ePaper FeatherWing Quote Display

Created by Dan Cogliano

https://learn.adafruit.com/epaper-display-featherwing-quote-display

Last updated on 2024-06-03 02:44:03 PM EDT

Table of Contents

Overview 3
• Parts
Arduino Setup 5
Connections 13
How it Works 14
• Going Further

Overview

Build this quote display using the Adafruit HUZZAH32 Feather and ePaper FeatherWing. Add a LiPo battery and you have a portable quote display.

The great thing about ePaper displays is they keep their display even when power is disconnected. Combine that with the HUZZAH 32 Feather, which can power itself off and wake up at a specified interval, and you have a great symbiotic relationship. A battery powered ePaper display and HUZZAH Feather can continuously display an image for weeks and even months, depending on the battery size and how frequently the HUZZAH powers up to update the display.

However, there are a few shortcomings in this relationship. An ePaper display is limited to black and white (and red for some ePaper displays), and it can not display fast moving graphics like those used in games and videos, since the refresh rate can be several seconds to display a static image. So ePaper displays work best with static images that update infrequently. This quote display is a great use of an ePaper display with its static image.

Another shortcoming with an ePaper display is that it requires a memory area for drawing text and graphics before it is shown on the display. With an Adafruit 2.13 ePaper display, this will take 7.6 KB of memory (250 x 122 pixels x 2 colors), leaving less memory available for your sketch to run in. Also, the larger the display, the more memory needed, making large ePaper displays impracticable for most microcontrollers.

Fortunately, Adafruit has come up with an answer to this shortcoming. By adding display memory to the ePaper Featherwing, graphics memory is offloaded from the microcontroller to the display module, making more memory available to your sketch.

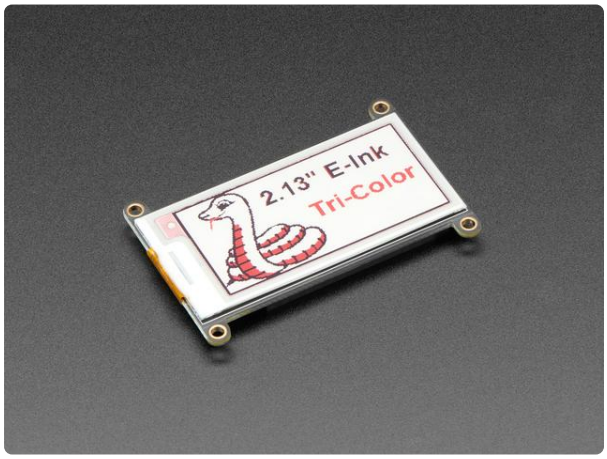
Parts

Building the quote display is actually very easy and requires just 3 parts: an ePaper FeatherWing display, a HUZZAH32 Feather and a battery for portable use. Pick one of the 2 ePaper FeatherWings Adafruit currently offers: a monochrome 2.13" display and a tri-color 2.13" display.

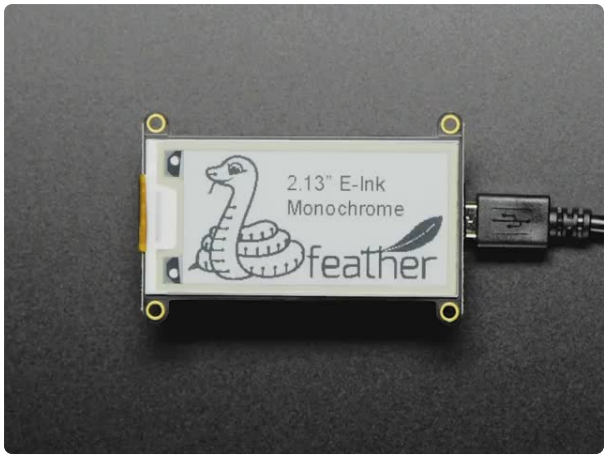
There are also a couple of choices of HUZZAH32 Feathers. If you want a quick, no solder project, save some time and go with the pre-soldered version. If you don't mind soldering and save a little bit of money, go with the loose headers version.

The battery makes this project portable and with the low-power features of both the HUZZAH32 and ePaper display, this display can show quotes for weeks or even months on a single charge. The length of time will depend on the size of the battery and how often the HUZZAH32 wakes up to show a new quote.

If you are interested in ePaper displays for other projects, check out the entire line of Adafruit's ePaper displays (https://adafru.it/ExU).

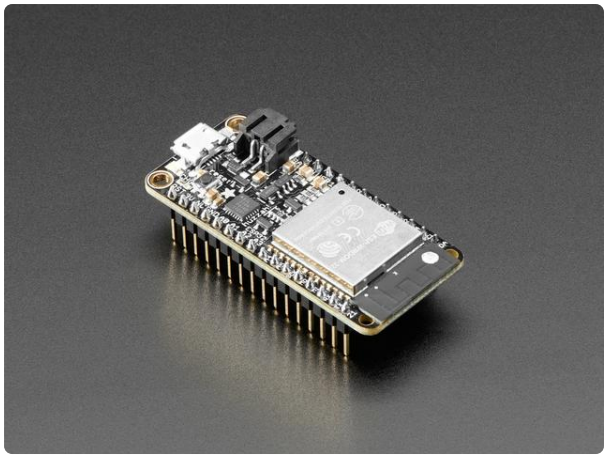


Adafruit 2.13" Tri-Color eInk / ePaper Display FeatherWing
Easy e-paper finally comes to your Feather, with this breakout that's designed to make it a breeze to add a tri-color eInk display. Chances are you've seen one of those...
https://www.adafruit.com/product/4128



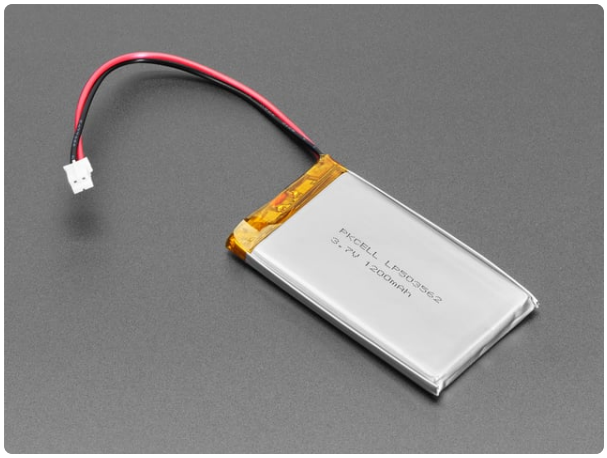
Adafruit 2.13" Monochrome eInk / ePaper Display FeatherWing

Easy e-paper finally comes to your Feather, with this breakout that's designed to make it a breeze to add a monochrome eInk display. Chances are you've seen one of those...
https://www.adafruit.com/product/4195



Adafruit HUZZAH32 – ESP32 Feather Board (pre-soldered)

Aww yeah, it's the Feather you have been waiting for, this time with pre-assembled headers! The HUZZAH32 is our ESP32-based Feather, made with the...
https://www.adafruit.com/product/3591



Lithium Ion Polymer Battery - 3.7v 1200mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/258

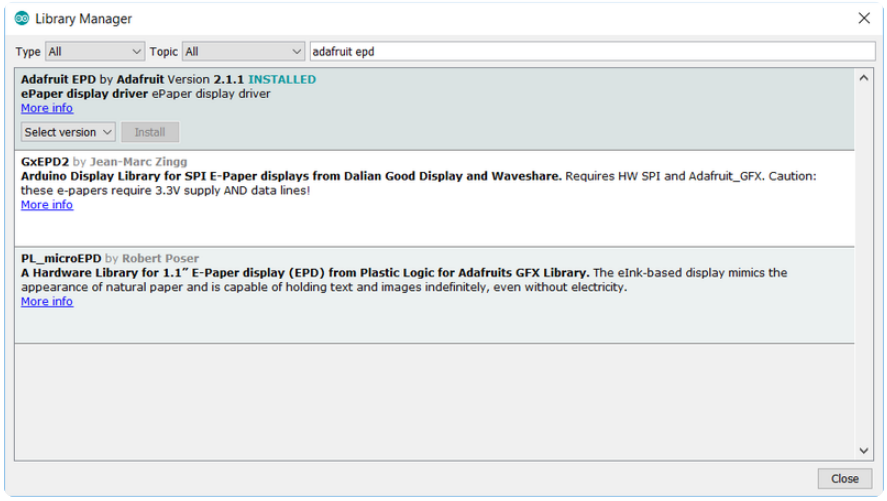
Arduino Setup

if you don't have it already, you will need the Arduino IDE installed on your computer to upload this sketch to the HUZZAH32. You will find information on installing Arduino in this learning guide (https://adafru.it/CfF).

You will then need to install three libraries that are needed by the sketch. the Adafruit EPD (ePaper display) library, the Adafruit GFX library, and the Benoit Blanchon's (bblanchon) JSON library. All of these libraries can be installed directly from the IDE.

Select the menu item, **Sketch** -> **Include Library** -> **Manage Libraries**. From the Library Manager Window, enter the words **adafruit epd** in the search box and you should see the Adafruit EPD library appear. Click the box where the library appears and then click the "Install" button to install the library.

In the same manner, install the **Adafruit GFX**, **Adafruit BusIO** and **JSON** libraries by searching the words **adafruit gfx**, **busio** and **arduinojson** (all one word), respectively, to install the libraries.



Once you have the Arduino IDE setup, you are ready to install the ePaper Quote Display. You can download the source by selecting Download Project Zip in the window below:

```
// SPDX-FileCopyrightText: 2019 Dan Cogliano for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*************************************************
 * Quote Display for Adafruit ePaper FeatherWings
 * For use with Adafruit tricolor and monochrome ePaper FeatherWings
 *
 * Adafruit invests time and resources providing this open source code.
 * Please support Adafruit and open source hardware by purchasing
 * products from Adafruit!
 *
 * Written by Dan Cogliano for Adafruit Industries
 * Copyright (c) 2019 Adafruit Industries
 *
 * Notes:
 * Update the secrets.h file with your WiFi details
 * Uncomment the ePaper display type you are using below.
 * Change the SLEEP setting to define the time between quotes
 */

#include <Adafruit_GFX.h>    // Core graphics library
#include <HTTPClient.h>
#include <ArduinoJson.h>     //https://github.com/bblanchon/ArduinoJson
#include <Adafruit_EPD.h>
#include "secrets.h"

// define the # of seconds to sleep before waking up and getting a new quote
#define SLEEP 3600 // 1 hour in seconds
```

```
// WiFi timeout in seconds
#define WIFI_TIMEOUT 30

// What fonts do you want to use?
#include <Fonts/FreeSans9pt7b.h>
//#include <Fonts/FreeSans12pt7b.h> // a font option for larger screens

// qfont is the font for the quote
const GFXfont *qfont = &FreeSans9pt7b;
//const GFXfont *qfont = &FreeSans12pt7b;

// afont is the font for the author's name
const GFXfont *afont = &FreeSans9pt7b;
//const GFXfont *afont = &FreeSans12pt7b;

// ofont is the font for the origin of the feed
const GFXfont *ofont = NULL;

// font to use if qfont is too large
// NULL means use the system font (which is small)
const GFXfont *smallfont = NULL;


// ESP32 settings
#define SD_CS       14
#define SRAM_CS     32
#define EPD_CS      15
#define EPD_DC      33
#define LEDPIN      13
#define LEDPINON    HIGH
#define LEDPINOFF   LOW

#define EPD_RESET   -1 // can set to -1 and share with microcontroller Reset!
#define EPD_BUSY    -1 // can set to -1 to not use a pin (will wait a fixed delay)

// Uncomment the following line if you are using 2.13" tricolor 212*104 EPD
//Adafruit_IL0373 epd(212, 104 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);
// Uncomment the following line if you are using 2.13" monochrome 250*122 EPD
Adafruit_SSD1675 epd(250, 122, EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);

// get string length in pixels
// set text font prior to calling this
int getStringLength(const char *str, int strlength = 0)
{
  char buff[1024];
  int16_t x, y;
  uint16_t w, h;
  if(strlength == 0)
  {
    strcpy(buff, str);
  }
  else
  {
    strncpy(buff, str, strlength);
    buff[strlength] = '\0';
  }
  epd.getTextBounds(buff, 0, 0, &x, &y, &w, &h);
  return(w);
}

// word wrap routine
// first time send string to wrap
// 2nd and additional times: use empty string
// returns substring of wrapped text.
char *wrapWord(const char *str, int linesize)
{
  static char buff[1024];
  int linestart = 0;
```

```
  static int lineend = 0;
  static int bufflen = 0;
  if(strlen(str) == 0)
  {
    // additional line from original string
    linestart = lineend + 1;
    lineend = bufflen;
    Serial.println("existing string to wrap, starting at position " +
String(linestart) + ": " + String(&buff[linestart]));
  }
  else
  {
    Serial.println("new string to wrap: " + String(str));
    memset(buff,0,sizeof(buff));
    // new string to wrap
    linestart = 0;
    strcpy(buff,str);
    lineend = strlen(buff);
    bufflen = strlen(buff);
  }
  uint16_t w;
  int lastwordpos = linestart;
  int wordpos = linestart + 1;
  while(true)
  {
    while(buff[wordpos] == ' ' && wordpos < bufflen)
      wordpos++;
    while(buff[wordpos] != ' ' && wordpos < bufflen)
      wordpos++;
    if(wordpos < bufflen)
      buff[wordpos] = '\0';
    w = getStringLength(&buff[linestart]);
    if(wordpos < bufflen)
      buff[wordpos] = ' ';
    if(w > linesize)
    {
      buff[lastwordpos] = '\0';
      lineend = lastwordpos;
      return &buff[linestart];
    }
    else if(wordpos >= bufflen)
    {
      // first word too long or end of string, send it anyway
      buff[wordpos] = '\0';
      lineend = wordpos;
      return &buff[linestart];
    }
    lastwordpos = wordpos;
    wordpos++;
  }
}

// return # of lines created from word wrap
int getLineCount(const char *str, int scrwidth)
{
  int linecount = 0;
  String line = wrapWord(str,scrwidth);

  while(line.length() > 0)
  {
    linecount++;
    line = wrapWord("",scrwidth);
  }
  return linecount;
}

int getLineHeight(const GFXfont *font = NULL)
{
  int height;
```

```
  if(font == NULL)
  {
    height = 12;
  }
  else
  {
    height = (uint8_t)pgm_read_byte(&font->yAdvance);
  }
  return height;
}

// Retrieve page response from given URL
String getURLResponse(String url)
{
  HTTPClient http;
  String jsonstring = "";
  Serial.println("getting url: " + url);
  if(http.begin(url))
  {
    Serial.print("[HTTP] GET...\n");
    // start connection and send HTTP header
    int httpCode = http.GET();

    // httpCode will be negative on error
    if (httpCode > 0) {
      // HTTP header has been sent and Server response header has been handled
      Serial.println("[HTTP] GET... code: " + String(httpCode));

      // file found at server
      if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
      jsonstring = http.getString();
      // use this string for testing very long quotes
      //jsonstring = "[{\"text\":\"Don't worry about what anybody else is going to
do… The best way to predict the future is to invent it. Really smart people with
reasonable funding can do just about anything that doesn't violate too many of
Newton's Laws!\",\"author\":\"Alan Kay\"}]";
      Serial.println(jsonstring);
    }
    } else {
      Serial.println("[HTTP] GET... failed, error: " +
http.errorToString(httpCode));
    }
    http.end();
  }
  else {
    Serial.println("[HTTP] Unable to connect");
  }
  return jsonstring;
}

void getQuote(String &quote, String &author)
{
  StaticJsonDocument<1024> doc;
  String url = "https://www.adafruit.com/api/quotes.php";
  String jsonquote = getURLResponse(url);
  if(jsonquote.length() > 0)
  {
    // remove start and end brackets, jsonBuffer is confused by them
    jsonquote = jsonquote.substring(1,jsonquote.length()-1);
    Serial.println("using: " + jsonquote);
    DeserializationError error = deserializeJson(doc, jsonquote);
    if (error)
    {
      Serial.println("json parseObject() failed");
      Serial.println("bad json: " + jsonquote);
      quote = "json parseObject() failed";
    }
    else
    {
```

```
      String tquote = doc["text"];
      String tauthor = doc["author"];
      quote = tquote;
      author = tauthor;
    }
  }
  else
  {
    quote = "Error retrieving URL";
  }
}

void printQuote(String &quote)
{
  int x = 0;
  int y = 0;
  bool bsmallfont = false;
  epd.setTextColor(EPD_BLACK);
  epd.setFont(qfont);
  epd.setTextSize(1);

  int scrwidth = epd.width() - 8;
  Serial.println("Screen width is " + String(scrwidth));
  Serial.println("Screen height is " + String(epd.height()));
  int linecount = getLineCount(quote.c_str(),scrwidth);
  int lineheightquote = getLineHeight(qfont);
  int lineheightauthor = getLineHeight(afont);
  int lineheightother = getLineHeight(ofont);
  int maxlines = (epd.height() - (lineheightauthor + lineheightother)) /
lineheightquote;
  Serial.println("maxlines is " + String(maxlines));
  Serial.println("line height is " +String(lineheightquote));
  Serial.println("linecount is " +String(linecount));
  int topmargin = 0;
  if(linecount > maxlines)
  {
    // too long for default font size
    // next attempt, reduce lineheight to .8 size
    lineheightquote = .8 * lineheightquote;
    maxlines = (epd.height() - (lineheightauthor + lineheightother)) /
lineheightquote;
    if(linecount > maxlines)
    {
      // next attempt, use small font
      epd.setFont(smallfont);
      bsmallfont = true;
      epd.setTextSize(1);
      lineheightquote = getLineHeight(smallfont);
      maxlines = (epd.height() - (lineheightauthor + lineheightother)) /
lineheightquote;
      linecount = getLineCount(quote.c_str(),scrwidth);
      if(linecount > maxlines)
      {
        // final attempt, last resort is to reduce the lineheight to make it fit
        lineheightquote = (epd.height() - (lineheightauthor + lineheightother)) /
linecount;
      }
    }
    Serial.println("maxlines has changed to " + String(maxlines));
    Serial.println("line height has changed to " +String(lineheightquote));
    Serial.println("linecount has changed to " +String(linecount));
  }
  if(linecount <= maxlines)
  {

    topmargin = (epd.height() - (lineheightauthor + lineheightother) -
linecount*lineheightquote)/2;
    if(!bsmallfont)
      topmargin+=lineheightquote-4;
```

```
    //Serial.println("topmargin = " + String(topmargin));
  }
  String line = wrapWord(quote.c_str(),scrwidth);

  int counter = 0;
  epd.setTextColor(EPD_BLACK);
  while(line.length() > 0)
  {
    counter++;
    Serial.println("printing line " + String(counter) + ": '" + line + String("'"));
    epd.setCursor(x +4, y + topmargin);
    epd.print(line);
    y += lineheightquote;
    line = wrapWord("",scrwidth);
  }
}

void printAuthor(String author)
{
  epd.setTextColor(EPD_BLACK);
  epd.setFont(afont);
  int lineheightauthor = getLineHeight(afont);
  int lineheightother = getLineHeight(ofont);
  int x = getStringLength(author.c_str());
  // draw line above author
  epd.drawLine(epd.width() - x - 10, epd.height() - (lineheightauthor +
lineheightother) + 2, epd.width(), epd.height() - (lineheightauthor +
lineheightother) + 2, EPD_RED);
  epd.drawLine(epd.width() - x - 10, epd.height() - (lineheightauthor +
lineheightother) + 2, epd.width() - x - 10,epd.height() - lineheightother -
lineheightauthor/3, EPD_RED);
  epd.drawLine(0, epd.height() - lineheightother - lineheightauthor/3, epd.width() -
x - 10,epd.height() - lineheightother - lineheightauthor/3, EPD_RED);
  // draw author text
  int cursorx = epd.width() - x - 4;
  int cursory = epd.height() - lineheightother - 2;
  if(afont == NULL)
  {
    cursory = epd.height() - lineheightother - lineheightauthor - 2 ;
  }
  epd.setCursor(cursorx, cursory);
  epd.print(author);
}

void printOther(String other)
{
  epd.setFont(ofont);
  int lineheightother = getLineHeight(ofont);
  int ypos = epd.height() - 2;
  if (ofont == NULL)
    ypos = epd.height() - (lineheightother - 2);
  epd.setTextColor(EPD_BLACK);
  epd.setCursor(4,ypos);
  epd.print(other);
}

void setup() {
  Serial.begin(115200);
  //while(!Serial);

  pinMode(LEDPIN, OUTPUT);
  digitalWrite(LEDPIN, LEDPINON);

  epd.begin();
  Serial.println("ePaper display initialized");
  epd.clearBuffer();
  epd.setTextWrap(false);

  Serial.print("Connecting to WiFi ");
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  int counter = 0;
  while (WiFi.status() != WL_CONNECTED && counter < WIFI_TIMEOUT) {
    delay(1000);
    Serial.print(".");
    counter++;
  }

  String quote, author;
  if(WiFi.status() == WL_CONNECTED)
  {
    Serial.println("connected");
    getQuote(quote, author);
  }
  else
  {
    quote = "WiFi connection timed out, try again later";
    Serial.println(quote);
  }

  printQuote(String("\"") + quote + String("\""));
  printAuthor(author);
  printOther("adafruit.com/quotes");

  epd.display();
  Serial.println("done, going to sleep...");
  // power down ePaper display
  epd.powerDown();
  // put microcontroller to sleep, wake up after specified time
  ESP.deepSleep(SLEEP * 1e6);
}

void loop() {
  // should never get here, setup() puts the CPU to sleep
}
```

There are two code files in this project. The **secrets.h** file must be modified by you to specify the login and password for the Wi-Fi connection from the HUZZAH32. Modify it to match your WiFi connection.

```
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#ifndef _SECRET_H
#define _SECRET_H

// define your WIFI SSID and password in this file

#define WIFI_SSID "your_SSID"
#define WIFI_PASSWORD "your_password"

#endif
```

The file **adafruit_feather_quote.ino** is the primary sketch for displaying the quotes. There are a couple items here that need be modified in this file. First, select the correct ePaper device you are using. Look for the lines beginning with `Adafruit_` near the top of the file and remove the leading double slashes for the device you are using, and put the double quotes in front of the line for the device you are not using.

For example, if you are using the monochrome ePaper display, the lines would look like this:

```
/* Uncomment the following line if you are using 2.13" tricolor EPD */
//Adafruit_IL0373 epd(212, 104 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);
/* Uncomment the following line if you are using 2.13" monochrome 250*122 EPD */
Adafruit_SSD1675 epd(250, 122, EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);
```

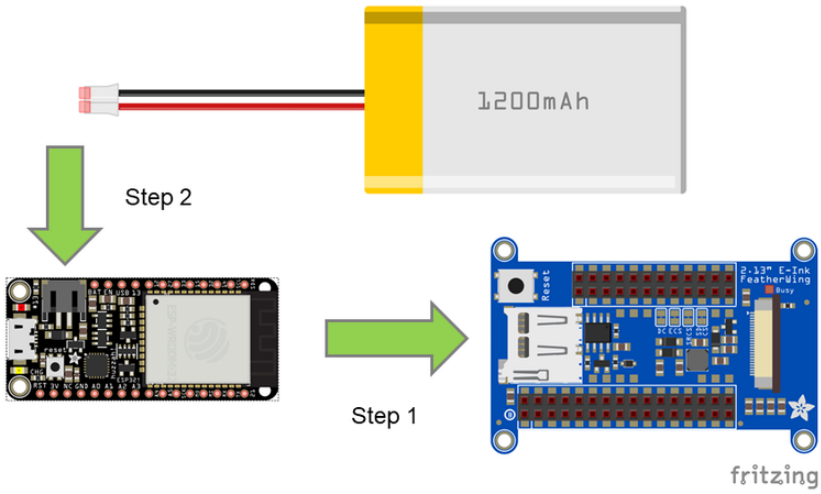
Make sure you are using the correct epd() item for the display you are using and comment out the other one by prefixing the other line with a double slash, "//",.

There is also a sleep timer setting at the top of the file that tells the hardware how long to sleep before waking up again to update the display. This value is in seconds and the default sleep setting is one hour of 3600 seconds (60 minutes x 60 seconds). You can always wake up the HUZZAH32 manually by pressing the reset button on the FeatherWing.

```
#define SLEEP 3600 // 1 hour in seconds
```

For larger ePaper displays, you may want to use a larger font. The default font was chosen for small ePaper displays, but larger displays can use a larger font. One font option is included in the sketch that can be uncommented. A full list of fonts (https://adafru.it/kAf) to choose from with the Adafruit GFX library can be found in the Adafruit GFX Learning guide (https://adafru.it/DtY).

Connections



Assembly of the parts could not be simpler:

First, plug the Feather board into the back of the e-ink display.

Next, plug the battery into the JST connector on the side of the Feather board.

Done!

How it Works

The quotes come from the Adafruit web site (https://adafru.it/ExV) using the HUZZAH32's WiFi connection. These are the same quotes that appear at the bottom of the packing list you receive when you order an item from Adafruit. There is a different quote each time a quote is requested, and new quotes may be added in the future without any changes needed to the sketch.

The sketch receives the quote and its author in JSON format using an Adafruit API. The JSON format is a standard method for formatting data when transmitting structured data between two points.

Once the data arrives, it is up to the sketch to format the quote and author on the ePaper device. This includes word wrapping the quote into multiple lines. This can get tricky with smaller ePaper devices, which can run out of room for longer quotes. The Quote Display will reduce the font size and space between the lines to help make the quote fit on the display. This quote fitting works for different size displays, including the monochrome and the tri-color ePaper FeatherWings, which are different sizes by a few thousand pixels.

Once the ePaper display is updated with the new quote and author, the device powers down the FeatherWing and puts the HUZZAH32 in a deep sleep. An interval is specified to indicate when to wake up the HUZZAH32 the next time to update its display. Even in this low power mode, the ePaper display continues to show the quote, making it a very battery friendly display.

Going Further

What new project will you work on next? Perhaps the ePaper quote display will inspire you or spark your imagination for your next project (after this one of course!).