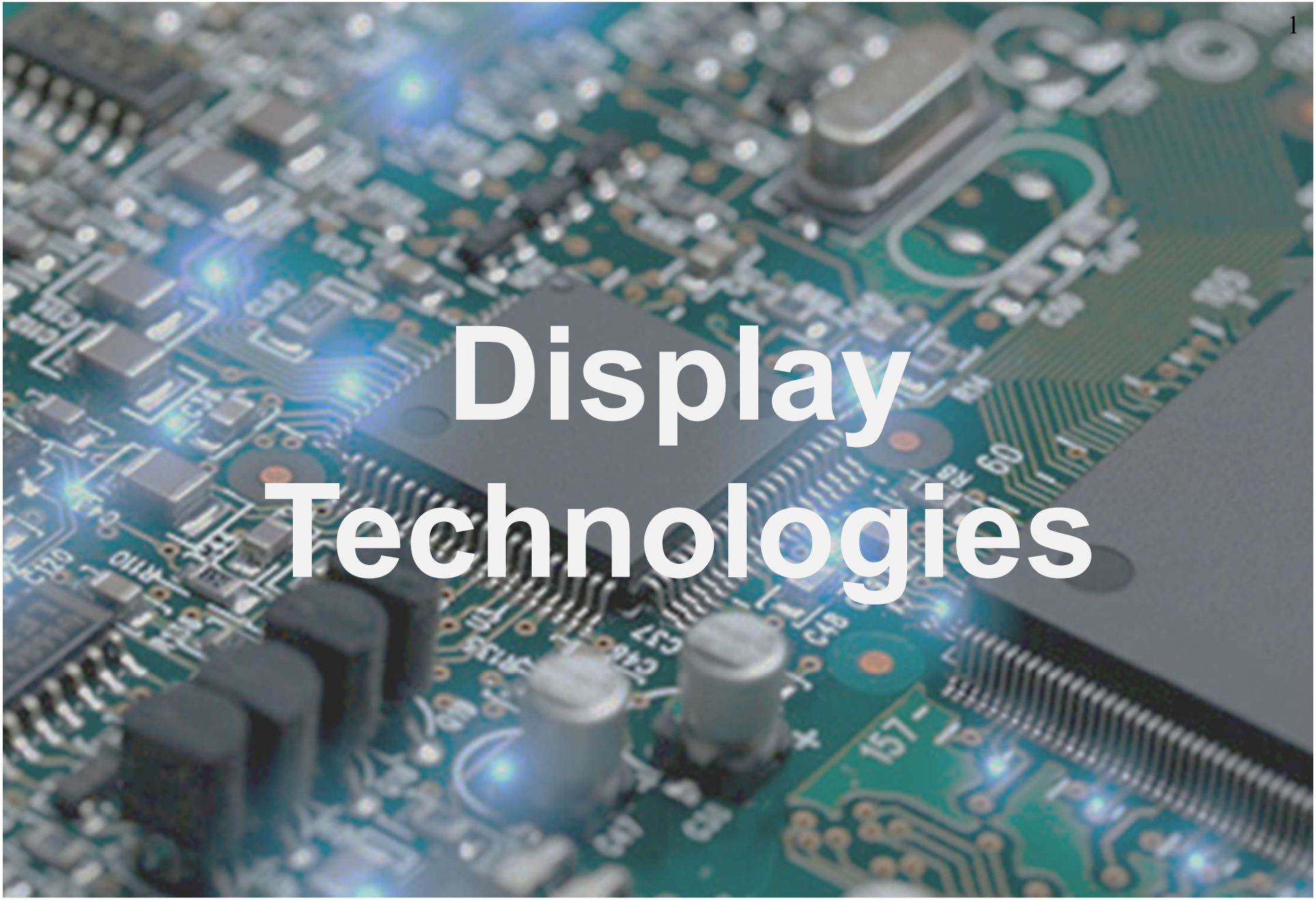


Display Technologies

Display Technologies

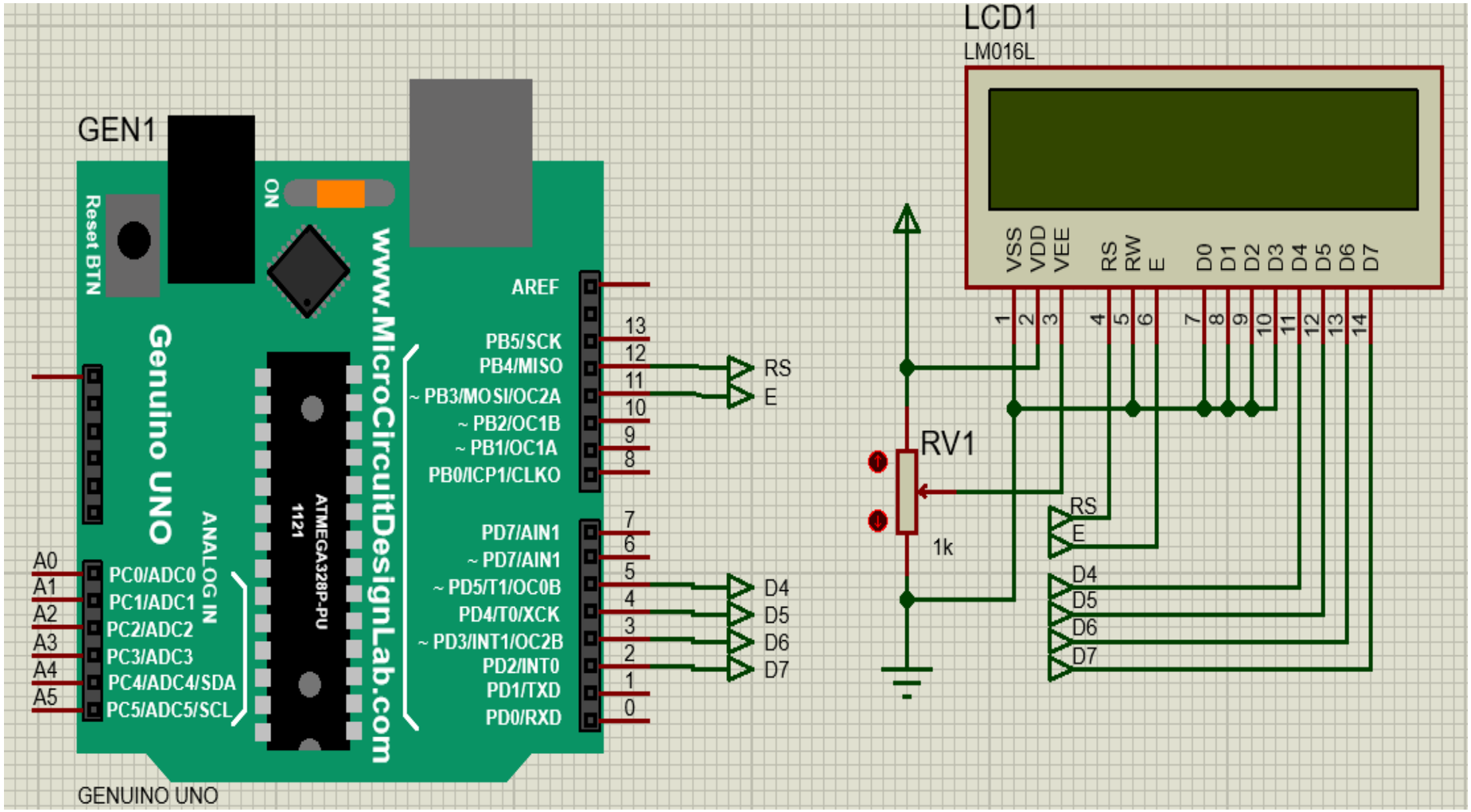
Contents

- Objectives
- Liquid Crystal Display 16X02
- Liquid Crystal Display 16X02 I2C
- ST7735 Display
- ST7789 Display
- OLED SSD1306
- Nokia 5110 Graphic LCD

Objectives

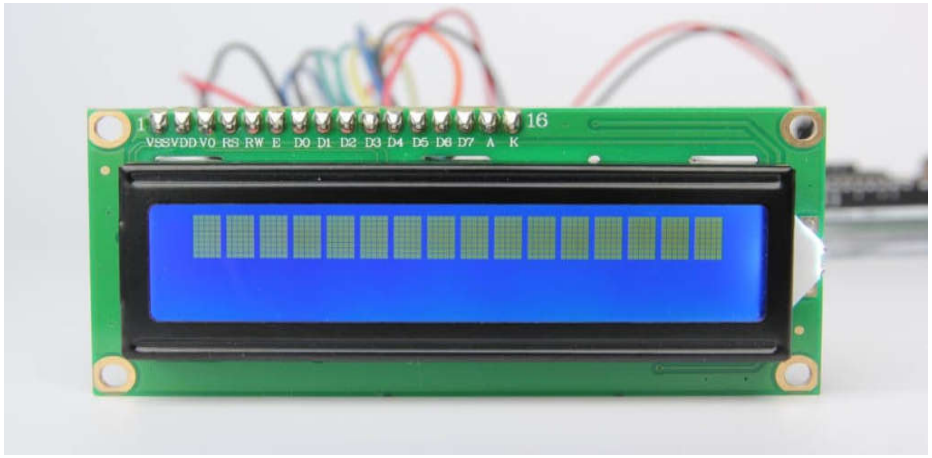
- In this tutorial, you will learn:
  - To understand how to implement Liquid Crystal Display.
  - To know how to make Liquid Crystal Display I2C projects.
  - To become familiar using ST7735R TFT Display.
  - To understand how to make ST7789 IPS Display projects.
  - To know how to use OLED SSD1306 Display.
  - To be able to start Nokia 5110 Graphic LCD.

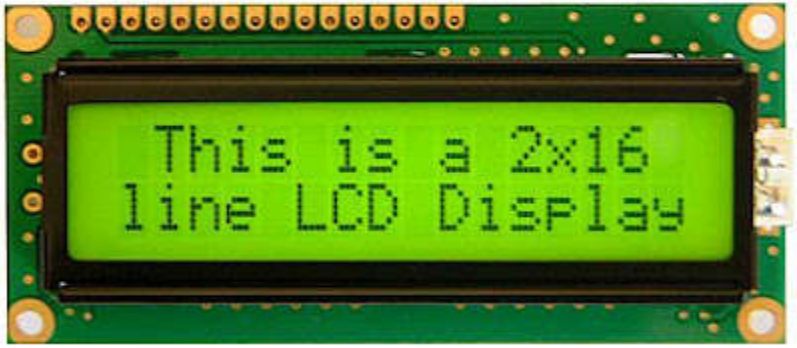
Liquid Crystal Display 16X02



Liquid Crystal Display 16X02

-Common Display
-Parallel interface from Hitachi HD44780
-Variants (16x4, 16x1,8x16, 20x4)
-Text or characters only
-Pixels=5x8 pixels
-Background-blue/green

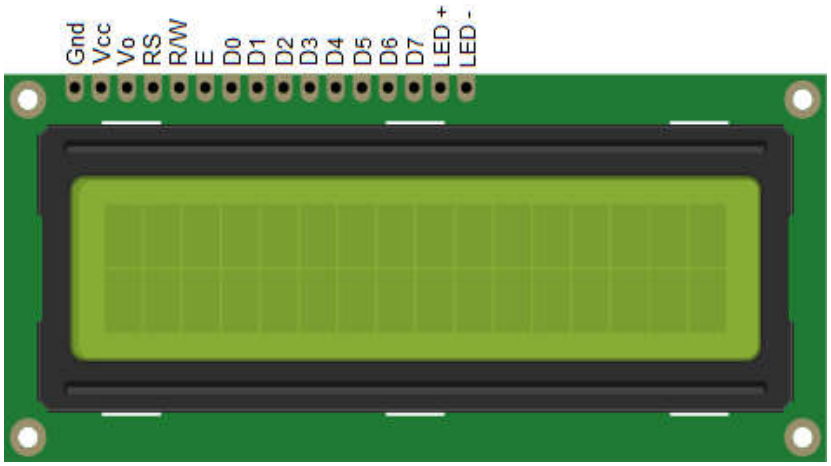


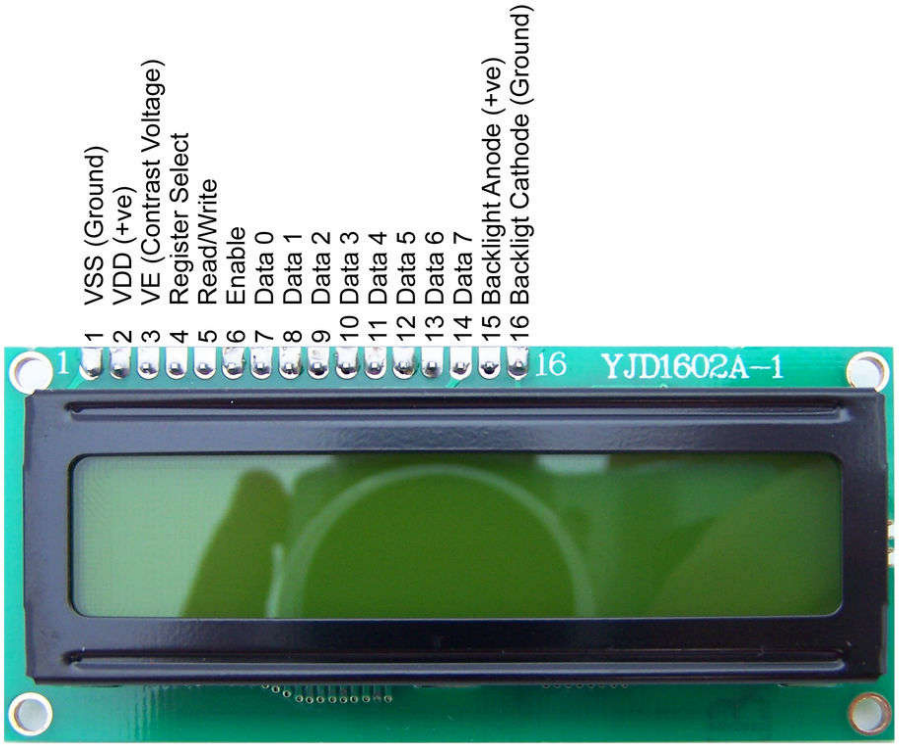


Liquid Crystal Display 16X02

16x2 Character LCD Pinouts

1.GND/VSS- ground of LCD
2.VCC/VDD- +5V supply
3. contrast of LCD using potentiometer to adjust
4.RS (Register Select)- to send commancds or data
5.R/W (Read/Write)- to control reading or sending data. Usually it is tied this pin in ground to disable read mode because we only using write mode.
6.E/EN (Enable)- to enable display.
- Enale LOW- dont care the LCD Display
- Enable HIGH- process the incoming data
7-14.Data (D0-D7)- 8bit-parallel data pins
- 4bit data can be used
15.LED+/Anode (A)- + or anode pin for backlight
16.LED-/Cathode (K)- - (GND)or cathode pin for backlight

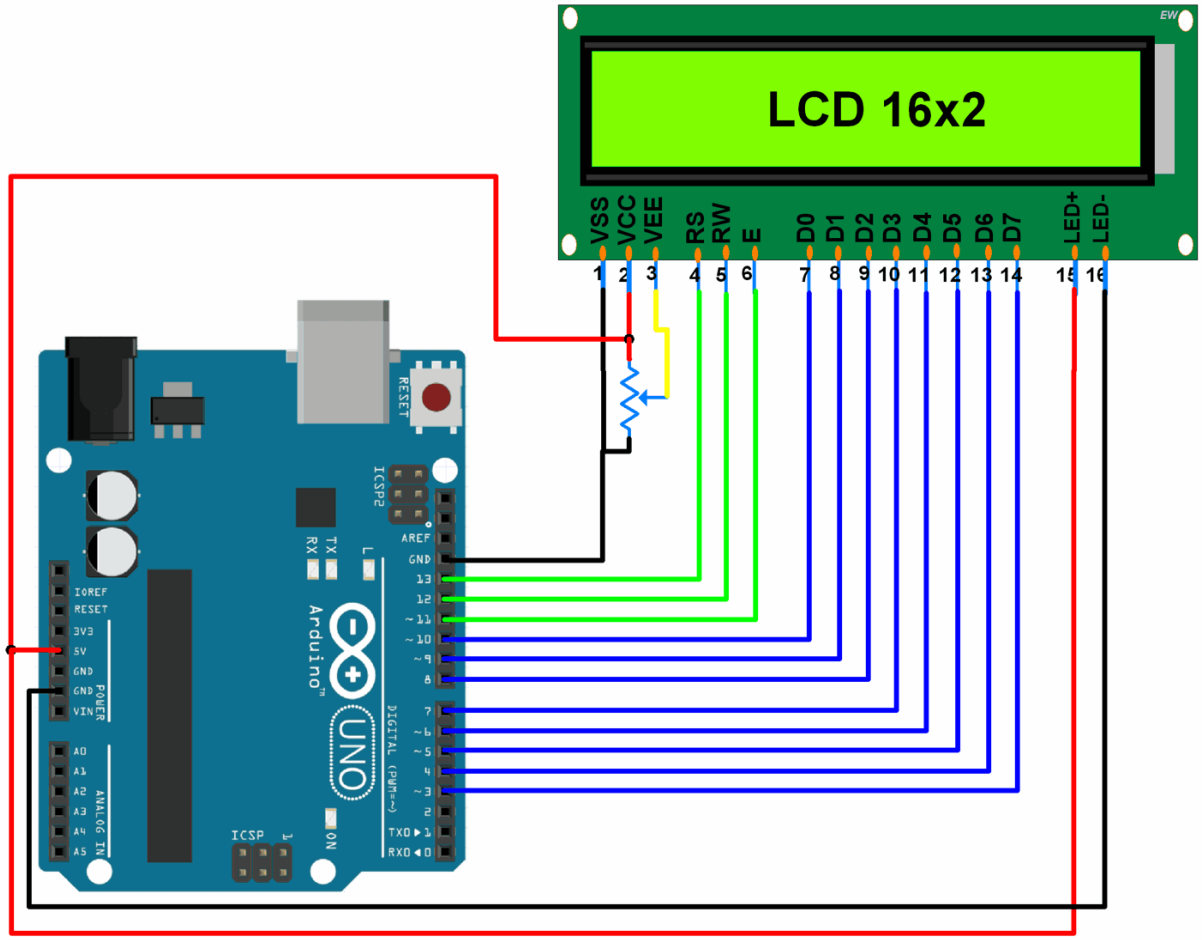




Liquid Crystal Display 16X02

Connections to Arduino Uno

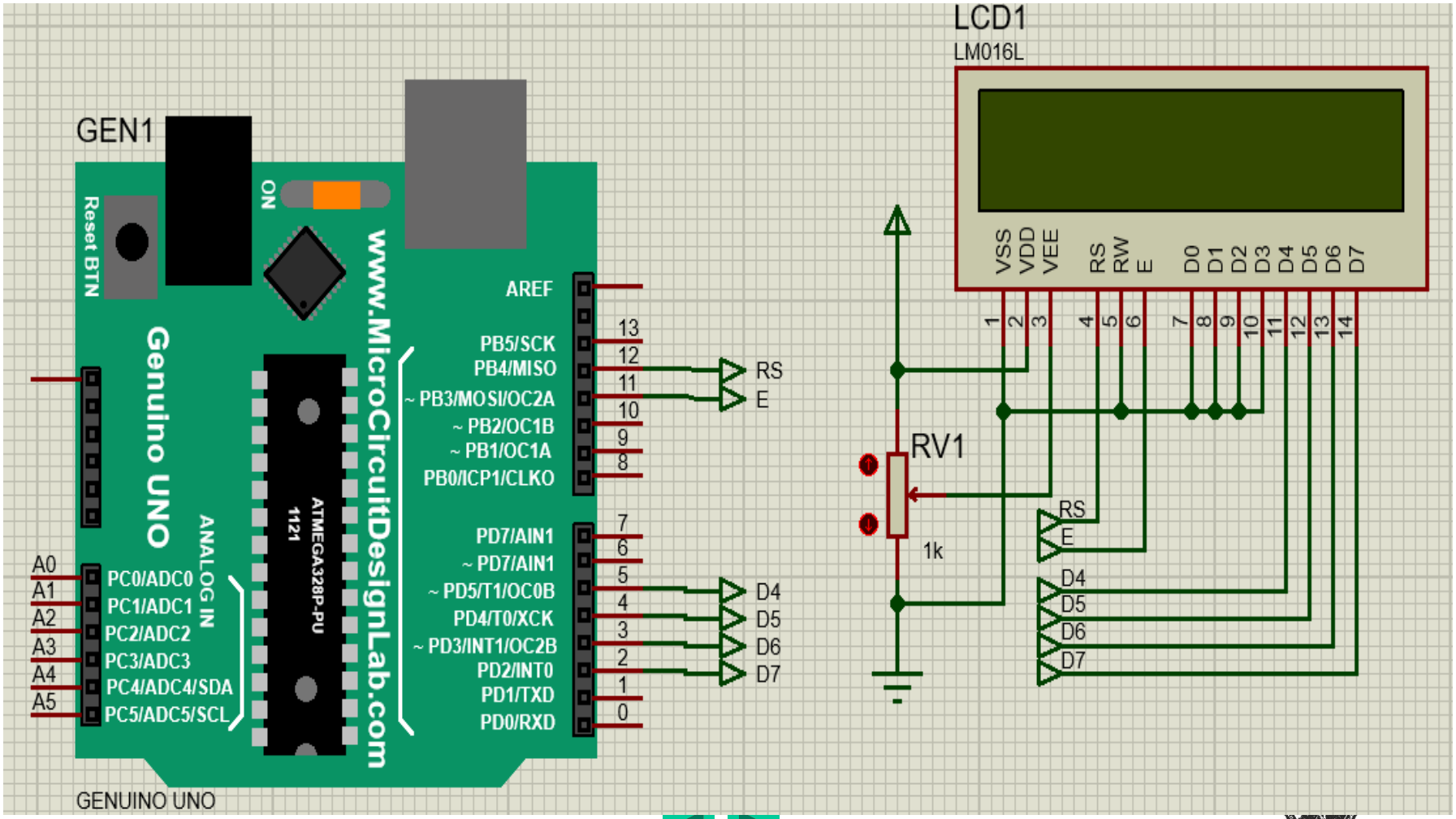
8-bit Mode



Liquid Crystal Display 16X02

Connections to Arduino Uno

4-bit Mode



Liquid Crystal Display 16X02

LiquidCrystal()

The LiquidCrystal() function sets the pins the Arduino uses to connect to the LCD. You can use any of the Arduino's digital pins to control the LCD. Just put the Arduino pin numbers inside the parentheses in this order:

LiquidCrystal(RS, E, D4, D5, D6, D7)

RS, E, D4, D5, D6, D7 are the LCD pins.

For example, say you want LCD pin D7 to connect to Arduino pin 12. Just put "12" in place of D7 in the function like this:

LiquidCrystal(RS, E, D4, D5, D6, 12)

This function needs to be placed before the void setup() section of the program.

Liquid Crystal Display 16X02

lcd.begin()

This function sets the dimensions of the LCD. It needs to be placed before any other LiquidCrystal function in the void setup() section of the program. The number of rows and columns are specified as lcd.begin(columns, rows). For a 16×2 LCD, you would use lcd.begin(16, 2), and for a 20×4 LCD you would use lcd.begin(20, 4).

lcd.clear()

This function clears any text or data already displayed on the LCD. If you use lcd.clear() with lcd.print() and the delay() function in the void loop() section.

lcd.home()

This function places the cursor in the upper left hand corner of the screen, and prints any subsequent text from that position.

lcd.setCursor()

Similar, but more useful than lcd.home() is lcd.setCursor(). This function places the cursor (and any printed text) at any position on the screen. It can be used in the void setup() or void loop() section of your program.

Liquid Crystal Display 16X02

lcd.write()

You can use this function to write different types of data to the LCD, for example the reading from a temperature sensor, or the coordinates from a GPS module. You can also use it to print custom characters that you create yourself (more on this below). Use lcd.write() in the void setup() or void loop() section of your program.

lcd.print()

This function is used to print text to the LCD. It can be used in the void setup() section or the void loop() section of the program.

To print letters and words, place quotation marks (” “) around the text.

lcd.cursor()

This function creates a visible cursor. The cursor is a horizontal line placed below the next character to be printed to the LCD.

lcd.blink()

This function creates a block style cursor that blinks on and off at approximately 500 milliseconds per cycle. Use it in the void loop() section. The function lcd.noBlink() disables the blinking block cursor.

Liquid Crystal Display 16X02

lcd.display()

This function turns on any text or cursors that have been printed to the LCD screen. The function lcd.noDisplay() turns off any text or cursors printed to the LCD, without clearing it from the LCD's memory.

These two functions can be used together in the void loop() section to create a blinking text effect.

lcd.scrollDisplayLeft()

This function takes anything printed to the LCD and moves it to the left. It should be used in the void loop() section with a delay command following it. The function will move the text 40 spaces to the left before it loops back to the first character.

lcd.scrollDisplayRight()

This function behaves like lcd.scrollDisplayLeft(), but moves the text to the right.

lcd.autoscroll()

This function takes a string of text and scrolls it from right to left in increments of the character count of the string. For example, if you have a string of text that is 3 characters long, it will shift the text 3 spaces to the left with each step.

Liquid Crystal Display 16X02

lcd.noAutoscroll()

lcd.noAutoscroll() turns the lcd.autoscroll() function off. Use this function before or after lcd.autoscroll() in the void loop() section to create sequences of scrolling text or animations.

lcd.rightToLeft()

This function sets the direction that text is printed to the screen. The default mode is from left to right using the command lcd.leftToRight(), but you may find some cases where it's useful to output text in the reverse direction.

lcd.createChar()

This command allows you to create your own custom characters. Each character of a 16×2 LCD has a 5 pixel width and an 8 pixel height. Up to 8 different custom characters can be defined in a single program. To design your own characters, you'll need to make a binary matrix of your custom character from an LCD character generator or map it yourself.

Liquid Crystal Display 16X02

```
// Example LCD1602_Ex501.ino
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // create an lcd object and assign the pins

void setup()
{
    lcd.begin(16, 2); // Set the display to 16 columns and 2 rows
}

void loop()
{
    // run the 7 demo routines
    basicPrintDemo();
    displayOnOffDemo();
    setCursorDemo();
    scrollLeftDemo();
    scrollRightDemo();
    cursorDemo();
    createGlyphDemo();
}

void basicPrintDemo()
{
    lcd.clear(); // Clear the display
    lcd.print("Hello World!:-)"); // print some text
    lcd.setCursor(0,1);
    lcd.print("1234567890ABCDEF");
    delay(2000);
}
```

Liquid Crystal Display 16X02

```
void displayOnOffDemo()
{
    lcd.clear(); // Clear the display
    lcd.print("Blink On/Off"); // print some text
    for(int x=0; x < 3; x++)
    { // loop 3 times
        lcd.noDisplay(); // turn display off
        delay(1000);
        lcd.display(); // turn it back on again
        delay(1000);
    }
}

void setCursorDemo()
{
    lcd.clear(); // Clear the display
    lcd.print("Cursor Demo"); // print some text
    delay(1000);
    lcd.clear(); // Clear the display
    lcd.setCursor(7,0); // cursor at column 5 row 0
    lcd.print("7,0");
    delay(2000);
    lcd.setCursor(8,1); // cursor at column 10 row 1
    lcd.print("8,1");
    delay(2000);
    lcd.setCursor(1,1); // cursor at column 3 row 1
    lcd.print("1,1");
    delay(2000);
}
```

Liquid Crystal Display 16X02

```
void scrollLeftDemo()
{
    lcd.clear(); // Clear the display
    lcd.print("Scroll to Left");
    delay(1000);
    lcd.clear(); // Clear the display
    lcd.setCursor(7,0);
    lcd.print("LCD 16X02");
    lcd.setCursor(9,1);
    lcd.print("Arduino");
    delay(1000);
    for(int x=0; x<16; x++)
    {
        lcd.scrollDisplayLeft(); // scroll display left 16 times
        delay(250);
    }
}

void scrollRightDemo()
{
    lcd.clear(); // Clear the display
    lcd.print("Scroll to Right");
    lcd.setCursor(0,1);
    lcd.print("Demo");
    delay(1000);
    lcd.clear(); // Clear the display
    lcd.print("LCD 16X02");
    lcd.setCursor(0,1);
    lcd.print("Arduino");
    delay(1000);
    for(int x=0; x<16; x++)
    {
        lcd.scrollDisplayRight(); // scroll display right 16 times
        delay(250);
    }
}
```

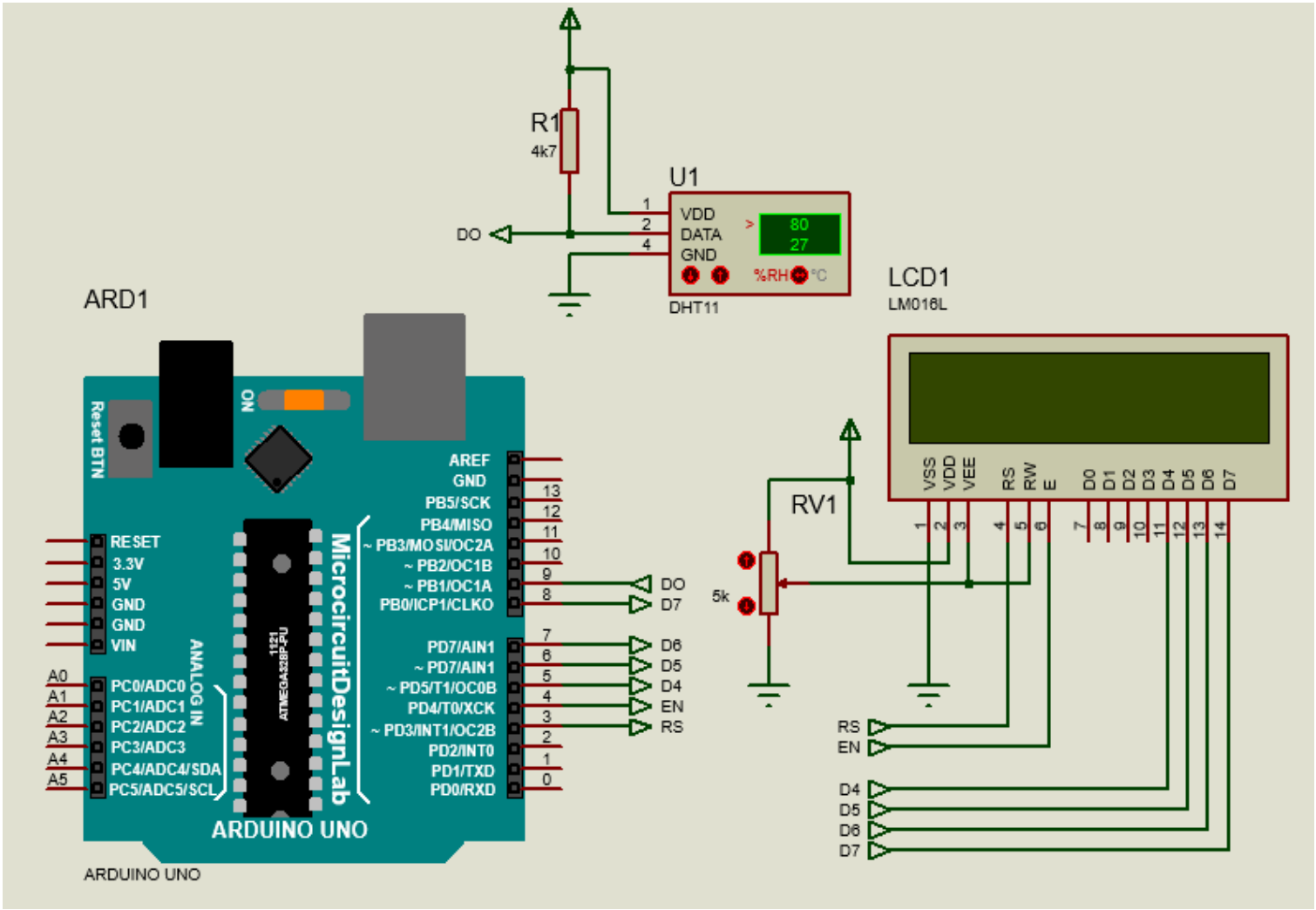
Liquid Crystal Display 16X02

```
void cursorDemo()
{
    lcd.clear(); // Clear the display
    lcd.cursor(); // Enable cursor visible
    lcd.print("Cursor On");
    delay(3000);
    lcd.clear(); // Clear the display
    lcd.noCursor(); // cursor invisible
    lcd.print("Cursor Off");
    delay(3000);
    lcd.clear(); // Clear the display
    lcd.cursor(); // cursor visible
    lcd.blink(); // cursor blinking
    lcd.print("Cursor Blink On");
    delay(3000);
    lcd.noCursor(); // cursor invisible
    lcd.noBlink(); // blink off
}
```

Liquid Crystal Display 16X02

```
void createGlyphDemo()
{
    lcd.clear();
    byte happy[8] =
    { // create byte array with happy face
    B00000,
    B00000,
    B10001,
    B00000,
    B10001,
    B01110,
    B00000,
    B00000};
    byte sad[8] =
    { // create byte array with sad face
    B00000,
    B00000,
    B10001,
    B00000,
    B01110,
    B10001,
    B00000,
    B00000};
    lcd.createChar(0, happy); // create custom character 0
    lcd.createChar(1, sad); // create custom character 1

  for(int x=0; x<5; x++)
    { // loop animation 5 times
      lcd.setCursor(8,0);
      lcd.write(byte(0));// write custom char 0
      delay(1000);
      lcd.setCursor(8,0);
      lcd.write(byte(1)); // write custom char 1
      delay(1000);
    }
}
```

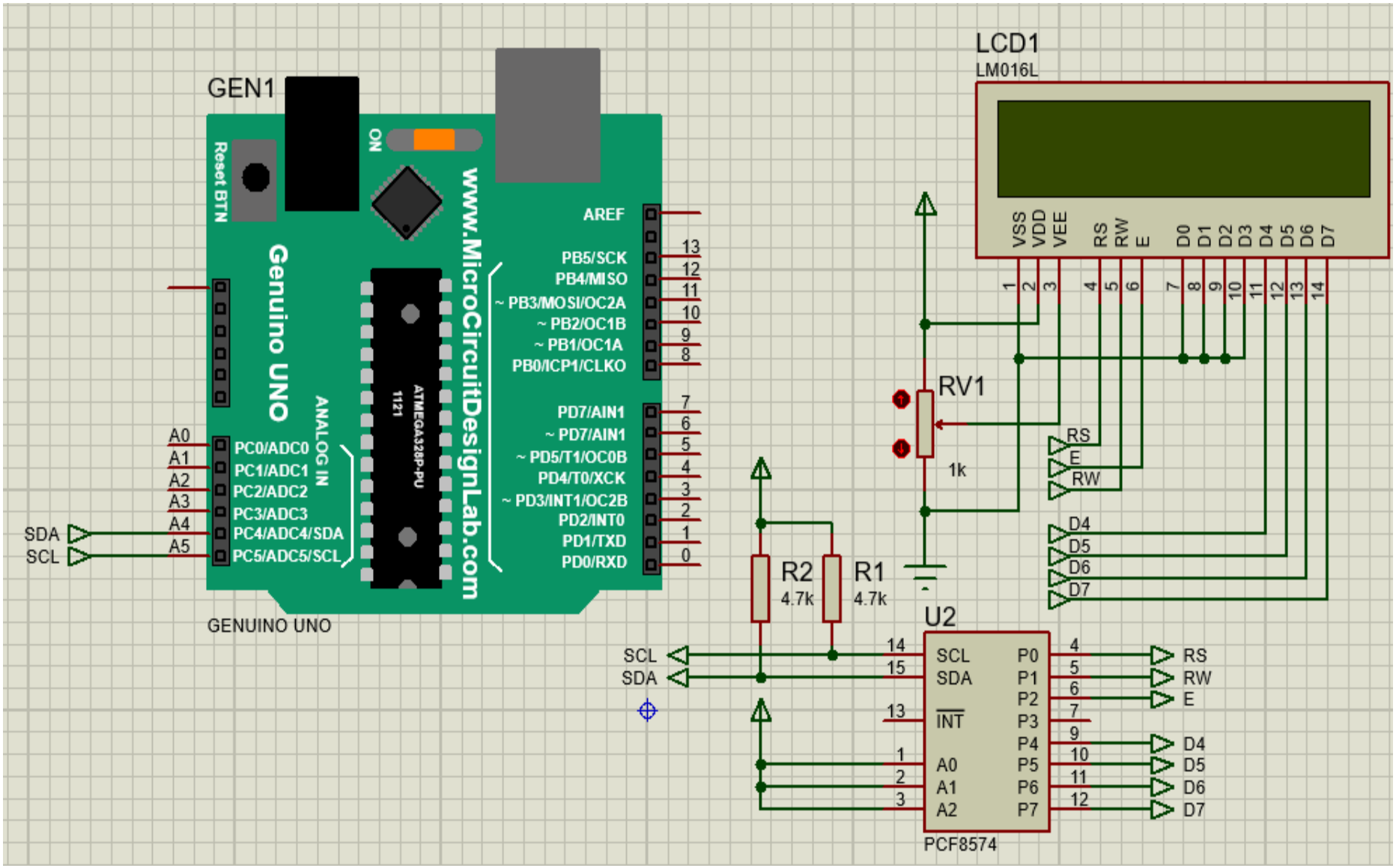
Liquid Crystal Display 16X02 DHT11



Liquid Crystal Display 16X02 DHT11

```
#include <dht.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(3, 4, 5, 6, 7, 8);

dht DHT;

#define DHT11_PIN 9

void setup(){
  lcd.begin(16, 2);
}

void loop()
{
  int chk = DHT.read11(DHT11_PIN);
  lcd.setCursor(0,0);
  lcd.print("Temp: ");
  lcd.print(DHT.temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("Humidity: ");
  lcd.print(DHT.humidity);
  lcd.print("%");
  delay(2000);

}//Code Listing DHT11_LCD1602.ino
```

Liquid Crystal Display 16X02 I2C



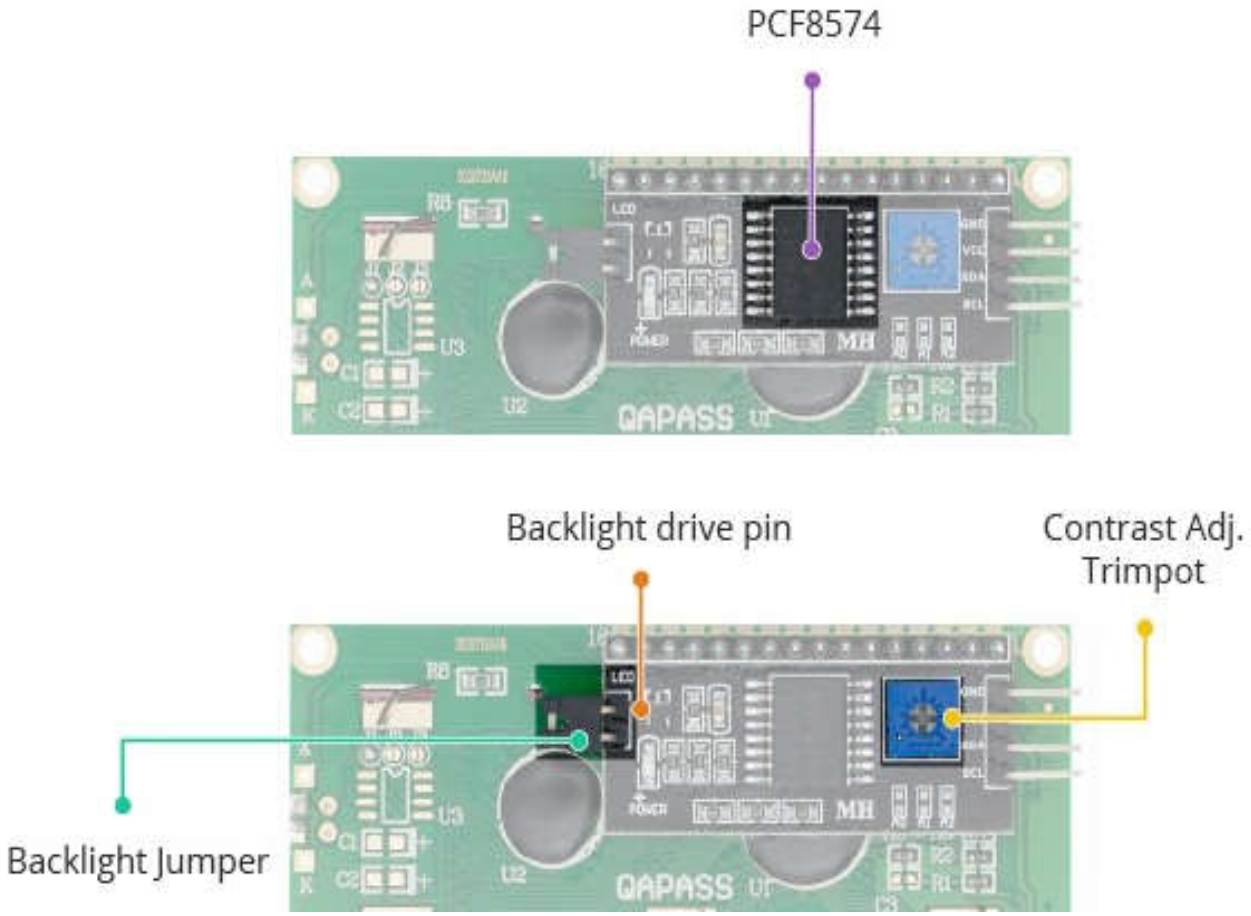
Liquid Crystal Display 16X02 I2C

I2C LCD Overview

- to save I/O pins of Arduino
- it takes only 2 I/O's A4 and A5 (SDA and SCL)

Liquid Crystal Display 16X02 I2C

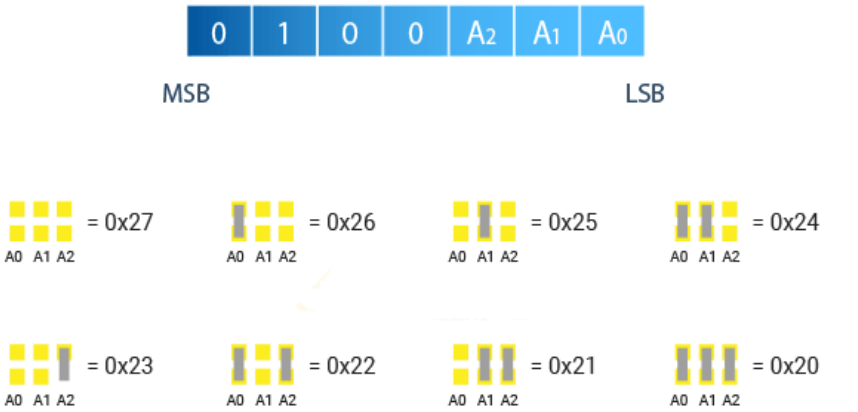
I2C LCD Adapter 8-bit I/O expander chip PCF8574



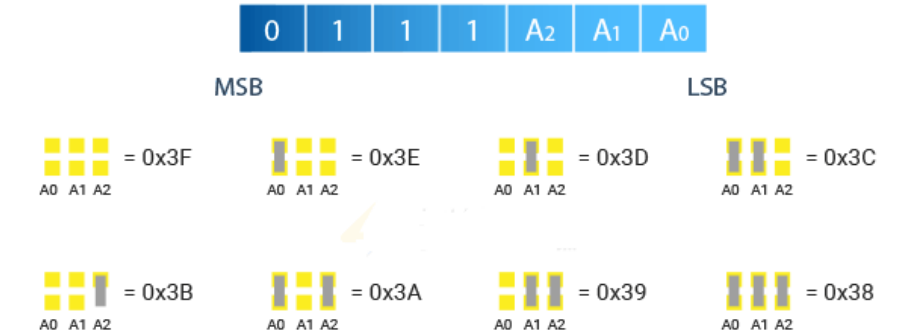
Liquid Crystal Display 16X02 I2C

I2C Address-depends on IC used

Texas Instrument:

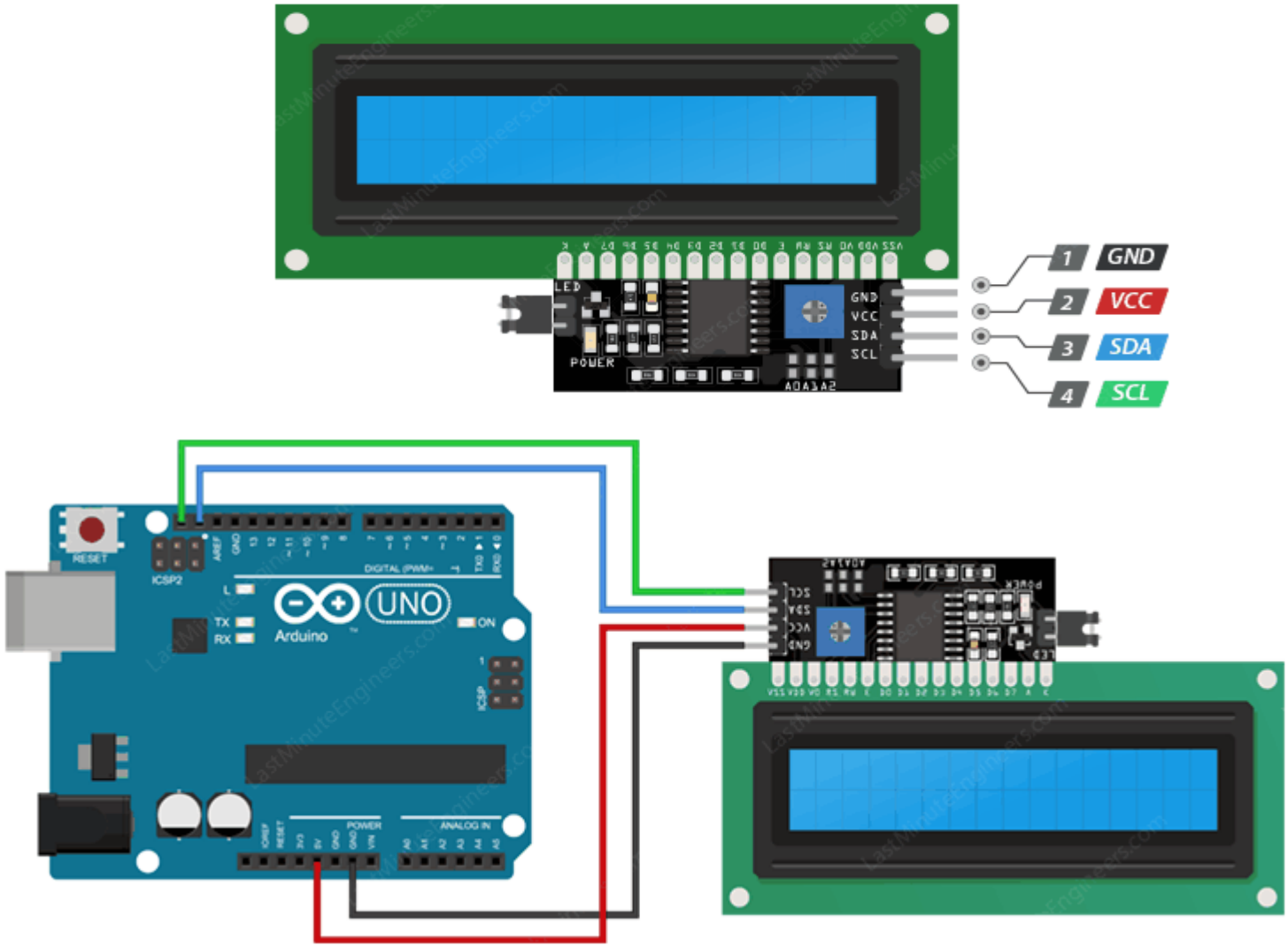


NXP:



Liquid Crystal Display 16X02 I2C

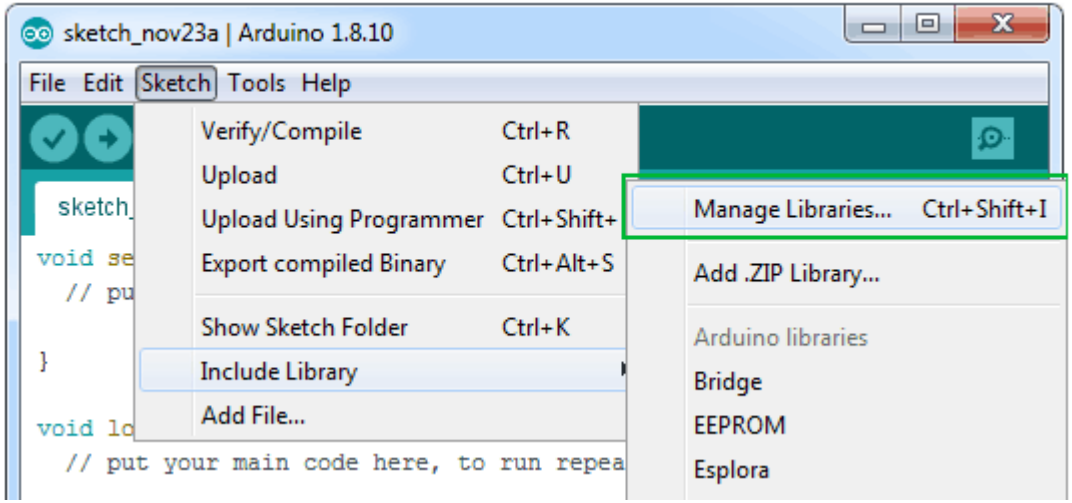
I2C LCD Display Pinout



Liquid Crystal Display 16X02 I2C

Library Installation

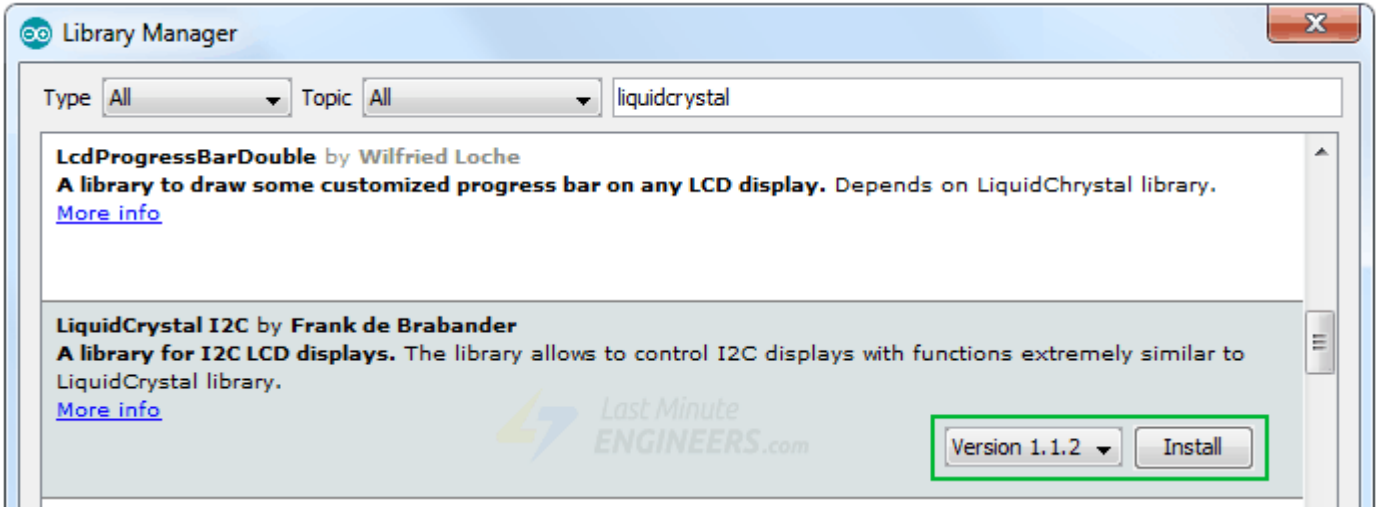
Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.



Liquid Crystal Display 16X02 I2C

Library Installation

Typing ‘liquidcrystal’. There should be a couple entries. Look for LiquidCrystal I2C library by Frank de Brabander. Click on that entry, and then select Install.



Liquid Crystal Display 16X02 I2C

I2C Address Checking

LCD probably has an I2C address 0x27Hex or 0x3FHex. Nevertheless it is recommended that you find out the actual I2C of the LCD before using. Luckily there is a simple way to do this, thanks to Nick Gammon‘s I2C code.

Liquid Crystal Display 16X02 I2C

I2C Address Checking

```
#include <Wire.h>

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

  // Leonardo: wait for serial port to connect
  while (!Serial)
    {
    }

  Serial.println ();
  Serial.println ("I2C scanner. Scanning ...");
  byte count = 0;

  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0)
      {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      delay (1);  // maybe unneeded?
      } // end of good response
  } // end of for loop
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");
}  // end of setup

void loop() {}
```


Liquid Crystal Display 16X02 I2C

Hello World I2C LCD

```
#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() {
}
```

Liquid Crystal Display 16X02 I2C

Hello World I2C LCD

Liquid Crystal Display 16X02 I2C

Other useful functions of the library

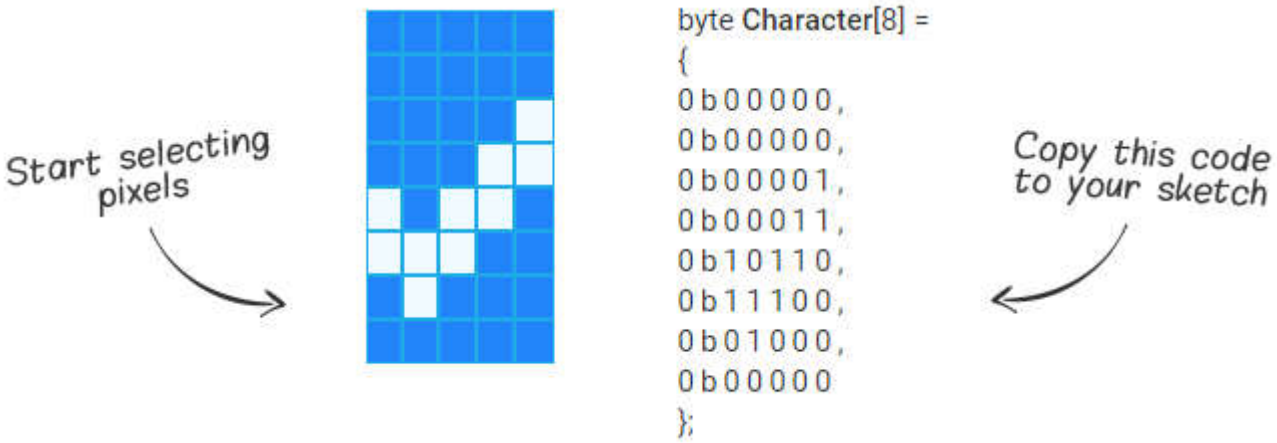
home() – positions the cursor in the top-left corner of the LCD without clearing the display.
cursor() – displays the LCD cursor, an underscore (line) at the position of the next character to be printed.
noCursor() – hides the LCD cursor.
blink() – creates a blinking block style LCD cursor: a blinking rectangle of 5×8 pixels at the position of the next character to be printed.
noBlink() – disables the blinking block style LCD cursor.
display() – turns on the LCD screen and displays the characters that were previously printed on the display.
noDisplay() – turns off the LCD screen. Simply turning off the LCD screen does not clear data from the LCD memory. This means that it will be shown again when the display() function is called.
scrollDisplayLeft() – scrolls the contents of the display one space to the left. If you want to scroll the text continuously, you need to use this function inside a loop.
scrollDisplayRight() – scrolls the contents of the display one space to the right.
autoscroll() – turns on automatic scrolling of the LCD. If the current text direction is left-to-right (default), the display scrolls to the left, if the current direction is right-to-left, the display scrolls to the right.
noAutoscroll() – turns off automatic scrolling.

Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

-to create your own custom characters (glyph) and symbols for your LCD.

-custom character the createChar() function is used. This function accepts an array of 8 bytes. Each byte (only 5 bits are considered) in the array defines one row of the character in the 5×8 matrix.



Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

```
#include <LiquidCrystal_I2C.h>

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

// make some custom characters:
byte Heart[8] = {
0b00000,
0b01010,
0b11111,
0b11111,
0b01110,
0b00100,
0b00000,
0b00000
};

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

Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

```
byte Alien[8] = {
0b11111,
0b10101,
0b11111,
0b11111,
0b01110,
0b01010,
0b11011,
0b00000
};

byte Check[8] = {
0b00000,
0b00001,
0b00011,
0b10110,
0b11100,
0b01000,
0b00000,
0b00000
};
```

Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

```
byte Speaker[8] = {
0b00001,
0b00011,
0b01111,
0b01111,
0b01111,
0b00011,
0b00001,
0b00000
};


byte Sound[8] = {
0b00001,
0b00011,
0b00101,
0b01001,
0b01001,
0b01011,
0b11011,
0b11000
};
```

Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

```
byte Skull[8] = {
0b00000,
0b01110,
0b10101,
0b11011,
0b01110,
0b01110,
0b00000,
0b00000
};

byte Lock[8] = {
0b01110,
0b10001,
0b10001,
0b11111,
0b11011,
0b11011,
0b11111,
0b00000
};
```

Liquid Crystal Display 16X02 I2C

Create and Display Custom Characters

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

  // create a 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);

  // Clears the LCD screen
  lcd.clear();

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

// Print All the custom characters
```

Liquid Crystal Display 16X02 I2C

Create and Display 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);
}
```

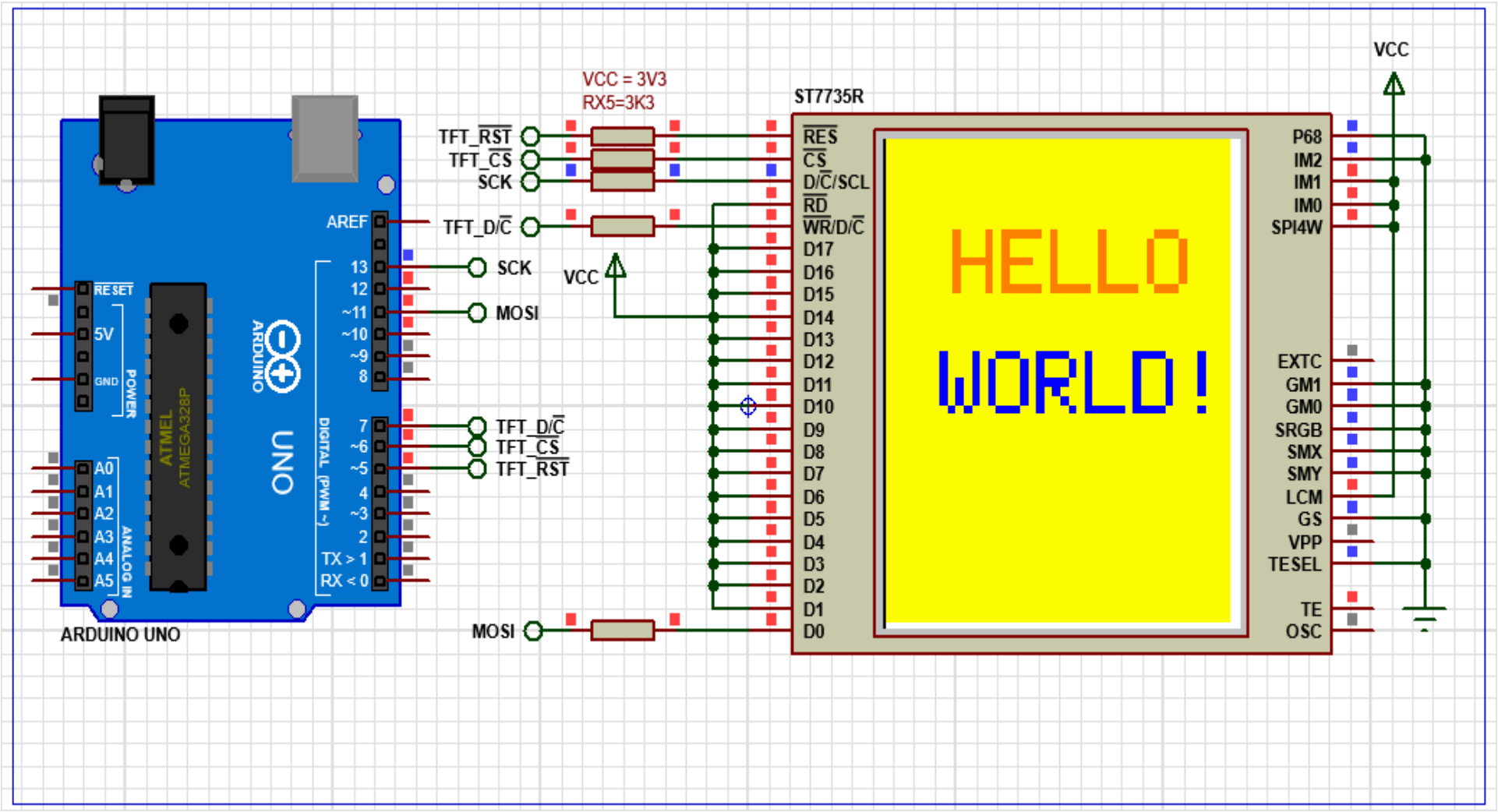
Liquid Crystal Display 16X02 I2C

```
/*
  Example LCD1602_I2C.ino
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// LCD address and geometry and library initialization
const byte lcdAddr = 0x27;  // Address of I2C backpack
const byte lcdCols = 16;    // Number of character in a row
const byte lcdRows = 2;     // Number of lines
//const byte lcdAddr = 0x3F;  // Address of I2C backpack
//const byte lcdCols = 20;    // Number of character in a row
//const byte lcdRows = 4;     // Number of lines

LiquidCrystal_I2C lcd(lcdAddr, lcdCols, lcdRows);

// Demo parameters
const char demoText[]= “#OneIECEP!";
const unsigned int scrollDelay = 500;   // Miliseconds before scrolling next
char
const unsigned int demoDelay = 2000;    // Miliseconds between demo loops
byte textLen;                           // Number of visible characters in
the text

void setup() {
  textLen = sizeof(demoText) - 1;
  lcd.init();
  lcd.backlight();
  lcd.print(demoText);
  delay(demoDelay);
}
```

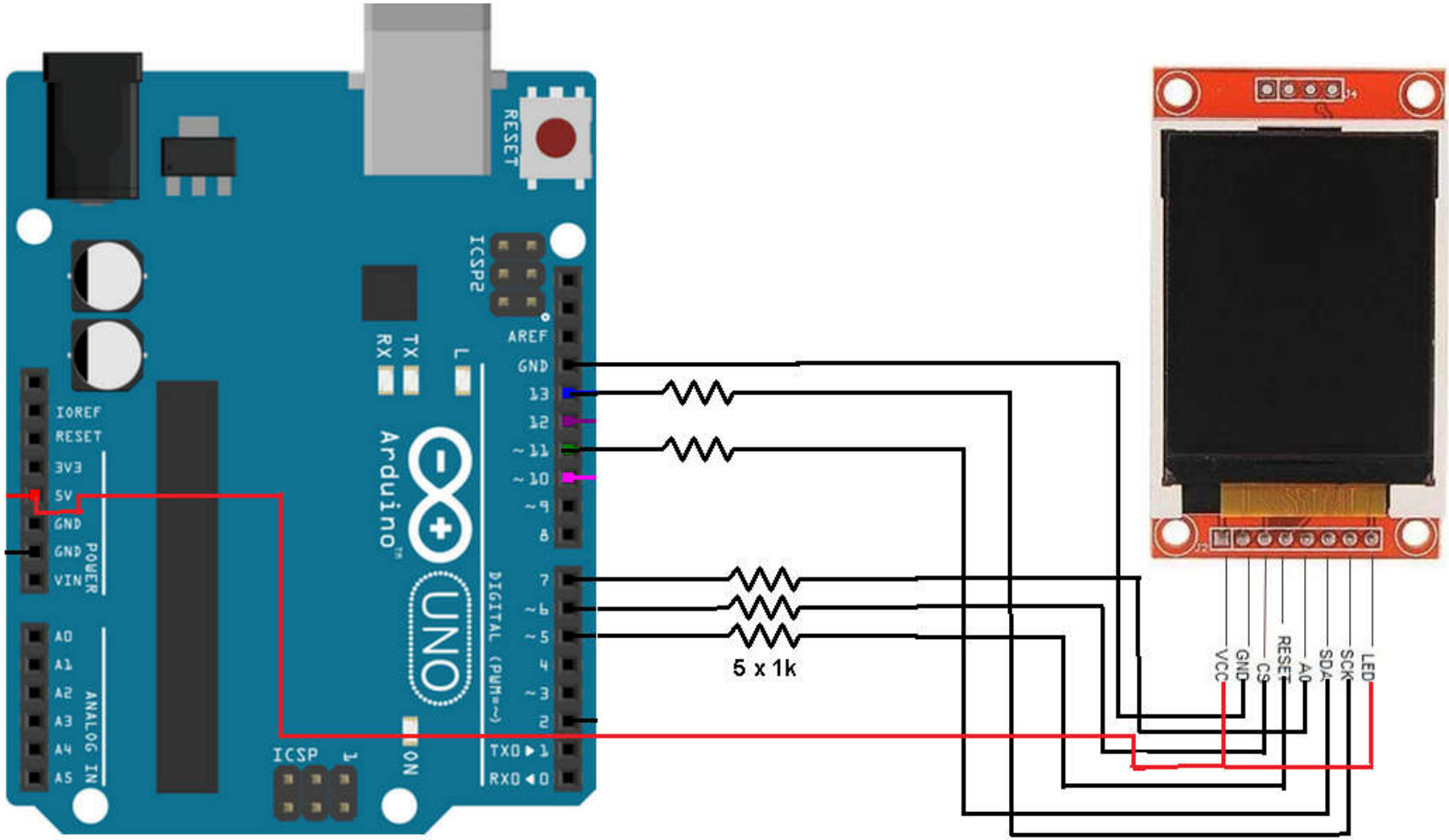
Liquid Crystal Display 16X02 I2C

```
void loop()
{
  // Scroll entire text in a row to the left outside the screen
  for (byte positionCounter = 0; positionCounter < textLen;
positionCounter++) {
    lcd.scrollDisplayLeft();
    delay(scrollDelay);
  }
  // Scroll hidden text through entire row to the right outside the screen
  for (byte positionCounter = 0; positionCounter < textLen + lcdCols;
positionCounter++)
  {
    lcd.scrollDisplayRight();
    delay(scrollDelay);
  }
  // Scroll text to the right back to original position
  for (byte positionCounter = 0; positionCounter < lcdCols;
positionCounter++)
  {
    lcd.scrollDisplayLeft();
    delay(scrollDelay);
  }
  delay(demoDelay);
}
```

ST7735 Display



ST7735 Display



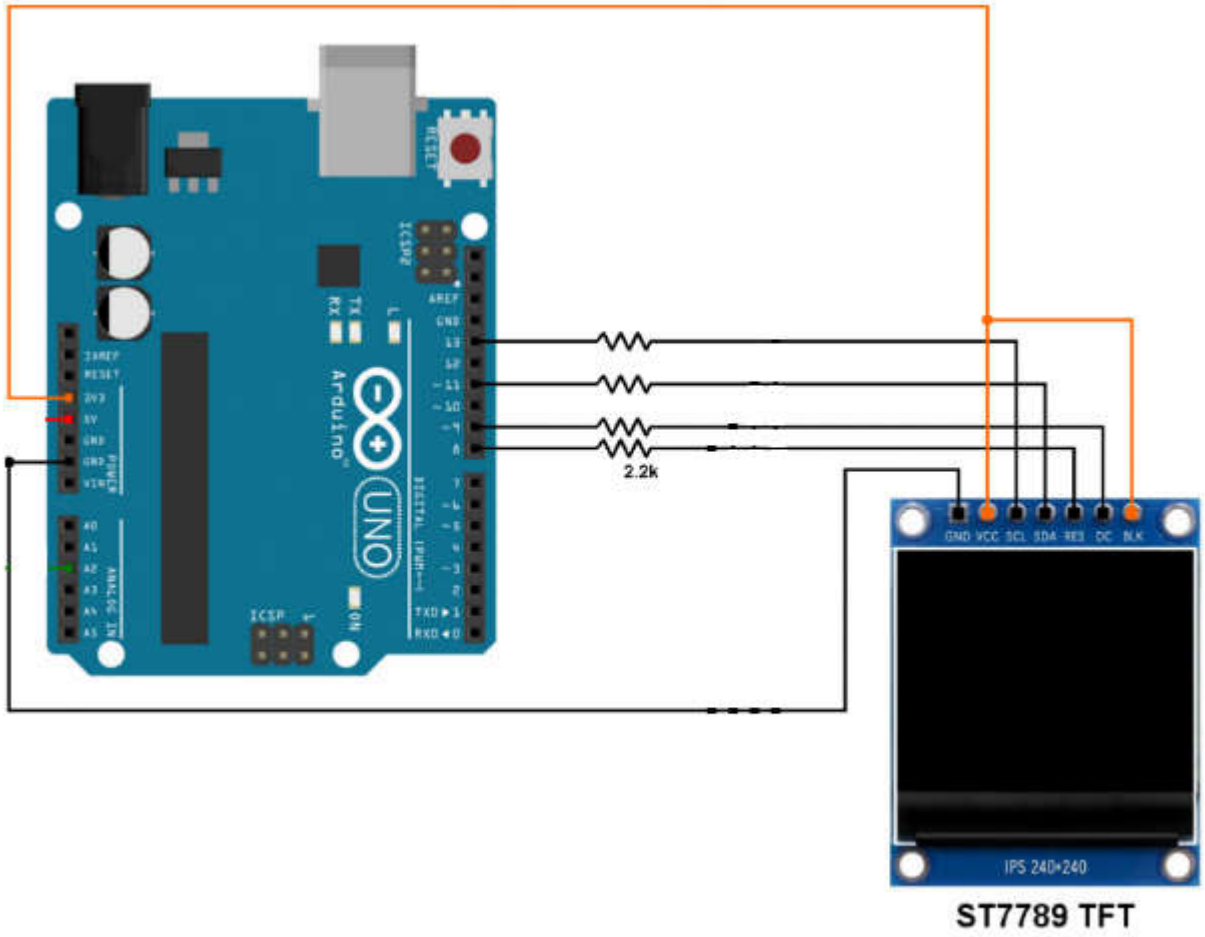
ST7735 Display

```
// Example ST7735Test1.ino
#include <SPI.h>
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_ST7735.h> // Hardware-specific library

#define TFT_CS         6
#define TFT_RST        5 // Or set to -1 and connect to Arduino RESET pin
#define TFT_DC         7

Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup(void) {
  //tft.initR(INITR_144GREENTAB); // Init ST7735R chip, green tab
  tft.initR(INITR_BLACKTAB);
  //tft.fillScreen(ST77XX_BLACK);
  tft.setRotation(0); // set display orientation
}
void loop()
{
  tft.fillScreen(ST77XX_YELLOW);
  print_text(25,30,"HELLO",3,ST77XX_ORANGE);
  print_text(20,70,"WORLD!",3,ST77XX_BLUE);
  delay(5000);

  tft.fillScreen(ST77XX_BLACK);
  tft.fillRoundRect(25, 10, 78, 60, 8, ST77XX_WHITE);
  tft.fillTriangle(42, 20, 42, 60, 90, 40, ST77XX_RED);
  delay(5000);

  tft.fillScreen(ST77XX_GREEN);
  tft.drawRect(5,5,120,120,ST77XX_RED);
  tft.drawFastHLine(5,60,120,ST77XX_RED);
  tft.drawFastVLine(60,5,120,ST77XX_RED);
  delay(5000);
}
```

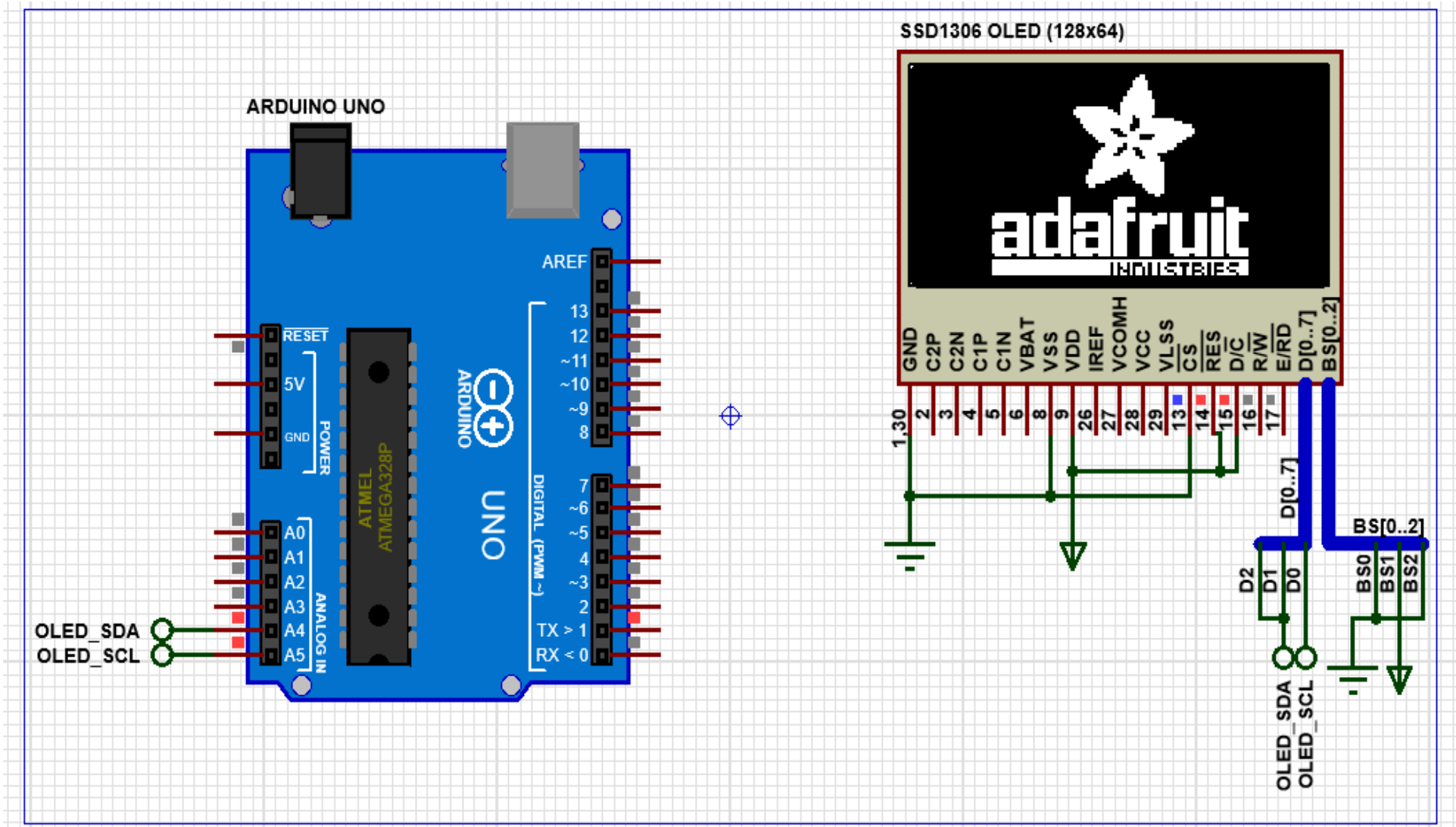
ST7735 Display

```
void print_text(byte x_pos, byte y_pos, char *text, byte text_size, uint16_t color)
{
  tft.setCursor(x_pos, y_pos);
  tft.setTextSize(text_size);
  tft.setTextColor(color);
  tft.setTextWrap(true);
  tft.print(text);
}
```

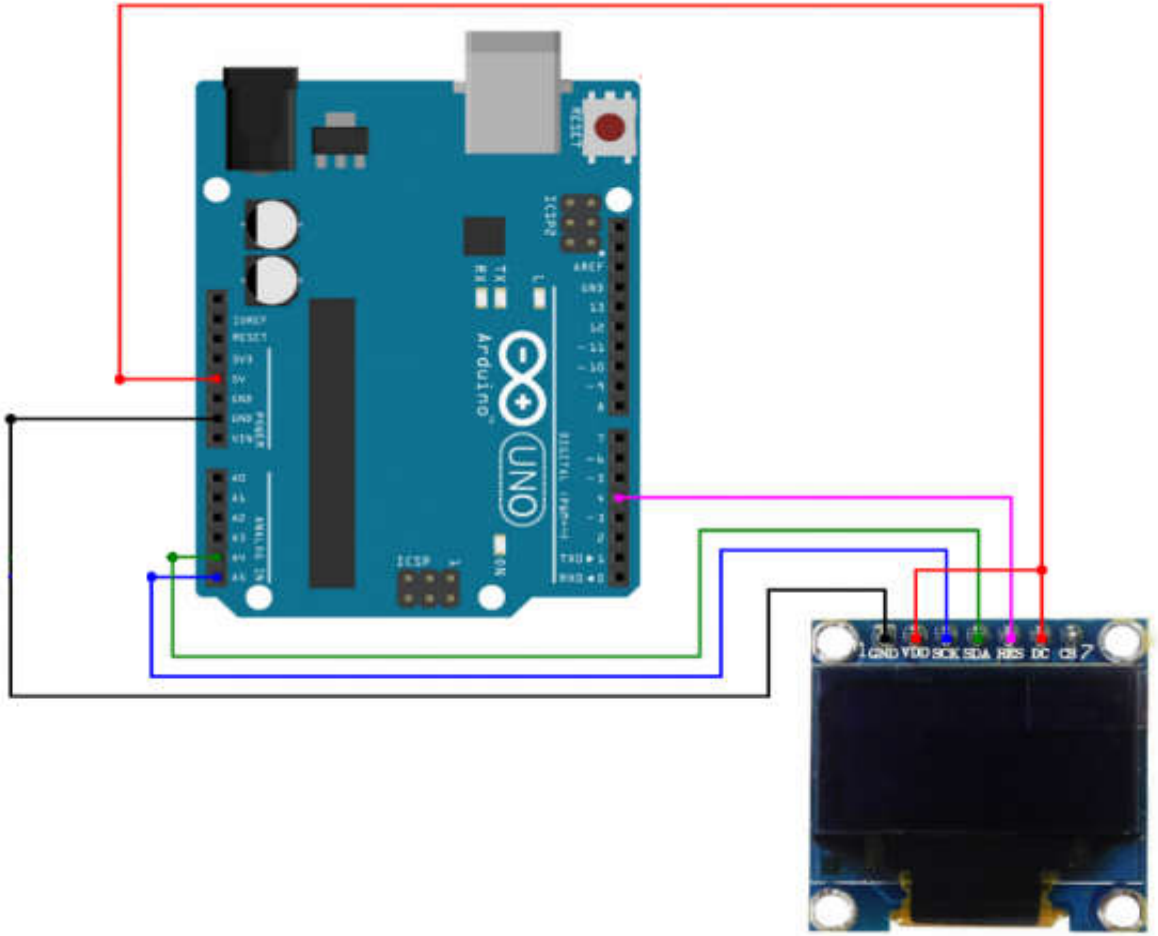
ST7789 Display



OLED SSD1306 I2C



OLED SSD1306 I2C



OLED SSD1306 I2C

```
/*************************************************************************
 *This is an example for our Monochrome OLEDs based on SSD1306 drivers
 *This example is for a 128x64 pixel display using I2C to communicate
 *3 pins are required to interface (two I2C and one reset).
 *Ex. ssd1306_12x64_i2c_test1.ino
*************************************************************************/
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
//#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LOGO_HEIGHT   16
#define LOGO_WIDTH    16
static const unsigned char PROGMEM logo_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
```


OLED SSD1306 I2C

```
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

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

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3D)) { // Address 0x3D for 128x64
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();
  delay(2000); // Pause for 2 seconds

  // Clear the buffer
  display.clearDisplay();

  // Draw a single pixel in white
  display.drawPixel(10, 10, SSD1306_WHITE);

  // Show the display buffer on the screen. You MUST call display() after
  // drawing commands to make them visible on screen!
  display.display();
  delay(2000);
  // display.display() is NOT necessary after every single drawing command,
  // unless that's what you want...rather, you can batch up a bunch of
  // drawing operations and then update the screen all at once by calling
  // display.display(). These examples demonstrate both approaches...
```

OLED SSD1306 I2C

```
 testdrawline();      // Draw many lines

  testdrawrect();      // Draw rectangles (outlines)

  testfillrect();      // Draw rectangles (filled)

  testdrawcircle();    // Draw circles (outlines)

  testfillcircle();    // Draw circles (filled)

  testdrawroundrect(); // Draw rounded rectangles (outlines)

  testfillroundrect(); // Draw rounded rectangles (filled)

  testdrawtriangle();  // Draw triangles (outlines)

  testfilltriangle();  // Draw triangles (filled)

  testdrawchar();      // Draw characters of the default font

  testdrawstyles();    // Draw 'stylized' characters

  testscrolltext();    // Draw scrolling text

  testdrawbitmap();    // Draw a small bitmap image

  // Invert and restore display, pausing in-between
  display.invertDisplay(true);
  delay(1000);
  display.invertDisplay(false);
  delay(1000);

  testanimate(logo_bmp, LOGO_WIDTH, LOGO_HEIGHT); // Animate bitmaps
}
```

OLED SSD1306 I2C

```
 void loop() {
}

void testdrawline() {
  int16_t i;

  display.clearDisplay(); // Clear display buffer

  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, 0, i, display.height()-1, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn line
    delay(1);
  }
  for(i=0; i<display.height(); i+=4) {
    display.drawLine(0, 0, display.width()-1, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(250);

  display.clearDisplay();

  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, display.height()-1, i, 0, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=display.height()-1; i>=0; i-=4) {
    display.drawLine(0, display.height()-1, display.width()-1, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(250);
```

OLED SSD1306 I2C

```
 display.clearDisplay();

  for(i=display.width()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, i, 0, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=display.height()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, 0, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(250);

  display.clearDisplay();

  for(i=0; i<display.height(); i+=4) {
    display.drawLine(display.width()-1, 0, 0, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=0; i<display.width(); i+=4) {
    display.drawLine(display.width()-1, 0, i, display.height()-1, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000); // Pause for 2 seconds
}
```

OLED SSD1306 I2C

```
void testdrawrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
}
  void testfillrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate white/black
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
} void testdrawcircle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}
```

OLED SSD1306 I2C

```
void testdrawrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
}
  void testfillrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate white/black
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }

  delay(2000);
} void testdrawcircle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}
```

OLED SSD1306 I2C

```
void testfillroundrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    // The INVERSE color is used so round-rects alternate white/black
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawtriangle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=5) {
    display.drawTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_WHITE);
    display.display();
    delay(1);
  }

  delay(2000);
}
```

OLED SSD1306 I2C

```
void testfilltriangle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=5) {
    // The INVERSE color is used so triangles alternate white/black
    display.fillTriangle(
      display.width()/2  , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }

  delay(2000);
}

void testdrawchar(void) {
  display.clearDisplay();

  display.setTextSize(1);      // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE); // Draw white text
  display.setCursor(0, 0);     // Start at top-left corner
  display.cp437(true);         // Use full 256 char 'Code Page 437' font
// Not all the characters will fit on the display. This is normal.
  // Library will draw what it can and the rest will be clipped.
  for(int16_t i=0; i<256; i++) {
    if(i == '\n') display.write(' ');
    else          display.write(i);
  }

  display.display();
  delay(2000);
}
```

OLED SSD1306 I2C

```
void testdrawstyles(void) {
  display.clearDisplay();

  display.setTextSize(1);             // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);        // Draw white text
  display.setCursor(0,0);             // Start at top-left corner
  display.println(F("Hello, world!"));

  display.setTextColor(SSD1306_BLACK, SSD1306_WHITE); // Draw 'inverse' text
  display.println(3.141592);

  display.setTextSize(2);             // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.print(F("0x")); display.println(0xDEADBEEF, HEX);

  display.display();
  delay(2000);
}
```

OLED SSD1306 I2C

```
void testdrawbitmap(void) {
  display.clearDisplay();

  display.drawBitmap(
    (display.width()  - LOGO_WIDTH ) / 2,
    (display.height() - LOGO_HEIGHT) / 2,
    logo_bmp, LOGO_WIDTH, LOGO_HEIGHT, 1);
  display.display();
  delay(1000);
}

#define XPOS   0 // Indexes into the 'icons' array in function below
#define YPOS   1
#define DELTAY 2

void testanimate(const uint8_t *bitmap, uint8_t w, uint8_t h) {
  int8_t f, icons[NUMFLAKES][3];

  // Initialize 'snowflake' positions
  for(f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
    icons[f][YPOS]   = -LOGO_HEIGHT;
    icons[f][DELTAY] = random(1, 6);
    Serial.print(F("x: "));
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(F(" y: "));
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(F(" dy: "));
    Serial.println(icons[f][DELTAY], DEC);
  }
```

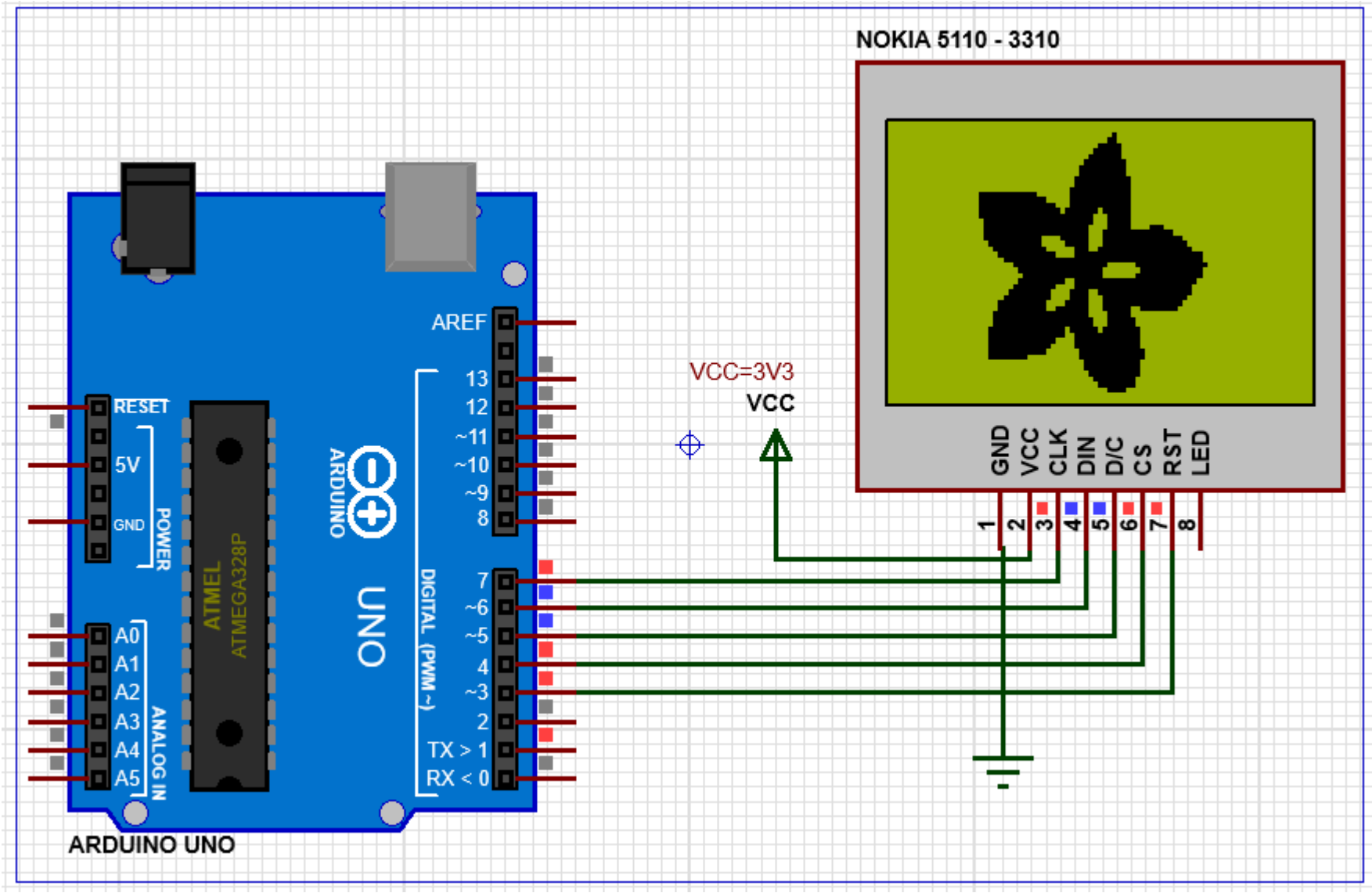
OLED SSD1306 I2C

```
void testscrolltext(void) {
  display.clearDisplay();

  display.setTextSize(2); // Draw 2X-scale text
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(10, 0);
  display.println(F("scroll"));
  display.display();      // Show initial text
  delay(100);

  // Scroll in various directions, pausing in-between:
  display.startscrollright(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrollleft(0x00, 0x0F);
  delay(2000);
  display.stopscroll();
  delay(1000);
  display.startscrolldiagright(0x00, 0x07);
  delay(2000);
  display.startscrolldiagleft(0x00, 0x07);
  delay(2000);
  display.stopscroll();
  delay(1000);
}
```

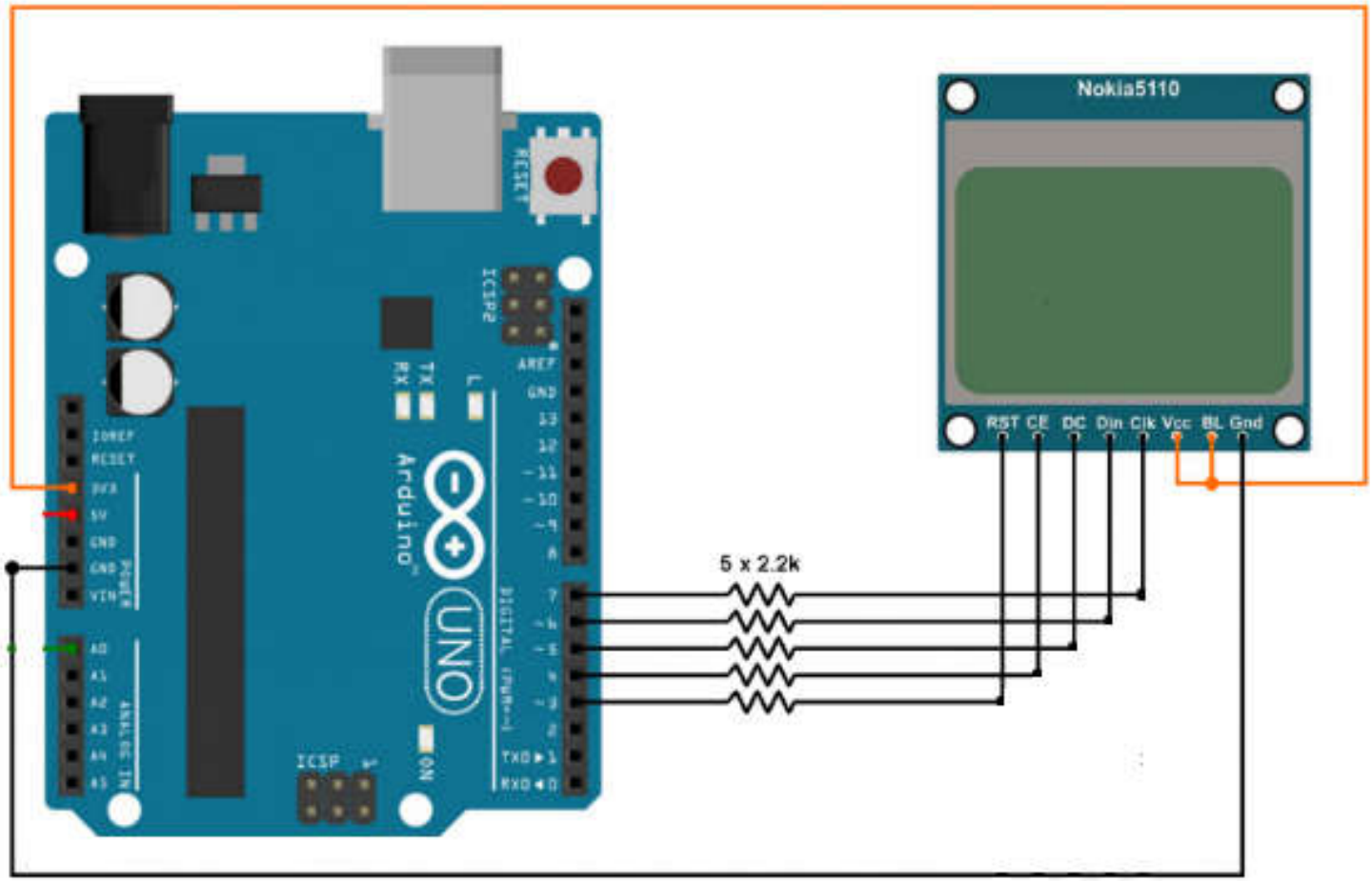
OLED SSD1306 I2C

```
for(;;) { // Loop forever...
    display.clearDisplay(); // Clear the display buffer

    // Draw each snowflake:
    for(f=0; f< NUMFLAKES; f++) {
      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], bitmap, w, h, SSD1306_WHITE);
    }

    display.display(); // Show the display buffer on the screen
    delay(200);        // Pause for 1/10 second

    // Then update coordinates of each flake...
    for(f=0; f< NUMFLAKES; f++) {
      icons[f][YPOS] += icons[f][DELTAY];
      // If snowflake is off the bottom of the screen...
      if (icons[f][YPOS] >= display.height()) {
        // Reinitialize to a random position, just off the top
        icons[f][XPOS]   = random(1 - LOGO_WIDTH, display.width());
        icons[f][YPOS]   = -LOGO_HEIGHT;
        icons[f][DELTAY] = random(1, 6);
      }
    }
  }
}
```

NOKIA 5110 GRAPHIC LCD



NOKIA 5110 GRAPHIC LCD



NOKIA 5110 GRAPHIC LCD

```
/*********************************************************************
//This is an example sketch for our Monochrome Nokia 5110 LCD Displays
//These displays use SPI to communicate, 4 or 5 pins are required to
//interface
//Ex. pcdtest_nokia5110.ino
*********************************************************************/
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

// Software SPI (slower updates, more flexible pin options):
// pin 7 - Serial clock out (SCLK)
// pin 6 - Serial data out (DIN)
// pin 5 - Data/Command select (D/C)
// pin 4 - LCD chip select (CS)
// pin 3 - LCD reset (RST)
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

// Hardware SPI (faster, but must use certain hardware pins):
// SCK is LCD serial clock (SCLK) - this is pin 13 on Arduino Uno
// MOSI is LCD DIN - this is pin 11 on an Arduino Uno
// pin 5 - Data/Command select (D/C)
// pin 4 - LCD chip select (CS)
// pin 3 - LCD reset (RST)
// Adafruit_PCD8544 display = Adafruit_PCD8544(5, 4, 3);
// Note with hardware SPI MISO and SS pins aren't used but will still be read
// and written to during SPI transfer.  Be careful sharing these pins!

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2


#define LOGO16_GLCD_HEIGHT 16
#define LOGO16_GLCD_WIDTH  16
```

```
static const unsigned char PROGMEM logo16_glcd_bmp[] =
{ B00000000, B11000000,
  B00000001, B11000000,
  B00000001, B11000000,
  B00000011, B11100000,
  B11110011, B11100000,
  B11111110, B11111000,
  B01111110, B11111111,
  B00110011, B10011111,
  B00011111, B11111100,
  B00001101, B01110000,
  B00011011, B10100000,
  B00111111, B11100000,
  B00111111, B11110000,
  B01111100, B11110000,
  B01110000, B01110000,
  B00000000, B00110000 };

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

  display.begin();
  // init done

  // you can change the contrast around to adapt the display
  // for the best viewing!
  display.setContrast(50);

  display.display(); // show splashscreen
  delay(2000);
  display.clearDisplay();   // clears the screen and buffer

  // draw a single pixel
  display.drawPixel(10, 10, BLACK);
  display.display();
  delay(2000);
  display.clearDisplay();
```


NOKIA 5110 GRAPHIC LCD

```
// draw many lines
  testdrawline();
  display.display();
  delay(2000);
  display.clearDisplay();

  // draw rectangles
  testdrawrect();
  display.display();
  delay(2000);
  display.clearDisplay();

  // draw multiple rectangles
  testfillrect();
  display.display();
  delay(2000);
  display.clearDisplay();

  // draw mulitple circles
  testdrawcircle();
  display.display();
  delay(2000);
  display.clearDisplay();

  // draw a circle, 10 pixel radius
  display.fillCircle(display.width()/2, display.height()/2, 10, BLACK);
  display.display();
  delay(2000);
  display.clearDisplay();

  testdrawroundrect();
  delay(2000);
  display.clearDisplay();

  testfillroundrect();
  delay(2000);
  display.clearDisplay();
```

```
  testdrawtriangle();
  delay(2000);
  display.clearDisplay();

  testfilltriangle();
  delay(2000);
  display.clearDisplay();

  // draw the first ~12 characters in the font
  testdrawchar();
  display.display();
  delay(2000);
  display.clearDisplay();

  // text display tests
  display.setTextSize(1);
  display.setTextColor(BLACK);
  display.setCursor(0,0);
  display.println("Hello, world!");
  display.setTextColor(WHITE, BLACK); // 'inverted' text
  display.println(3.141592);
  display.setTextSize(2);
  display.setTextColor(BLACK);
  display.print("0x"); display.println(0xDEADBEEF, HEX);
  display.display();
  delay(2000);

  // rotation example
  display.clearDisplay();
  display.setRotation(1);  // rotate 90 degrees counter clockwise, can also use values of 2 and 3
to go further.
```

NOKIA 5110 GRAPHIC LCD

```
display.setTextSize(1);
  display.setTextColor(BLACK);
  display.setCursor(0,0);
  display.println("Rotation");
  display.setTextSize(2);
  display.println("Example!");
  display.display();
  delay(2000);

  // revert back to no rotation
  display.setRotation(0);

  // miniature bitmap display
  display.clearDisplay();
  display.drawBitmap(30, 16,  logo16_glcd_bmp, 16, 16, 1);
  display.display();

  // invert the display
  display.invertDisplay(true);
  delay(1000);
  display.invertDisplay(false);
  delay(1000);

  // draw a bitmap icon and 'animate' movement
  testdrawbitmap(logo16_glcd_bmp, LOGO16_GLCD_WIDTH, LOGO16_GLCD_HEIGHT);
}


void loop() {

}
```

NOKIA 5110 GRAPHIC LCD

```
void testdrawbitmap(const uint8_t *bitmap, uint8_t w, uint8_t h) {
  uint8_t icons[NUMFLAKES][3];
  randomSeed(666);     // whatever seed

  // initialize
  for (uint8_t f=0; f< NUMFLAKES; f++) {
    icons[f][XPOS] = random(display.width());
    icons[f][YPOS] = 0;
    icons[f][DELTAY] = random(5) + 1;

    Serial.print("x: ");
    Serial.print(icons[f][XPOS], DEC);
    Serial.print(" y: ");
    Serial.print(icons[f][YPOS], DEC);
    Serial.print(" dy: ");
    Serial.println(icons[f][DELTAY], DEC);
  }

  while (1) {
    // draw each icon
    for (uint8_t f=0; f< NUMFLAKES; f++) {
      display.drawBitmap(icons[f][XPOS], icons[f][YPOS], logo16_glcd_bmp, w, h, BLACK);
    }
    display.display();
    delay(200);

    // then erase it + move it
    for (uint8_t f=0; f< NUMFLAKES; f++) {
      display.drawBitmap(icons[f][XPOS], icons[f][YPOS],  logo16_glcd_bmp, w, h, WHITE);
      // move it
      icons[f][YPOS] += icons[f][DELTAY];
      // if its gone, reinit
      if (icons[f][YPOS] > display.height()) {
           icons[f][XPOS] = random(display.width());
           icons[f][YPOS] = 0;
           icons[f][DELTAY] = random(5) + 1;
```

NOKIA 5110 GRAPHIC LCD

```
 }
    }
   }
}


void testdrawchar(void) {
  display.setTextSize(1);
  display.setTextColor(BLACK);
  display.setCursor(0,0);

  for (uint8_t i=0; i < 168; i++) {
    if (i == '\n') continue;
    display.write(i);
    //if ((i > 0) && (i % 14 == 0))
      //display.println();
  }
  display.display();
}

void testdrawcircle(void) {
  for (int16_t i=0; i<display.height(); i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, BLACK);
    display.display();
  }
}

void testfillrect(void) {
  uint8_t color = 1;
  for (int16_t i=0; i<display.height()/2; i+=3) {
    // alternate colors
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, color%2);
    display.display();
    color++;
  }
}
```

NOKIA 5110 GRAPHIC LCD

```
void testdrawtriangle(void) {
  for (int16_t i=0; i<min(display.width(),display.height())/2; i+=5) {
    display.drawTriangle(display.width()/2, display.height()/2-i,
                     display.width()/2-i, display.height()/2+i,
                     display.width()/2+i, display.height()/2+i, BLACK);
    display.display();
  }
}

void testfilltriangle(void) {
  uint8_t color = BLACK;
  for (int16_t i=min(display.width(),display.height())/2; i>0; i-=5) {
    display.fillTriangle(display.width()/2, display.height()/2-i,
                     display.width()/2-i, display.height()/2+i,
                     display.width()/2+i, display.height()/2+i, color);
    if (color == WHITE) color = BLACK;
    else color = WHITE;
    display.display();
  }
}

void testdrawroundrect(void) {
  for (int16_t i=0; i<display.height()/2-2; i+=2) {
    display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i, display.height()/4,
BLACK);
    display.display();
  }
}

void testfillroundrect(void) {
  uint8_t color = BLACK;
  for (int16_t i=0; i<display.height()/2-2; i+=2) {
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i, display.height()/4,
color);
    if (color == WHITE) color = BLACK;
    else color = WHITE;
    display.display();
```


NOKIA 5110 GRAPHIC LCD

```
 }
}

void testdrawrect(void) {
  for (int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i, BLACK);
    display.display();
  }
}

void testdrawline() {
  for (int16_t i=0; i<display.width(); i+=4) {
    display.drawLine(0, 0, i, display.height()-1, BLACK);
    display.display();
  }
  for (int16_t i=0; i<display.height(); i+=4) {
    display.drawLine(0, 0, display.width()-1, i, BLACK);
    display.display();
  }
  delay(250);

  display.clearDisplay();
  for (int16_t i=0; i<display.width(); i+=4) {
    display.drawLine(0, display.height()-1, i, 0, BLACK);
    display.display();
  }
  for (int8_t i=display.height()-1; i>=0; i-=4) {
    display.drawLine(0, display.height()-1, display.width()-1, i, BLACK);
    display.display();
  }
  delay(250);

  display.clearDisplay();
  for (int16_t i=display.width()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, i, 0, BLACK);
    display.display();
```


```
 }
  for (int16_t i=display.height()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, 0, i, BLACK);
    display.display();
  }
  delay(250);

  display.clearDisplay();
  for (int16_t i=0; i<display.height(); i+=4) {
    display.drawLine(display.width()-1, 0, 0, i, BLACK);
    display.display();
  }
  for (int16_t i=0; i<display.width(); i+=4) {
    display.drawLine(display.width()-1, 0, i, display.height()-1, BLACK);
    display.display();
  }
  delay(250);
}
```