LIQUID CRYSTAL DISPLAY (LCD) with I2C Adapter

1
16
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
Hello World!
LCD Tutorial
Last Minute ENGINEERS.com

If you've ever tried connecting an LCD display to an Arduino, you've probably noticed it requires quite a few Arduino pins. Even when using 4-bit mode, you still need seven connections – which takes up about half of the Arduino's available digital I/O pins.

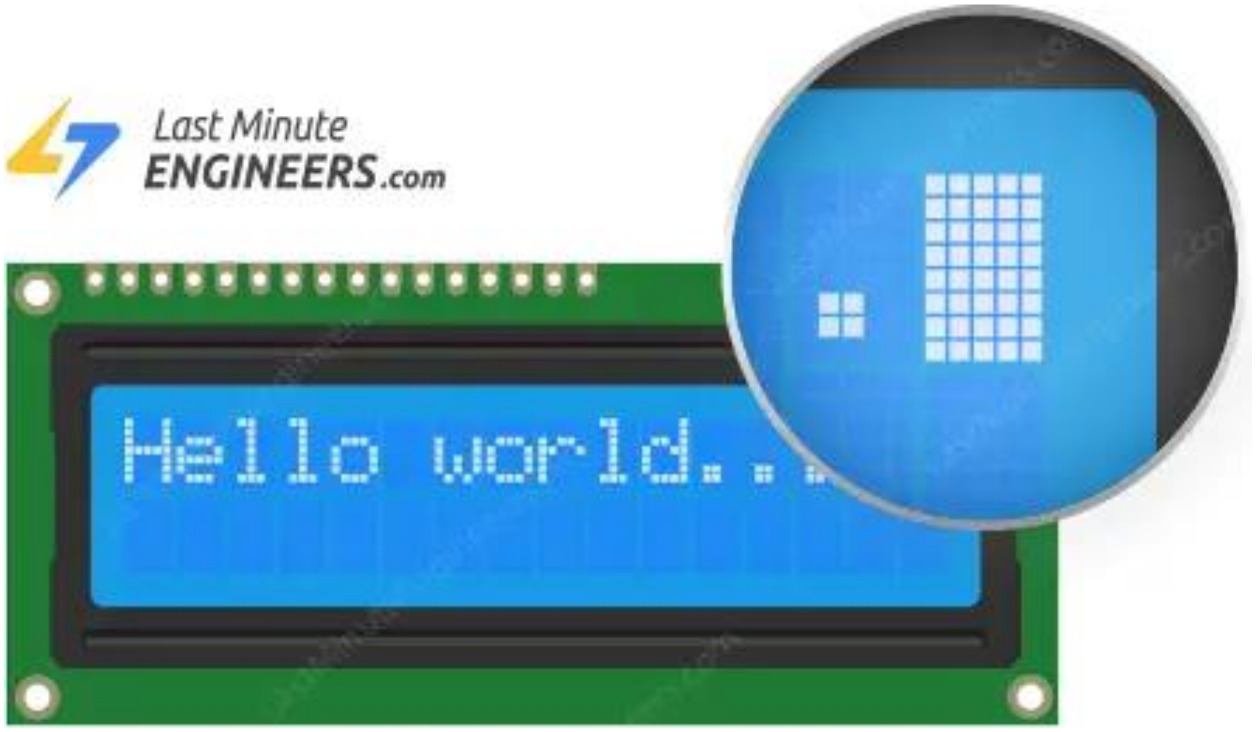
A better solution is to use an I2C LCD display. It only needs two pins that aren't even part of the regular digital I/O pin set. Even better, these pins can be shared with other I2C devices. This makes your wiring much simpler and saves those valuable pins for connecting other sensors or components in your projects.

Hardware Overview

A typical I2C LCD display consists of two main parts: an HD44780-based character LCD display and an I2C LCD adapter. Let's explore both of these components in more detail.

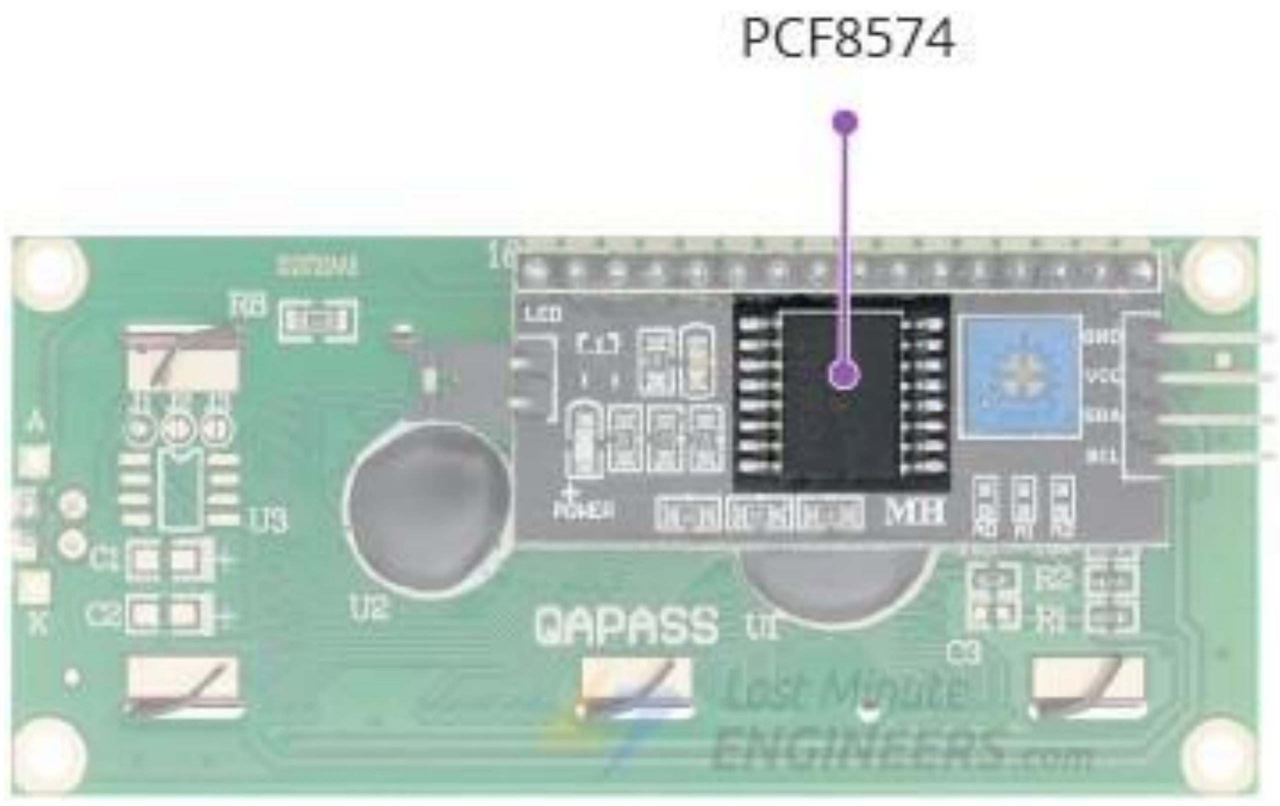
- ### Character LCD Display

Character LCDs are specially designed to display letters, numbers, and symbols. A 16×2-character LCD, for example, can show 16 characters across each line, with two lines total.

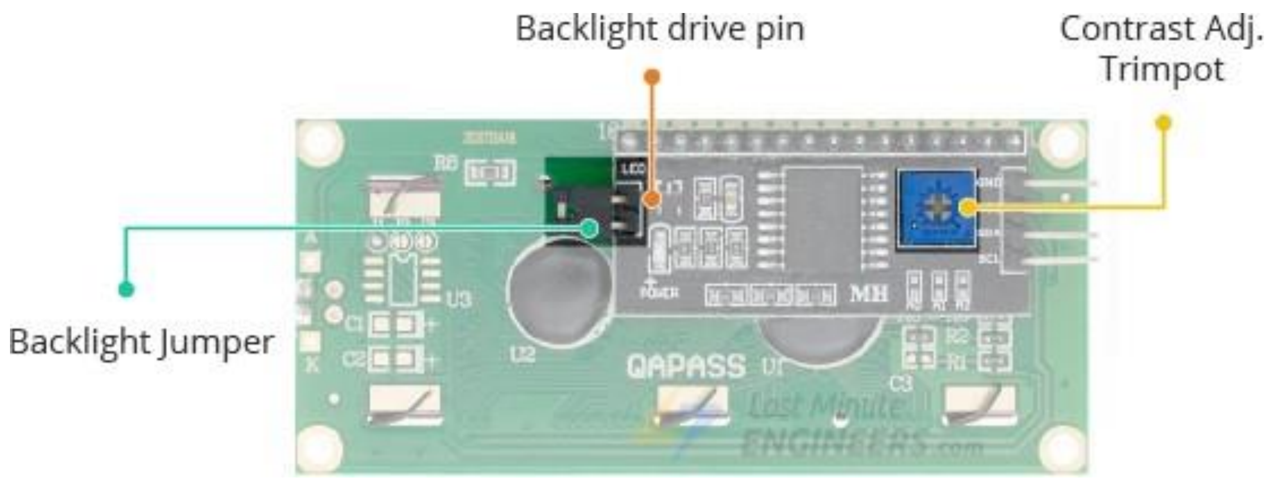


- ### I2C LCD Adapter

The key component of this adapter is an 8-bit I/O expander chip called PCF8574. This clever chip converts the I2C data from your Arduino into the parallel data that the LCD display needs to function.



The adapter board also includes a small trimpot that lets you make fine adjustments to the display's contrast.

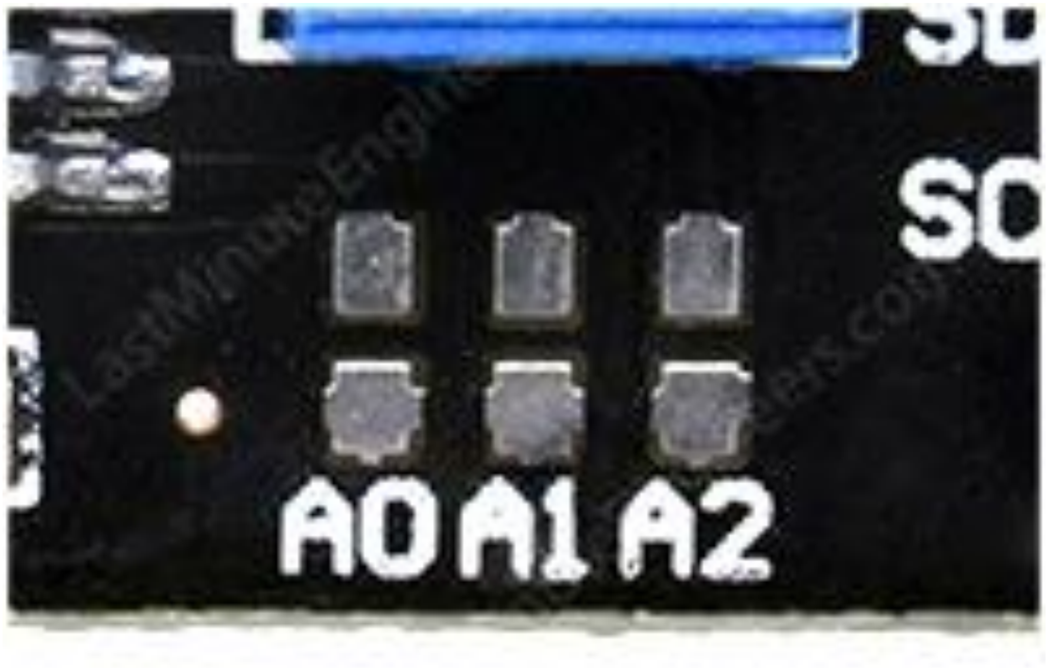


You'll notice there's a jumper on the board that supplies power to the backlight. If you want to control how bright the backlight is, you can simply remove this jumper and connect an external voltage source to the header pin marked 'LED'.

- ## I2C Address of LCD

If you have multiple devices on the same I2C bus, you may need to set a different I2C address for the LCD adapter to avoid conflicts with other I2C devices.

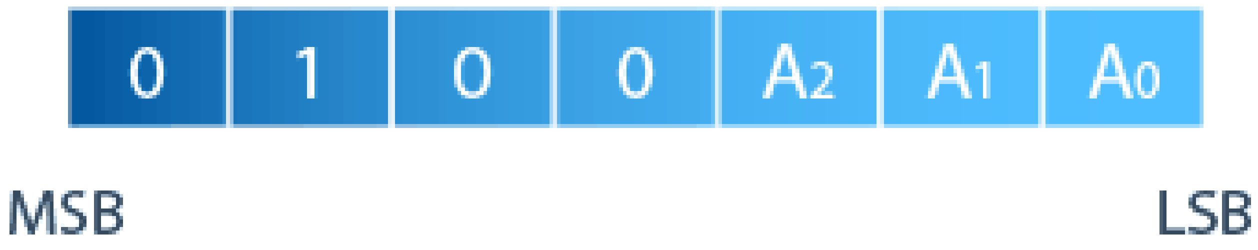
For this purpose, the adapter comes with three solder jumpers/pads (labeled A0, A1, and A2). For a single LCD, you can leave these untouched; however, if you connect multiple I2C devices or LCDs, you'll need to make sure each has a unique address by modifying these jumpers. You can change the address by shorting a jumper with a small blob of solder.



An important thing to know is that different companies, like Texas Instruments and NXP Semiconductors, make the same PCF8574 chip. The I2C address of your LCD depends on which company manufactured the chip.

If your LCD has Texas Instruments' PCF8574 chip:

According to Texas Instruments' datasheet, the three address selection bits (A0, A1, and A2) are located at the end of the 7-bit I2C address register.

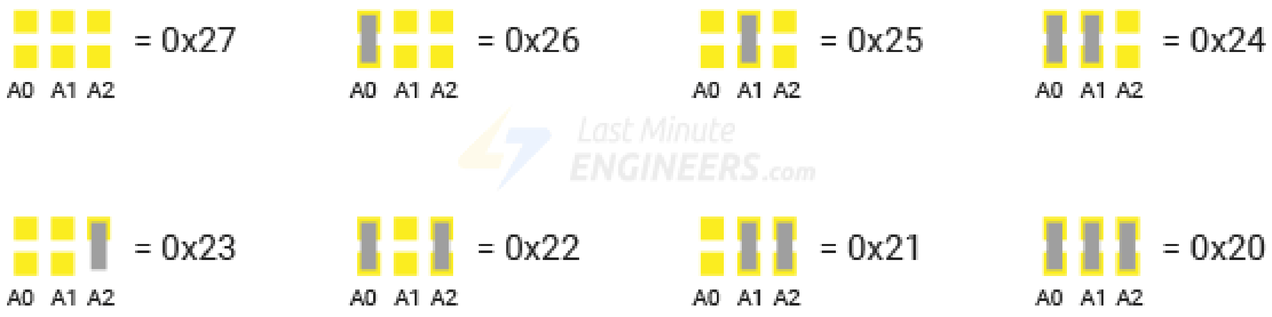


Since there are three address inputs that can be either HIGH or LOW, there are eight possible combinations (2^3 = 8) of addresses.

All three address inputs are pulled HIGH by default using onboard pullup resistors. This gives the PCF8574 a default I2C address of 0x27.

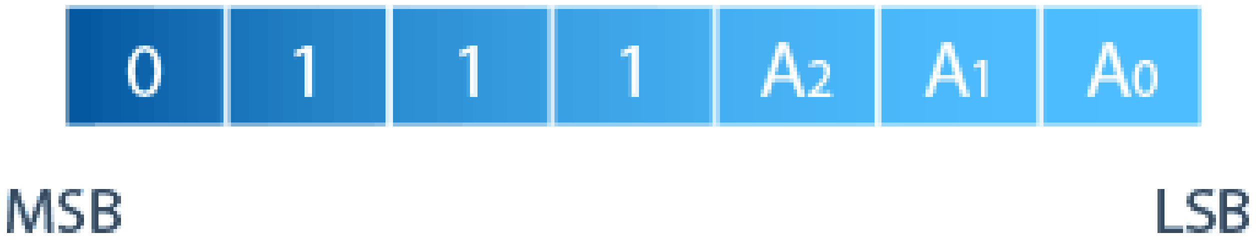
When you short a solder jumper, you pull that address input LOW. If you were to short all three jumpers, the address would change to 0x20. So the range of possible addresses goes from 0x20 to 0x27.

You can set different I2C addresses according to this table:



If your LCD has NXP's PCF8574 chip:

According to NXP Semiconductors' datasheet, the three address selection bits (A0, A1, and A2) are also located at the end of the 7-bit I2C address register. However, the remaining bits in the address register are different from the Texas Instruments chip.

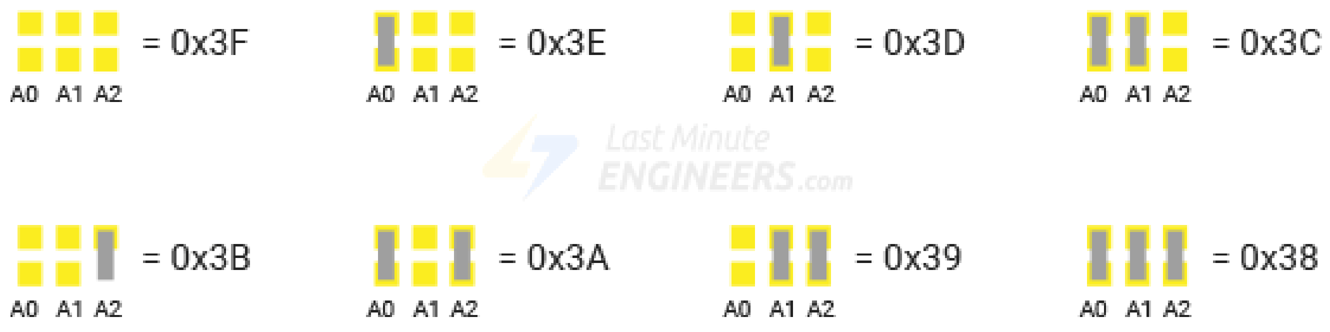


Just like with the Texas Instruments chip, there are eight possible address combinations since the three inputs can be either HIGH or LOW.

All three address inputs are pulled HIGH by default using onboard pullup resistors. This gives the PCF8574 a default I2C address of 0x3F.

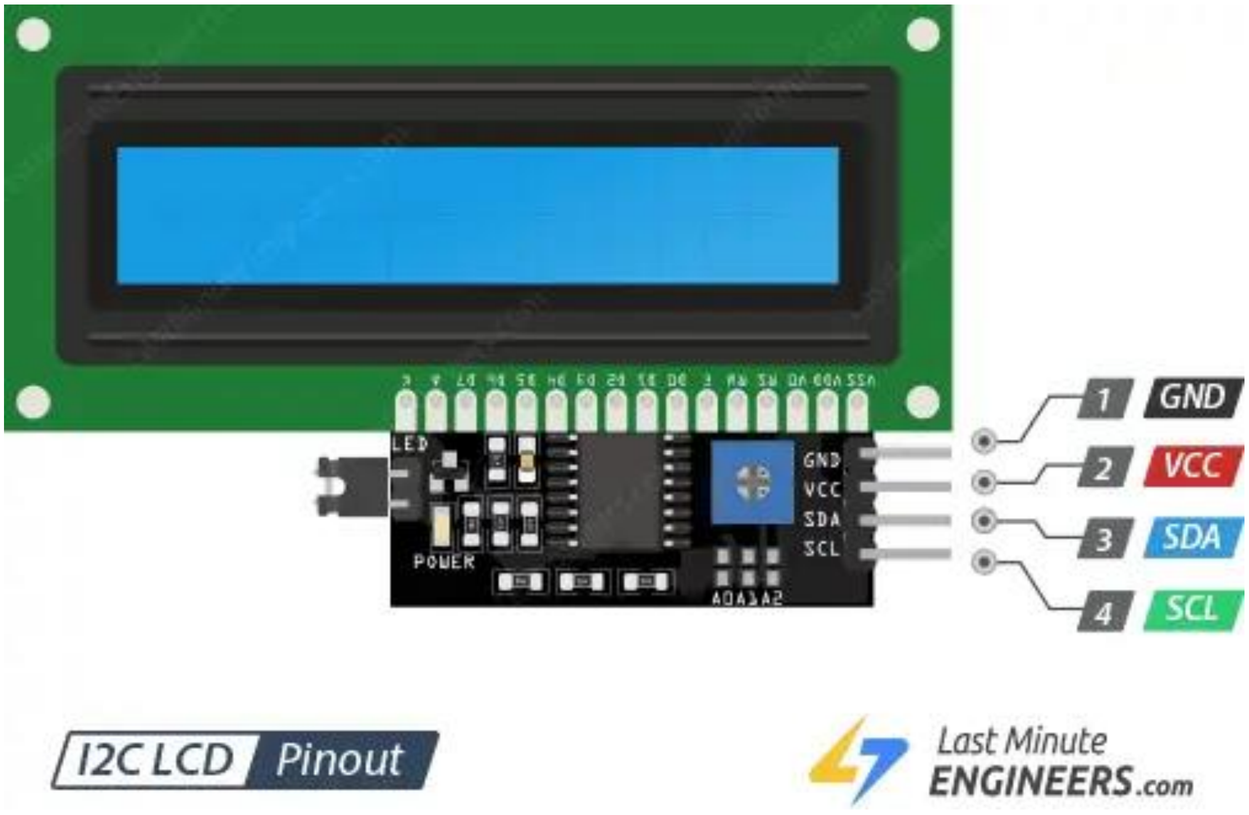
When you short a solder jumper, you pull that address input LOW. If you were to short all three jumpers, the address would change to 0x38. So the range of possible addresses goes from 0x38 to 0x3F.

You can set different I2C addresses according to this table:



I2C LCD Display Pinout

The I2C LCD Display has only four pins. The following is the pinout:



GND is a ground pin.

VCC is the power supply pin. Connect it to the 5V output of the Arduino or an external 5V power supply.

SDA is the I2C data pin.

SCL is the I2C clock pin.

Wiring an I2C LCD Display to an Arduino

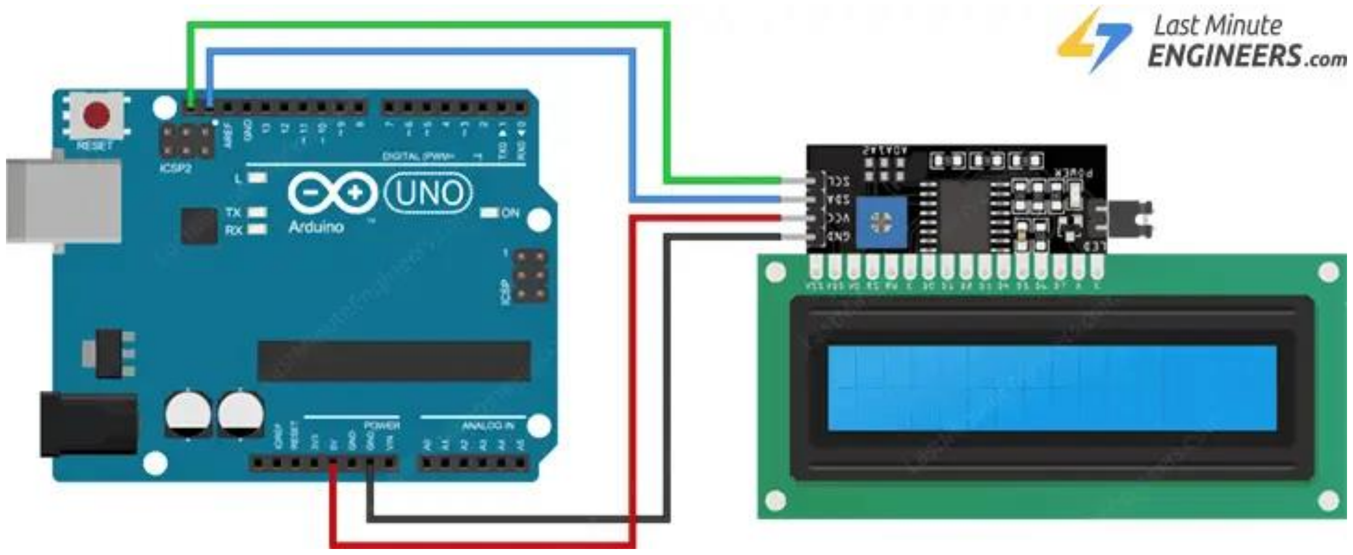
Connecting an I2C LCD is much simpler than connecting a standard LCD. You only need to connect four pins instead of many more.

First, connect the VCC pin to the Arduino’s 5V output and the GND pin to the Arduino’s ground.

Next, we need to connect the pins used for I2C communication. It’s important to know that different Arduino boards have different I2C pins, and these must be connected correctly. On Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) pins are also found on the pin headers near the AREF pin. However, internally, they are the same as the A4 (SDA) and A5 (SCL) pins.

I2C LCD	ARDUINO
VCC	5V
GND	GND
SCL	SCL OR A5
SDA	SDA OR A4

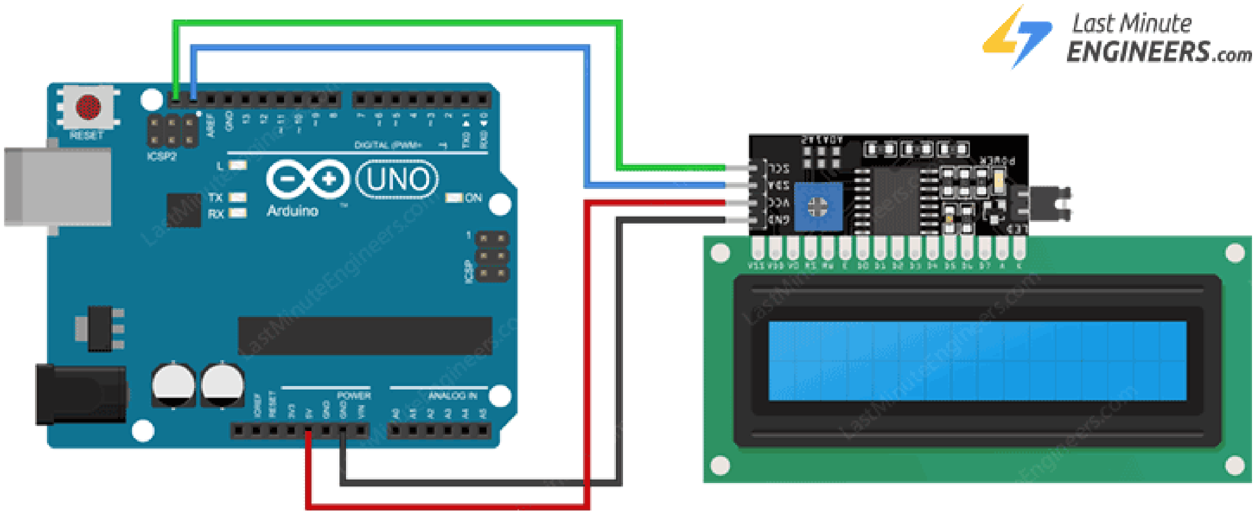
This diagram shows you exactly how to connect everything:



- **Adjusting The LCD Contrast**

After you finish wiring the LCD to your Arduino, you'll need to adjust the contrast of the LCD display. Look for a small blue trimpot on the I2C adapter board – this controls the LCD's contrast.

Now, power up your Arduino. You should see the backlight glow up right away. Take a small screwdriver and gently turn the potentiometer knob. As you adjust it, you'll start to see the first row of rectangles appear on the screen. These rectangles are where characters will show up. If you can see those rectangles clearly, congratulations—your LCD is working perfectly and is ready for you to display text!

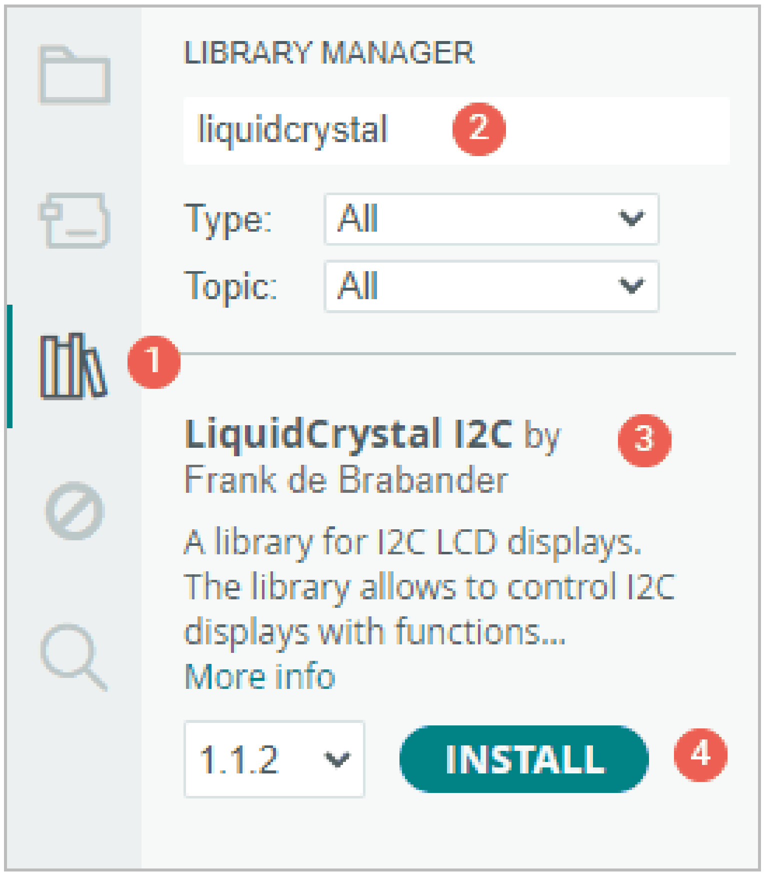


- **Library Installation**

To control the I2C LCD, we will use the Arduino LiquidCrystal_I2C library. This library makes writing to the LCD much easier because it handles all the complicated I2C communication in the background. There are several versions of this library available, but one of the most reliable ones is the version created by Frank de Brabander.

To install the library:

1. First open your Arduino IDE program. Then click on the Library Manager icon on the left sidebar.
2. Type “liquidcrystal” in the search box to filter your results.
3. Look for the “LiquidCrystal I2C library” library created by Frank de Brabander.
4. Click the Install button to add it to your Arduino IDE.



liquidcrystal

Type: All

Topic: All

LiquidCrystal I2C by Frank de Brabander

A library for I2C LCD displays. The library allows to control I2C displays with functions...

1.1.2

Basic Arduino Sketch – Hello World

With the hardware connected and the library installed, we can write a simple Arduino sketch to display text on the LCD. In this example, we'll print 'Hello World!' on the first line of the LCD and 'LCD Tutorial' on the second line.

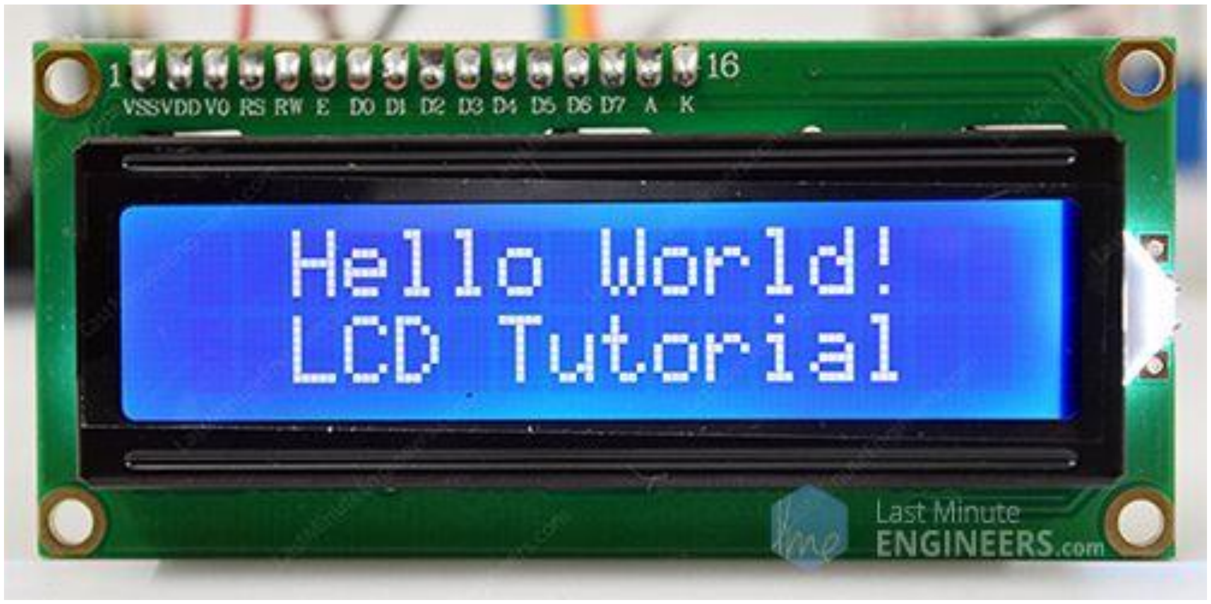
Before you upload the sketch, you need to make two important changes to make it work with your specific LCD. You must enter the correct I2C address of your LCD (which we found earlier) and specify the display dimensions in the `LiquidCrystal_I2C()` constructor. If you're using a 16×2 character LCD, enter 16 and 2; if you're using a 20×4 character LCD, enter 20 and 4.

```
// enter the I2C address and the dimensions of your LCD here
LiquidCrystal_I2C lcd(0x3F, 16, 2);
```

Once you've made these changes, you're ready to try the complete sketch:

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2);  // set the LCD address to 0x3F for a 16
chars and 2 line display

void setup() {
  lcd.init();
  lcd.clear();
  lcd.backlight();      // Make sure backlight is on

  // Print a message on both lines of the LCD.
  lcd.setCursor(2,0);   //Set cursor to character 2 on line 0
  lcd.print("Hello world!");

  lcd.setCursor(2,1);   //Move cursor to character 2 on line 1
  lcd.print("LCD Tutorial");
}

void loop() {
}
```

After uploading this code to your Arduino, this is what you should see on the screen:


1
VSS VDD V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
16
Hello World!
LCD Tutorial
Last Minute ENGINEERS.com

- **Code Explanation:**

The sketch begins by including the LiquidCrystal_I2C library, which gives us all the functions we need to control the LCD.

```
#include <LiquidCrystal_I2C.h>
```

Next, we create an object of the LiquidCrystal_I2C class. The LiquidCrystal_I2C constructor needs three pieces of information: the I2C address, the number of columns, and the number of rows of your display.

```
LiquidCrystal_I2C lcd(0x3F,16,2);
```

In the setup function, we call three important functions. First, the `init()` function initializes the interface to the LCD. Second, the `clear()` function erases anything on the LCD screen and moves the cursor to the upper-left corner. Third, the `backlight()` function turns on the LCD's backlight so we can see the text.

```
lcd.init();
lcd.clear();
lcd.backlight();
```

The `setCursor(2, 0)` function moves the cursor to the third column of the first row. (Remember that counting starts at 0, so column 2 is actually the third column.) The cursor position tells the LCD where to place new text on the screen. The upper left corner is considered position (0,0).

```
lcd.setCursor(2,0);
```

Next, we use the `print()` function to display the text "Hello world!" on the LCD.

```
lcd.print("Hello world!");
```

Similarly, the next two lines of code move the cursor to the third column of the second row and print 'LCD Tutorial' on the LCD.

```
lcd.setCursor(2,1);
lcd.print("LCD Tutorial");
```

Other useful functions of the LiquidCrystal_I2C Library

The LiquidCrystal_I2C object gives you many helpful functions to control your LCD. Here are some of the most useful ones:

- `lcd.home()` moves the cursor back to the upper-left corner of the LCD (the first position on the first row). Unlike `clear()`, it doesn't erase what's already on the screen – it just moves the cursor to the starting position.

- `lcd.blink()` and `lcd.noBlink()` turn on or off a blinking block cursor. When turned on with `blink()`, you'll see a solid block that flashes on and off at the current cursor position. This is great for getting the user's attention or showing where text will appear next. If you don't want this blinking block, use `noBlink()` to turn it off.

- `lcd.cursor()` and `lcd.noCursor()` control whether an underscore line (`_`) appears at the position where the next character will be written. The `cursor()` function shows this line, while `noCursor()` hides it. This is different from the blinking block – it's just a simple line showing where the next character will go.

- `lcd.display()` and `lcd.noDisplay()` let you turn the entire display on or off without erasing anything. When you use `noDisplay()`, the screen goes blank, but all the text stays stored in the LCD's memory. When you call `display()` again, everything reappears! This is perfect for creating blinking effects or saving power when the display isn't needed.

Try experimenting with these functions in your own code to see how they work. They'll help you create more interactive and dynamic displays for your projects!

REFERENCE:

https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/