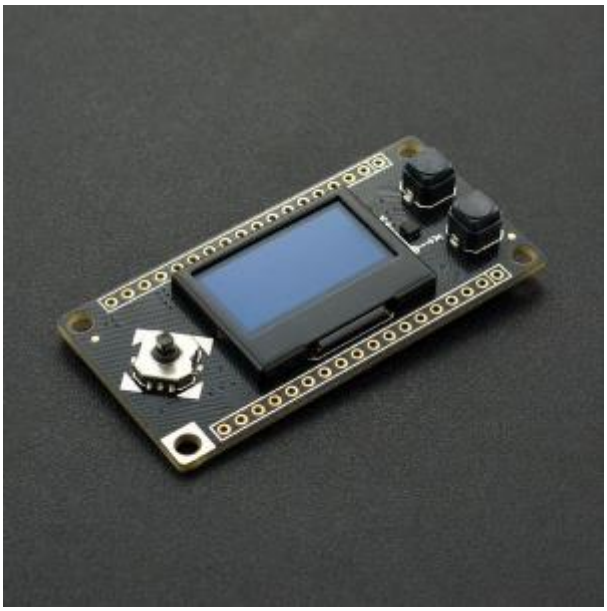
# FireBeetle Covers-OLED12864 Display SKU: DFR0507



FireBeetle Covers-OLED12864 Display

## Contents

# Introduction

DFRobot FireBeetle firefly series is low-power development component designed for the IoT. This FireBeetle covers display module is equipped with 128x64 resolution OLED and new version of SSD1360 driver, uses I2C interface, supports for the Arduino library and microPython, fully compatible with the Gravity I2C OLED-2864 screen. OLED screen has a protective frame to protect the screen, it prevents the finger from being scratched on the edge of the glass. The OLED 12864 display module also integrates the GT30L24A3W Chinese / foreign font library chip and the BMA220 three-axis accelerometer. In addition, the FireBeetle Covers-OLED12864 display features an analog directional key and two independent digital keys A and B.

# Specification

- Voltage Range: 3.7V-5.5V
- 2 User Buttons: Using digital port D4, D8 detection mode
- 5-way Switch: Using analog port A0 detection mode
- Three- axis Accelerometer BMA220: IIC
- Data Bus: IIC
- Dimension: 0.079x0.079x0.035in/2x2x0.9mm
- Resolution: 6 bit
- Range: 2/4/8/16g
- Power Consumption: 250uA
- OLED
- Model: UG-2864HLBEG01
- Dimension: 0.96in
- Color: blue
- Data Bus: IIC
- Pixel:128x64
- Font Chip: GT30L24A3W
- Data Bus: SPI
- Character Set:
- GB18030 Simplified Chinese/Traditional Chinese
- KSC5601 Korean
- JIS0208 Japanese
- 180 Foreign Font
- Support for Multinational Unicode
- IS08859 and CODE PAGE
- Chinese Character Size:12 dot matrix, 16dot matrix, 24dot matrix
- Foreign Character Size:16 dot matrix, 24 dot matrix
- Working Current: 12mA

# Function Diagram

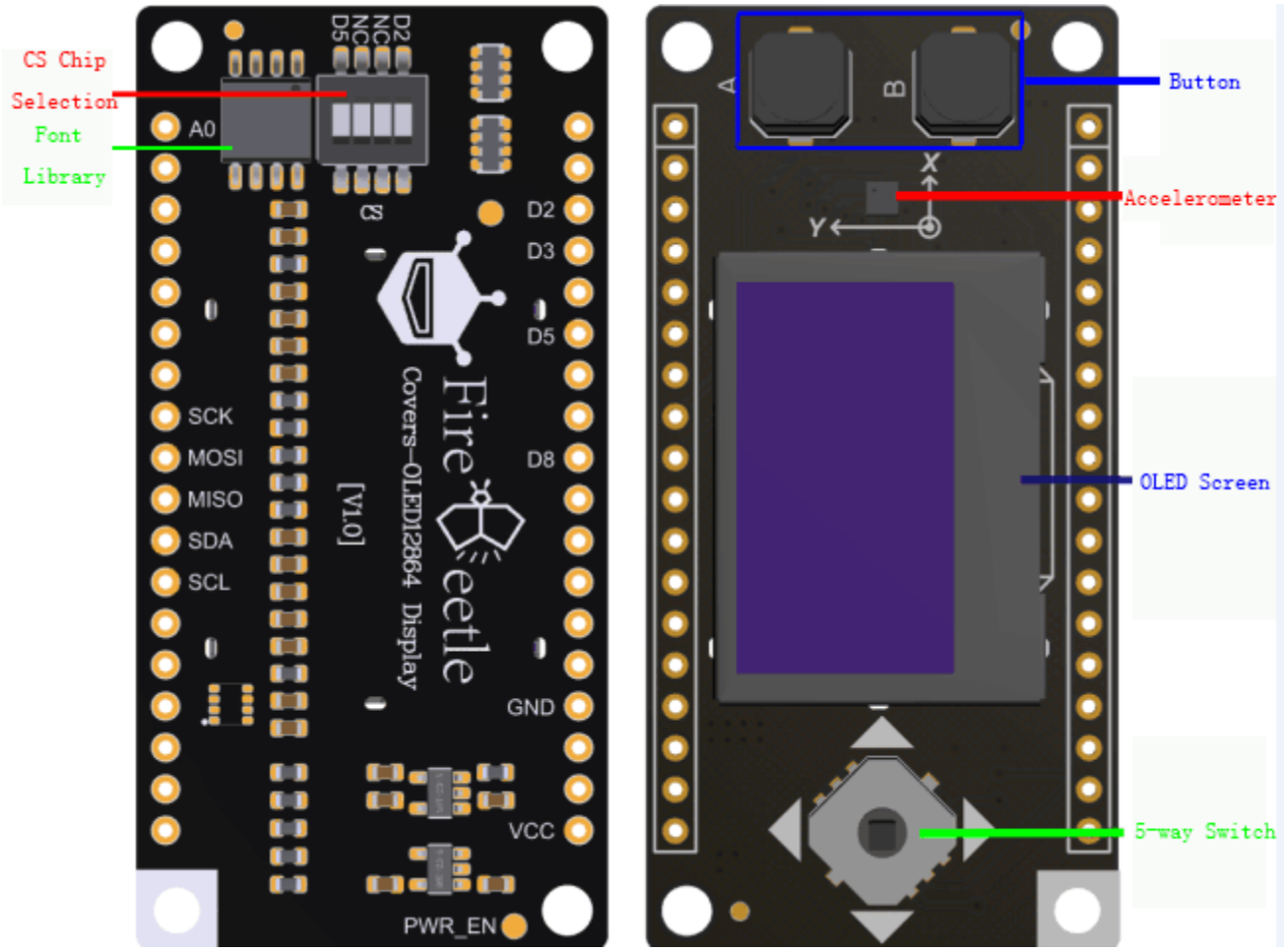



Fig1: FireBeetle Covers-OLED12864 Display Functional Module

- A<->D4
- B<->D8
- 5-way Switch<->A0

# Pin Out

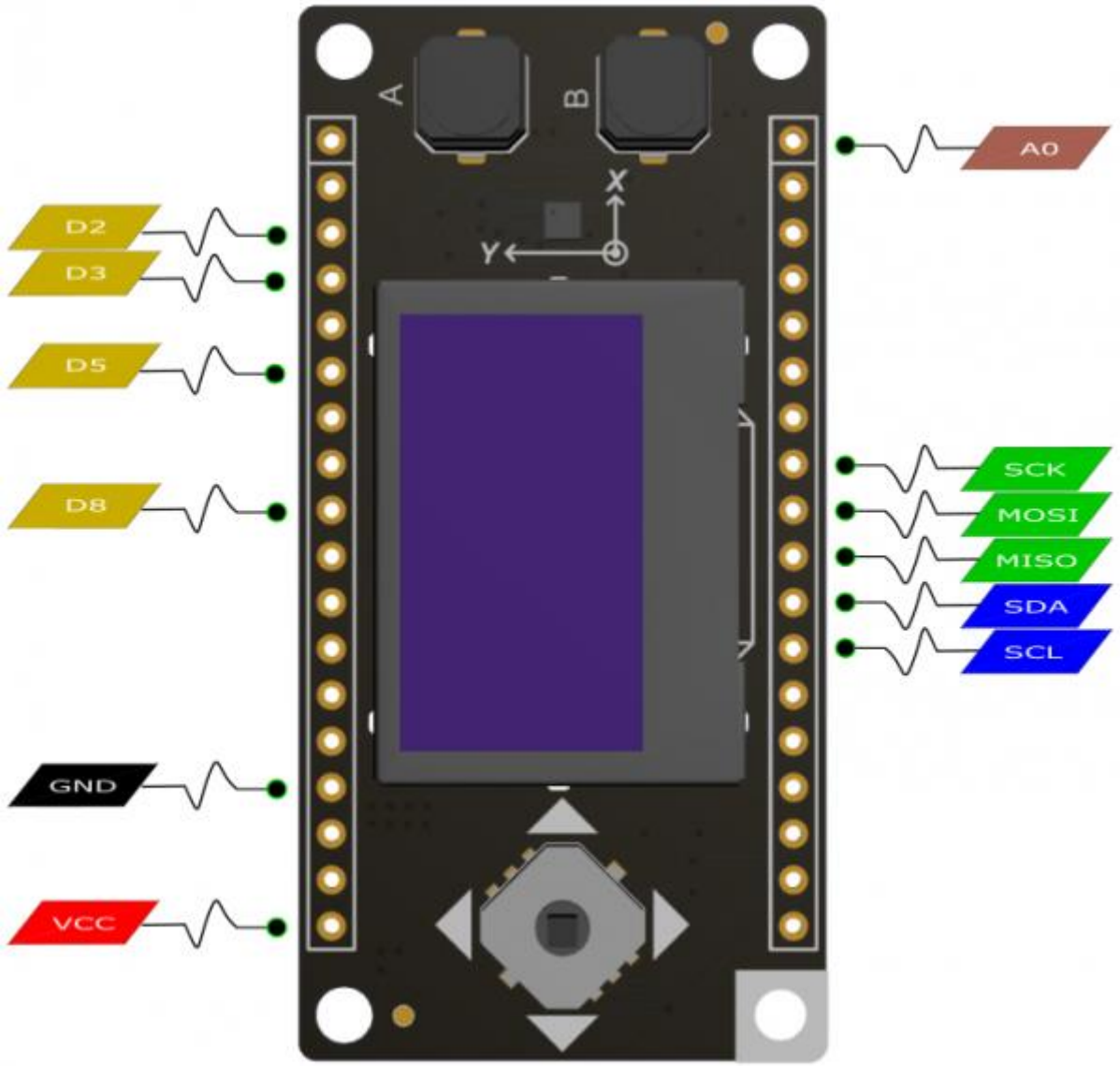



Fig2: FireBeetle Covers-OLED12864 Display Pinout

NOTE: NC do not need to connect, and the VCC is the power supply voltage output.

# Tutorial

## Preparation

- Hardware
- 1 x ESP32
- 1 x FireBeetle Covers-OLED12864
- Software
- Arduino IDE 1.8+
- Please download FireBeetle Covers-OLED12864 Display library first.
- Do you need to install any additional libraries? Explain it here How to install Libraries in Arduino IDE

NOTE: All of the following examples are belong to DFRobot_OLED12864 library files

## Image

- Open Dfrobot_OLED12864 Image Demo

```
void setup() {
 #include "DFRobot_OLED12864.h"

// Include custom images
#include "images.h"

// Initialize the OLED display using Wire library
DFRobot_OLED12864  display(0x3c);

void setup()
{
        Serial.begin(115200);
        Serial.println();
        Serial.println();
        // Initialising the UI will init the display too.
```

```
        display.init();
        display.flipScreenVertically();// flip vertical
        display.clear();
        drawImageDemo();
        display.display();
}

void drawImageDemo()
{
  display.drawXbm(0, 0, Picture_width, Picture_height, Picture_bits);
}

void loop()
{
}
```

- Function: After download this demo the screen will display our logo (notice that the picture file "images.h" has been in the project folder, if you need to replace the picture, you can use The Dot Factory to generate bitmaps)
- Function Declaration:
- Create an object and write the I2C address

```
DFRobot _OLED12864  display(0x3c)
```

- Initialize OLED and library

```
init()
```

- Flip screen vertically

```
flipScreenVertically
```

- Clear data

```
clear()
```

- Import the specified width-high data at the x, y-axis position, starting at the top left corner.

```
drawXbm(0, 0, Picture_width, Picture_height, Picture_bits)
```

- Flush the data from OLED to the screen. If not called, the data will only be stored in the OLED and will not be displayed.

```
display()
```

## Paint

- Open DFRobot_OLED12864 Drawing Demo

```
#include "DFRobot_OLED12864.h"

// Initialize the OLED display using Wire library
DFRobot_OLED12864  display(0x3c);

void drawLines()
{
  for (int16_t i=0; i<DISPLAY_WIDTH; i+=4) {
    display.drawLine(0, 0, i, DISPLAY_HEIGHT-1);
    display.display();
    delay(10);
  }
  for (int16_t i=0; i<DISPLAY_HEIGHT; i+=4) {
    display.drawLine(0, 0, DISPLAY_WIDTH-1, i);
    display.display();
    delay(10);
  }
```

```
  delay(250);

  display.clear();
  for (int16_t i=0; i<DISPLAY_WIDTH; i+=4) {
    display.drawLine(0, DISPLAY_HEIGHT-1, i, 0);
    display.display();
    delay(10);
  }
  for (int16_t i=DISPLAY_HEIGHT-1; i>=0; i-=4) {
    display.drawLine(0, DISPLAY_HEIGHT-1, DISPLAY_WIDTH-1, i);
    display.display();
    delay(10);
  }
  delay(250);

  display.clear();
  for (int16_t i=DISPLAY_WIDTH-1; i>=0; i-=4) {
    display.drawLine(DISPLAY_WIDTH-1, DISPLAY_HEIGHT-1, i, 0);
    display.display();
    delay(10);
  }
  for (int16_t i=DISPLAY_HEIGHT-1; i>=0; i-=4) {
    display.drawLine(DISPLAY_WIDTH-1, DISPLAY_HEIGHT-1, 0, i);
    display.display();
    delay(10);
  }
  delay(250);
  display.clear();
  for (int16_t i=0; i<DISPLAY_HEIGHT; i+=4) {
    display.drawLine(DISPLAY_WIDTH-1, 0, 0, i);
    display.display();
    delay(10);
  }
  for (int16_t i=0; i<DISPLAY_WIDTH; i+=4) {
```

```
    display.drawLine(DISPLAY_WIDTH-1, 0, i, DISPLAY_HEIGHT-1);
    display.display();
    delay(10);
  }
  delay(250);
}

void drawRect(void)
{
  for (int16_t i=0; i<DISPLAY_HEIGHT/2; i+=2) {
    display.drawRect(i, i, DISPLAY_WIDTH-2*i, DISPLAY_HEIGHT-2*i);
    display.display();
    delay(10);
  }
}

void fillRect(void)
{
  uint8_t color = 1;
  for (int16_t i=0; i<DISPLAY_HEIGHT/2; i+=3) {
    display.setColor((color % 2 == 0) ? BLACK : WHITE); // alternate colors
    display.fillRect(i, i, DISPLAY_WIDTH - i*2, DISPLAY_HEIGHT - i*2);
    display.display();
    delay(10);
    color++;
  }
  // Reset back to WHITE
  display.setColor(WHITE);
}

void drawCircle(void)
{
  for (int16_t i=0; i<DISPLAY_HEIGHT; i+=2) {
    display.drawCircle(DISPLAY_WIDTH/2, DISPLAY_HEIGHT/2, i);
```

```
    display.display();
    delay(10);
  }
  delay(1000);
  display.clear();

  // This will draw the part of the circel in quadrant 1
  // Quadrants are numberd like this:
  //   0010 | 0001
  //  ------|-----
  //   0100 | 1000
  //
  display.drawCircleQuads(DISPLAY_WIDTH/2, DISPLAY_HEIGHT/2, DISPLAY_HEIGHT/4
, 0b00000001);
  display.display();
  delay(200);
  display.drawCircleQuads(DISPLAY_WIDTH/2, DISPLAY_HEIGHT/2, DISPLAY_HEIGHT/4
, 0b00000011);
  display.display();
  delay(200);
  display.drawCircleQuads(DISPLAY_WIDTH/2, DISPLAY_HEIGHT/2, DISPLAY_HEIGHT/4
, 0b00000111);
  display.display();
  delay(200);
  display.drawCircleQuads(DISPLAY_WIDTH/2, DISPLAY_HEIGHT/2, DISPLAY_HEIGHT/4
, 0b00001111);
  display.display();
}

void printBuffer(void)
{
  // Initialize the log buffer
  // allocate memory to store 8 lines of text and 30 chars per line.
  display.setLogBuffer(5, 30);

  // Some test data
```

```
  const char* test[] = {
      "Hello",
      "World" ,
      "----",
      "Show off",
      "how",
      "the log buffer",
      "is",
      "working.",
      "Even",
      "scrolling is",
      "working"
  };

  for (uint8_t i = 0; i < 11; i++) {
    display.clear();
    // Print to the screen
    display.println(test[i]);
    // Draw it to the internal screen buffer
    display.drawLogBuffer(0, 0);
    // Display it on the screen
    display.display();
    delay(500);
  }
}

void setup()
{
  display.init();

  // display.flipScreenVertically();

  display.setContrast(255);
```

```
  drawLines();
  delay(1000);
  display.clear();

  drawRect();
  delay(1000);
  display.clear();

  fillRect();
  delay(1000);
  display.clear();

  drawCircle();
  delay(1000);
  display.clear();

  printBuffer();
  delay(1000);
  display.clear();
}

void loop() { }
```

- The program will do some acts of drawing, display some characters at the end, and then stop
- Function Declaration:
- Set contrast

```
 setContrast(contrast)
```

- Draw lines

```
 drawLines()
```

- Draw rectangle

```
drawRect()
```

- Draw circle

```
drawCircle()
```

- Print character

```
printBuffer()
```

# Clock

- Open DFRobot_OLED12864 Clock Demo

```
#include <TimeLib.h>
#include "DFRobot_OLED12864.h" // alias for `#include "DFRobot_OLED12864Wire.h"`

// Include the UI lib
#include "OLEDDisplayUi.h"

// Include custom images
#include "images.h"

DFRobot_OLED12864  display(0x3c);

OLEDDisplayUi ui ( &display );

int screenW = 128;
int screenH = 64;
int clockCenterX = screenW/2;
int clockCenterY = ((screenH-16)/2)+16;   // top yellow part is 16 px height
```

```
int clockRadius = 23;

// utility function for digital clock display: prints leading 0
String twoDigits(int digits)
{
  if(digits < 10) {
    String i = '0'+String(digits);
    return i;
  }
  else {
    return String(digits);
  }
}

void clockOverlay(OLEDDisplay *display, OLEDDisplayUiState* state)
{

}

void analogClockFrame(OLEDDisplay *display, OLEDDisplayUiState* state, int16_
t x, int16_t y)
{
        //  ui.disableIndicator();

        // Draw the clock face
        //  display->drawCircle(clockCenterX + x, clockCenterY + y, clockRadiu
s);
        display->drawCircle(clockCenterX + x, clockCenterY + y, 2);
        //
        //hour ticks
        for( int z=0; z < 360;z= z + 30 ){
        //Begin at 0° and stop at 360°
                float angle = z ;
                angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
                int x2 = ( clockCenterX + ( sin(angle) * clockRadius ) );
```

```
                int y2 = ( clockCenterY - ( cos(angle) * clockRadius ) );
                int x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clo
ckRadius / 8 ) ) ) );
                int y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clo
ckRadius / 8 ) ) ) );
                display->drawLine( x2 + x , y2 + y , x3 + x , y3 + y);
        }

        // display second hand
        float angle = second() * 6 ;
        angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
        int x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius
/ 5 ) ) ) );
        int y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius
/ 5 ) ) ) );
        display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 +
y);
        //
        // display minute hand
        angle = minute() * 6 ;
        angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
        x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 4
) ) ) );
        y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 4
) ) ) );
        display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 +
y);
        //
        // display hour hand
        angle = hour() * 30 + int( ( minute() / 12 ) * 6 )   ;
        angle = ( angle / 57.29577951 ) ; //Convert degrees to radians
        x3 = ( clockCenterX + ( sin(angle) * ( clockRadius - ( clockRadius / 2
) ) ) );
        y3 = ( clockCenterY - ( cos(angle) * ( clockRadius - ( clockRadius / 2
) ) ) );
        display->drawLine( clockCenterX + x , clockCenterY + y , x3 + x , y3 +
y);
}
```

```
void digitalClockFrame(OLEDDisplay *display, OLEDDisplayUiState* state, int16
_t x, int16_t y)
{
  String timenow = String(hour())+":"+twoDigits(minute())+":"+twoDigits(secon
d());
  display->setTextAlignment(TEXT_ALIGN_CENTER);
  display->setFont(ArialMT_Plain_24);
  display->drawString(clockCenterX + x , clockCenterY + y, timenow );
}

// This array keeps function pointers to all frames
// frames are the single views that slide in
FrameCallback frames[] = { analogClockFrame, digitalClockFrame };

// how many frames are there?
int frameCount = 2;

// Overlays are statically drawn on top of a frame eg. a clock
OverlayCallback overlays[] = { clockOverlay };
int overlaysCount = 1;

void setup()
{
  Serial.begin(115200);
  Serial.println();

        // The ESP is capable of rendering 60fps in 80Mhz mode
        // but that won't give you much time for anything else
        // run it in 160Mhz mode or just set it to 30 fps
  ui.setTargetFPS(60);

        // Customize the active and inactive symbol
  ui.setActiveSymbol(activeSymbol);
  ui.setInactiveSymbol(inactiveSymbol);
```

```
  // You can change this to
  // TOP, LEFT, BOTTOM, RIGHT
  ui.setIndicatorPosition(TOP);

  // Defines where the first frame is located in the bar.
  ui.setIndicatorDirection(LEFT_RIGHT);

  // You can change the transition that is used
  // SLIDE_LEFT, SLIDE_RIGHT, SLIDE_UP, SLIDE_DOWN
  ui.setFrameAnimation(SLIDE_LEFT);

  // Add frames
  ui.setFrames(frames, frameCount);

  // Add overlays
  ui.setOverlays(overlays, overlaysCount);

  // Initialising the UI will init the display too.
  ui.init();

  display.flipScreenVertically();

  unsigned long secsSinceStart = millis();
  // Unix time starts on Jan 1 1970. In seconds, that's 2208988800:
  const unsigned long seventyYears = 2208988800UL;
  // subtract seventy years:
  unsigned long epoch = secsSinceStart - seventyYears * SECS_PER_HOUR;
  setTime(epoch);
}


void loop()
{
  int remainingTimeBudget = ui.update();
```

```
  if (remainingTimeBudget > 0) {
    // You can do some work here
    // Don't do stuff if you are below your
    // time budget.
    delay(remainingTimeBudget);

  }
}
```

- Function:Switch between an analog dial frame and a digital clock frame every 5 seconds.
- Function Declaration:
- Create an UI object and specify its OLED object

```
OLEDDisplayUi ui ( &display );
```

- Set FPS

```
setTargetFPS(fps)
```

- Set active screen identification

```
setActiveSymbol(activeSymbol)
```

- Set inactive screen identification

```
setInactiveSymbol(inactiveSymbol)
```

- Set the position of the indicator

```
setIndicatorPosition(pos)
```

- Set the direction of indicator

```
setIndicatorDirection(direction)
```

- Set the slide direction of the frame

```
setFrameAnimation(direction)
```

- Set frames

```
setFrames(frames, count)
```

- Set overlays

```
setOverlays(overlays, count)
```

- Set time

```
setTime(time)
```

- Update frame

```
update()
```

- Set the time to automatically Update the Screen(the unit is ms)

```
setTimePerFrame(time)
```

- Allow automatic scrolling

```
enableAutoTransition()
```

- Prohibit automatic scrolling

```
disableAutoTransition()
```

- Set the content of display

```
transitionToFrame(frame)
```

## Progress Bar

```
#include "DFRobot_OLED12864.h" // alias for `#include "DFRobot_OLED12864Wire.h"`



// Initialize the OLED display using Wire library
DFRobot_OLED12864  display(0x3c);

int counter = 1;

void setup()
{
  Serial.begin(115200);
  Serial.println();
  Serial.println();

  // Initialising the UI will init the display too.
  display.init();
  display.flipScreenVertically();

}

void drawProgressBarDemo()
{
  int progress = (counter / 5) % 100;
```

```
  // draw the progress bar
  display.drawProgressBar(0, 32, 120, 10, progress);

  // draw the percentage as String
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.drawString(64, 15, String(progress) + "%");
}

void loop()
{
        // clear the display
        display.clear();
        // draw the current demo method
        drawProgressBarDemo();

        // write the buffer to the display
        display.display();
        counter++;
        delay(10);
}
```

- Function: Display the progress bar on the screen
- Function Declaration:
- Draw progress bar

```
drawProgressBar(x, y, width, height, progress)
```

- Formatting fonts

```
setTextAlignment(alignment)
```

- Draw string in specified location

```
drawString(x, y, string)
```

# UI

- Open DFRobot_OLED12864 UI Demo

```
#include "DFRobot_OLED12864.h"

// Include the UI lib
#include "OLEDDisplayUi.h"

// Include custom images
#include "images.h"

// Initialize the OLED display using Wire library
DFRobot_OLED12864  display(0x3c);

OLEDDisplayUi ui(&display);

void msOverlay(OLEDDisplay *display, OLEDDisplayUiState* state)
{
  display->setTextAlignment(TEXT_ALIGN_RIGHT);
  display->setFont(ArialMT_Plain_10);
  display->drawString(128, 0, String(millis()));
}

void drawFrame1(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, i
nt16_t y)
{
  // draw an xbm image.
  // Please note that everything that should be transitioned
  // needs to be drawn relative to x and y

  display->drawXbm(x + 34, y + 14, WiFi_Logo_width, WiFi_Logo_height, WiFi_Lo
go_bits);
```

```
}


void drawFrame2(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, i
nt16_t y)
{
  // Demonstrates the 3 included default sizes. The fonts come from DFRobot_O
LED12864Fonts.h file
  // Besides the default fonts there will be a program to convert TrueType fo
nts into this format
  display->setTextAlignment(TEXT_ALIGN_LEFT);
  display->setFont(ArialMT_Plain_10);
  display->drawString(0 + x, 10 + y, "Arial 10");

  display->setFont(ArialMT_Plain_16);
  display->drawString(0 + x, 20 + y, "Arial 16");

  display->setFont(ArialMT_Plain_24);
  display->drawString(0 + x, 34 + y, "Arial 24");
}


void drawFrame3(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, i
nt16_t y)
{
  // Text alignment demo
  display->setFont(ArialMT_Plain_10);

  // The coordinates define the left starting point of the text
  display->setTextAlignment(TEXT_ALIGN_LEFT);
  display->drawString(0 + x, 11 + y, "Left aligned (0,10)");

  // The coordinates define the center of the text
  display->setTextAlignment(TEXT_ALIGN_CENTER);
  display->drawString(64 + x, 22 + y, "Center aligned (64,22)");

  // The coordinates define the right end of the text
  display->setTextAlignment(TEXT_ALIGN_RIGHT);
```

```
  display->drawString(128 + x, 33 + y, "Right aligned (128,33)");
}

void drawFrame4(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, int16_t y)
{
  // Demo for drawStringMaxWidth:
  // with the third parameter you can define the width after which words will be wrapped.
  // Currently only spaces and "-" are allowed for wrapping
  display->setTextAlignment(TEXT_ALIGN_LEFT);
  display->setFont(ArialMT_Plain_10);
  display->drawStringMaxWidth(0 + x, 10 + y, 128, "Lorem ipsum\n dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore.");
}

void drawFrame5(OLEDDisplay *display, OLEDDisplayUiState* state, int16_t x, int16_t y)
{


}

// This array keeps function pointers to all frames
// frames are the single views that slide in
FrameCallback frames[] = { drawFrame1, drawFrame2, drawFrame3, drawFrame4, drawFrame5 };

// how many frames are there?
int frameCount = 5;

// Overlays are statically drawn on top of a frame eg. a clock
OverlayCallback overlays[] = { msOverlay };
int overlaysCount = 1;

void setup()
{
```

```
    Serial.begin(115200);
    Serial.println();
    Serial.println();

          // The ESP is capable of rendering 60fps in 80Mhz mode
          // but that won't give you much time for anything else
          // run it in 160Mhz mode or just set it to 30 fps
    ui.setTargetFPS(60);

          // Customize the active and inactive symbol
    ui.setActiveSymbol(activeSymbol);
    ui.setInactiveSymbol(inactiveSymbol);

    // You can change this to
    // TOP, LEFT, BOTTOM, RIGHT
    ui.setIndicatorPosition(BOTTOM);

    // Defines where the first frame is located in the bar.
    ui.setIndicatorDirection(LEFT_RIGHT);

    // You can change the transition that is used
    // SLIDE_LEFT, SLIDE_RIGHT, SLIDE_UP, SLIDE_DOWN
    ui.setFrameAnimation(SLIDE_LEFT);

    // Add frames
    ui.setFrames(frames, frameCount);

    // Add overlays
    ui.setOverlays(overlays, overlaysCount);

    // Initialising the UI will init the display too.
    ui.init();

    display.flipScreenVertically();
```

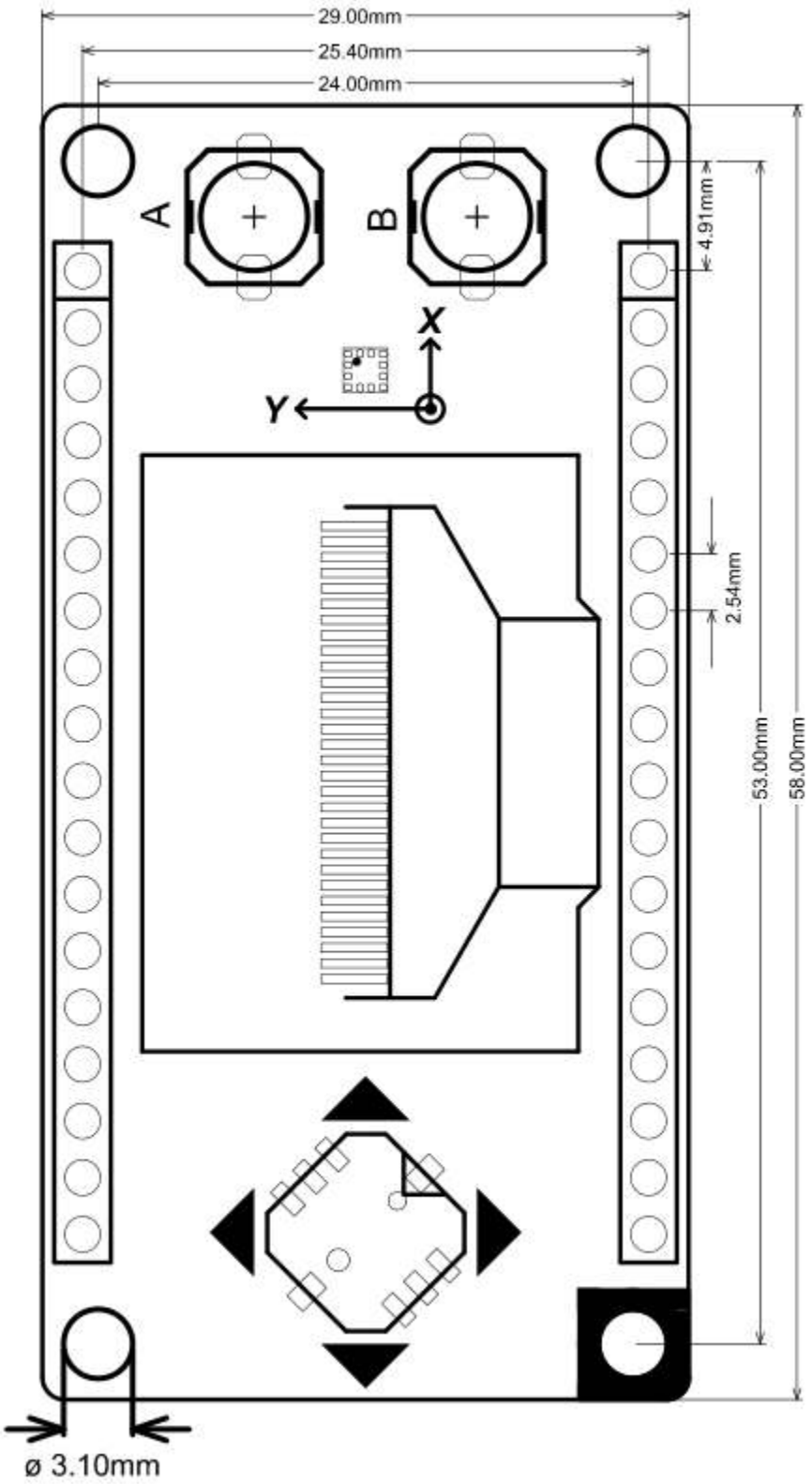
```
}



void loop()
{
  int remainingTimeBudget = ui.update();

  if (remainingTimeBudget > 0) {
    // You can do some work here
    // Don't do stuff if you are below your
    // time budget.
    delay(remainingTimeBudget);
  }
}
```

- Function: Switch between 5 frames
- Function declaration:
- User can edit the frame

```
drawFrame5(*display, *state, x, y)
```

# Dimension

- Pin Spacing: 2.54mm/0.1in
- Mounting hole spacing: 24mm/53mm 0.94/2.09in
- Mounting hole size: 3.1mm/0.12in
- Board Size: 29.00mm×58.00mm/1.14x2.28in
- Thickness:1.6mm/0.06in




FireBeetle Covers-OLED12864 Display Dimensions

# FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum**.

# More Documents

- Hardware Design