How to Create Custom LCD Characters for Arduino

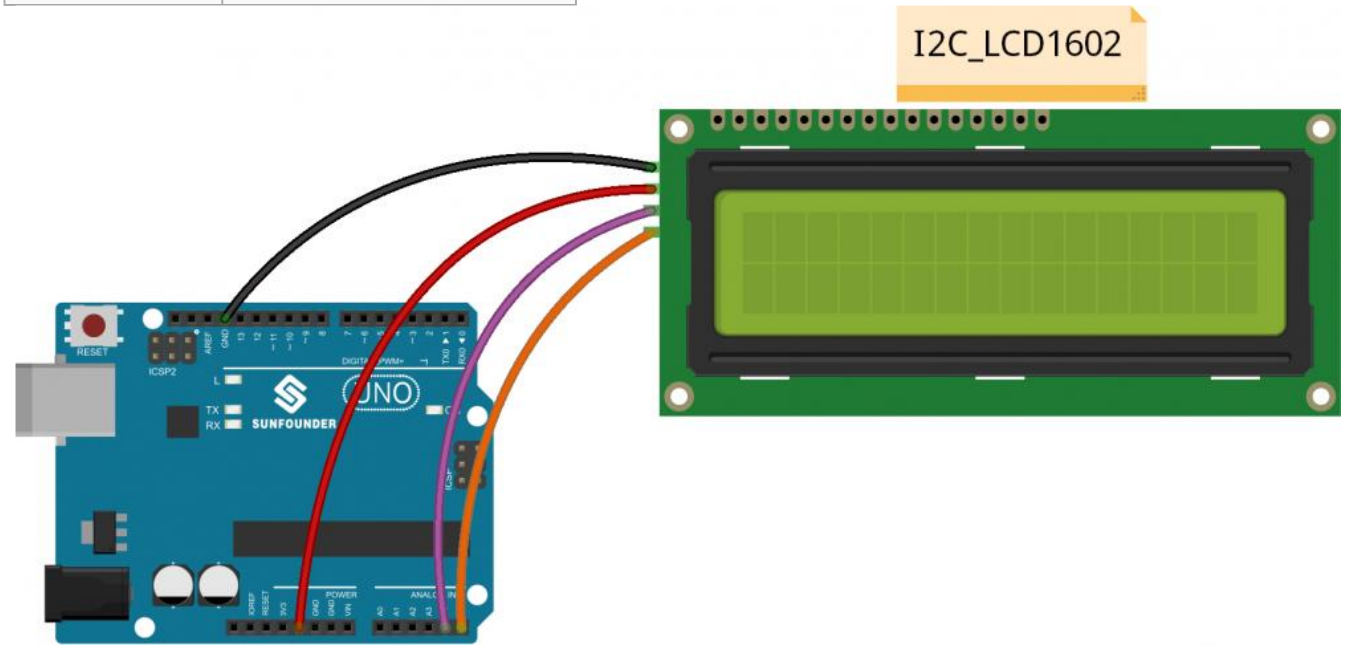
The I2C LCD1602 is a popular display module capable of showing letters, numbers, and standard characters, which are easily programmable with available libraries. However, to display custom symbols or unique patterns, you need to design these using the LCD's 8x5 pixel matrix.

Custom characters are created by defining a byte array, with each byte representing a row in the matrix. By setting bits in these bytes, you can craft various designs, from simple shapes to intricate icons. These custom designs are stored in the LCD's memory and can be displayed like standard characters.

This approach to creating and displaying custom symbols on the LCD1602 adds a creative and personalized element to electronic projects, making it ideal for those looking to go beyond standard text and numbers.

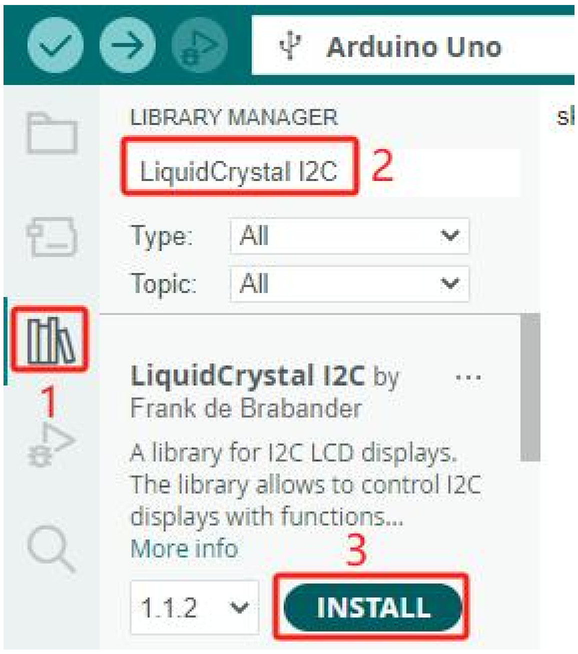
Build the Circuit

I2C LCD1602	Arduino Board
GND	GND
VCC	5V
SDA	A4 /pin 20 mega2560
SCL	A5 /pin 21 mega2560



Install the Library

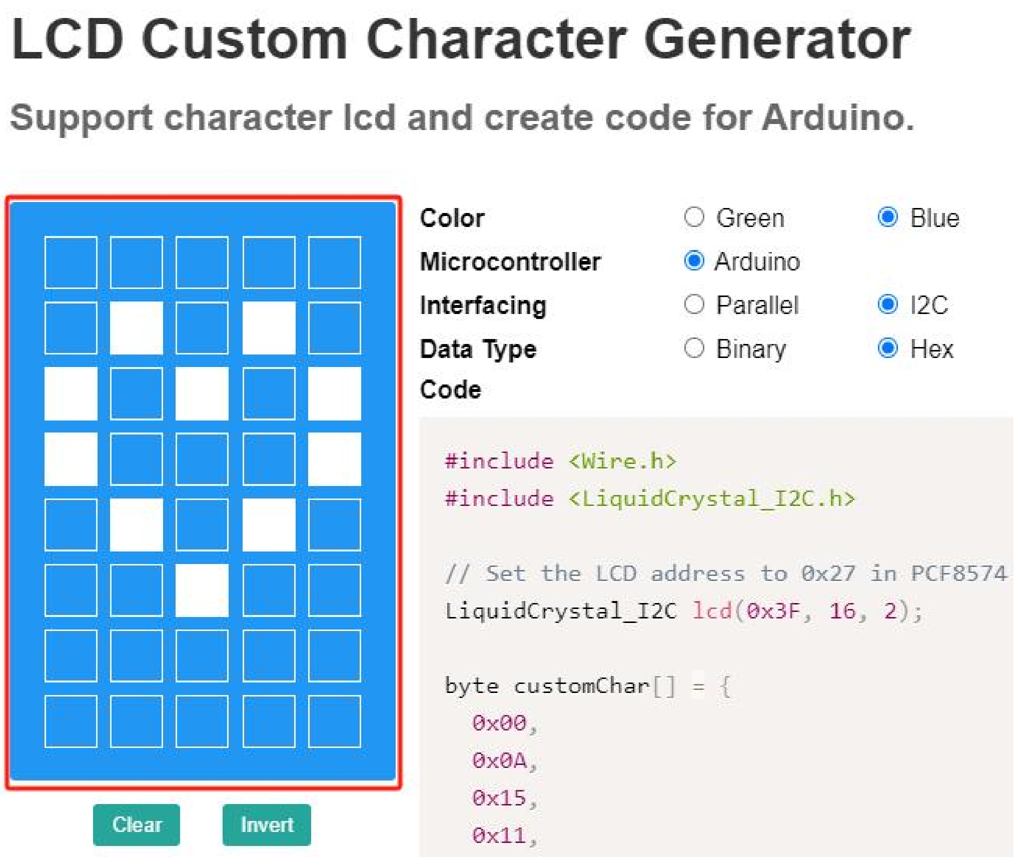
Open the **Library Manager**, enter **LiquidCrystal I2C**, and click **INSTALL** when it appears.


Arduino Uno
LIBRARY MANAGER
LiquidCrystal I2C
2
Type: All
Topic: All
1
LiquidCrystal I2C by Frank de Brabander
A library for I2C LCD displays. The library allows to control I2C displays with functions...
More info
3
1.1.2
INSTALL

Steps

1. Visit the link: https://maxpromer.github.io/LCD-Character-Creator/

2. Draw the pattern you want. For example, here I have drawn a heart-shaped pattern.



LCD Custom Character Generator

Support character lcd and create code for Arduino.

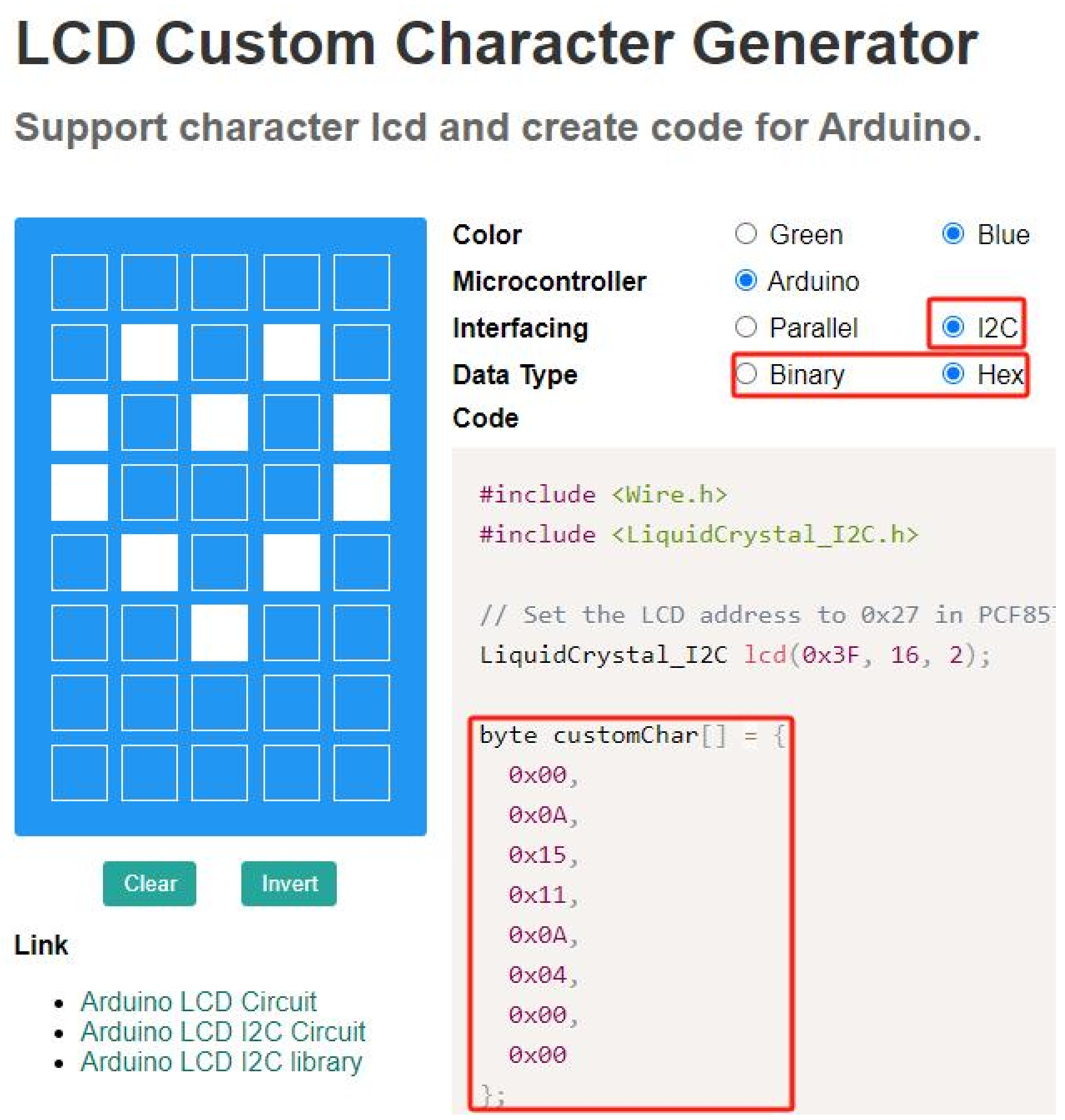
Color: ○ Green [✓ Blue]
Microcontroller: [✓ Arduino]
Interfacing: ○ Parallel [✓ I2C]
Data Type: ○ Binary [✓ Hex]
Code

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

// Set the LCD address to 0x27 in PCF8574
LiquidCrystal_I2C lcd(0x3F, 16, 2);

byte customChar[] = {
  0x00,
  0x0A,
  0x15,
  0x11,
```

3. Set the Interfacing to I2C. Choose the Data Type as either Binary or Hex. After that, you will get the desired array for your custom character.


LCD Custom Character Generator
Support character lcd and create code for Arduino.
Clear
Invert
Link
Arduino LCD Circuit
Arduino LCD I2C Circuit
Arduino LCD I2C library

Color ( ) Green [✓ Blue]
Microcontroller [✓ Arduino]
Interfacing ( ) Parallel [✓ I2C]
Data Type ( ) Binary [✓ Hex]
Code

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

// Set the LCD address to 0x27 in PCF85
LiquidCrystal_I2C lcd(0x3F, 16, 2);

byte customChar[] = {
  0x00,
  0x0A,
  0x15,
  0x11,
  0x0A,
  0x04,
  0x00,
  0x00
};
```

4. Copy this array into the code below, replacing the **customChar[]** array.

```
#include <Wire.h> // Include the Wire library for I2C communication
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal_I2C
library for controlling I2C LCD displays

// Initialize an LCD object with I2C address set to 0x27, and specify
its size as 16 columns and 2 rows
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define a custom character using an 8x5 pixel matrix
byte customChar[] = {
  0x00, // Row 1: All pixels off
  0x0A, // Row 2: Turn on pixels 2 and 4
  0x15, // Row 3: Alternating pixels on
  0x11, // Row 4: Turn on pixels 1 and 5
  0x0A, // Row 5: Turn on pixels 2 and 4
  0x04, // Row 6: Only pixel 3 is on
  0x00, // Row 7: All pixels off
  0x00  // Row 8: All pixels off
};

void setup() {
  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  lcd.createChar(0, customChar); // Create the custom character and
store it in location 0
  lcd.setCursor(7,0); // Move the cursor to the 0 row 7 col
  lcd.write(0); // Display the custom character stored in location 0
}

void loop() {
  // The loop is empty and does nothing in this case
}
```

5. After selecting your board and port, click 'Upload' to upload the code. Then, you will be able to see the heart-shaped pattern displayed on the I2C LCD1602.

More

You can also create several patterns to achieve an animated effect. For example, you could make a human figure move from the left side to the right side of the display.

```
#include <Wire.h> // Include the Wire library for I2C communication
#include <LiquidCrystal_I2C.h> // Include the LiquidCrystal_I2C
library for controlling I2C LCD displays

// Initialize an LCD object with the I2C address set to 0x27, and specify
its size as 16 columns and 2 rows
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define the custom characters for the animation in hexadecimal
byte man1[] = {
  0x04, // Row 1: Middle pixel on
  0x04, // Row 2: Middle pixel on
  0x15, // Row 3: Pixels 1, 3, and 5 on
  0x0E, // Row 4: Pixels 2, 3, and 4 on
  0x04, // Row 5: Middle pixel on
  0x0A, // Row 6: Pixels 2 and 4 on
  0x11, // Row 7: Pixels 1 and 5 on
  0x00  // Row 8: All pixels off
};

byte man2[] = {
  0x00, // Row 1: All pixels off
  0x04, // Row 2: Middle pixel on
  0x15, // Row 3: Pixels 1, 3, and 5 on
  0x0E, // Row 4: Pixels 2, 3, and 4 on
  0x04, // Row 5: Middle pixel on
  0x0A, // Row 6: Pixels 2 and 4 on
  0x11, // Row 7: Pixels 1 and 5 on
  0x00  // Row 8: All pixels off
};

byte man3[] = {
  0x00, // Row 1: All pixels off
  0x00, // Row 2: All pixels off
  0x15, // Row 3: Pixels 1, 3, and 5 on
  0x0E, // Row 4: Pixels 2, 3, and 4 on
  0x04, // Row 5: Middle pixel on
  0x0A, // Row 6: Pixels 2 and 4 on
  0x11, // Row 7: Pixels 1 and 5 on
  0x04  // Row 8: Middle pixel on
};

void setup() {
  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  // Create the custom characters and store them in the LCD's memory
  lcd.createChar(0, man1);
  lcd.createChar(1, man2);
  lcd.createChar(2, man3);
}

void loop() {
  for (int i = 0; i < 3; i++) {
    lcd.clear(); // Clear the LCD screen
    lcd.setCursor(i*5, 0); // Set the cursor to different positions to
simulate movement
    lcd.write(i); // Write the custom character to the LCD
    delay(1000); // Delay to make the animation visible
  }
}
```