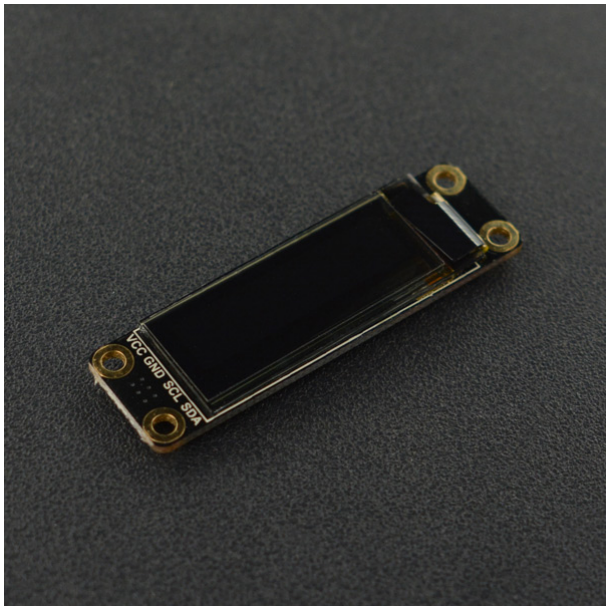



(https://www.dfrobot.com/product-2019.html?search=DFR0647&description=true)

## Introduction

This OLED display module uses the SSD1306 drive chip, has 128x32 self-illuminating white pixels, and adopts the back chip-mounting pad design. Small size, the length is 41mm and the width is only 12mm. With its small size, simple wiring and low power consumption, it can be applied to many display applications, such as wearable display devices, Mini small game consoles, desktop widgets, etc.

The display module adopts a high-contrast, low-power OLED display. Full-screen lighting consumes approximately 25mA (3.3V) 25mA (5V) and full-screen off power consumption: approx. 2mA (3.3V) 2mA (5V). It is compatible with controllers such as Arduino UNO, Leonardo, ESP32/ESP8266, FireBettle-M0 and so on. Internal power-on reset processing, IIC communication methods and the chip-mounting pad, so that your welding line will be more convenient and simpler. The module has an ultra-narrow PCB design that is more flexible to match more applications.

> ⚠ **NOTE**: Turn the pixels off when you are not using the display to prevent the use of some pixels too long to darken them, which will cause the brightness uniformity problem.

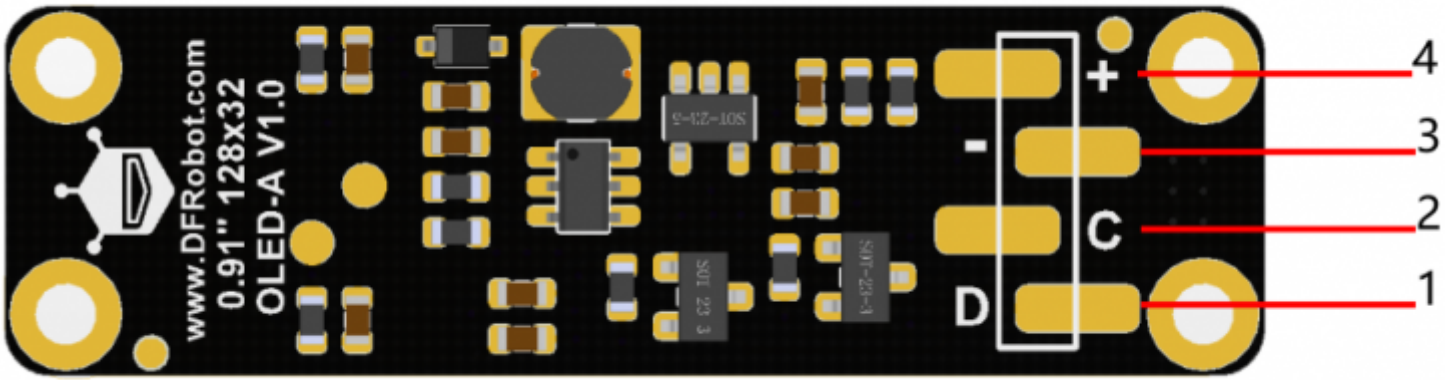
## Features

- Chip-mounting pad
- Self-illuminating OLED display
- Controllable pixels and low power consumption
- Supports power-on reset

## Specification

- Operating voltage: 3.3V to 5V
- Color: white pixels
- Number of pixels: 128 columns x 32 rows
- Interface: I2C
- Drive chip: SSD1306
- Brightness: 180 (Typ) cd/m2
- Full-screen lighting power: approx. 25mA (3.3V) 25mA (5V); Full-screen off power consumption: approx. 2mA (3.3V) 2mA (5V)
- Operating temperature: -30°C to 70°C
- Display area: 22.38 x 5.58 mm / 0.88″x 0.22″
- Mounting hole diameter: 2 mm
- Size: 41x12 mm / 1.61″x 0.47″
- Weight: 2.2 g

## Board Overview




| Num | Label | Description |
|---|---|---|
| 1 | SDA | IIC System Data Line |
| 2 | SCL | IIC System Clock Line |
| 3 | GND | - |
| 4 | VCC | + |

# Tutorial

> **NOTE**:
>
> - The IIC address is 0x3C
> - It is recommended to use Arduino 1.8.9 and above.
> - When welding, please try to keep the pin or line on the pad at arm's length from the mounting hole.

## Requirements

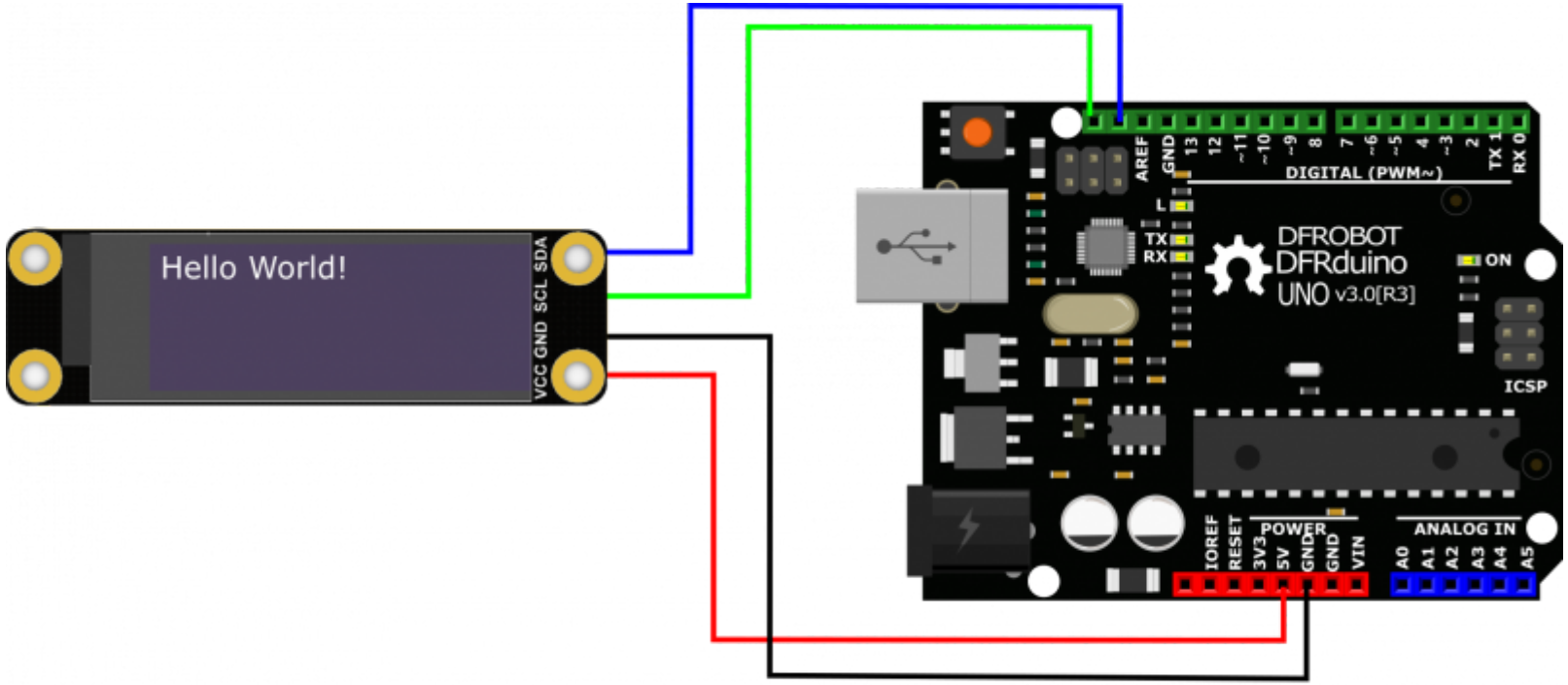
- **Hardware**
  - DFRduino UNO R3 (https://www.dfrobot.com/product-838.html) (or similar) x 1
  - 0.91" 128x32 OLED-A x 1
  - M-M/F-M/F-F Jumper wires
- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)
  - Download and install the **0.91" 128x32 OLED-A u8g2 Library** (https://github.com/DFRobot/U8g2_Arduino/archive/master.zip) (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))
  - Download and install the [**0.91" 128x32 OLED-A DFRobot_GDL Library**](Not released yet)

1.u8g2 API Functions (https://github.com/olikraus/u8g2/wiki/u8g2reference#updatedisplayarea=%E5%BA%93%E6%96%87%E4%BB%B6)

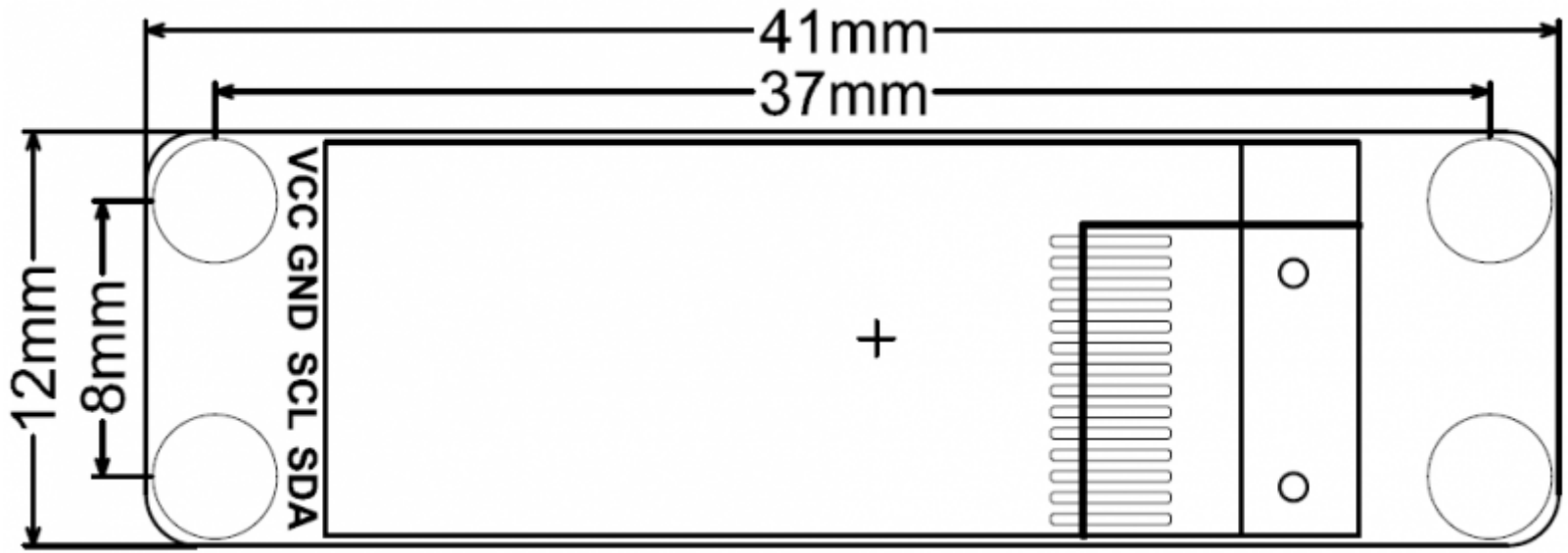
2.u8g2 Font Parameters (https://github.com/olikraus/u8g2/wiki/fntlistall)

3.DFRobot_GDL API Functions (https://www.dfrobot.com/)

## Connection Diagram



## Dimension




## Sample Code1

1. U8G2 supports multi-languages, such as Chinese, English, Japanese, Korean and so on.
2. U8G2 supports multiple fonts: fonts and icons of different pixels, so that users can adjust the fonts and icons as they want.

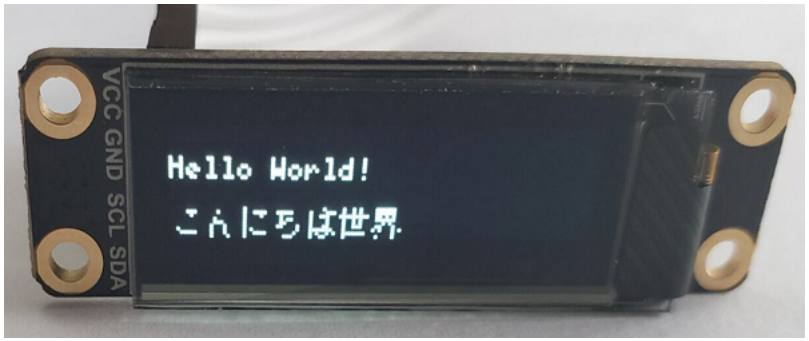
Please go to u8g2wiki for more fonts. There are many good-looking icons, multi-language words, etc.

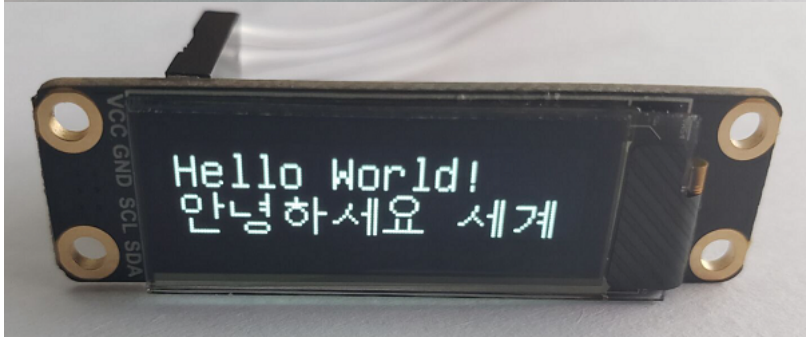
```
/*!
 * @file Language.ino
 * @brief Display multiple languages in U8G2
 * @n A demo for displaying “hello world! ” in Chinese, English, Korean, Japanese
 * @n U8G2 Font GitHub Address: https://github.com/olikraus/u8g2/wiki/fntlistall
 *
 * @copyright  Copyright (c) 2010 DFRobot Co.Ltd (https://www.dfrobot.com)
 * @licence     The MIT License (MIT)
 * @author [Ajax](Ajax.zhong@dfrobot.com)
 * @version  V1.0
 * @date  2019-11-29
 * @get from https://www.dfrobot.com
 * @url https://github.com/DFRobot/U8g2_Arduino
*/
#include <Arduino.h>
#include <U8g2lib.h>

#include <Wire.h>

/*
 *IIC Constructor
 *@param  rotation:     U8G2_R0 No rotation, horizontal, draw from left to right
                        U8G2_R1 Rotate 90 degrees clockwise, draw from top to  bottom
                        U8G2_R2 Rotate 180 degrees clockwise, draw from right to left
                        U8G2_R3 Rotate 270 degrees clockwise, draw from bottom to top.
                        U8G2_MIRROR Display image content normally (v2.6.x and above)   Note: U8G2_MIRROR needs to be used
 *@param reset: U8x8_PIN_NONE Empty pin, reset pin will not be used.
 *
*/
U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);  //M0/ESP32/ESP8266/mega2560/Uno/Leonardo
```

```
void setup(void) {
  u8g2.begin();    //init
  u8g2.enableUTF8Print();        // Enable UTF8 support for Arduino print () function.
}


void loop(void)
{
  /*
   *The font takes up a lot of memory, so please use it with caution. Get your own Chinese encode for displaying only seve
   *Display by drawXBM or use controller with larger memory
   *Chinese Font: require a controller with larger memory than Leonardo
   *Japanese Font: require a controller with larger memory than UNO
   *Korean Font: Arduino INO files of the current version do not support for displaying Korean, but it can displayed proper
  */
  u8g2.setFont(u8g2_font_unifont_t_chinese2);  //Set all fonts in “你好世界” to chinese2
  //u8g2.setFont(u8g2_font_b10_t_japanese1);  // Japanese 1 includes all fonts in “こんにちは世界”：Lerning level 1-6
  //u8g2.setFont(u8g2_font_unifont_t_korean1);  // Korean 1 includes all fonts in “안녕하세요세계”: Lerning level 1-2

  /*@brief Set font direction  of all strings setFontDirection(uint8_t dir)
   *@param dir=0, rotate 0 degree
                dir=1, rotate 90 degrees
                dir=2, rotate 180 degrees
                dir=3, rotate 270 degrees
   *@param When completed font setting, re-set the cursor position to display normally. Refer to API description for more d
  */
  u8g2.setFontDirection(0);

  /*
   * firstPage Change the current page number position to 0
   * Revise content in firstPage and nextPage, re-render everything every time
   * This method consumes less ram space than sendBuffer
  */

  u8g2.firstPage();
  do {
```

```
    u8g2.setCursor(/* x=*/0, /* y=*/15);    //Define the cursor of print function, any output of the print function will s
    u8g2.print("Hello World!");
    u8g2.setCursor(0, 30);
    u8g2.print("你好世界");         // Chinese "Hello World"
    //u8g2.print("こんにちは世界");         // Japanese "Hello World"
    //u8g2.print("안녕하세요 세계");   // Korean "Hello World"
  } while ( u8g2.nextPage() );
  delay(1000);
}
```

## Expected Result1


## Sample Code2

U8G2 supports a variety of graphics, such as rectangles, circles, ellipses, triangles, etc.

```
/*!

  GraphicsTest.ino

  Universal 8bit Graphics Library (https://github.com/olikraus/u8g2/)

  Copyright (c) 2016, olikraus@gmail.com
  All rights reserved.

  Modify by [Ajax](Ajax.zhong@dfrobot.com)
  url https://github.com/DFRobot/U8g2_Arduino
  2019-11-29

*/
#include <Arduino.h>
#include <U8g2lib.h>

#include <Wire.h>


/*
  U8g2lib Example Overview:
    Frame Buffer Examples: clearBuffer/sendBuffer. Fast, but may not work with all Arduino boards because of RAM consumptio
    Page Buffer Examples: firstPage/nextPage. Less RAM usage, should work with all Arduino boards.
    U8x8 Text Only Example: No RAM usage, direct communication with display controller. No graphics, 8x8 Text only.

*/

// Please UNCOMMENT one of the contructor lines below
// U8g2 Contructor List (Frame Buffer)
// The complete list is available here: https://github.com/olikraus/u8g2/wiki/u8g2setupcpp
```

```
// The complete list is available here: https://github.com/olikraus/u8g2/wiki/u8g2setupcpp
// Please update the pin numbers according to your setup. Use U8X8_PIN_NONE if the reset pin is not connected

U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE); //  M0/ESP32/ESP8266/mega2560/Uno/Leonard


void u8g2_prepare(void) {
  u8g2.setFont(u8g2_font_6x10_tf);    //Set the font to "u8g2_font_6x10_tf"
  u8g2.setFontRefHeightExtendedText();//Ascent will be the largest ascent of "A", "1" or "(" of the current font. Descent
  u8g2.setDrawColor(1);        //Defines the bit value (color index) for all drawing functions. All drawing functions will
  u8g2.setFontPosTop();    /*When you use drawStr to display strings, the default criteria is to display the lower-left co
                         XXXX  */
  u8g2.setFontDirection(0);    //Set the screen orientation: 0 -- for normal display
}

/*
 * Draw a border theme
*/
void u8g2_box_title(uint8_t a) {
  u8g2.drawStr( 10+a*2, 5, "U8g2");//Draw string "U8g2"
  u8g2.drawStr( 10, 20, "GraphicsTest");
  u8g2.drawFrame(0,0,u8g2.getDisplayWidth(),u8g2.getDisplayHeight() );//Start drawing an empty box of width w and height h
}

/*
 * Draw solid squares and hollow squares
*/
void u8g2_box_frame(uint8_t a) {
  u8g2.drawStr(0,0,"drawBox-drawFrame");
  u8g2.drawBox(5+a,9,50,10); //Start drawing a solid square with a width w and a height h at a position with coordinates o
  u8g2.drawFrame(5+a,20,50,10);//Start drawing a hollow square with a width w and a height h at a position with coordinate
}

/*
 * Draw solid and hollow circles
*/
void u8g2_circle_disc(uint8_t a){
```

```
  u8g2.drawStr(0,0,"drawDisc-drawCircle");
  u8g2.drawCircle(10+a,20,10);  //Draw a solid circle with a radius of 10 at the position (10+ a, 20)
  u8g2.drawDisc(118-a,20,10);   //Draw a hollow circle with a radius of 10 at the position (118-a, 20)
}


/*
 * Draw solid and hollow boxes
*/
void u8g2_RBox_RFrame(uint8_t a){
  u8g2.drawStr(0,0,"drawRBox-drawRFrame");
  u8g2.drawRBox(5+a,9,30,20,a+1);   //At the position (5 + a, 9) start drawing with a width of 40 and a height of 30 a fra
  u8g2.drawRFrame(90-a,9,30,20,a+1);  //At the position (90-a,9) start drawing with a width of 25 and a height of 40 a fra
}

/*
 * Draw rays
*/
void u8g2_Hline(uint8_t a){
  u8g2.drawStr(0,0,"drawHLine");
  u8g2.drawHLine(1,10,40+a*5); //Draw a ray at coordinates (1,10)
  u8g2.drawHLine(10,20,40+a*5);
  u8g2.drawHLine(20,30,40+a*5);
}

/*
 * Draw segments
*/
void u8g2_line(uint8_t a){
  u8g2.drawStr(0,0,"drawLine");
  u8g2.drawLine(10+a,10,80-a,32); //Draw a line between two points. (Argument is two-point coordinates)
  u8g2.drawLine(10+a*2,10,80-a*2,32);
  u8g2.drawLine(10+a*3,10,80-a*3,32);
  u8g2.drawLine(10+a*4,10,80-a*4,32);
}

/*
```

```
 * Draw solid triangles and hollow triangles
*/
void u8g2_triangle(uint8_t a) {
  uint16_t offset = a;
  u8g2.drawStr( 0, 0, "drawTriangle");

  u8g2.drawTriangle(14,7, 45,30, 10,40);  //Draw a triangle (solid polygon). (Argument is triangle three vertex coordinate
  u8g2.drawTriangle(14+offset,7-offset, 45+offset,30-offset, 57+offset,10-offset);
  u8g2.drawTriangle(57+offset*2,10, 45+offset*2,30, 86+offset*2,53);
  u8g2.drawTriangle(10+offset,40+offset, 45+offset,30+offset, 86+offset,53+offset);
}

/*
 * Show characters in the ASCII code table
*/
void u8g2_ascii_1() {
  char s[2] = " ";
  uint8_t x, y;
  u8g2.drawStr( 0, 0, "ASCII page 1");
  for( y = 0; y < 2; y++ ) {
    for( x = 0; x < 16; x++ ) {
      s[0] = y*16 + x + 32;
      u8g2.drawStr(x*7, y*10+10, s);
    }
  }
}

void u8g2_ascii_2() {
  char s[2] = " ";
  uint8_t x, y;
  u8g2.drawStr( 0, 0, "ASCII page 2");
  for( y = 0; y < 2; y++ ) {
    for( x = 0; x < 16; x++ ) {
      s[0] = y*16 + x + 160;
      u8g2.drawStr(x*7, y*10+10, s);
    }
  }
}
```

```
}

/*
 * Draw a string icon in a UTF-8 encoding
*/
void u8g2_extra_page(uint8_t a)
{
  u8g2.drawStr( 0, 0, "Unicode");
  u8g2.setFont(u8g2_font_unifont_t_symbols);
  u8g2.setFontPosTop();
  u8g2.drawUTF8(0, 9, "☀ ☁");//Start drawing a string icon encoded as UTF-8 at the location (0,24)
  switch(a) {
    case 0:
    case 1:
    case 2:
    case 3:
      u8g2.drawUTF8(a*3, 20, "☂");
      break;
    case 4:
    case 5:
    case 6:
    case 7:
      u8g2.drawUTF8(a*3, 20, "☔");
      break;
  }
}

/*
 * Show the reverse display of font. Which means the font is displayed interchangeable with
 * the background color. (For example, it would have been black on white, replaced by white
 * on black)
*/
void u8g2_xor(uint8_t a) {
  uint8_t i;
  u8g2.drawStr( 0, 0, "XOR");
  u8g2.setFontMode(1);
```

```
  u8g2.setDrawColor(2);
  for( i = 0; i < 5; i++)
  {
    u8g2.drawBox(10+i*16, 18, 21,10);
  }
  u8g2.drawStr( 5+a, 19, "XOR XOR XOR XOR");
}

/*
 *This is the data of converting a graph to bitmap for using the drawXBMP () to display.
*/
#define cross_width 24
#define cross_height 24
static const unsigned char cross_bits[] U8X8_PROGMEM  = {
  0x00, 0x18, 0x00, 0x00, 0x24, 0x00, 0x00, 0x24, 0x00, 0x00, 0x42, 0x00,
  0x00, 0x42, 0x00, 0x00, 0x42, 0x00, 0x00, 0x81, 0x00, 0x00, 0x81, 0x00,
  0xC0, 0x00, 0x03, 0x38, 0x3C, 0x1C, 0x06, 0x42, 0x60, 0x01, 0x42, 0x80,
  0x01, 0x42, 0x80, 0x06, 0x42, 0x60, 0x38, 0x3C, 0x1C, 0xC0, 0x00, 0x03,
  0x00, 0x81, 0x00, 0x00, 0x81, 0x00, 0x00, 0x42, 0x00, 0x00, 0x42, 0x00,
  0x00, 0x42, 0x00, 0x00, 0x24, 0x00, 0x00, 0x24, 0x00, 0x00, 0x18, 0x00, };

#define cross_fill_width 24
#define cross_fill_height 24
static const unsigned char cross_fill_bits[] U8X8_PROGMEM  = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x18, 0x64, 0x00, 0x26,
  0x84, 0x00, 0x21, 0x08, 0x81, 0x10, 0x08, 0x42, 0x10, 0x10, 0x3C, 0x08,
  0x20, 0x00, 0x04, 0x40, 0x00, 0x02, 0x80, 0x00, 0x01, 0x80, 0x18, 0x01,
  0x80, 0x18, 0x01, 0x80, 0x00, 0x01, 0x40, 0x00, 0x02, 0x20, 0x00, 0x04,
  0x10, 0x3C, 0x08, 0x08, 0x42, 0x10, 0x08, 0x81, 0x10, 0x84, 0x00, 0x21,
  0x64, 0x00, 0x26, 0x18, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, };

#define cross_block_width 14
#define cross_block_height 14
static const unsigned char cross_block_bits[] U8X8_PROGMEM  = {
  0xFF, 0x3F, 0x01, 0x20, 0x01, 0x20, 0x01, 0x20, 0x01, 0x20, 0x01, 0x20,
  0xC1, 0x20, 0xC1, 0x20, 0x01, 0x20, 0x01, 0x20, 0x01, 0x20, 0x01, 0x20,
```

```
  0x01, 0x20, 0xFF, 0x3F, };

/*
 * Draw a bitmap
*/
void u8g2_bitmap_overlay(uint8_t a) {
  uint8_t frame_size = 28;
  u8g2.drawStr(0, 0, "Bitmap overlay");
  u8g2.drawStr(0, frame_size + 12, "Solid / transparent");
/*
 * Set the pattern of the bitmap to define whether the background color is written to the bitmap function
 * (Mode0/solid, is_transparent = 0).
 * Or not write the background color to the bitmap function. (Mode1/solid, is_transparent = 1).
 * The default mode is 0(fixed mode).
*/
  u8g2.setBitmapMode(false /* solid */);
  u8g2.drawFrame(0, 10, frame_size, frame_size);
/*
 * The position (x,y) is the upper-left corner of the bitmap. XBM contains a monochrome 1-bit bitmap.
 * The current color index is used for drawing (refers to setColorIndex) pixel value1.
*/
  u8g2.drawXBMP(2, 12, cross_width, cross_height, cross_bits);
  if(a & 4)
    u8g2.drawXBMP(7, 17, cross_block_width, cross_block_height, cross_block_bits);//Draw a bitmap

  u8g2.setBitmapMode(true /* transparent*/);
  u8g2.drawFrame(frame_size + 5, 10, frame_size, frame_size);
  u8g2.drawXBMP(frame_size + 7, 12, cross_width, cross_height, cross_bits);
  if(a & 4)
    u8g2.drawXBMP(frame_size + 12, 17, cross_block_width, cross_block_height, cross_block_bits);
}

//Define the initial variable for the drawing state
uint8_t draw_state = 0;

/*
```

```
 * Draw functions: call other functions in an orderly manner.
*/
void draw(void) {
  u8g2_prepare();
  switch(draw_state >> 3) {
    case 0: u8g2_box_title(draw_state&7); break;
    case 1: u8g2_box_frame(draw_state&7); break;
    case 2: u8g2_circle_disc(draw_state&7); break;
    case 3: u8g2_RBox_RFrame(draw_state&7); break;
    case 4: u8g2_Hline(draw_state&7); break;
    case 5: u8g2_line(draw_state&7); break;
    case 6: u8g2_triangle(draw_state&7); break;
    case 7: u8g2_ascii_1(); break;
    case 8: u8g2_ascii_2(); break;
    case 9: u8g2_extra_page(draw_state&7); break;
    case 10: u8g2_xor(draw_state&7); break;
    case 11: u8g2_bitmap_overlay(draw_state&7); break;
  }
}

void setup(void) {
  u8g2.begin(); //Initialize the function
}

void loop(void) {
  //Picture loop
  u8g2.firstPage();
  do {
    draw();
  } while( u8g2.nextPage() );

  //Increase in state variables
  draw_state++;
  if ( draw_state >= 14*8 )
    draw_state = 0;

  delay(150);
```

```
  delay(150);
}
```

## Expected Result2


# Compatibility Test

| MCU | Pass | Failed | Not Tested | Remark |
|---|---|---|---|---|
| FireBeetle-ESP32 | √ | | | |
| FireBeetle-ESP8266 | √ | | | |
| Arduino Uno | √ | | | |
| Leonardo | √ | | | |
| Mega2560 | √ | | | |
| Arduino M0 | √ | | | |

## FAQ

| Q&A | Some general Arduino Problems/FAQ/Tips |
| --- | --- |
| Q | There are two drives u8g2 and DFRobot_GDL are provided, should I install both of them? |
| A | You just need to select one to install. Please note that u8g2 is a powerful lcd library for black and white liquid crystals, and thre are functions like built-in capacitive resistance touch, simple UI and u8g2 partial functions in DFRobot_SDL, it supports both black and white, and colorful LCD. |
| A | For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (https://www.dfrobot.com/forum/). |

## More Documents

- [DFR0647]Schematic (https://dfimg.dfrobot.com/nobody/wiki/7f5b3755d782059958ddd25e08e319b3.pdf)
- [DFR0647]Dimension (https://dfimg.dfrobot.com/nobody/wiki/c49387ef45a934f2fff400025cea2b19.png)
- [DFR0647]Datasheet (https://dfimg.dfrobot.com/nobody/wiki/8d5cd6e028ddc95e45b4966046b7a331.pdf)

Get **0.91" 128x32 OLED-A** (https://www.dfrobot.com/product-2019.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/distributor)

**Turn to the Top**