<https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/>

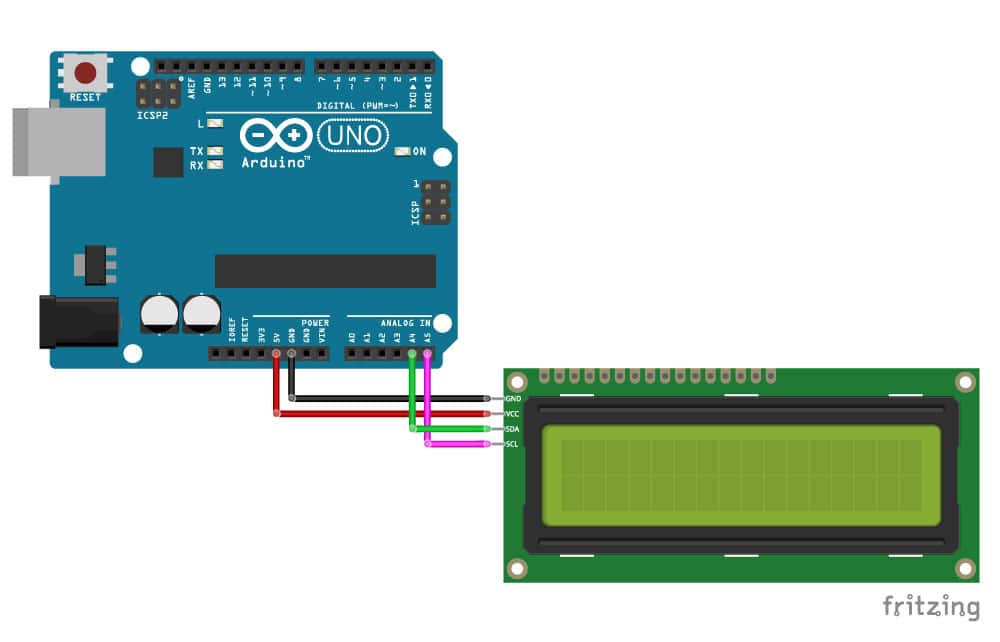
# How to control a character I2C LCD with Arduino

GND - GND

5v - VCC

A4 – SDA

A5 - SCL



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.

## Installing the LiquidCrystal\_I2C Arduino library

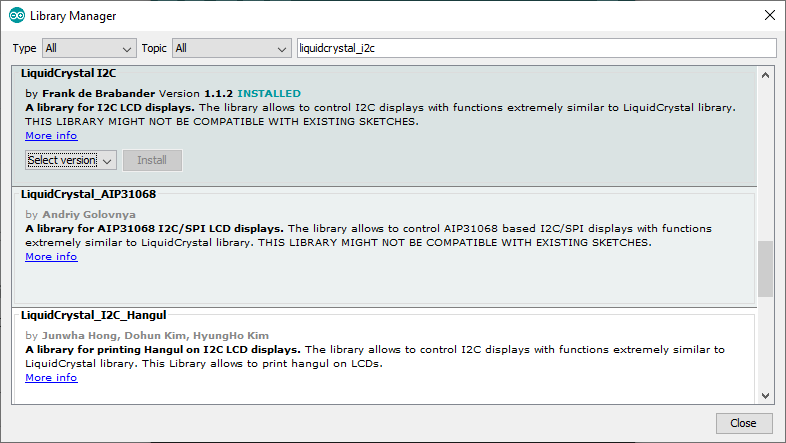
<https://www.makerguides.com/wp-content/uploads/2019/02/LiquidCrystal_I2C-master.zip>

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.

The LiquidCrystal\_I2C library works in combination with the **Wire.h** library which allows you to communicate with I2C devices. This library comes pre-installed with the Arduino IDE.

To install this library, go to Tools > Manage Libraries (Ctrl + Shift + I on Windows) in the [Arduino IDE](https://www.arduino.cc/en/main/software). The Library Manager will open and update the list of installed libraries.

Now search for ‘liquidcrystal\_i2c’ and look for the library by **Frank de Brabander**. Select the latest version and then click Install.



The library does include some examples that you can use, but you will have to modify them to match your hardware setup.

## 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 copy the code 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)

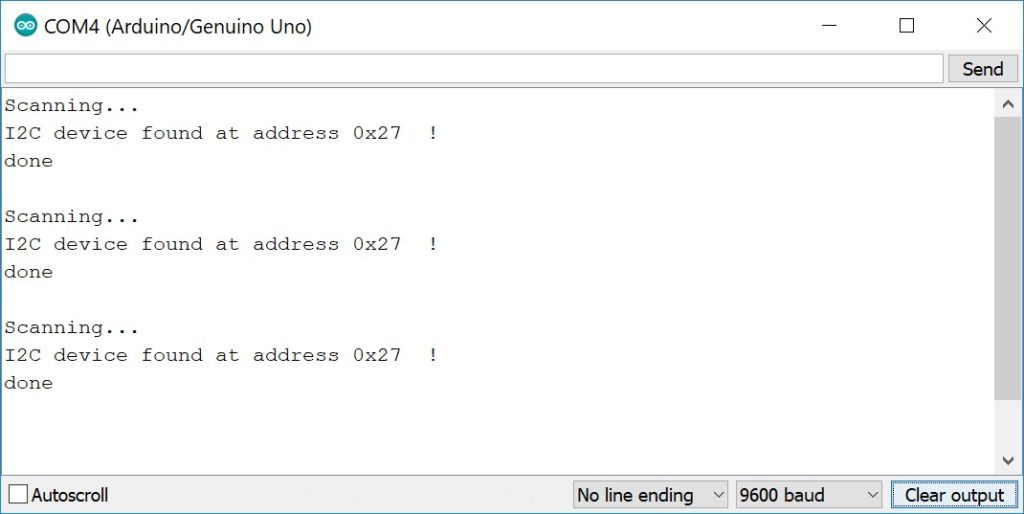
Serial.println("No I2C devices found\n");

else

Serial.println("done\n");

delay(5000);

}



## Basic Arduino example code for I2C LCD

/\* 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, 16, 2); // Change to (0x27,20,4) for 20x4 LCD.

void setup() {

// Initiate the LCD:

lcd.init();

lcd.backlight();

}

void loop() {

// Print 'Hello World!' on the first line of the LCD:

lcd.setCursor(2, 0); // Set the cursor on the third 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");

}

### 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 20×4 LCD, change this line to LiquidCrystal\_I2C(0x27,20,4);

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, 16, 2); // Change to (0x27,20,4) for 20x4 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 third column and the first row of the LCD with lcd.setCursor(2,0). 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](https://www.arduino.cc/reference/en/language/variables/data-types/string/). When you want to print numbers, no quotation marks are necessary. For example lcd.print(12345).

void loop() {

lcd.setCursor(2, 0); // Set the cursor on the third 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.

lcd.print("LCD tutorial"); // Print the string "LCD tutorial".

}

<https://www.makerguides.com/hc-sr04-arduino-tutorial/#example-code-hc-sr04-with-dht11-and-i2c-lcd>

### 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.

### 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!”

### 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.

### 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.