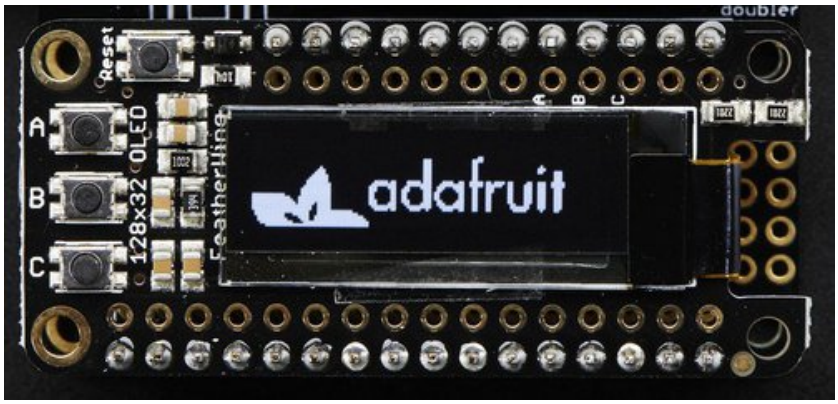
Adafruit OLED FeatherWing

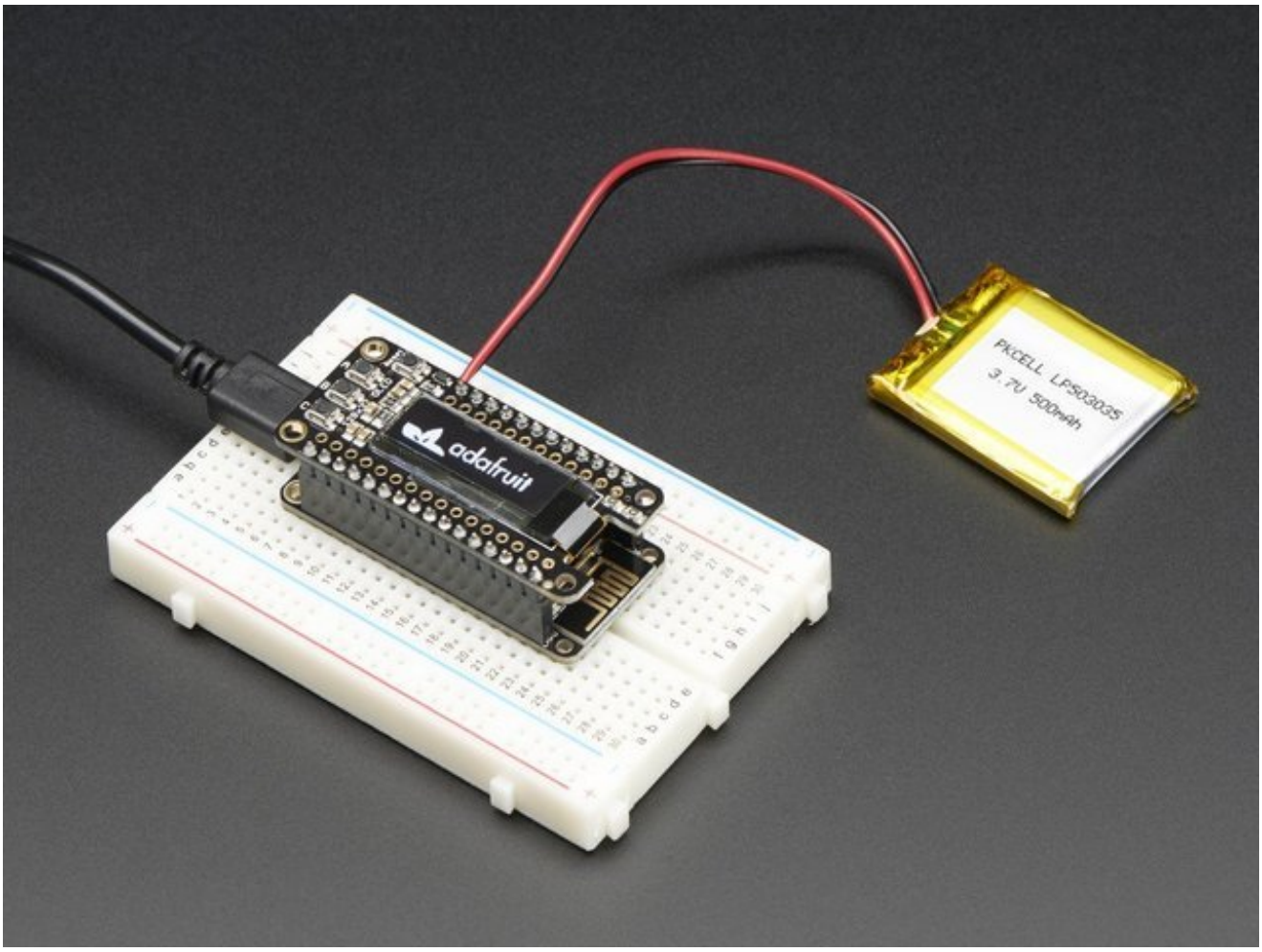
Created by lady ada



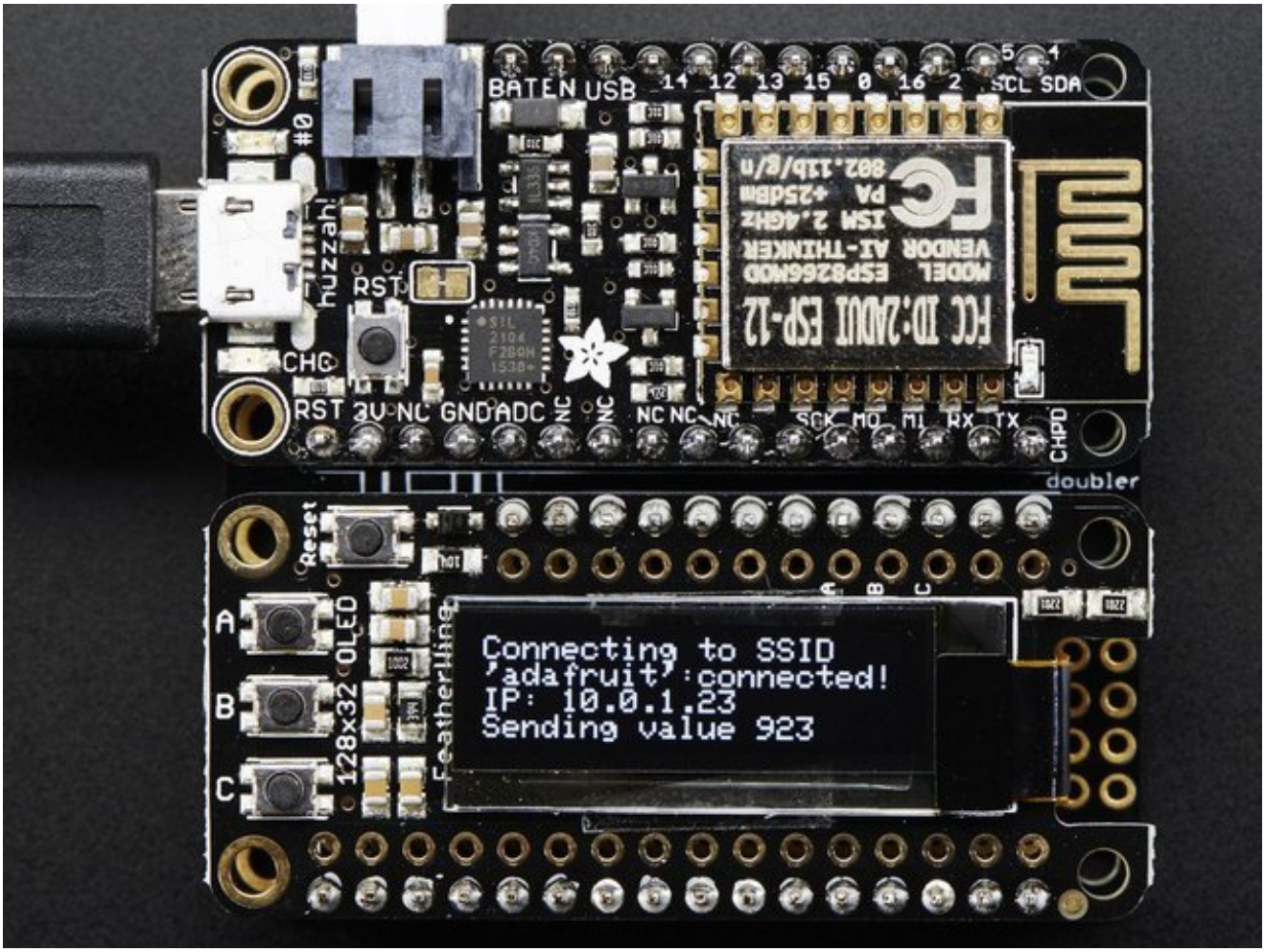
Last updated on 2016-09-15 07:13:44 PM UTC

Guide Contents

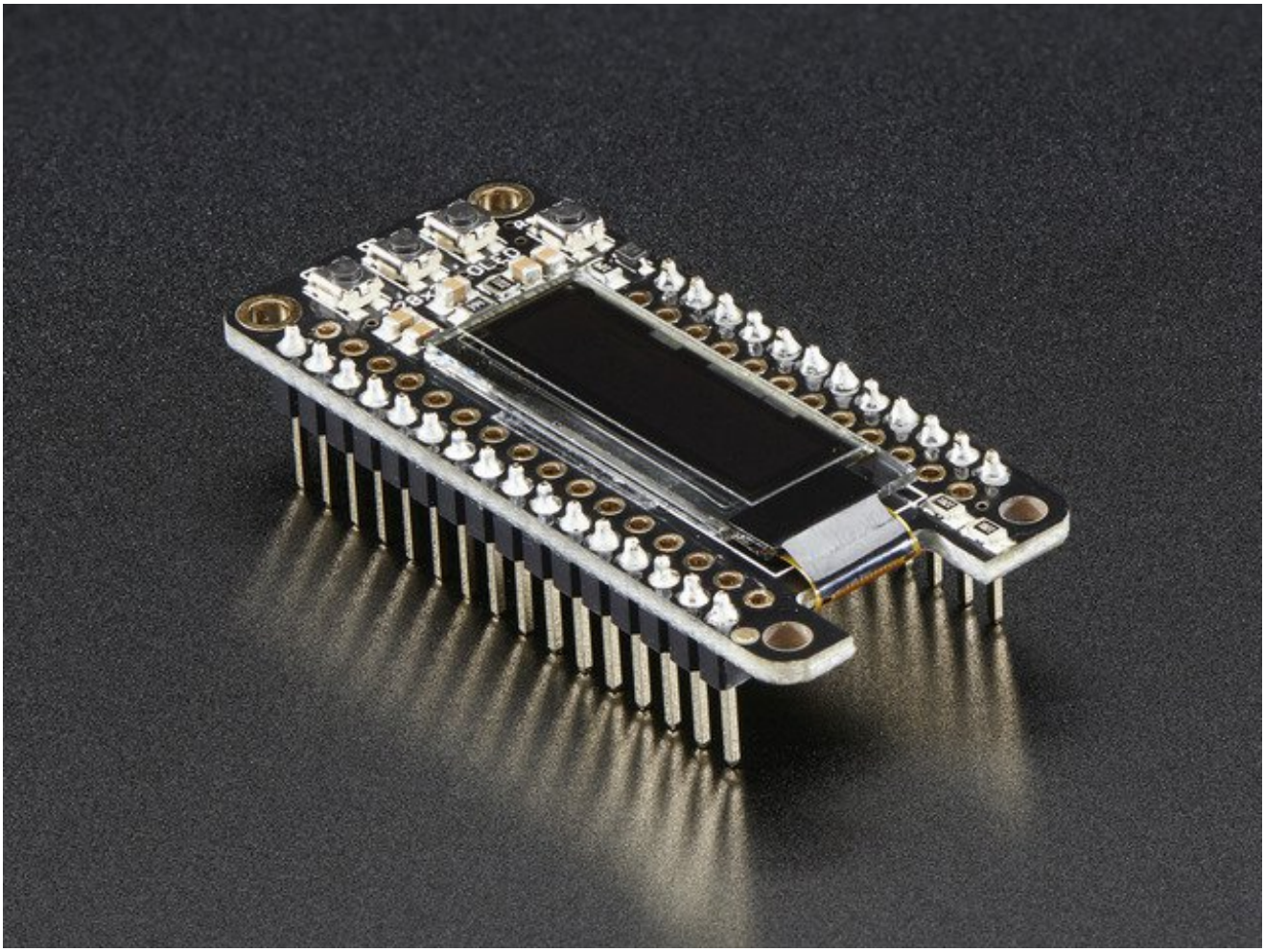
Overview



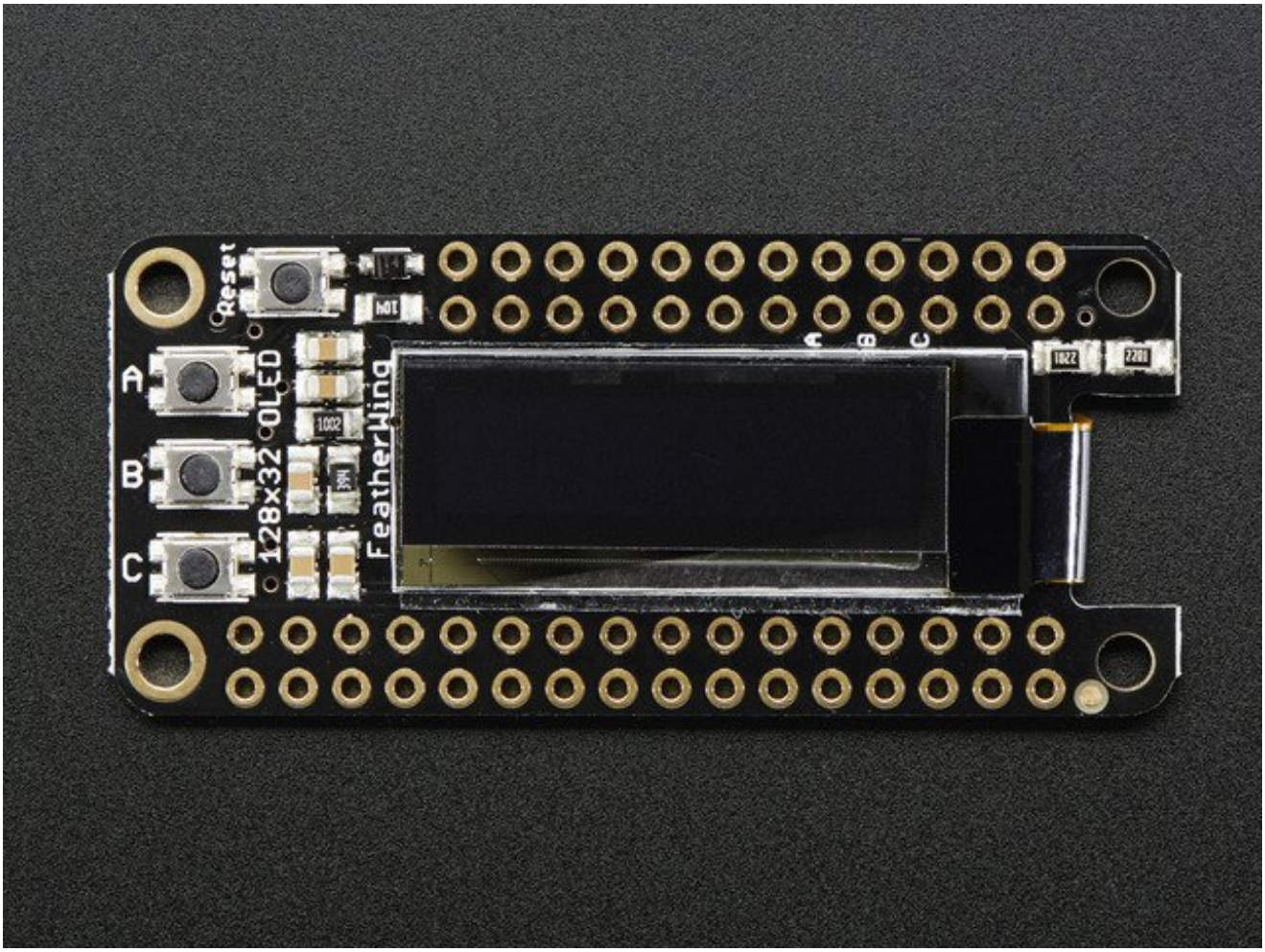
A Feather board without ambition is a Feather board without FeatherWings! This is the **FeatherWing OLED**: it adds a 128x32 monochrome OLED plus 3 user buttons to*any* Feather main board. Using our Feather Stacking Headers (http://adafru.it/2830) or Feather Female Headers (http://adafru.it/2886) you can connect a FeatherWing on top of your Feather board and let the board take flight!



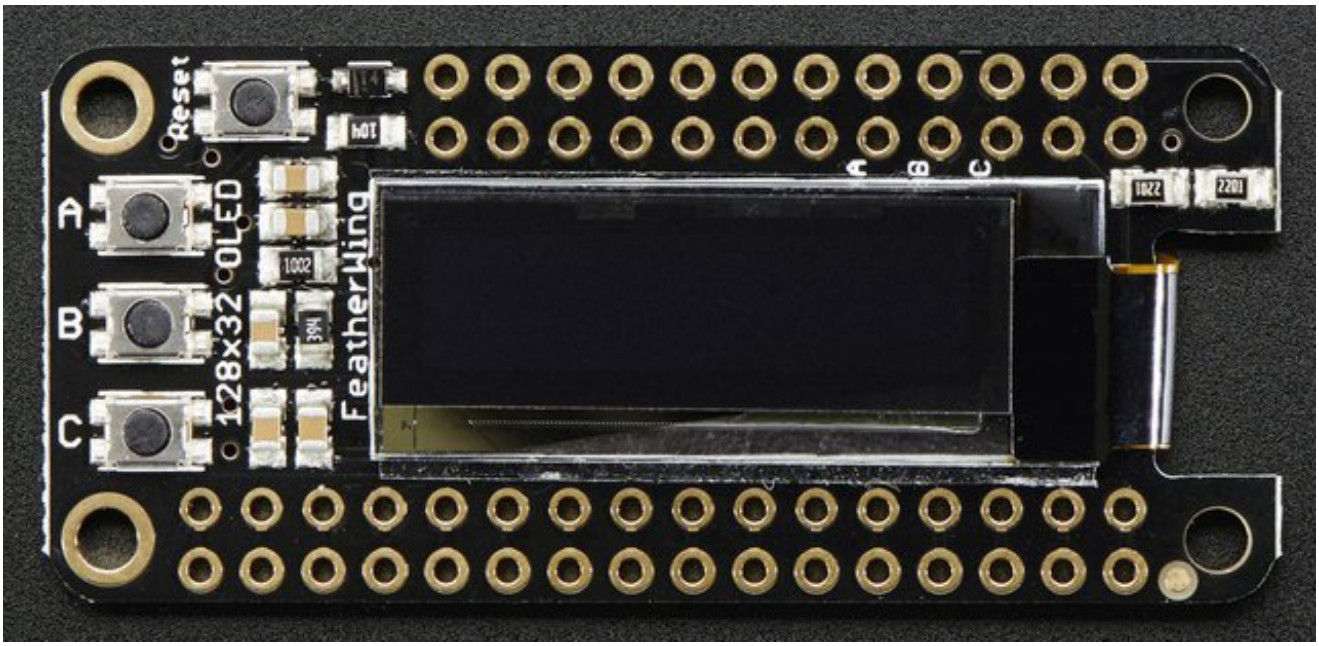
These displays are small, only about 1" diagonal, but very readable due to the high contrast of an OLED display. This screen is made of 128x32 individual white OLED pixels and because the display makes its own light, no backlight is required. This reduces the power required to run the OLED and is why the display has such high contrast; we really like this miniature display for its crispness! We also toss on a reset button and three mini tactile buttons called A B and C so you can add a mini user interface to your feather.



Tested works with all of our Feather 32u4, M0. WICED and ESP8266 boards. The OLED uses only the two I2C pins on the Feather, and you can pretty much stack it with any other FeatherWing, even ones that use I2C since that is a shared bus.

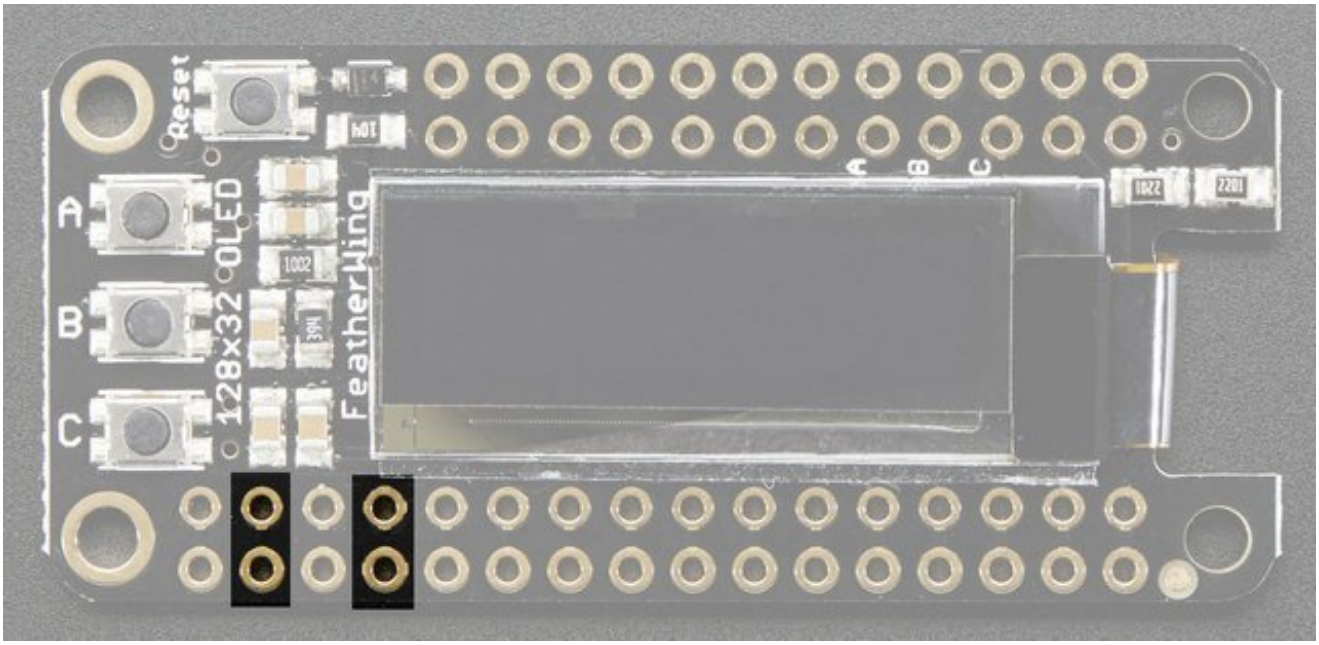

Reset
A
B
C
128x32 OLED
FeatherWing

Pinouts



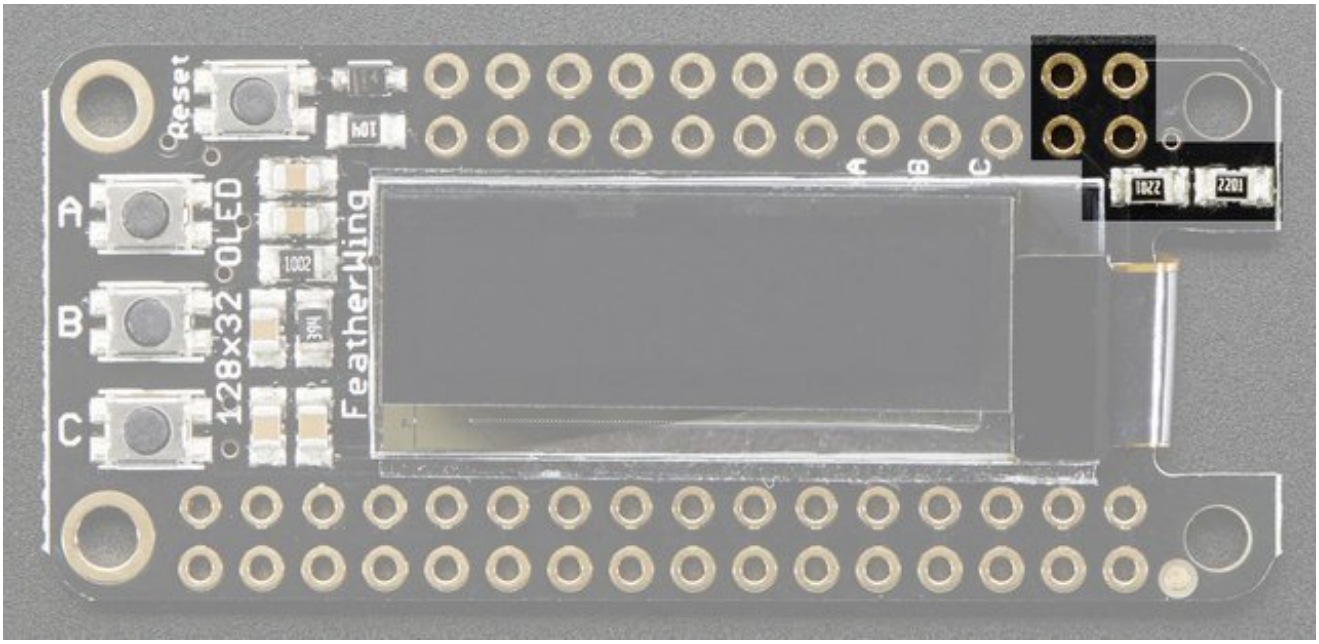
The OLED FeatherWing plugs into any Feather and adds a cute little display. To make it as cross-platform compatible as possible, we use only I2C to control the display. This is not as fast as SPI but it uses only two pins, can share the I2C bus and is fine for the small 128x32 pixel OLED.

Power Pins



OLED displays do not have a backlight, and are fairly low power, this display will draw about 10mA when in use. The display uses 3V power and logic so we just connect to the 3V and GND pins from the feather, as indicated above.

I2C Data Pins

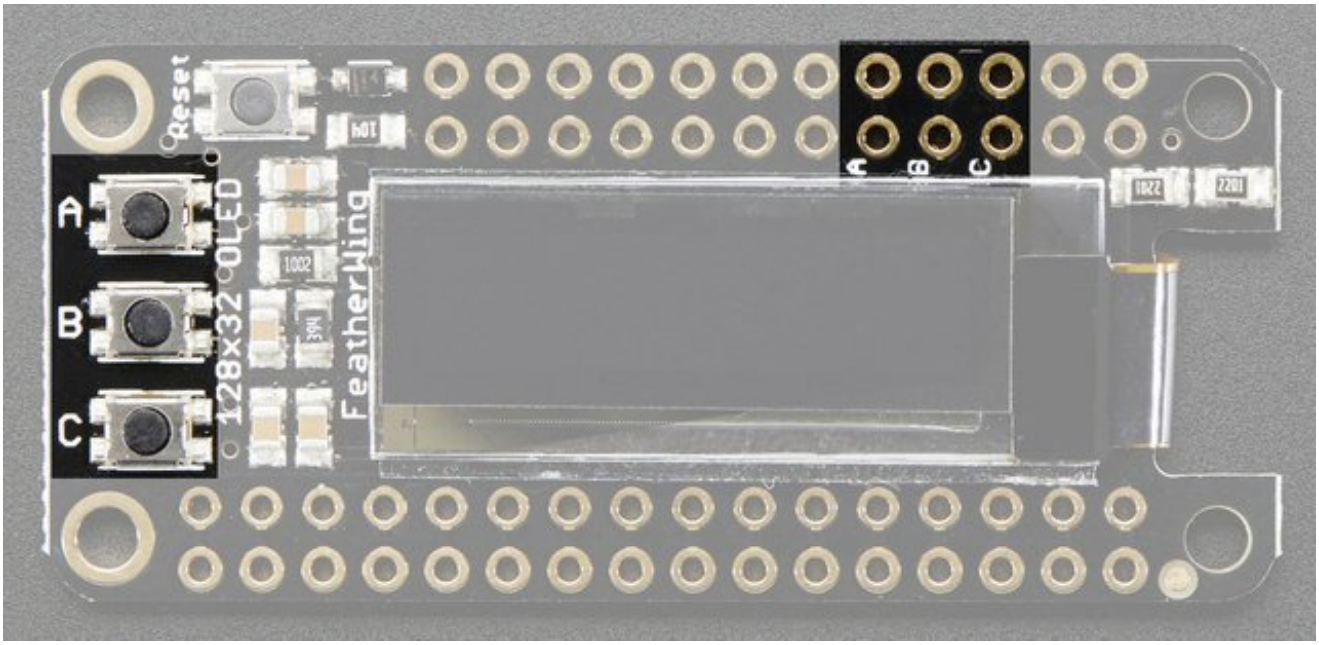


The cute little OLED does all of the data transfer over the I2C pins, highlighed above **SDA** and **SCL**. No other pins are required. There are two 2.2K pullups to 3V on each.

These pins can be shared with other I2C devices.

The I2C address is **0x38** and cannot be changed

Optional Buttons



We had a little bit of space so we added three mini tactile buttons that you can use for user interface. We label them **A B** and **C** because each Feather has slightly different pin numbering schemes and we wanted to make it 'universal'

If you're using Atmega32u4 or ATSAMD21 M0 Feather

- Button A is **#9**
- Button B is **#6**
- Button C is **#5**

If you're using ESP8266:

- Button A is **#0**
- Button B is **#16**
- Button C is **#2**

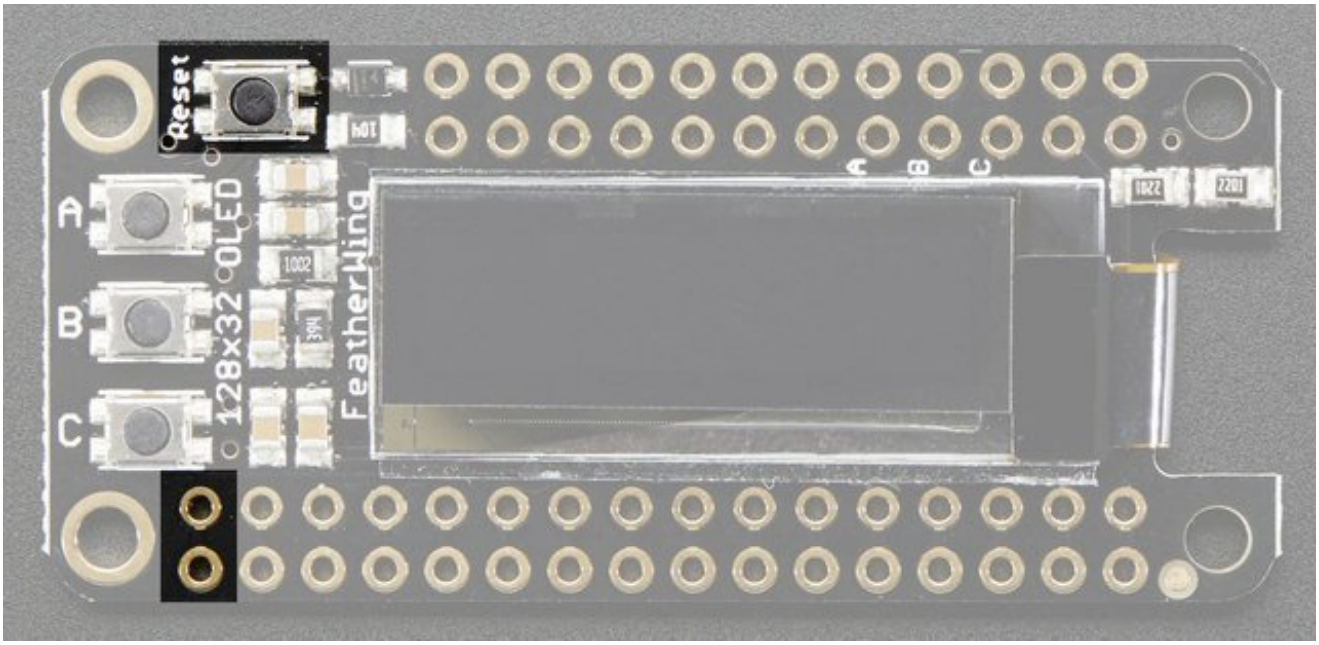
If you're using WICED/STM32 Feather

- Button A is **#PA15**
- Button B is **#PC7**
- Button C is **#PC5**

Button B has a 100K pullup on it so it will work with the ESP8266(which does not have

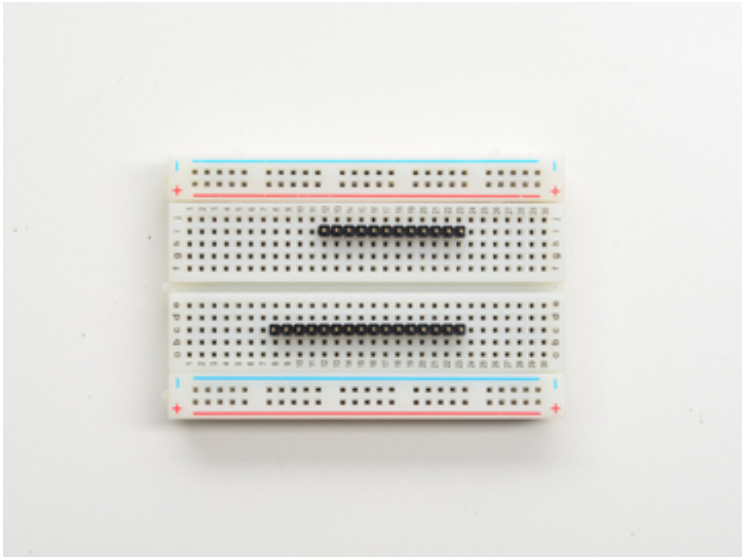
an internal pullup available on that pin). You will need to set up a pullup on all other pins for the buttons to work.

Reset Button



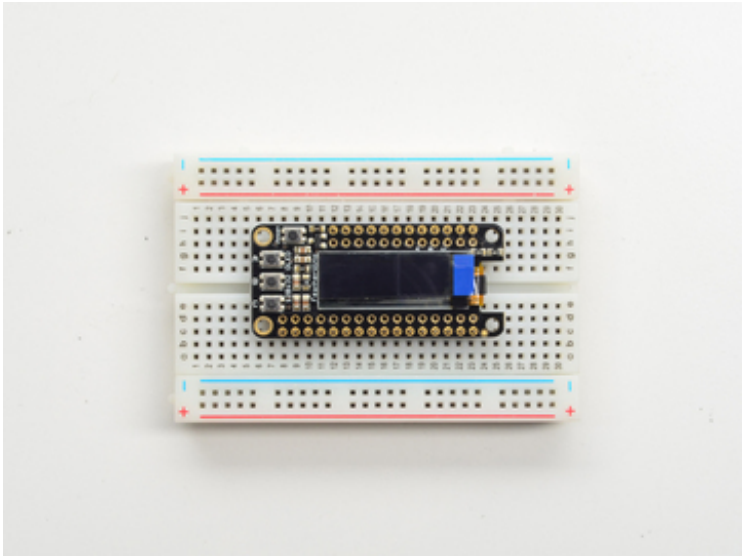
Sometimes its nice to be able to restart your program, so we also have a reset button. It is tied to the **RST** pin marked above.

Assembly

- 

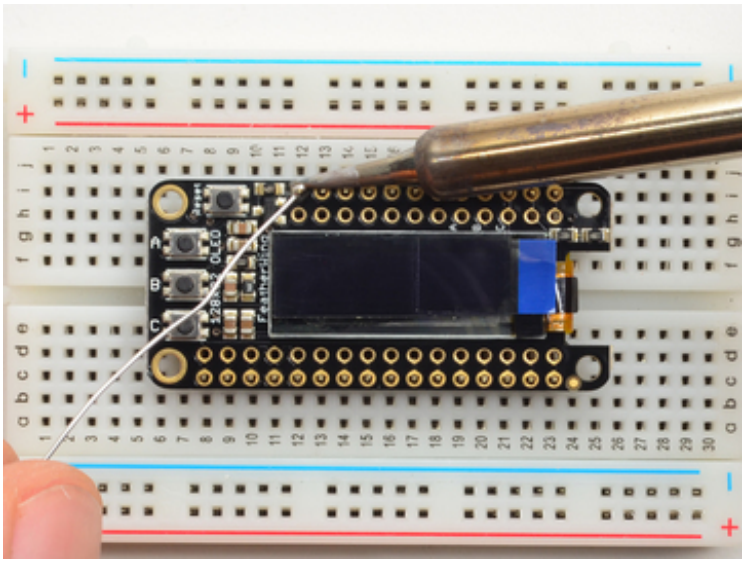
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

- 

Add the FeatherWing:

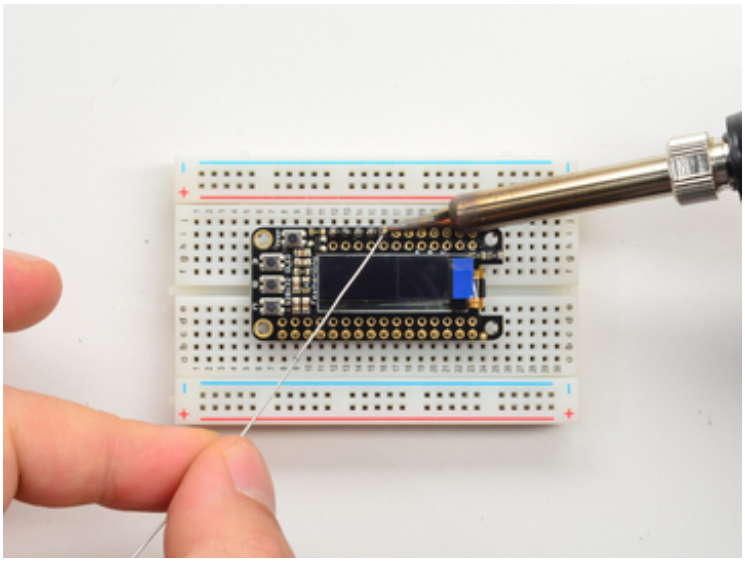
Place the featherwing over the pins so that the short pins poke through the two rows of breakout pads

- 

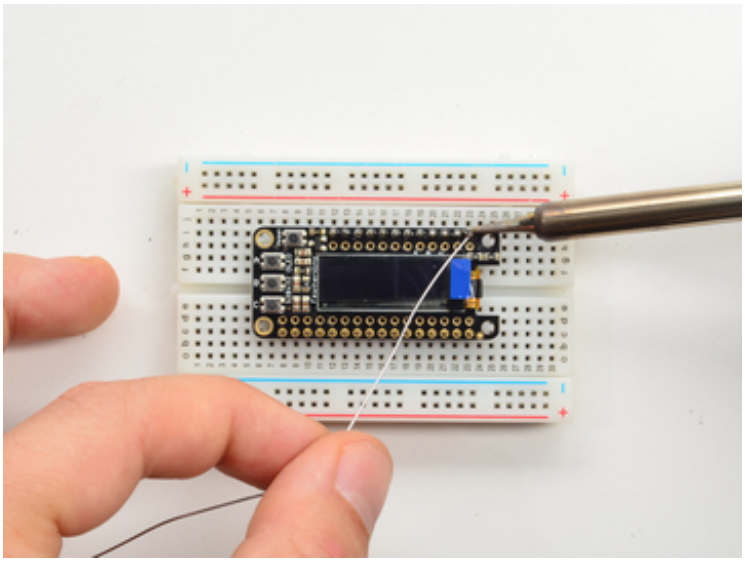
And Solder!

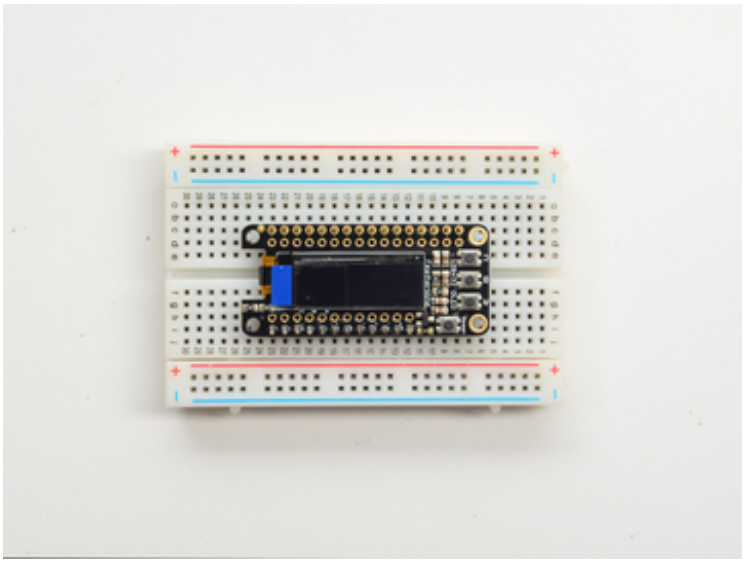
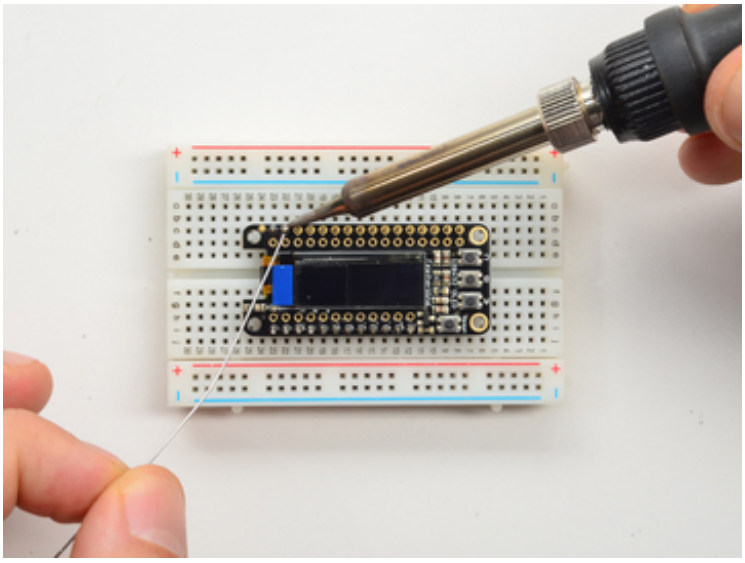
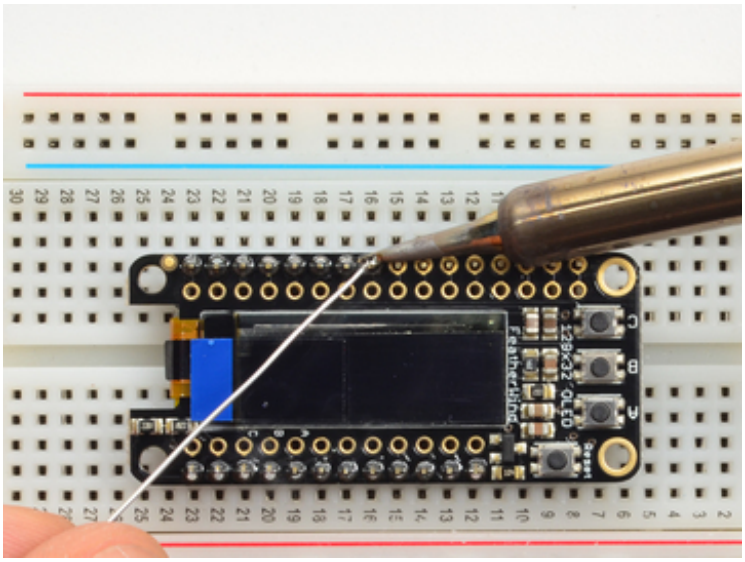
Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](http://adafru.it/aTk)).

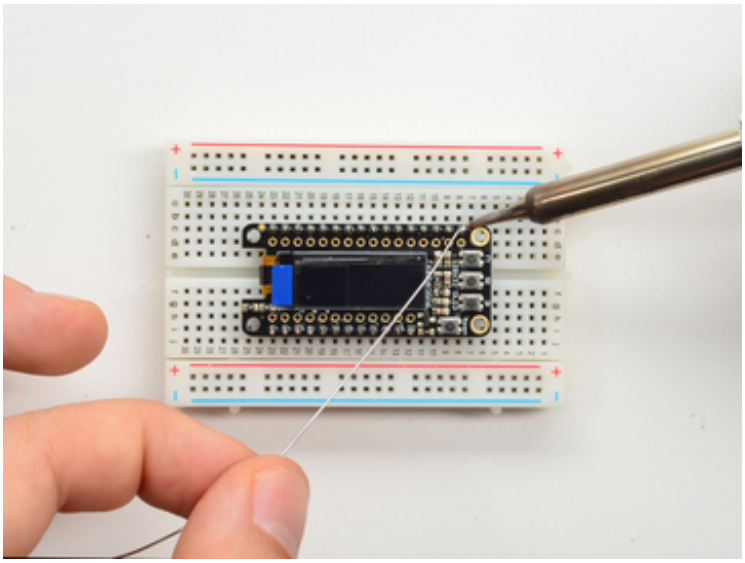
- 

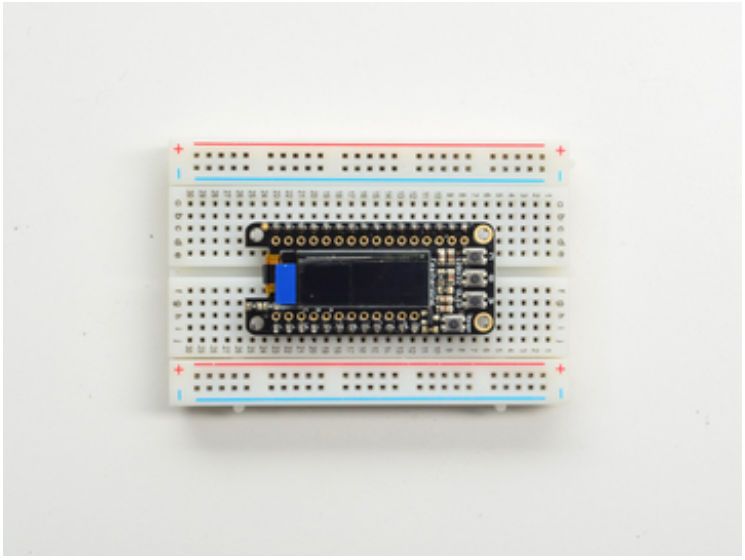
Start by soldering the first row of header

- 

- 
- 
- 

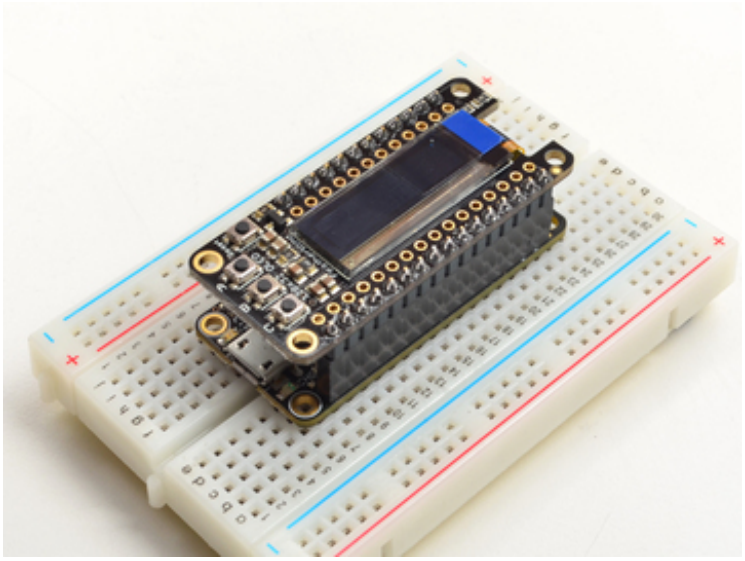
Now flip around and solder the other row completely

- 

- 

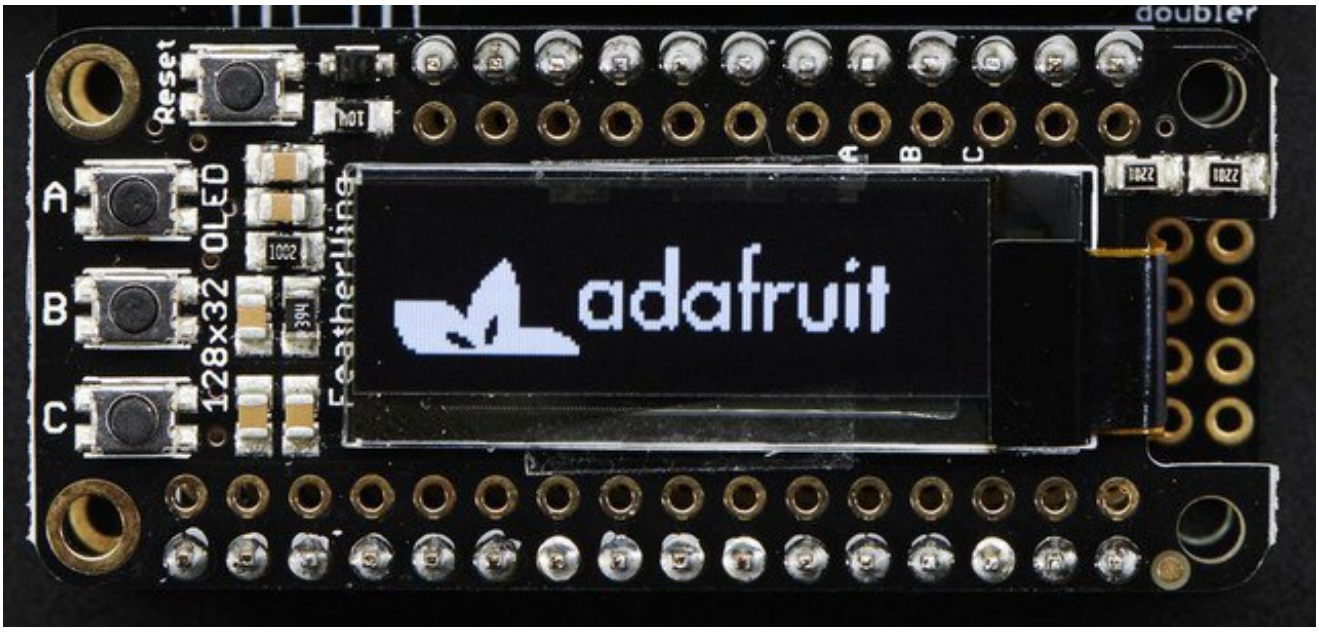
You're done with the two header strips.

Check your solder joints visually and continue onto the next steps

- 

OK You're done! You can now plug your FeatherWing into your Feather and get your OLED on!

Usage



The OLED display we use is well supported and works for all Feathers, all you need is a little library support and you will be drawing in no time!

Install Adafruit SSD1306 Library

Start by installing the support library for the OLED display, you'll need it to talk to the OLED controller chip. We have the Adafruit SSD1306 library repository on GitHub (http://adafru.it/aHq) if you're interested in looking at the code.

Start by downloading the library. You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip:

Download Adafruit_SSD1306 Library
http://adafru.it/e3E

Rename the uncompressed folder **Adafruit_SSD1306** and check that the **Adafruit_SSD1306** folder contains **Adafruit_SSD1306.cpp** and **Adafruit_SSD1306.h**

Place the **Adafruit_SSD1306** library folder your ***arduinosketchfolder*/libraries/** folder. You may need to create the**libraries** subfolder if its your first library. Restart the IDE.We also have a great tutorial on Arduino library installation at:

http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (http://adafru.it/aYM)

Install Adafruit GFX

You will need to do the same for the Adafurit_GFX library available here (http://adafru.it/aJa)

Download Adafruit GFX Library
http://adafru.it/cBB

Rename the uncompressed folder **Adafruit_GFX** and check that the **Adafruit_GFX** folder contains **Adafruit_GFX.cpp** and **Adafruit_GFX.h**

Place the **Adafruit_GFX** library folder your ***arduinosketchfolder***/**libraries**/ folder like you did with the SSD1306 library

Run Example Code

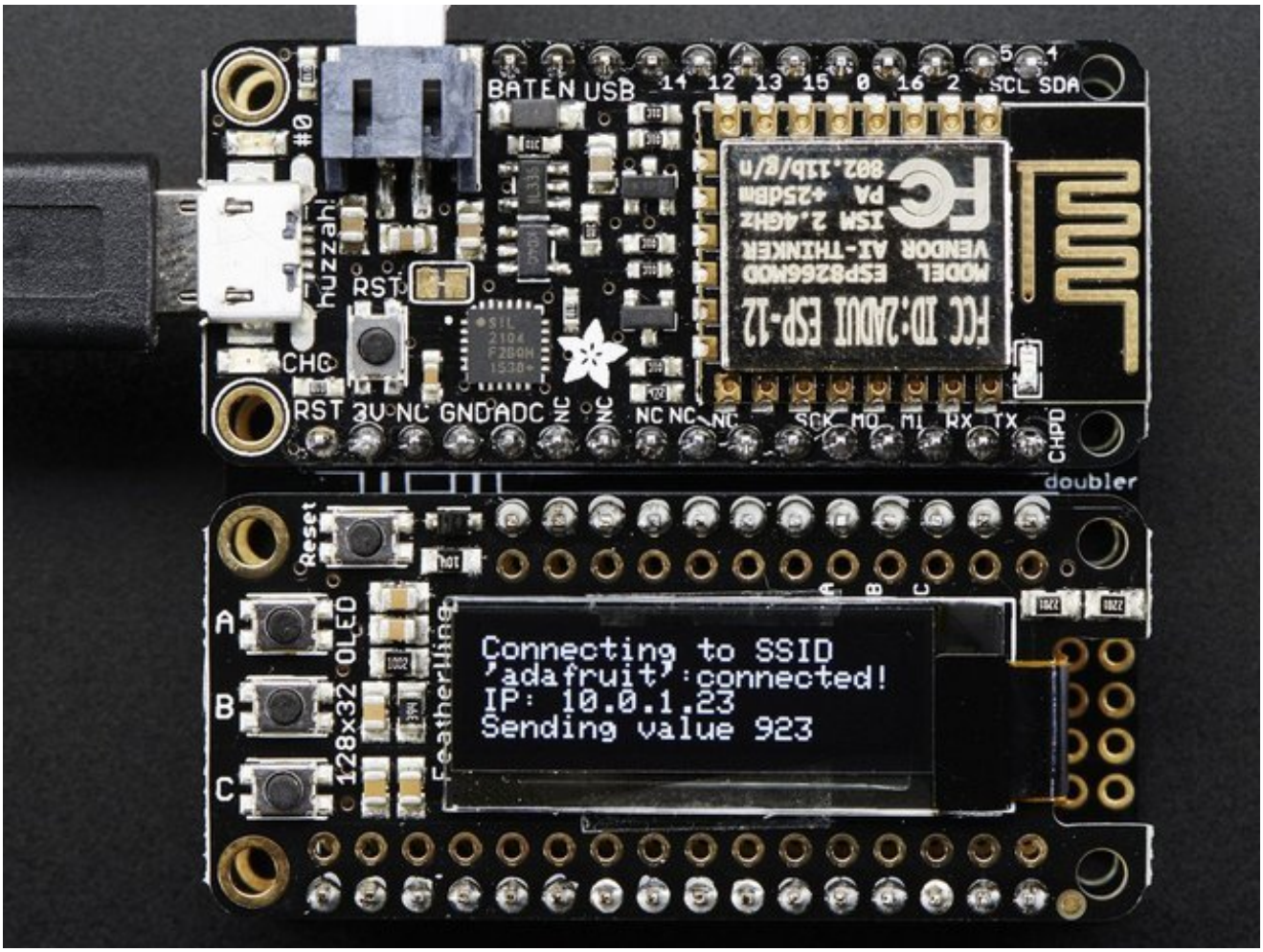
We have a basic demo that works with all Feathers, so restart the IDE and compile/upload this sketch

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display = Adafruit_SSD1306();

#if defined(ESP8266)
  #define BUTTON_A 0
  #define BUTTON_B 16
  #define BUTTON_C 2
  #define LED      0
#elif defined(ARDUINO_STM32F2_FEATHER)
  #define BUTTON_A PA15
  #define BUTTON_B PC7
  #define BUTTON_C PC5
  #define LED PB5
#elif defined(TEENSYDUINO)
  #define BUTTON_A 4
  #define BUTTON_B 3
  #define BUTTON_C 8
  #define LED 13
#else
  #define BUTTON_A 9
```

```
  #define BUTTON_B 6
  #define BUTTON_C 5
  #define LED      13
#endif

#if (SSD1306_LCDHEIGHT != 32)
 #error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

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

  Serial.println("OLED FeatherWing test");
  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);  // initialize with the I2C addr 0x3C (for the 128x32)
  // init done
  Serial.println("OLED begun");

  // Show image buffer on the display hardware.
  // Since the buffer is intialized with an Adafruit splashscreen
  // internally, this will display the splashscreen.
  display.display();
  delay(1000);

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

  Serial.println("IO test");

  pinMode(BUTTON_A, INPUT_PULLUP);
  pinMode(BUTTON_B, INPUT_PULLUP);
  pinMode(BUTTON_C, INPUT_PULLUP);

  // text display tests
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.print("Connecting to SSID\n'adafruit':");
  display.print("connected!");
  display.println("IP: 10.0.1.23");
  display.println("Sending val #0");
  display.setCursor(0,0);
  display.display(); // actually display all of the above
}


void loop() {
  if (! digitalRead(BUTTON_A)) display.print("A");
  if (! digitalRead(BUTTON_B)) display.print("B");
  if (! digitalRead(BUTTON_C)) display.print("C");
```

```
  delay(10);
  yield();
  display.display();
}
```

You should see the OLED display a splash screen then spit out some text. If you press the **A B** or **C** buttons it will also print those out



Do more!

You can use any of the Adafruit GFX library commands to draw onto your OLED, that means that you get all sorts of shapes, fonts, lines, etc available. [Check out GFX for all the underlying graphics support functions and how they work](http://adafru.it/doL) (http://adafru.it/doL)

Dont forget you have to call display() after drawing to push the buffer out to the display!

FeatherOLED Library

To make it easier to work with the OLED FeatherWing with specific Feather boards we've published an **optional** BETA library called Adafruit_FeatherOLED (http://adafru.it/m3b).

The library consists of a base class called Adafruit_FeatherOLED which breaks the 128x32 displays in four rows of eight pixels each, and you can extend the base class with your own OLED implementation to render custom icons, etc.

The top and bottom eight pixels are reserved for status icons and data rendered by the helper class, and the middle 16 pixels can be used to display custom messages or data.

Downloading the Library

You can download the latest version of the library by clicking on the following link, and then unzipping this file in your Arduino/libraries folder as Arduino/libraries/Adafruit_FeatherOLED:

Download Adafruit_FeatherOLED from Github
http://adafru.it/nbf

Adafruit_FeatherOLED Base Class

The **Adafruit_FeatherOLED** class exposes the following functions that will be available in every specialised instance of this library:

void init (void)

Initialises the OLED display, clearing it's contents and configuring the low level hardware. This function must be called before any other actions takes place with this library.

void setBattery (float vbat)

Sets the battery level in volts using a floating point value. For example:.setBattery(4.27F).

void setBatteryVisible (bool enable)

Enables or disables the battery display on the OLED. Setting 'true' in this function will cause the battery icon to be visible, otherwise set it to 'false' to disable the icon.

void setBatteryIcon (bool enable)

Setting this to 'true' will cause an icon to be displayed for the battery level. Setting this to 'false' will cause the raw voltage level to be displayed as a text string.

void clearMsgArea (void)

Calling this function will clear the middle 128x16 pixels message area on the OLED display.

Adafruit_FeatherOLED_WiFi

This sub-class is designed to display basic WiFi details like your IP address, connection status and the RSSI level for your connection. It can be used with boards like the Adafruit WICED Feather (http://adafru.it/3056), which includes a sample sketch (http://adafru.it/nbg) for this class in the default example folder.

void setConnected (bool conn)

Sets the 'connected' status to either true or false, which will cause the appropriate icon to be updated on the bottom 8 pixel status bar.

void setConnectedVisible (bool enable)

Enables or disables the connected icon in the bottom 8 pixel status bar. Setting this to 'true' enables the connected icon.

void setRSSI (int rssi)

Sets the RSSI level for the connection, which is a value describing the relative signal strength in dBm between two connected wireless devices. This is a negative integer value where the smaller the number the better the signal strength (meaning -90dBm is much weaker than -65dBm).

void setRSSIVisible (bool enable)

Enables or disables the RSSI icon in the top 8 pixel status bar. Setting this to 'true' enables the RSSI icon.

void setIPAddress (uint32_t addr)

Sets the IP address for the device, which will be displayed in the bottom 8 pixel status bar

of the OLED display. A single 32-bit integer is sent to this function, where each 8-bit component of the 4 byte IP address is seperated and printed out.

void setIPAddressVisible (bool enable)

Enables or disables the IP address in the bottom 8 pixel status bar. Setting this to 'true' enables the IP address.

void refreshIcons (void)

Calling this function will redraw all of the status icons, and should be called if you have made any changes to the values using the helper functions like .setRSSI, .setIPAddress, etc.

Adafruit_FeatherOLED_WiFi Example

The following example shows how the WiFi helper class can be used with the Adafruit WICED Feather (http://adafru.it/3056) to display basic information about the connection as well as custom messages.

It should give you the following output, and any connection errors will be displayed in the central message area:


```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_FeatherOLED.h>
#include <Adafruit_FeatherOLED_WiFi.h>
#include <Adafruit_Sensor.h>
#include <adafruit_feather.h>
#include <adafruit_mqtt.h>
#include <adafruit_aio.h>

#define WLAN_SSID             "SSID"
```

```
#define WLAN_PASS               "PASSWORD"

#define VBAT_ENABLED            1
#define VBAT_PIN                PA1

#define SENSOR_TSL2561_ENABLED  0
#if SENSOR_TSL2561_ENABLED
  #include <Adafruit_TSL2561_U.h>
  bool _tslFound = false;
  Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345);
#endif

#define AIO_ENABLED             0
#define AIO_USERNAME            "...your AIO username (see https://accounts.adafruit.com)..."
#define AIO_KEY                 "...your AIO key..."

#define FEED_VBAT               "vbat"
#define FEED_TSL2561_LUX        "lux"

AdafruitAIO                     aio(AIO_USERNAME, AIO_KEY);
AdafruitAIOFeedGauge<float>     feedVBAT(&aio, FEED_VBAT);
AdafruitAIOFeedGauge<float>     feedLUX (&aio, FEED_TSL2561_LUX);

Adafruit_FeatherOLED_WiFi  oled = Adafruit_FeatherOLED_WiFi();

/**************************************************************************/
/*!
    @brief  Connect to the AP
    @return Error code
*/
/**************************************************************************/
bool connectAP()
{
  oled.refreshIcons();
  oled.clearMsgArea();
  oled.println("Connecting to ...");
  oled.println(WLAN_SSID);
  oled.display();

  // Attempt to connect to the AP
  if ( Feather.connect(WLAN_SSID, WLAN_PASS) )
  {
    int8_t rssi = Feather.RSSI();
    uint32_t ipAddress = Feather.localIP();
    oled.setConnected(true);
    oled.setRSSI(rssi);
    oled.setIPAddress(ipAddress);
    oled.refreshIcons();
    oled.clearMsgArea();
  }
  else
```

```
  {
    // Display the error message
    err_t err = Feather.errno();
    oled.setConnected(false);
    oled.refreshIcons();
    oled.clearMsgArea();
    oled.println("Connection Error:");
    switch (err)
    {
      case ERROR_WWD_ACCESS_POINT_NOT_FOUND:
        // SSID wasn't found when scanning for APs
        oled.println("Invalid SSID");
        break;
      case ERROR_WWD_INVALID_KEY:
        // Invalid SSID passkey
        oled.println("Invalid Password");
        break;
      default:
        // The most likely cause of errors at this point is that
        // you are just out of the device/AP operating range
        oled.print(Feather.errno());
        oled.print(":");
        oled.println(Feather.errstr());
        oled.refreshIcons(); // Refresh icons in case the text ran over
        break;
    }
    oled.display();
    // Return false to indicate that we received an error (available in feather.errno)
    return false;
  }

  return true;
}

/**************************************************************************/
/*!
    @brief
*/
/**************************************************************************/
void updateVbat()
{
  int   vbatADC   = 0;        // The raw ADC value off the voltage div
  float vbatFloat = 0.0F;     // The ADC equivalent in millivolts
  float vbatLSB   = 0.80566F;  // mV per LSB

  // Read the analog in value:
  vbatADC = analogRead(VBAT_PIN);
  vbatADC = analogRead(VBAT_PIN);

  // Multiply the ADC by mV per LSB, and then
  // double the output to compensate for the
```

```
  // 10K+10K voltage divider
  vbatFloat = ((float)vbatADC * vbatLSB) * 2.0F;

  oled.setBattery(vbatFloat/1000);

  // Push VBAT out to MQTT if possible
  if (AIO_ENABLED && aio.connected())
  {
    feedVBAT = vbatFloat/1000;
  }
}

/**************************************************************************/
/*!
    @brief  The setup function runs once when the board comes out of reset
*/
/**************************************************************************/
void setup()
{
  // Wait for Serial Monitor
//  while(!Serial) delay(1);

  // Setup the LED pin
  pinMode(BOARD_LED_PIN, OUTPUT);

  // Setup the OLED display
  oled.init();
  oled.clearDisplay();

  // Initialize tsl sensor if enabled
  #if SENSOR_TSL2561_ENABLED
  if (tsl.begin())
  {
    _tslFound = true;
    tsl.enableAutoRange(true);
    tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but low resolution */
    // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);  /* medium resolution and speed   */
    // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS);  /* 16-bit data but slowest conversions */
  }
  #endif

  // Get a VBAT reading before refreshing if VBAT is available
  #if VBAT_ENABLED
  pinMode(VBAT_PIN, INPUT_ANALOG);
  oled.setBatteryIcon(true);
  updateVbat();
  #endif

  // Refresh the screen
  oled.refreshIcons();
  oled.clearMsgArea();
```

```
  // Try to connect to the AP
  if ( !connectAP() )
  {
    // Enter a while(1) loop here since any connection error
    // is handled in .connectAP() above
    while(1);
  }

  #if AIO_ENABLED
  // Attempt to connect to a Broker
  oled.clearMsgArea();
  oled.println("io.adafruit.com");
  oled.display();

  // Connect to AIO server
  if ( aio.connect() )
  {
    oled.println("Connected!");
    oled.display();
  }else
  {
    oled.print("Failed! Error: ");
    oled.println(aio.errno(), HEX);
    oled.display();
    delay(3000);
  }

  // Follow feed if enabled
  if ( VBAT_ENABLED )
    feedVBAT.follow(aio_vbat_callback);

  // Follow feed if enabled
  if ( SENSOR_TSL2561_ENABLED )
    feedLUX.follow(aio_vbat_callback);

  #endif
}

/**************************************************************************/
/*!
    @brief  This loop function runs over and over again
*/
/**************************************************************************/
void loop()
{
  // Update the battery level
  if ( VBAT_ENABLED )
    updateVbat();

  if ( Feather.connected() )
```

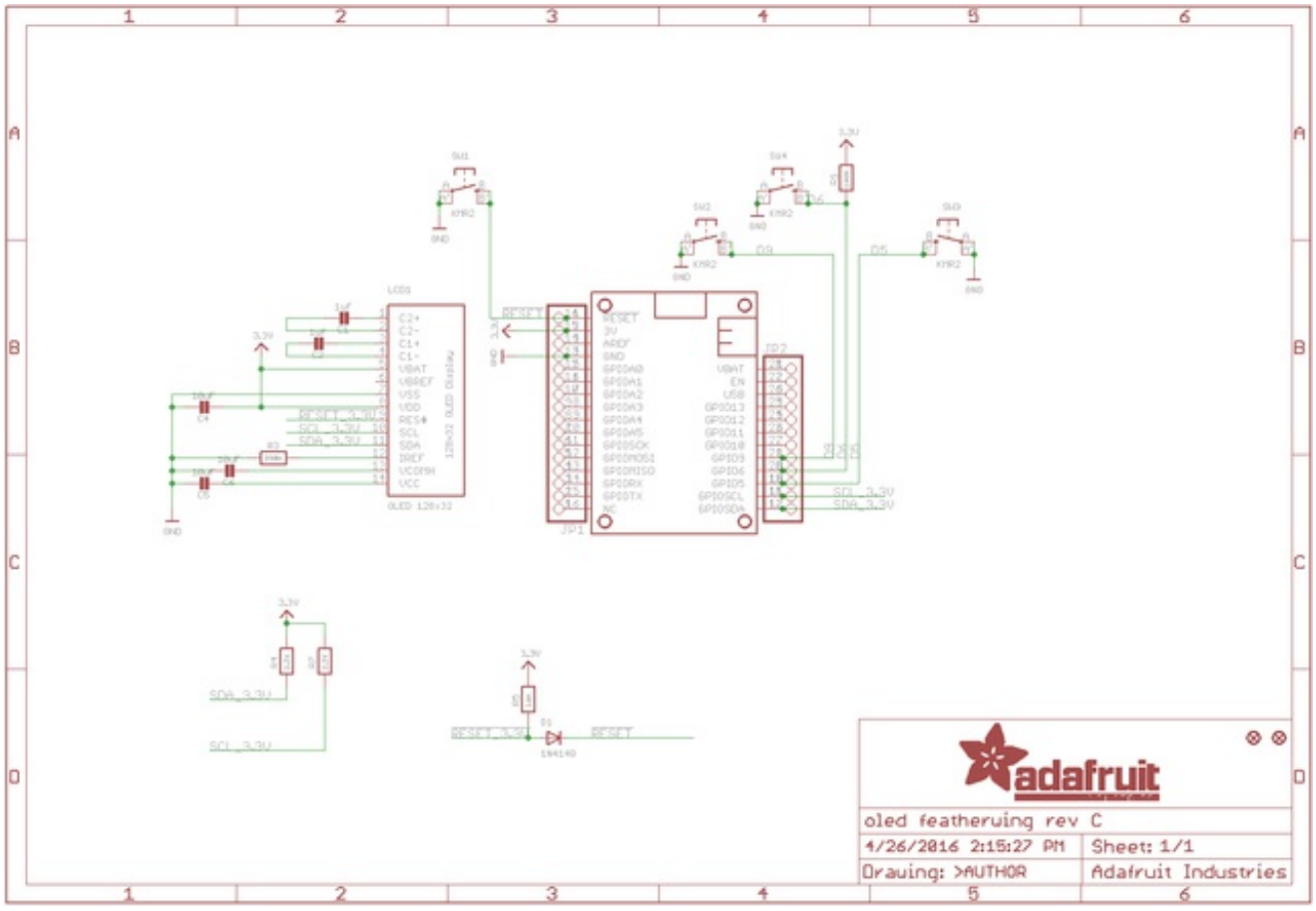
```
 {
  // Update the RSSI value
  int8_t rssi = Feather.RSSI();
  oled.setRSSI(rssi);

  // Get a light sample and publish to MQTT if available
  #if SENSOR_TSL2561_ENABLED
  if (_tslFound)
  {
    oled.clearMsgArea();
    sensors_event_t event;
    // Get a new data sample
    bool sensor_data = tsl.getEvent(&event);
    if (sensor_data)
    {
      if (AIO_ENABLED && aio.connected())
      {
        feedLUX = event.light;

        oled.clearMsgArea();
        oled.print("Lux -> AIO: ");
        oled.println(event.light);
        oled.display();
      }
    }
    else
    {
      oled.clearMsgArea();
      oled.println("Sensor failed");
      oled.display();
      Serial.println("Sensor failed");
    }
  }
  #endif
 }
 else
 {
  // The connection was lost ... reset the status icons
  oled.setConnected(false);
  oled.setRSSI(0);
  oled.setIPAddress(0);
  oled.clearMsgArea();
 }

 oled.refreshIcons();
 togglePin(BOARD_LED_PIN);
 delay(10000);
}

/*************************************************************************/
/*!
```

```
    @brief AIO callback when there is new value with Feed VBAT
*/
/**************************************************************************/
void aio_vbat_callback(float value)
{
//  oled.println("AIO VBAT: ");
//  oled.display();
}

/**************************************************************************/
/*!
    @brief AIO callback when there is new value with Feed LUX
*/
/**************************************************************************/
void aio_lux_callback(float value)
{
//  oled.println("AIO LUX:");
//  oled.display();
}
```

Download

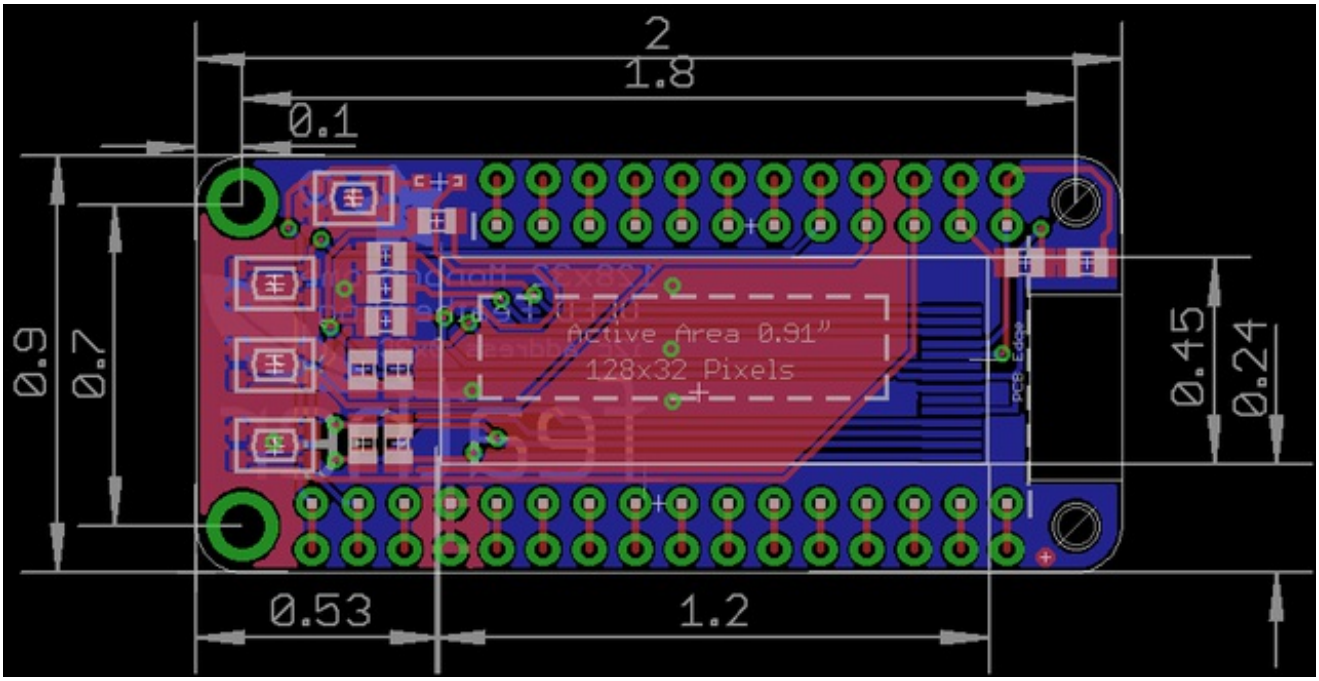
- SSD1306 (http://adafru.it/aJK) Datasheet
- OLED UG-2832HSWEG02 Datasheet (http://adafru.it/qrf)
- PCB Files in EagleCAD format (http://adafru.it/nbc)
- Fritzing object available in the Adafruit Fritzing Library(http://adafru.it/aP3)

Schematics



Fabrication Print

Dimensions in inches


2
1.8
0.1
0.9
0.7
Active Area 0.91"
128x32 Pixels
0.45
0.24
0.53
1.2