**Erasmus+ KA210-VET**

**Small-scale partnerships in vocational**

**education and training**

**Project Title: “Using Arduinos in Vocational Training”**

**Project Acronym: “UsingARDinVET”**

**Project No: “2023-1-RO01-KA210-VET-000156616”**

**LCD Module and Training KIT**

****

**LCD Module and Training KIT**

 In this module we want to explain how to display status messages or sensor readings of Arduino on LCD displays . They are extremely common and a fast way to add a readable interface to your project.

An LCD is short for Liquid Crystal Display. It is a display unit which uses liquid crystals to produce a visible image. When current is applied to this special kind of crystal, it turns opaque blocking the backlight that lives behind the screen. As a result that particular area will become dark compared to other. And that’s how characters are displayed on the screen.

**Interfacing 16×2 Character LCD Module with Arduino**

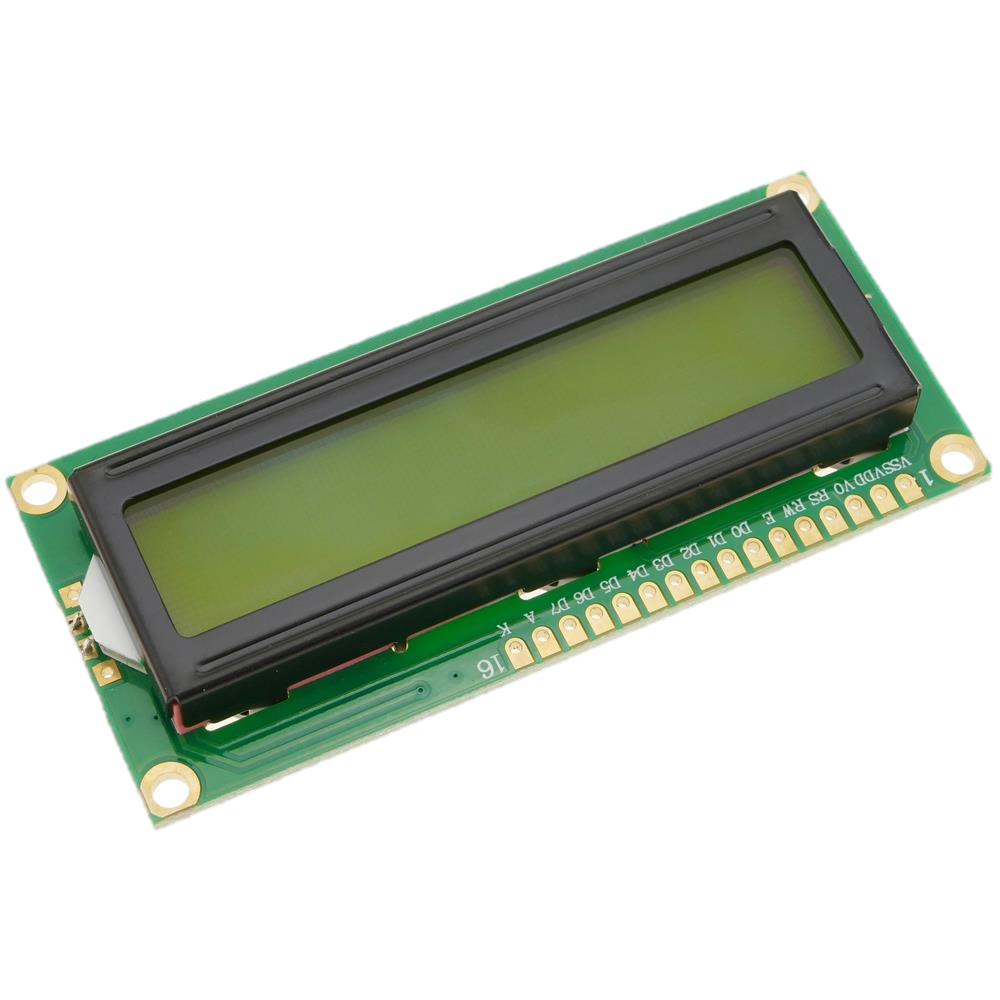
There are different kind of LCD displays that you can connect to your Arduino. The most common one is based on parallel interface LCD controller chip from Hitachi called the HD44780.

These LCDs are ideal for displaying text/characters only, hence the name ‘Character LCD’. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row.

There is a little rectangles for each character on the display, each of these rectangles is a grid of 5×8 pixels.

Although they display only text, they do come in many sizes and colors: for example, 16×1, 16×4, 20×4, with white text on blue background, with black text on green and many more.

The Arduino community has already developed a library to handle HD44780 LCDs (**LiquidCrystal Library)**; so we’ll have them interfaced in a few time.



Below is the LCD pinout:

* **VSS:**is a ground pin and should be connected to the ground of Arduino;
* **VDD:** connected to 5 V;

* **VD:**for contrast adjustment (connected to the central pin of the potentiometer) ;
* **RS:** controls in which lcd memory zone the sent data are stored;
* **R/W:**pin to select read/write mode;
* **E:** if enabled, allows the LCD module to perform special instructions;

* **D0 to** **D7:** data transmission;
* **A and K:**anode and cathode to provide backlight to the LCD module.

**Arduino - LCD Functions (Commands)**

LiquidCrystal lcd() - Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function’s parameters. Syntax:

LiquidCrystal(rs, enable, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)

LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Parameters:

rs: the number of the Arduino pin that is connected to the RS pin on the LCD

rw: the number of the Arduino pin that is connected to the RW pin on the LCD (optional)

enable: the number of the Arduino pin that is connected to the enable pin on the LCD

d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

lcd.begin() - Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands. Syntax:

lcd.begin(cols, rows)

Parameters:

lcd: a variable of type LiquidCrystal

cols: the number of columns that the display has

rows: the number of rows that the display has

lcd.print() - Prints text to the LCD. Syntax:

lcd.print(data)

lcd.print(data, BASE)

Parameters:

lcd: a variable of type LiquidCrystal

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns

byte print() will return the number of bytes written, though reading that number is optional

lcd.setCursor() - Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed. Syntax:

lcd.setCursor(col, row)

Parameters:

lcd: a variable of type LiquidCrystal

col: the column at which to position the cursor (with 0 being the first column)

row: the row at which to position the cursor (with 0 being the first row)

lcd.clear(); - Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax: lcd.clear()

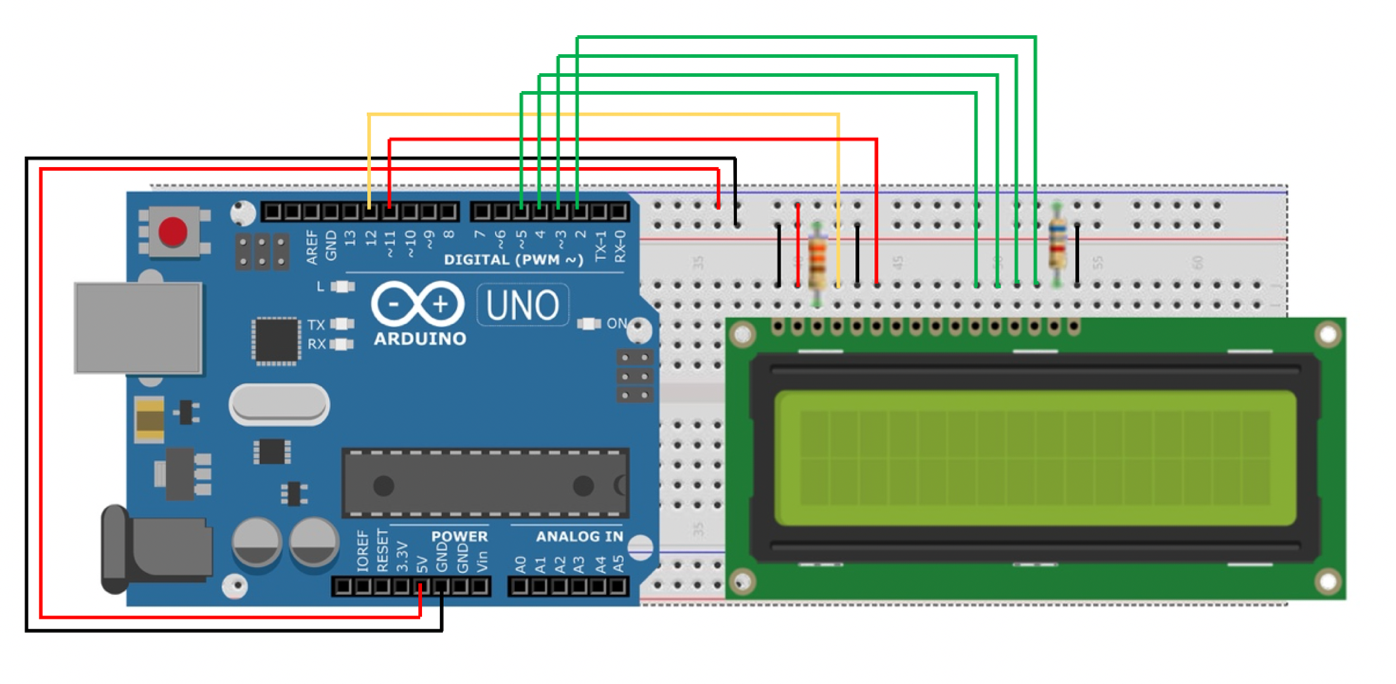
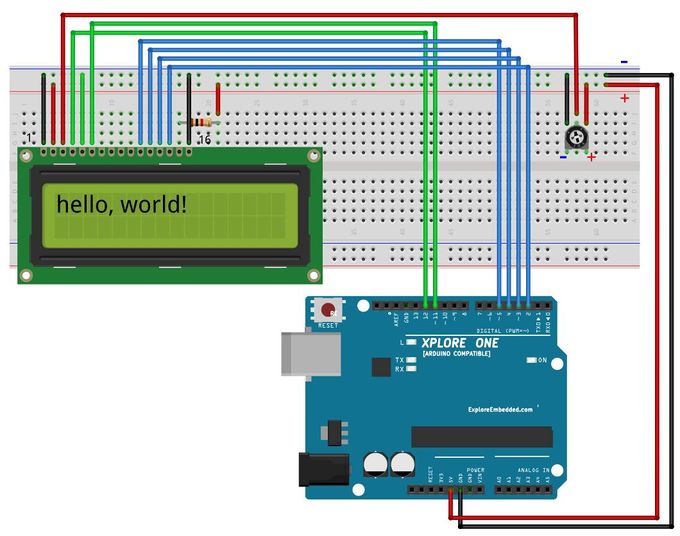
Parameters: lcd: a variable of type LiquidCrystal

**Circuit 1:**

**Circuit title:**  "Print a message to the LCD”

**Circuit Explanation:**It is possible to connect an LCD display to the microcontroller and print the desired messages on the screen.

**Note:** We have to include the library LiquidCrystal.h to use the LCD functions descripted below.



/\* Print a message \*/

#include <LiquidCrystal.h> // include the library code

//constants for the number of rows and columns in the LCD

const int numRows = 2;

const int numCols = 16;

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()

{

lcd.begin(numCols, numRows);

lcd.print("hello, world!"); // Print a message to the LCD.

}

void loop() {

}

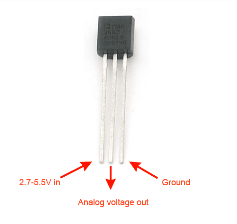
**Circuit 2:**

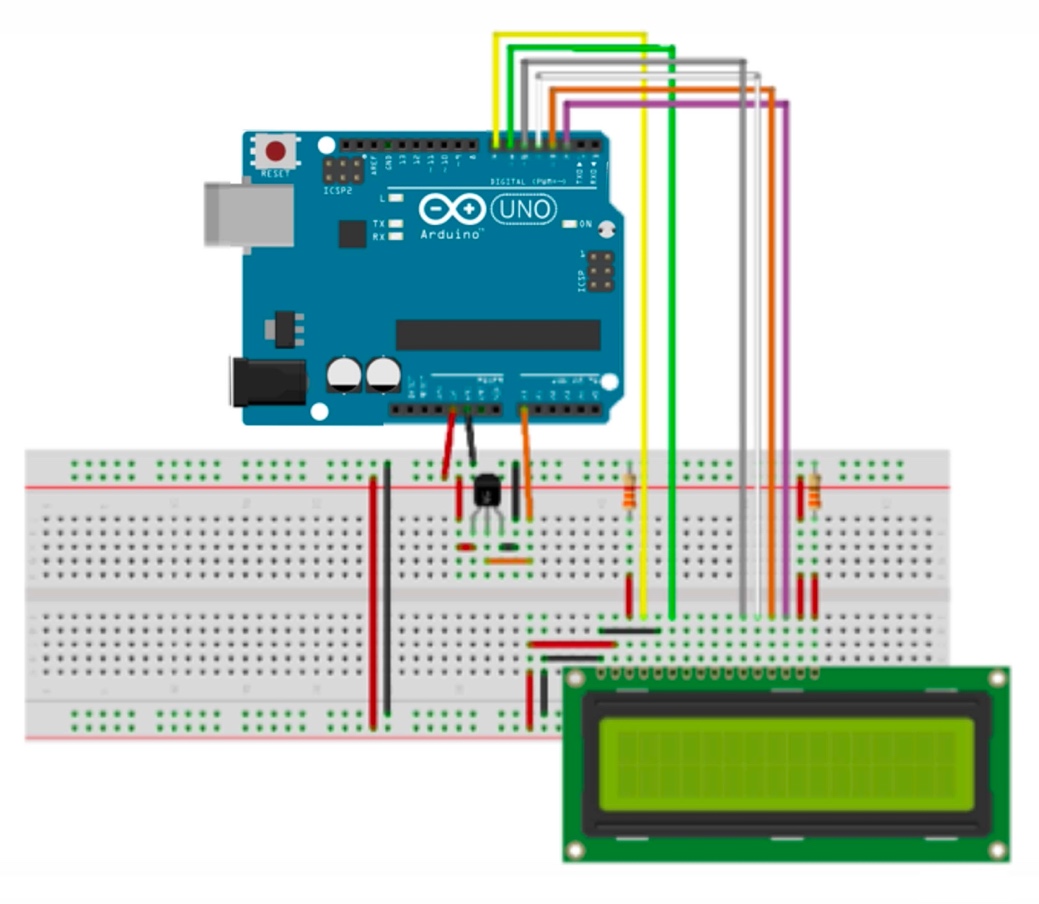
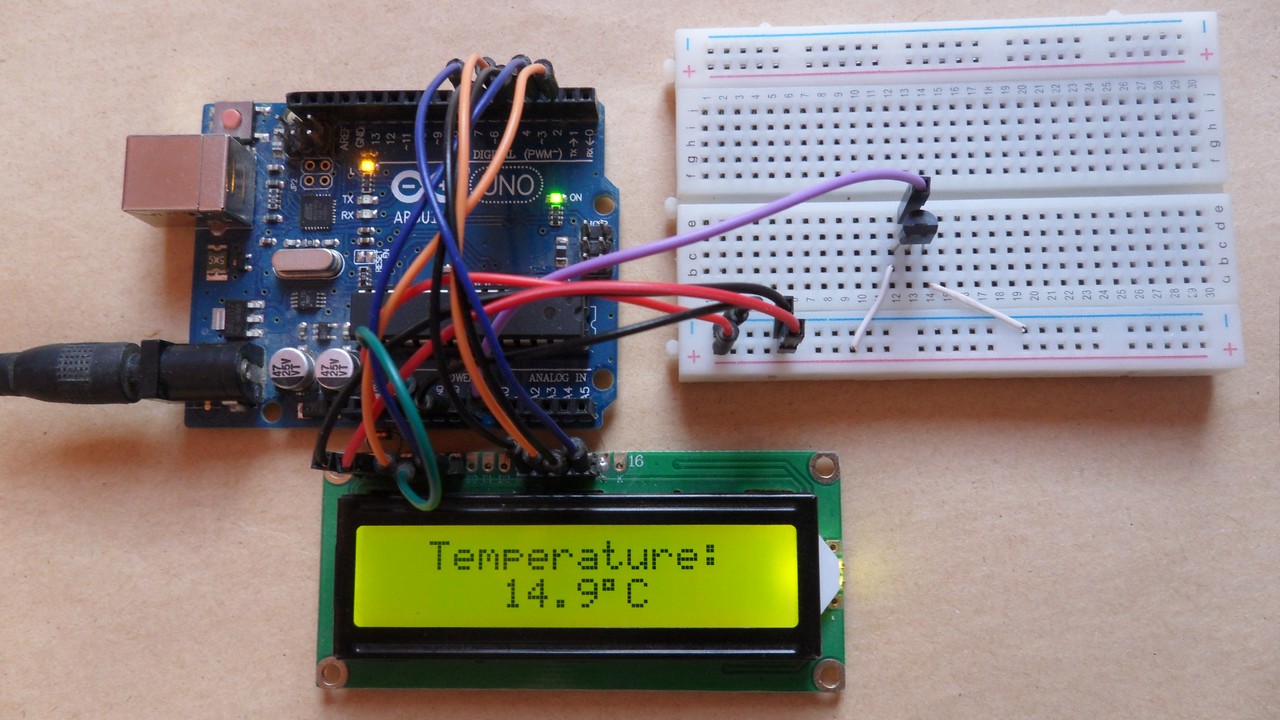
**Circuit title:** "Digital Thermometer”

**Circuit Explanation: To** create a digital thermometer with Arduino. The temperature will be taken with the LM35 temperature sensor and must be displayed on an LCD display and updated every second.

The Vo pin of the LCD display adjusts the contrast of the characters displayed in the display. As mentioned above, it must be connected to a 3.3 kΩ resistance connected to GND. However, in some displays you may need to connect the Vo pin directly to GND

**Note:** the LM35 sensor has 3 terminals: one for the power supply, one for mass and one for the output of the voltage proportional to the detected temperature, which is equal to 10 mV for each degree centigrade, and is calibrated in degrees Celsius.





/\* Digital Thermometer \*/

#include <LiquidCrystal.h> //Library to drive LCD display

#define pin\_temp A0 //Temperature sensor Vout foot connection pin

float temp = 0; //Variable in which the detected temperature will be stored

LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Initializing the library with LCD display pins

void setup()

{

lcd.begin(16, 2); //Setting the number of columns and rows in the display LCD  lcd.setCursor(0, 0); //Move the cursor over the first row (row 0) and the first column lcd.print ("Temperature:"); Print the message 'Temperature:' on the first line

//\*Imposed ADC Vref at 1.1V

(for greater accuracy in temperature calculation)

IMPORTANT: If you use Arduino Mega replace INTERNAL with INTERNAL1V1 \*/ analogReference(INTERNAL);

}

void loop()

{

/\*calculate the temperature =============================================\*/

temp = 0;

for (int i = 0; i < 5; i++) { //It executes the next statement 5 times

temp += (analogRead(pin\_temp) / 9.31); // It calculates temperature and sum at variable 'temp'

}

temp /= 5; //It calculates the mathematical average of temperature values

/\*=================================================================\*/

/\*I see the temperature on the LCD display =============================================\*/

lcd.setCursor(0, 1); //lcd.print(temp); Move the cursor over the first column and the second row lcd.print(temp);

// Mold on LCD display temperature

lcd.print(" C"); // Mold a space and the 'C' font on the display

/\*===========================================\*/

 delay(1000); //Delay by one second (it can be changed)

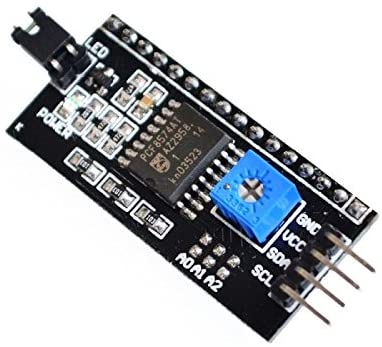
}

**Interfacing I2C LCD with Arduino**

If you want to connect an LCD display with Arduino, you have to consume a lot of pins on the Arduino. Even in 4-bit mode, the Arduino still requires a total of seven connections, which is half of the available digital I/O pins.

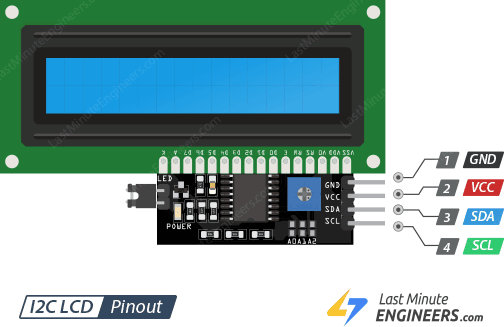
The solution is to use an LCD display that interfaces with I2C protocol. It only consumes two I/O pins which can be not even part of a digital I/O pin set and can be shared with other I2C devices as well.

The most diffuse I2C LCD display consists of a HD44780 based character LCD display and an I2C LCD adapter.



The most important part of the adapter is the 8-Bit I/O Expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required by the LCD display. The board also comes with a small potentiometer to make fine adjustments to the contrast of the display. If you are using more than a device on the same I2C bus, you may need to set a different I2C address for the board, so that it does not conflict with another I2C device. For this reason, the board has three solder jumpers (A0, A1 and A2).

An I2C LCD has only 4 pins that interface it to the Arduino.



The pinout is as follows:

* **GND:** is a ground pin and should be connected to the ground of Arduino;
* **VCC:** supplies power to the module and the LCD. Connect it to the 5V output of the Arduino or a separate power supply;
* **SDA:** is a Serial Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino;
* **SCL:** is a Serial Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

On the Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. They are also known as A5 (SCL) and A4 (SDA). Refer the below table to identify the correct pins depending on the Arduino model.

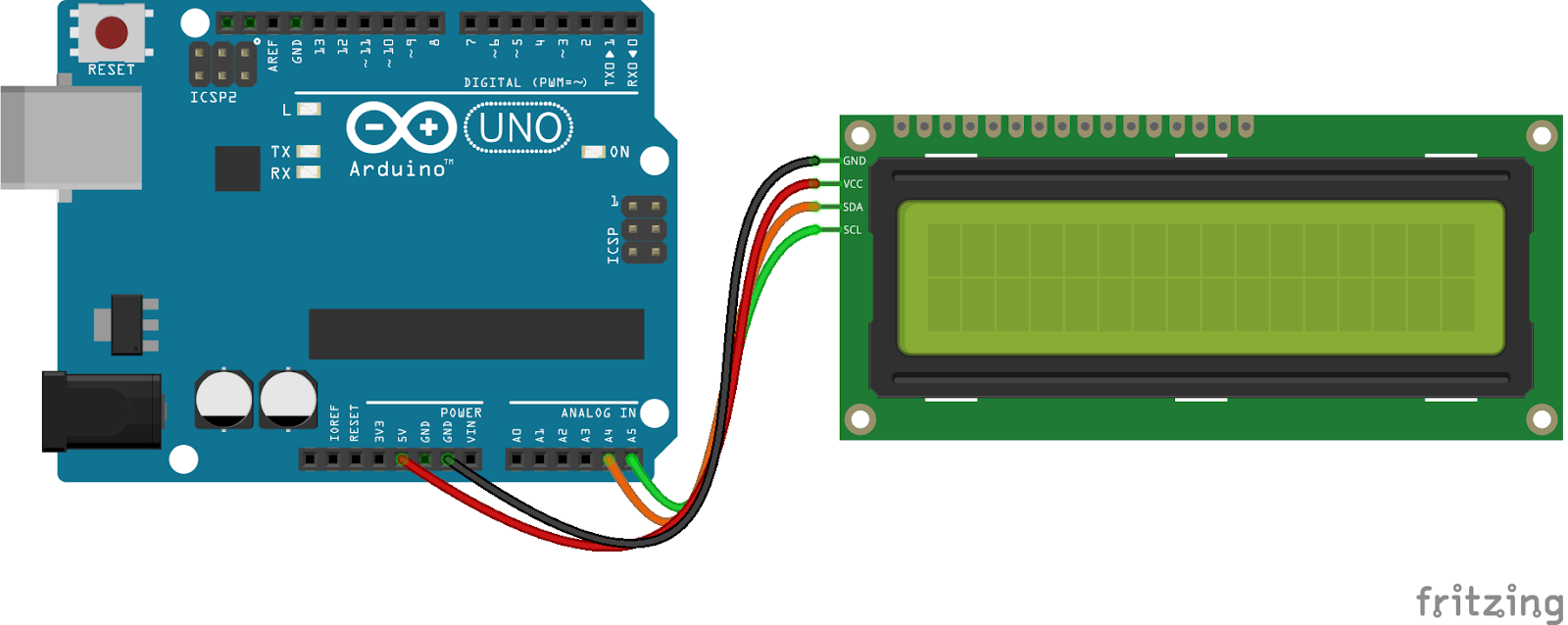
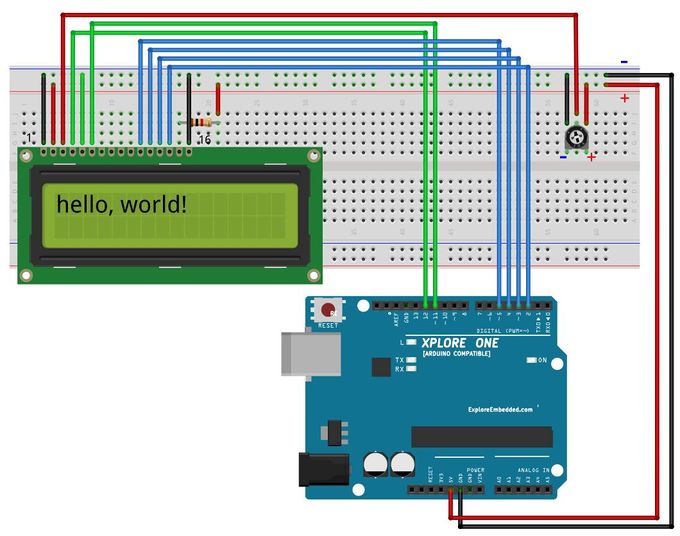
|  |  |  |
| --- | --- | --- |
|  | SCL | SDA |
| Arduino Uno | A5 | A4 |
| Arduino Nano | A5 | A4 |
| Arduino Mega | 21 | 20 |
| Leonardo/Micro | 3 | 2 |

**Circuit 3:**

**Circuit title:**  "Print a message to the I2C LCD”

**Circuit Explanation:**It is possible to connect an LCD display to the microcontroller with only 4 pins and print the desired messages on the screen.

**Note:** We have to install a library called LiquidCrystal\_I2C. This library is an improved version of the LiquidCrystal library that comes packaged with your Arduino IDE.



/\* Print a message on a I2C Display \*/

#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("with I2C protocol!");

}

void loop() {

}

**Interfacing OLED Graphic Display Module with Arduino**

Another possibility to use the I2C protocol is choosing an OLED (Organic Light-Emitting Diode) display. They’re super-light and thin, and produce a brighter and crisper picture.



An OLED display works without a backlight. This is why the display has such high contrast, extremely wide viewing angle and can display deep black levels. Absence of backlight significantly reduces the power required to run the OLED.

As we can see from the figure the pinout is the classic four-pin I2C interface already seen above.

The Arduino community has already developed a few libraries to handle these OLED displays, such as Adafruit’s SSD1306 library. To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.

**Arduino - OLED Graphic Display Module Functions (Commands)**

pinMode();

display.begin()

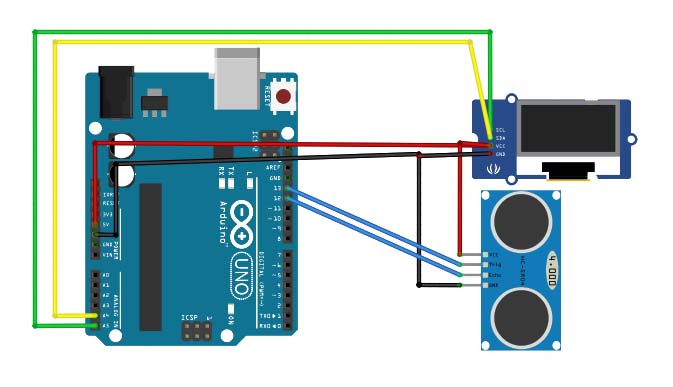
display.clearDisplay();

**Circuit 4:**

**Circuit title:**  "Distance sensor”

**Circuit Explanation:**The distance will be taken with the HC-SR04 ultrasonic sensor and be displayed on a OLED display.

**Note:** There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground).



/\* Distance sensor \*/

#include <SPI.h> // this library allows you to communicate with SPI devices, with the Arduino as the master device.

#include <Wire.h> // this library allows you to communicate with I2C / TWI devices

#include <Adafruit\_GFX.h> // the OLED display’s libraries

#include <Adafruit\_SSD1306.h>

#define CommonSenseMetricSystem

#define trigPin 13 // define the pins of the sensor

#define echoPin 12

void setup() {

Serial.begin (9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

display.begin(SSD1306\_SWITCHCAPVCC, 0x3C); //initialize with the I2C addr 0x3C (128x64)

display.clearDisplay();

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2) / 29.1;

display.setCursor(22,20); //oled display setting cursor

display.setTextSize(3); //size of the text

display.setTextColor(WHITE); //if you write black it erases things

display.println(distance); //print our variable

display.setCursor(85,20);

display.setTextSize(3);

display.println("cm");

display.display();

delay(500);

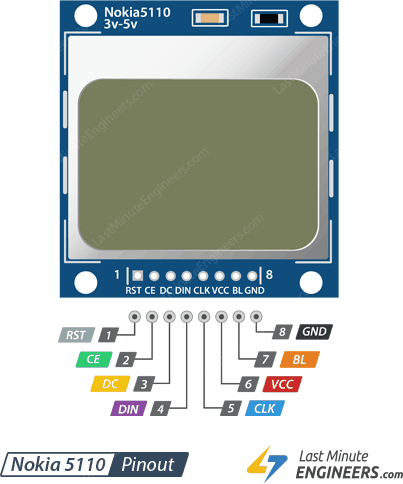
display.clearDisplay();

Serial.println(distance);//debug

}

**Interfacing Nokia 5110 Graphic LCD Display with Arduino**

You can interface Arduino with little LCDs similar to that Nokia used in their 3310 and 5110 cell phones. these displays are small (only about 1.5″), inexpensive, easy to use, fairly low power (as low as 6 to 7mA only) and can display text as well as bitmaps.



These are graphic display of 84×48 pixels. They interfaces to microcontrollers through a serial bus interface similar to SPI. The LCD also comes with a backlight in different colors such as red, green, blue and white. The backlight is nothing but four LEDs spaced around the edges of the display.

It has 8 pins that interface it to the Arduino, the pinout is as follows:

* **RST:** resets the display. It’s an active low pin meaning; you can reset the display by pulling it low. You can also connect this pin to the Arduino reset so that it will reset the screen automatically;
* **CE(Chip Enable**):  is used to select one of many connected devices sharing same SPI bus;
* **D/C(Data/Command):** pin tells the display whether the data it’s receiving is a command or displayable data;
* **DIN:** is a serial data pin for SPI interface;
* **CLK:**is a serial clock pin for SPI interface;
* **VCC:** supplies power for the LCD which we connect to the 3.3V volts pin on the Arduino;
* **BL(Backlight):**controls the backlight of the display. To control its brightness, you can add a potentiometer or connect this pin to any PWM-capable Arduino pin;
* **GND:** is a ground pin and should be connected to the ground of Arduino.

You can connect data transmission pins to any digital I/O pin. The LCD has 3v communication levels, so we cannot directly connect these pins to the Arduino. One way is to add resistors inline with each data transmission pin. Just add 10kΩ resistors between the CLK, DIN, D/C, and RST pins and a 1kΩ resistor between CE. The backlight(BL) pin is connected to 3.3V via 330Ω current limiting resistor. You can add a potentiometer or connect this pin to any PWM-capable Arduino pin, if you wish to control its brightness.

The Arduino community has already developed a few libraries to handle these NOKIA displays, such as Adafruit’s PCD8544 Nokia 5110 LCD library. To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.

**Circuit 5:**

**Circuit title:**  "Text Rotation”

**Circuit Explanation:**You can rotate the contents of the display by calling setRotation() function. It allows you to view your display in portrait mode, or flip it upside down.

**Note:** The function accepts only one parameter that corresponds to 4 cardinal rotations. This value can be any non-negative integer starting from 0. Each time you increase the value, the contents of the display are rotated 90 degrees counter clockwise. For example:

* 0 – Keeps the screen to the standard landscape orientation.
* 1 – Rotates the screen 90° to the right.
* 2 – Flips the screen upside down.
* 3 – Rotates the screen 90° to the left.



/\* Text Rotation \*/

#include <SPI.h> // this library allows you to communicate with SPI devices, with the Arduino as the master device.

#include <Adafruit\_GFX.h> // the OLED display’s libraries

#include <Adafruit\_PCD8544.h>

// Declare LCD object for software SPIAdafruit\_PCD8544(CLK,DIN,D/C,CE,RST);

Adafruit\_PCD8544 display = Adafruit\_PCD8544(7, 6, 5, 4, 3);

void setup() {

Serial.begin(9600);

//Initialize Display

display.begin();

display.setContrast(57); // you can change the contrast around to adapt the display

display.clearDisplay(); // clear the buffer.

// Text Rotation

while(1)

{

display.clearDisplay();

display.setRotation(rotatetext);

display.setTextSize(1);

display.setTextColor(BLACK);

display.setCursor(0,0);

display.println("Text Rotation");

display.display();

delay(1000);

display.clearDisplay();

rotatetext++;

}

}

void loop() {}

**Circuit 6**:

**Circuit title:** "Button counter"

**Circuit Explanation:** an OLED display showing a number, which can be incremented and decremented at the click of two different buttons.

Immagine che contiene testo, elettronico, screenshot

Descrizione generata automaticamente

#include <U8glib.h> //lcd libraries  
#include "U8glib.h"  
U8GLIB\_SSD1306\_128X64 u8g(U8G\_I2C\_OPT\_NONE|U8G\_I2C\_OPT\_DEV\_0); //display model

int button\_plus = 8, button\_minus = 7; //declaration pin buttons  
int state\_plus, state\_minus, number=0;

void setup() {

pinMode(button\_plus,INPUT);  
 pinMode(button\_minus,INPUT);

Serial.begin(9600);

u8g.setFont(u8g\_font\_fub25n); //font to be used for the write of the number  
}

void write\_number(void) {   
 u8g.setPrintPos(10,50);  
 u8g.print(number);  
}

void loop() {

//buttons  
 state\_plus = digitalRead(button\_plus);  
 state\_minus = digitalRead(button\_minus);

if(state\_plus==1)  
{  
 number++;  
 delay(50);   
}

if(state\_minus==1)  
 {  
 number--;  
 delay(50);  
 }

//write de number sul display

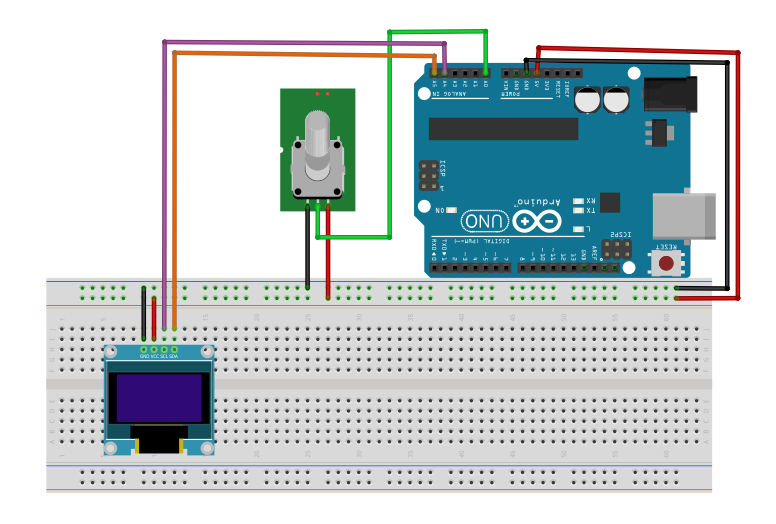
u8g.firstPage();   
 do {  
 write\_number();  
 } while ( u8g.nextPage() );

u8g.firstPage();  
}

**Circuit 7**:

**Circuit title: “**Counter with potentiometer"

**Circuit Explanation:** an OLED display that shows a number, incrementing and decrementing as a potentiometer is turned.



#include <U8glib.h> //lcd libraries

#include "U8glib.h"

U8GLIB\_SSD1306\_128X64 u8g(U8G\_I2C\_OPT\_NONE|U8G\_I2C\_OPT\_DEV\_0); //display model

int potentiometer = 0; //declaration and potentiometer

int state\_pot, stop\_pot, difference, number=0;

void setup() {

Serial.begin(9600);

u8g.setFont(u8g\_font\_fub25n);//font to be used for the write of the number

}

void write\_number(void) {

u8g.setPrintPos(10,50);

u8g.print(number);

}

void loop() {

state\_pot = analogRead(potentiometer);

//potentiometer

stop\_pot = state\_pot;//save the value when it is stop to see if the potentiometer turns clockwise or counterclockwise

delay(100);

state\_pot = analogRead(potentiometer);

difference = stop\_pot-state\_pot;

if((difference>2)||(difference<2))//is used to avoid making trades if the potentiometer is stop

{

number=number-difference/5;//if difference is greater than 0 then it turns clockwise and adds otherwise it subtracts

delay(100);

}

//write the number on display

u8g.firstPage();

do {

write\_number();

} while

( u8g.nextPage() );

u8g.firstPage();

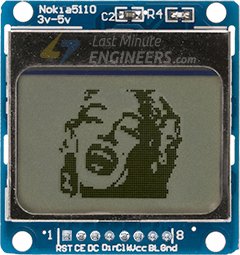
}

**Circuit 8:**

**Circuit title:**  "Marilyn Bmp Image”

**Circuit Explanation:**How to draw bitmap images to the Nokia 5110 LCD Display. In this example there is a portrait of Marilyn Monroe.

**Note:** The screen resolution of Nokia 5110 LCD display is 84×48 pixels, so images larger than that will not display correctly. To show bitmap image on the Nokia 5110 LCD display we need to call drawBitmap() function. It takes six parameters:. top left corner X coordinate, top left corner Y coordinate, byte array of monochrome bitmap, width of bitmap in pixels, height of bitmap in pixels and Color. In our example, the bitmap image is 84×48 in size. So, X & Y coordinates are set to 0 while width & height is set to 84 and 48.



/\* Marilyn Bmp Image \*/

#include <SPI.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_PCD8544.h>

Adafruit\_PCD8544 display = Adafruit\_PCD8544(7, 6, 5, 4, 3);

// 'Marilyn Monroe 84x48', 84x48px

const unsigned char MarilynMonroe [] PROGMEM = {

0x00, 0x00, 0x00, 0x7f, 0x00, 0x02, 0xfe, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xbe, 0x00,

0x00, 0x1f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0xe1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,

0x00, 0x00, 0x0f, 0xf1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xd8, 0xe0,

0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x07, 0xe0, 0x70, 0x00, 0x00, 0x00, 0x00, 0x03,

0x3f, 0xe0, 0x00, 0x07, 0xf0, 0x78, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x70, 0x00, 0x0f, 0xee,

0x7c, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x0f, 0xf7, 0x1c, 0x00, 0x00, 0x00, 0x00,

0x07, 0x80, 0x00, 0x0f, 0xc7, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x0f, 0xf3,

0xdf, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x08, 0x7d, 0xef, 0xff, 0xc0, 0x00, 0x00,

0x00, 0x7f, 0xff, 0x80, 0x30, 0x0f, 0xfc, 0xe0, 0xc0, 0x00, 0x00, 0x01, 0x9e, 0x73, 0xc0, 0xe0,

0x07, 0xf8, 0xc1, 0xc0, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xc0, 0x0f, 0xfd, 0xe1, 0x80, 0x00,

0x00, 0x03, 0xf8, 0x00, 0x01, 0x9c, 0x0f, 0xff, 0xc1, 0xc0, 0x00, 0x00, 0x02, 0xc0, 0x00, 0x01,

0x9f, 0xbf, 0xfe, 0x01, 0x40, 0x00, 0x00, 0x02, 0x60, 0x00, 0x03, 0x07, 0xef, 0xff, 0x01, 0x40,

0x00, 0x00, 0x00, 0x60, 0x00, 0x07, 0x01, 0xf7, 0xff, 0x80, 0xc0, 0x00, 0x00, 0x00, 0x50, 0x01,

0xdf, 0x00, 0x7f, 0xff, 0x1c, 0x80, 0x00, 0x00, 0x00, 0x40, 0x01, 0xff, 0x00, 0x1f, 0xff, 0x1e,

0xe0, 0x00, 0x00, 0x02, 0x08, 0x00, 0x3f, 0x80, 0x07, 0xef, 0x03, 0xe0, 0x00, 0x00, 0x06, 0x08,

0x00, 0x03, 0xc0, 0x07, 0xdf, 0x07, 0xc0, 0x00, 0x00, 0x06, 0x08, 0x0f, 0x81, 0x80, 0x1f, 0xdf,

0x1f, 0x80, 0x00, 0x00, 0x03, 0x08, 0x1f, 0x98, 0x00, 0x3f, 0xfe, 0x19, 0x80, 0x00, 0x00, 0x18,

0x08, 0x3f, 0xfe, 0x00, 0x7f, 0xfe, 0x3f, 0x00, 0x00, 0x00, 0x08, 0x08, 0x30, 0x3f, 0x00, 0xff,

0xff, 0x3f, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x76, 0x0f, 0x89, 0xff, 0xff, 0x9f, 0x00, 0x00, 0x00,

0x03, 0xe0, 0x7f, 0xc3, 0x81, 0xff, 0xfe, 0x9f, 0x80, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xf3, 0xc3,

0xff, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xfd, 0xc3, 0xff, 0xfe, 0x5e, 0x00, 0x00,

0x00, 0x03, 0xf0, 0x7f, 0xff, 0xc3, 0xff, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x71, 0xff,

0x87, 0xff, 0xe3, 0xff, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x7c, 0x3f, 0x87, 0xff, 0xe3, 0xfe, 0x00,

0x00, 0x00, 0x0f, 0xf0, 0x3c, 0xff, 0x05, 0xff, 0xf3, 0xfc, 0x00, 0x00, 0x00, 0x0f, 0xf0, 0x0f,

0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff, 0xf8,

0x00, 0x00, 0x00, 0x0c, 0x78, 0x00, 0x00, 0x13, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x0e, 0x78,

0x00, 0x00, 0x23, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0e, 0xf8, 0x00, 0x00, 0x47, 0xff, 0xff,

0xf0, 0x00, 0x00, 0x00, 0x0c, 0xfa, 0x00, 0x01, 0x8f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x08,

0x7b, 0x00, 0x03, 0x3f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x0c, 0x3b, 0xf8, 0x0f, 0xff, 0xff,

0xff, 0xe0, 0x00, 0x00, 0x00, 0x0f, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00,

0x07, 0xfb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x71, 0xff, 0xff, 0xff,

0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x41, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x00

};

void setup() {

Serial.begin(9600);

display.begin();

display.setContrast(57);

display.clearDisplay();

// Display bitmap

display.drawBitmap(0, 0, MarilynMonroe, 84, 48, BLACK);

display.display();

// Invert Display

//display.invertDisplay(1);

}

void loop() {}