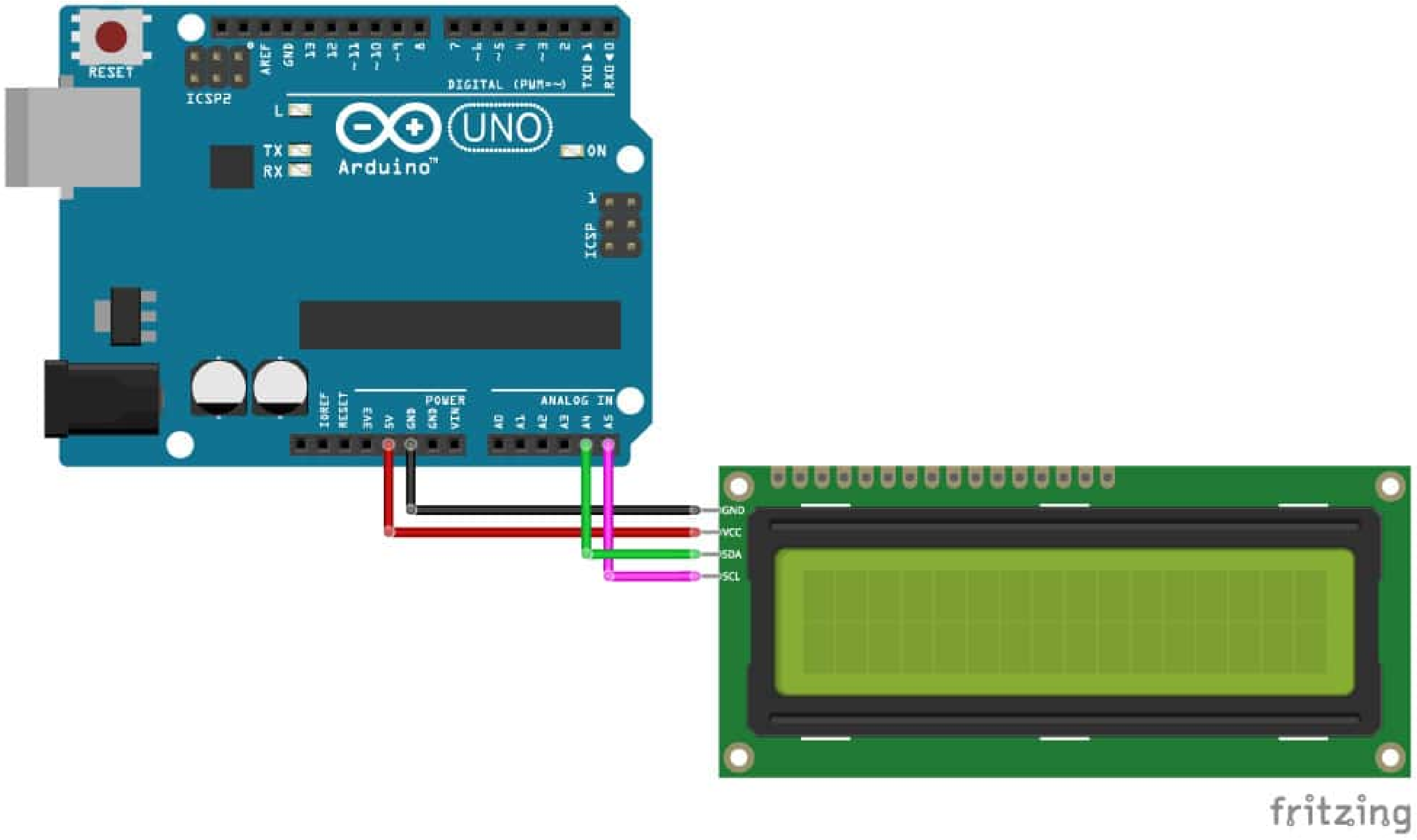
20×4 LCD Datasheet

HD44780 Datasheet

PCF8574 Datasheet

How to connect the I2C LCD to Arduino UNO

The wiring diagram below shows you how to connect the I2C LCD to the Arduino. Wiring an I2C LCD is a lot easier than connecting a standard LCD. You only need to connect 4 pins instead of 12.



I2C LCD with Arduino wiring diagram

The connections are also given in the table below.

I2C LCD Connections

I2C Character LCD	Arduino
GND	GND
VCC	5 V
SDA	A4
SDL	A5

If you are not using an Arduino Uno, the SDA and SCL pins can be at a different location. An Arduino UNO with the R3 layout (1.0 pinout), also has the SDA (data line) and SCL (clock line) pin headers close to the AREF pin. Check the table below for more details.

I2C pin locations for different Arduino boards

Board	SDA	SCL
Arduino Uno	A4	A5
Arduino Nano	A4	A5
Arduino Micro	2	3
Arduino Mega 2560	20	21
Arduino Leonardo	2	3
Arduino Due	20	21

Adjusting the contrast of the LCD

After you have wired up the LCD, you will need to adjust the contrast of the display. On the I2C module, you will find a potentiometer that you can turn with a small screwdriver.

Plug in the USB connector of the Arduino to power the LCD. You should see the backlight light up. Now rotate the potentiometer until one (16×2 LCD) or 2 rows (20×4 LCD) of rectangles appear.

You can tweak the contrast later if needed.

Once that is done, we can start programming the LCD.

Installing the LiquidCrystal_I2C Arduino library

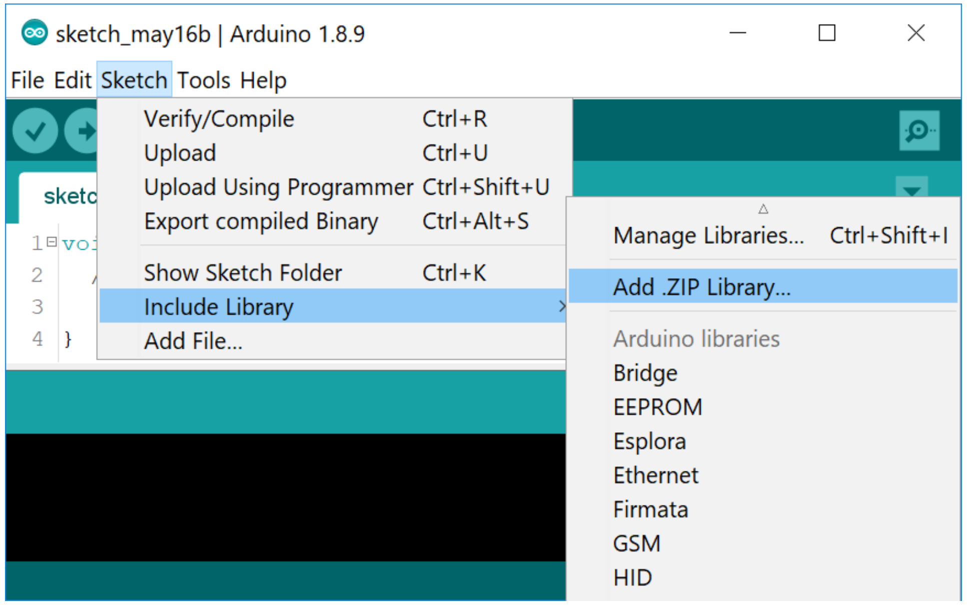
In this tutorial I will be using the **LiquidCrystal_I2C** library. This library has many built-in functions that make programming the LCD quite easy. The latest version of this library can be downloaded here on GitHub or click the button below.

LiquidCrystal_I2C.zip

Make sure that you have this exact library installed and delete any other libraries that have the same name (LiquidCrystal_I2C). Other libraries will probably work as well but might use slightly different names for the different functions.

This library works in combination with the **Wire.h** library which allows you to communicate with I2C devices. It should come pre-installed with the Arduino IDE.

You can install the library by going to **Sketch > Include Library > Add .ZIP Library…** in the Arduino IDE.


sketch_may16b | Arduino 1.8.9
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
HID
Installing a .ZIP Arduino library

The library does include some examples that you can use, but you will have to modify them to match your hardware setup. I have included many example codes below that you can use with the wiring setup I have shown earlier.

First I will show you some basic example code and then I will explain the functions in more detail.

How to find the I2C address of my LCD?

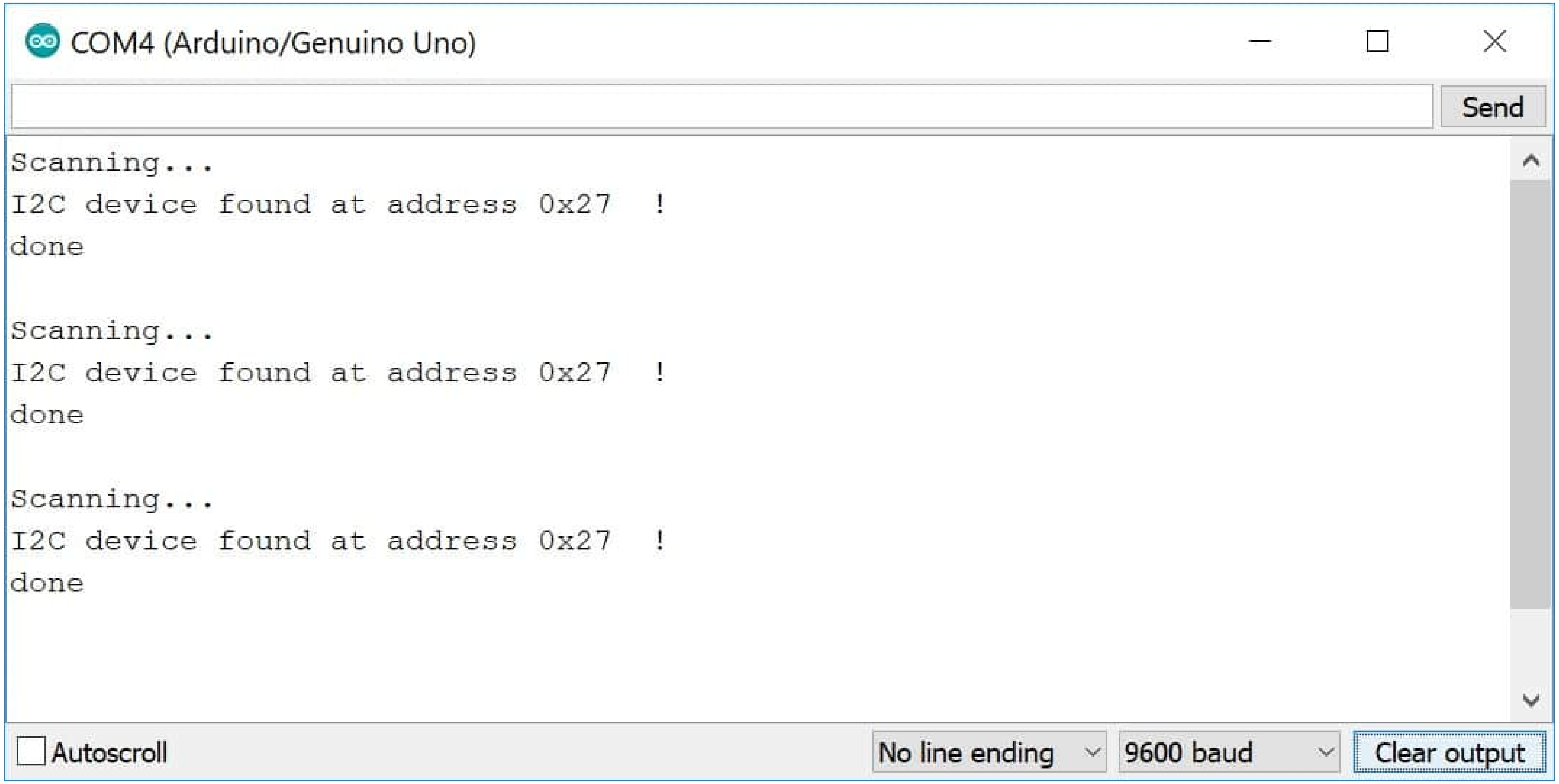
Most I2C LCDs ship with the default address ‘0x27’, but it can be different depending on the batch/manufacturer. If this is the case, you will need to find the actual address of the LCD before you can start using it.

On the Arduino website, you can find a simple example sketch that scans the I2C-bus for devices. If a device is found, it will display the address in the serial monitor.

You can open the code in a new window by clicking on the button in the top right corner of the code field.

```
/*I2C_scanner
  This sketch tests standard 7-bit addresses.
  Devices with higher bit address might not be seen properly.*/

#include <Wire.h>

void setup() {
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);
  Serial.println("\nI2C Scanner");
}

void loop() {
  byte error, address;
  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;
  for (address = 1; address < 127; address++ ) {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.print(address, HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error == 4) {
      Serial.print("Unknown error at address 0x");
      if (address < 16)
        Serial.print("0");
      Serial.println(address, HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");

  delay(5000);
}
```

If you upload this sketch to the Arduino and run it, you should see the following output in the Serial Monitor (Ctrl + Shift + M).



```
Scanning...
I2C device found at address 0x27  !
done

Scanning...
I2C device found at address 0x27  !
done

Scanning...
I2C device found at address 0x27  !
done
```

I2C address scanner Serial Monitor output

Write down the address you find, you will need it later when programming the LCD.

Basic Arduino example code for I2C LCD

You can upload the following example code to the Arduino using the Arduino IDE.

For this tutorial, I used this 20×4 I2C character LCD display, but you can use other I2C LCDs of different sizes as well.

This example sketch will display the classic 'Hello World!' on the first line of the LCD and 'LCD tutorial' on the second line. Next, I will explain how the code works.

```
/* I2C LCD with Arduino example code. More info: https://www.makerguides.com */

// Include the libraries:
// LiquidCrystal_I2C.h: https://github.com/johnrickman/LiquidCrystal_I2C
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD

// Wiring: SDA pin is connected to A4 and SCL pin to A5.
// Connect to LCD via I2C, default address 0x27 (A0-A2 not jumpered)
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4); // Change to (0x27,16,2) for 16x2 LCD.

void setup() {
  // Initiate the LCD:
  lcd.init();
  lcd.backlight();
}

void loop() {
  // Print 'Hello World!' on the first line of the LCD:
  lcd.setCursor(0, 0); // Set the cursor on the first column and first row.
  lcd.print("Hello World!"); // Print the string "Hello World!"
  lcd.setCursor(2, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
  lcd.print("LCD tutorial");
}
```

You should see the following output on the LCD:

(Coming soon)

How the code works

First, the required libraries are included. As mentioned earlier we need both the wire.h* and the LiquidCrystal_I2C library. In the rest of this tutorial, I will cover more of the built-in functions of this library.

*When using the latest version of the LiquidCrystal_I2C library it is no longer needed to include the wire.h library in your sketch. The other library imports wire.h automatically.

```
// Include the libraries:
// LiquidCrystal_I2C.h: https://github.com/johnrickman/LiquidCrystal_I2C
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD
```

The next step is to create an LCD object with the LiquidCrystal_I2C class and specify the address and dimensions. For this we use the function `LiquidCrystal_I2C(address, columns, rows)`. This is where you will need to change the default address to the address you found earlier if it happens to be different.

When using a 16×2 LCD, change this line to `LiquidCrystal_I2C(0x27,16,2);`

Note that we have called the display 'lcd'. You can give it a different name if you want like 'menu_display'. You will need to change 'lcd' to the new name in the rest of the sketch.

```
// Connect to LCD via I2C, default address 0x27 (A0-A2 not jumpered)
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4); // Change to (0x27,16,2) for 16x2 LCD.
```

In the setup, the LCD is initiated with `lcd.init()` and the backlight is turned on with `lcd.backlight()`.

```
void setup() {
  // Initiate the LCD:
  lcd.init();
  lcd.backlight();
}
```

In the loop section of the code, the cursor is set to the first column and the first row of the LCD with `lcd.setCursor(0,0)`. Note that counting starts at 0 and the first argument specifies the column. So `lcd.setCursor(2,1)` sets the cursor on the third column and the second row.

Next the string 'Hello World!' is printed with `lcd.print("Hello World!")`. Note that you need to place quotation marks (" ") around the text, since we are printing a text string. When you want to print numbers, no quotation marks are necessary. For example `lcd.print(12345)`.

```
void loop() {
  // Print 'Hello World!' on the first line of the LCD:
  lcd.setCursor(0, 0); // Set the cursor on the first column and first row.
  lcd.print("Hello World!"); // Print the string "Hello World!"
  lcd.setCursor(2, 1); //Set the cursor on the third column and the second row (counting starts at 0!).
  lcd.print("LCD tutorial");
}
```

If you want to see an example for displaying (changing) variables on the LCD, check out my tutorial for the HC-SR04 ultrasonic distance sensor:

- How to use a HC-SR04 Ultrasonic Distance Sensor with Arduino

Other functions of the LiquidCrystal_I2C library

The example sketch above shows you the basics of displaying text on the LCD. Now we will take a look at the other functions of the LiquidCrystal_I2C library.

clear()

Clears the LCD screen and positions the cursor in the upper-left corner (first row and first column) of the display. You can use this function to display different words in a loop.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
}

void loop() {
  lcd.clear();
  lcd.print("Monday");
  delay(2000);
  lcd.clear();
  lcd.print("13:45");
  delay(2000);
}
```

home()

Positions the cursor in the top-left corner of the LCD. Use `clear()` if you also want to clear the display.

cursor()

Displays the LCD cursor: an underscore (line) at the position of the next character to be printed.

noCursor()

Hides the LCD cursor. The following example creates a blinking cursor at the end of “Hello World!”.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("Hello World!");
}

void loop() {
  lcd.cursor();
  delay(500);
  lcd.noCursor();
  delay(500);
}
```

blink()

Creates a blinking block style LCD cursor: a blinking rectangle at the position of the next character to be printed.

noBlink()

Disables the block style LCD cursor. The following example displays the blinking cursor for 5 seconds and then disables it for 2 seconds.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("blink() example");
}

void loop() {
  lcd.blink();
  delay(5000);
  lcd.noBlink();
  delay(2000);
}
```

display()

This function turns on the LCD screen and displays any text or cursors that have been printed to the display.

noDisplay()

This function turns off any text or cursors printed to the LCD. The text/data is not cleared from the LCD memory. This means it will be shown again when the function `display()` is called.

The following example creates a blinking text effect.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("Blinking text");
}

void loop() {
  lcd.display();
  delay(2000);
  lcd.noDisplay();
  delay(2000);
}
```

write()

This function can be used to write a character to the LCD. See the section about creating and displaying custom characters below for more info.

scrollDisplayLeft()

Scrolls the contents of the display (text and cursor) one space to the left. You can use this function in the loop section of the code in combination with `delay(500)`, to create a scrolling text animation.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.print("Hello World!");
}

void loop() {
  lcd.scrollDisplayLeft();
  delay(500);
}
```

scrollDisplayRight()

Scrolls the contents of the display (text and cursor) one space to the right.

autoscroll()

This function turns on automatic scrolling of the LCD. This causes each character output to the display to push previous characters over by one space. If the current text direction is left-to-right (the default), the display scrolls to the left, if the current direction is right-to-left, the display scrolls to the right. This has the effect of outputting each new character to the same location on the LCD.

The following example sketch enables automatic scrolling and prints the character 0 to 9 at the position (20,0) of the LCD. Change this to (16,0) for a 16×2 LCD.

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

void setup() {
  lcd.init();
  lcd.backlight();
}

void loop() {
  lcd.autoscroll();
  lcd.setCursor(20, 0);
  for (int x = 0; x < 10; x++) {
    lcd.print(x);
    delay(500);
  }
  lcd.clear();
}
```

noAutoscroll()

Turns off automatic scrolling of the LCD.

leftToRight()

This function causes text to flow to the right from the cursor, as if the display is left-justified (default).

rightToLeft()

This function causes text to flow to the left from the cursor, as if the display is right-justified.

How to create and display custom characters?

With the function `createChar()` it is possible to create and display custom characters on the LCD. This is especially useful if you want to display a character that is not part of the standard ASCII character set.

Technical info: LCDs that are based on the Hitachi HD44780 LCD controller have two types of memories: CGROM and CGRAM (Character Generator ROM and RAM). CGROM generates all the 5 x 8 dot character patterns from the standard 8-bit character codes. CGRAM can generate user-defined character patterns.

For 5 x 8 dot displays, CGRAM can write **up to 8 custom characters** and for 5 x 10 dot displays 4. For more info see the datasheet.

Custom character example code

The following example sketch creates and displays eight custom characters (numbered 0 – 7).

You can open the code in a new window by clicking on the button in the top right corner of the code field.

```
/* Arduino example code to display custom characters on I2C character LCD. More info: www.makerguides.com */

// Include the library:
#include <LiquidCrystal_I2C.h>

// Create lcd object of class LiquidCrystal_I2C:
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4); // Change to (0x27,16,2) for 16x2 LCD.

// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};

byte Bell[] = {
  B00100,
  B01110,
  B01110,
  B01110,
  B11111,
  B00000,
  B00100,
  B00000
};

byte Alien[] = {
  B11111,
  B10101,
  B11111,
  B11111,
  B01110,
  B01010,
  B11011,
  B00000
};

byte Check[] = {
  B00000,
  B00001,
  B00011,
  B10110,
  B11100,
  B01000,
  B00000,
  B00000
};

byte Speaker[] = {
  B00001,
  B00011,
  B01111,
  B01111,
  B01111,
  B00011,
  B00001,
  B00000
};

byte Sound[] = {
  B00001,
  B00011,
  B00101,
  B01001,
  B01001,
  B01011,
  B11011,
  B11000
};

byte Skull[] = {
  B00000,
  B01110,
  B10101,
  B11011,
  B01110,
  B01110,
  B00000,
  B00000
};

byte Lock[] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11011,
  B11011,
  B11111,
  B00000
};

void setup() {
  // Initialize LCD and turn on the backlight:
  lcd.init();
  lcd.backlight();

  // Create new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);

  // Clear the LCD screen:
  lcd.clear();

  // Print a message to the lcd:
  lcd.print("Custom Character");
}

// Print all the custom characters:
void loop() {
  lcd.setCursor(0, 1);
  lcd.write(0);

  lcd.setCursor(2, 1);
  lcd.write(1);

  lcd.setCursor(4, 1);
  lcd.write(2);

  lcd.setCursor(6, 1);
  lcd.write(3);

  lcd.setCursor(8, 1);
  lcd.write(4);

  lcd.setCursor(10, 1);
  lcd.write(5);

  lcd.setCursor(12, 1);
  lcd.write(6);

  lcd.setCursor(14, 1);
  lcd.write(7);
}
```

You should see the following output on the LCD:

(Coming soon)

How the code works

After including the library and creating the LCD object, the custom character arrays are defined. Each array consists of 8 bytes, 1 byte for each row of the 5 x 8 led matrix. In this example, 8 custom characters are created.

```
// Make custom characters:
byte Heart[] = {
  B00000,
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000
};
```

When looking closely at the array, you will see the following. Each row consists of 5 numbers corresponding to the 5 pixels in a 5 x 8 dot character. A 0 means pixel off and a 1 means pixel on.

It is possible to edit each row by hand, but I recommend using this visual tool on GitHub. This application automatically creates the character array and you can click on the pixels to turn them on or off.

In the setup, the custom characters are created with `lcd.createChar(num, data)`.

The first argument in this function is the number of the custom character (0-7) and the second argument is the character array that we created.

```
  // Create new characters:
  lcd.createChar(0, Heart);
  lcd.createChar(1, Bell);
  lcd.createChar(2, Alien);
  lcd.createChar(3, Check);
  lcd.createChar(4, Speaker);
  lcd.createChar(5, Sound);
  lcd.createChar(6, Skull);
  lcd.createChar(7, Lock);
```

In the loop, all the characters are displayed with lcd.write(). As a parameter we use the number of the custom character that we want to display.

```
  lcd.setCursor(0, 1);
  lcd.write(0);
```

Conclusion

In this article I have shown you how to use a character I2C LCD with Arduino. I hope you found it useful and informative. If you did, please **share it with a friend** that also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with these LCDs. If you have any questions, suggestions or if you think that things are missing in this tutorial, **please leave a comment down below**.

Note that comments are held for moderation to prevent spam.

Beginner

Comments

Joe Simone says:
January 6, 2020 at 12:42 pm

Good job. Everything is much clearer to me now. Thank you.
I plan to build a PID Temperature Kiln controller .

Reply

Mostafa abdulaziz says:
September 28, 2019 at 11:04 am

Nice topic thanks

Reply

Rahul Padman says:
August 13, 2019 at 3:40 am

Can’t thank you enough for such an detailed explanation… Great help… 👍 👍

Reply

Richard says:
June 29, 2019 at 1:19 am

Thank you very much for taking the time.

Reply

Dr Bryan Roe says:
May 13, 2019 at 8:53 pm

Thanks very much for your reply Benne. Prior to picking up your reply I’d had a look at the various libraries which I had installed and suspected that was they were the issue. So I decided to download the version from your tutorial and it worked first time!
I am using a number of different LCD modules as part of my STEM Ambassador role to demonstrate what can be done with Arduinos.

Reply

Dr Bryan Roe says:
May 10, 2019 at 1:36 pm

I have tried to load/compile this file into my Arduino Uno, but repeatedly get an error message Liquid\Crystal_I2C – no such file or file directory.

I have a Liquid Crystal I2C library loaded into my IDE.

Help!

Reply

Benne de Bakker says:
May 10, 2019 at 2:03 pm

Hi Bryan,

Thank you for your comment. I suspect you have a different library installed than the one I used in this tutorial. There are several Liquid Crystal I2C libraries available on the web and some have the same name. If you download the library by clicking on the download button in this tutorial or via the GitHub link in the code it should work. I am not sure if you need to remove the previously installed library or not.

Benne

Reply

Trackbacks

TM1637 4-Digit 7-Segment Arduino Tutorial (3 Examples) says:

September 6, 2019 at 3:07 pm
[…] How to control a character I2C LCD with Arduino […]

TM1637 4-Digit 7-Segment Display Arduino Tutorial (3 Examples) says:
September 3, 2019 at 11:23 am
[…] How to control a character I2C LCD with Arduino […]

DHT11/DHT22 Sensors with Arduino Tutorial (2 Examples)says:
August 20, 2019 at 8:52 am
[…] How to control a character I2C LCD with Arduino […]

16x2 Character LCD Arduino Tutorial (wiring diagram + examples) says:
July 8, 2019 at 9:37 am
[…] How to control a character I2C LCD with Arduino […]

How to use HC-SR04 Ultrasonic Sensor with Arduino (5 examples)says:
June 16, 2019 at 12:38 pm
[…] How to control a character I2C LCD with Arduino […]