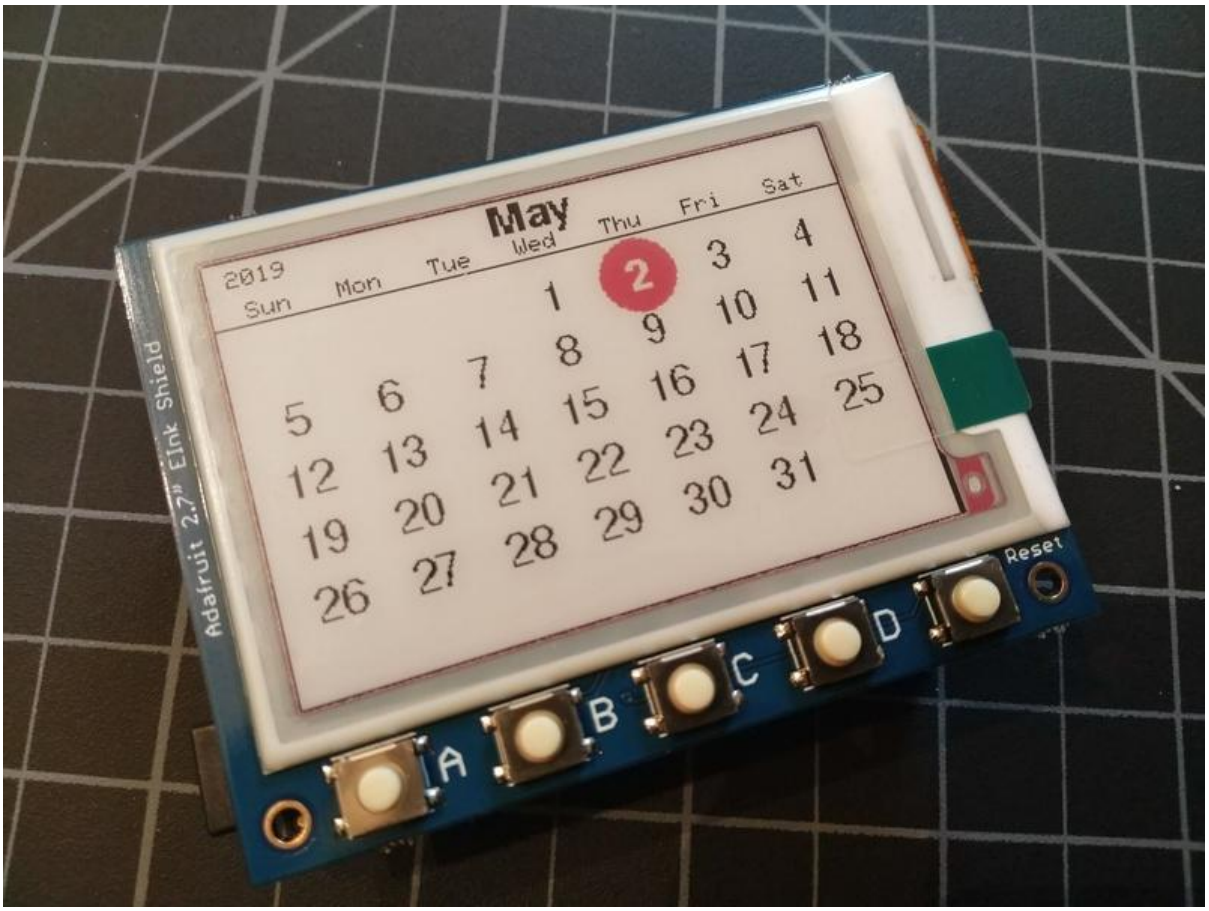
ePaper Calendar Featuring Metro M4 Express Airlift and Tri-Color ePaper Shield

Created by Dan Cogliano

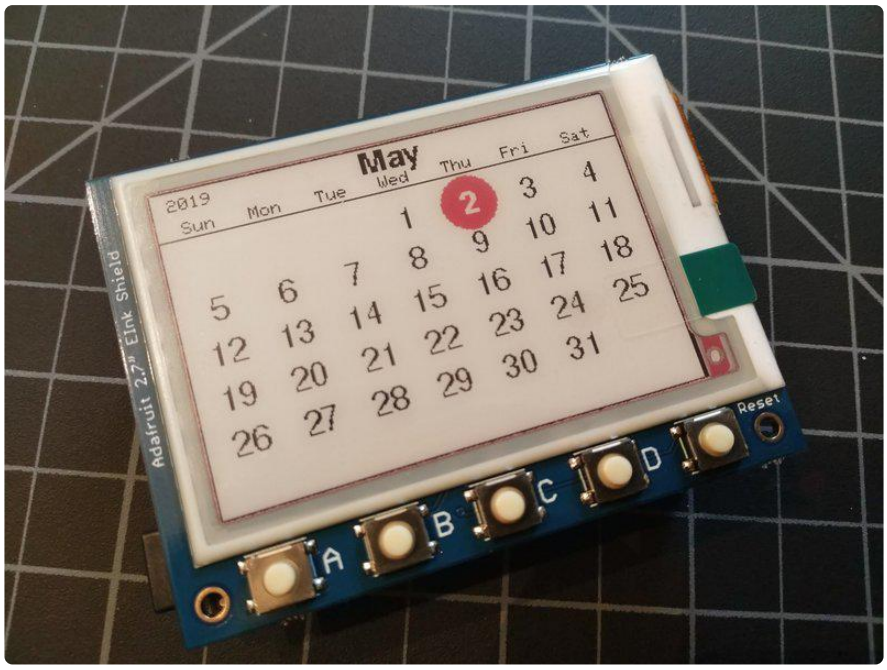


https://learn.adafruit.com/epaper-calendar-featuring-metro-m4-express-airlift-and-epaper-shield

Last updated on 2023-08-29 04:08:20 PM EDT

Table of Contents

Overview



Build this calendar display using the Adafruit Metro M4 Express Airlift and the Tri-Color ePaper Shield.

A wonderful thing happened when the Metro M4 Express Airlift was released. It opened Arduino sketches to the Internet, freeing them from the confines of their closed environment. It is now officially a "thing" in a world of the Internet of Things (IoT). For this project, the sketch utilizes the built-in Airlift coprocessor board of the Metro M4 Express Airlift to grab the current date and time from Adafruit.io. No Real Time Clock (RTC) is needed here, since Adafruit.io will happily give us the current date and time, and in our local timezone too.

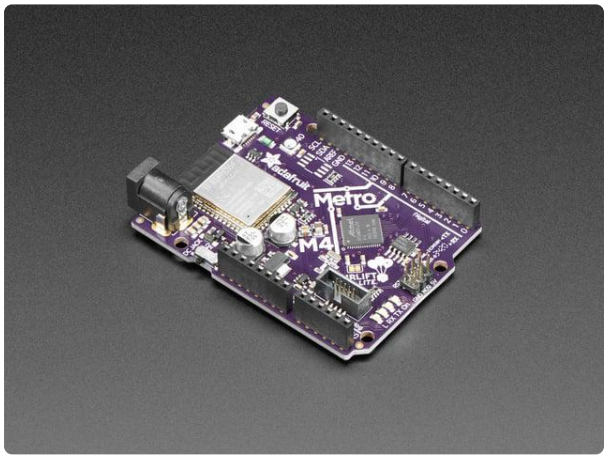
The Adafruit Tri-Color ePaper Shield used with this project is a 2.7" ePaper shield that displays black, white and red pixels. It easily connects to the Metro with no soldering needed since the headers are already assembled on both the Metro and the ePaper shield. The sketch displays the current month using the date and time pulled from Adafruit.io. The ePaper display keeps its display even when power is removed. Once a monthly calendar is displayed, you could unplug it and the display will continue to show the calendar.

Parts

Building this project requires no soldering and uses just two parts: the Adafruit Metro M4 Express AirLift Lite and the Adafruit 2.7" Tri-Color eInk / ePaper Shield with SRAM. To make this project portable, you could add a USB battery pack () or the Adafruit

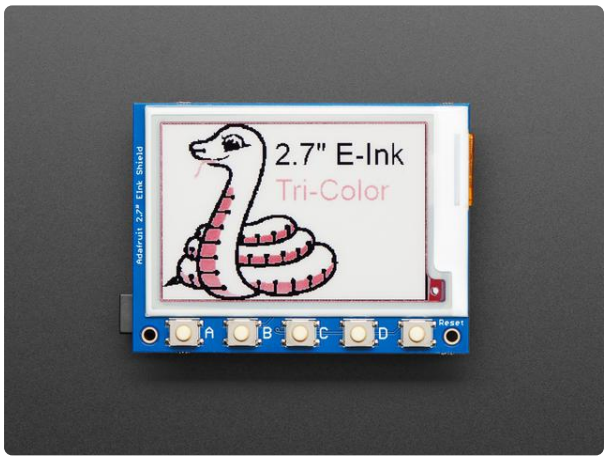
PowerBoost 500 Shield () and a Li-Po battery and insert it between the Metro and the ePaper shield.

If you are interested in ePaper displays for other projects, check out the entire line of Adafruit's ePaper displays ().



Adafruit Metro M4 Express AirLift (WiFi) - Lite

Give your next project a lift with AirLift - our witty name for the ESP32 co-processor that graces this Metro M4. You already know about the Adafruit Metro... https://www.adafruit.com/product/4000



Adafruit 2.7" Tri-Color eInk / ePaper Shield with SRAM

Easy e-paper finally comes to microcontrollers, 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/4229

USB cable - USB A to Micro-B

This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or... https://www.adafruit.com/product/592

Arduino Code

If you don't have it already, you will need the Arduino IDE installed on your computer to upload this sketch to the Metro M4 Express Airlift. You will find information on installing Arduino in this learning guide ().

Installing The Libraries

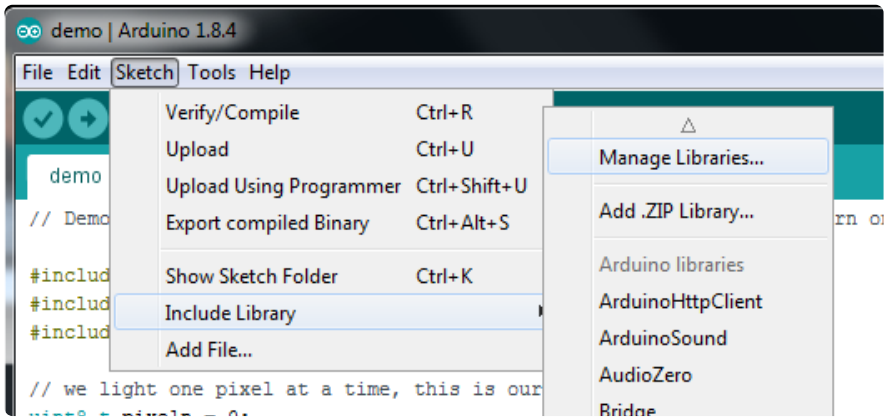
You will need to install these libraries, which are needed by the sketch:

- Adafruit EPD (ePaper display) library
- Adafruit GFX library
- Adafruit BusIO
- Adafruit NeoPixel library
- Adafruit variant of the WiFiNINA library

You can manually install the libraries needed for this sketch using the links below. The sketch uses a variant of the Arduino WiFiNINA library developed by Adafruit in order to support the Airlift coprocessor. Make sure you use this version of the library with your sketch, which is included in the links below.

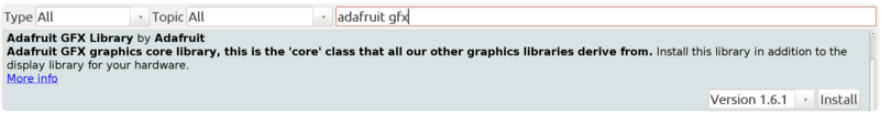
You will also need to setup an Adafruit IO account if you do not have one already. This learning guide about Airlift and Adafruit IO () will help you get started with Adafruit IO. You will need a user name and key in order for the sketch to retrieve the current date and time from Adafruit IO.

Open up the Arduino library manager:



Search for the Adafruit epd library and install it

Search for the Adafruit gfx library and install it



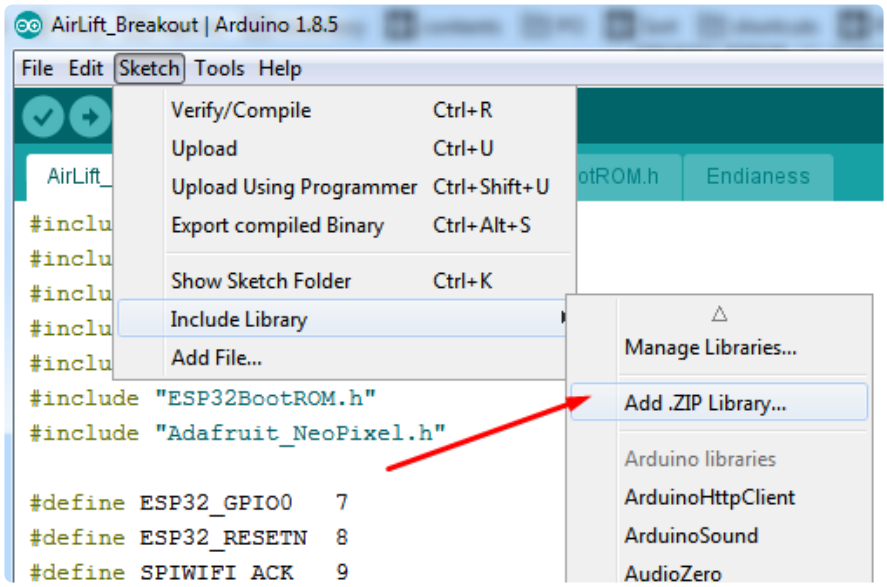
If using an earlier version of the Arduino IDE (prior to 1.8.10), also locate and install Ad afruit_BusIO (newer versions will install this dependency automatically).

Search for the Adafruit neopixel library and install it

To install the Adafruit WiFiNINA library, click the link below to download the library as a zip file:

Download Adafruit's version of WiFiNina

Within the Arduino IDE, select Install library from ZIP...



And select the zip files you just downloaded.

Installing the Sketch

There are two files used by this sketch: the secrets.h file and the main project sketch file. The secrets.h file contains the WiFi connection credentials as well as the Adafruit IO account details. This information must be entered before running the sketch.

You can download a skeleton of a secrets.h file from the window below, just fill in the details for the WiFi access point and Adafruit IO account.

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

#ifndef _SECRETS_H THEN
#define _SECRETS_H

// define your WIFI and Adadfruit IO credentials in this file

#define WIFI_SSID "your_SSID"
#define WIFI_PASSWORD "your_password"
#define AIO_USERNAME "your_adafruit_io_username"
#define AIO_KEY "your_adafruit_io_key"

#endif
```

The main project sketch is named adafruit_airlift_calendar.ino. A customization that you can do with the calendar is choose how to highlight the current day in the calendar. You have the option of showing the current day as a red circle, black circle, bold text or no highlighting, which displays the current day just like any other day. Uncomment the option you will want to use.

```
enum dayhighlight {RedCircle, BlackCircle, Bold, None};

// pick one of these options for displaying the current day in the current month

dayhighlight currentday = RedCircle;
//dayhighlight currentday = BlackCircle;
//dayhighlight currentday = Bold;
//dayhighlight currentday = None;
```

The sketch is available for download below.

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

/*************************************************
 * ePaper Tri Color Calendar Demo
 * For use with Adafruit Metro M4 Express Airlift and tricolor e-Paper Display
Shield
 *
 * Adafruit invests time and resources providing this open source code.
 * Please support Adafruit and open source hardware by purchasing
 * products from Adafruit.com!
 *
```

```
 * Written by Dan Cogliano for Adafruit Industries
 * Copyright (c) 2019 Adafruit Industries
 *
 * Notes:
 * Update the secrets.h file with your WiFi details and Adafruit IO credentials
 */
#include <time.h>
#include "secrets.h"
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_EPD.h>
#include <Adafruit_NeoPixel.h>

#include <Fonts/FreeSans9pt7b.h>
#include <Fonts/FreeSansBold9pt7b.h>

#include <SPI.h>
#include <WiFiNINA.h>

enum dayhighlight {RedCircle, BlackCircle, Bold, None};

// pick one of these options for displaying the current day in the current month

dayhighlight currentday = RedCircle;
//dayhighlight currentday = BlackCircle;
//dayhighlight currentday = Bold;
//dayhighlight currentday = None;

// Configure the pins used for the ESP32 connection
#if !defined(SPIWIFI_SS)  // if the wifi definition isnt in the board variant
  // Don't change the names of these #define's! they match the variant ones
  #define SPIWIFI     SPI
  #define SPIWIFI_SS    10  // Chip select pin
  #define SPIWIFI_ACK   7   // a.k.a BUSY or READY pin
  #define ESP32_RESETN  5   // Reset pin
  #define ESP32_GPIO0   -1  // Not connected
#endif

const char *wifi_ssid = WIFI_SSID;
const char *wifi_password = WIFI_PASSWORD;
const char *aio_username = AIO_USERNAME;
const char *aio_key = AIO_KEY;

#define SRAM_CS     8
#define EPD_CS      10
#define EPD_DC      9
#define EPD_RESET -1
#define EPD_BUSY -1

#define NEOPIXELPIN   40

/* This isfor the 2.7" tricolor EPD */
Adafruit_IL91874 gfx(264, 176 ,EPD_DC, EPD_RESET, EPD_CS, SRAM_CS, EPD_BUSY);

WiFiSSLClient client;

Adafruit_NeoPixel neopixel = Adafruit_NeoPixel(1, NEOPIXELPIN, NEO_GRB +
NEO_KHZ800);


struct tm *today;
struct tm *pickdate = new struct tm;

int8_t readButtons(void) {
  uint16_t reading = analogRead(A3);
  //Serial.println(reading);

  if (reading > 600) {
    return 0; // no buttons pressed
  }
```

```
  if (reading > 400) {
    return 4; // button D pressed
  }
  if (reading > 250) {
    return 3; // button C pressed
  }
  if (reading > 125) {
    return 2; // button B pressed
  }
  return 1; // Button A pressed
}

bool isLeapYear(int year) {
  if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
    return true;
  return false;
}

int getDaysInMonth(int month, int year) {
  int daysInMonth[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
  if (month != 2)
    return daysInMonth[(month-1)%12];
  if(isLeapYear(year))
    return 29;
  return 28;
}

int getDayOfWeek(int year, int month, int day)
{
  uint16_t months[] = {
    0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365         };   // days
until 1st of month

  uint32_t days = year * 365;        // days until year
  for (uint16_t i = 4; i < year; i += 4) if (isLeapYear(i) ) days++;     // adjust
leap years, test only multiple of 4 of course

  days += months[month-1] + day;    // add the days of this year
  if ((month > 2) && isLeapYear(year)) days++;  // adjust 1 if this year is a leap
year, but only after febr

  // make Sunday 0
  days--;
  if(days < 0)
    days+= 7;
  return days % 7;   // remove all multiples of 7
}

void drawCalendar(struct tm * today, struct tm * pickdate)
{
  char *dows[7] =
{"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};
  char *months[12] =
{"January","February","March","April","May","June","July","August","September","October","Nove

  pickdate->tm_wday = getDayOfWeek(pickdate->tm_year,pickdate->tm_mon,pickdate-
>tm_mday);

  Serial.println("drawing calendar for " + String(months[pickdate->tm_mon-1]) + " "
+ String(pickdate->tm_year));
  neopixel.setPixelColor(0, neopixel.Color(0, 255, 0));
  neopixel.show();

  // draw calendar
  String stryear = String(pickdate->tm_year);
  gfx.powerUp();
  gfx.clearBuffer();
  //gfx.setFont(&FreeSans9pt7b);
  gfx.setFont();
```

```
gfx.setTextColor(EPD_BLACK);
gfx.setCursor(8,4);
gfx.print(stryear);

String strmonth = months[pickdate->tm_mon-1];

int daysinmonth = getDaysInMonth(pickdate->tm_mon,pickdate->tm_year);
gfx.setTextColor(EPD_BLACK);
gfx.setFont(&FreeSansBold9pt7b);
int16_t fx, fy;
uint16_t w, h;
gfx.getTextBounds((char *)strmonth.c_str(), 0,0, &fx, &fy, &w, &h);

gfx.setCursor((gfx.width() - w)/2,14);
gfx.print(strmonth);

int curday = pickdate->tm_mday - pickdate->tm_wday;
while(curday > 1)
  curday -= 7;

int x = 0;
int y = 26;
for(int i = 0 ; i < 7; i++)
{
  x = 4 + i * (gfx.width()-8)/7;
  gfx.setTextColor(EPD_BLACK);
  gfx.setFont();
  gfx.getTextBounds(String(dows[i]).substring(0,3), 0,0, &fx, &fy, &w, &h);
  gfx.setCursor(x + (gfx.width()/7-w)/2, y-8);
  gfx.print(String(dows[i]).substring(0,3));
}
gfx.drawLine(0,27,gfx.width(),27,EPD_BLACK);
y = 45;
while(curday <= daysinmonth)
{
  for(int i = 0; i < 7; i++)
  {
    x = 4 + i * (gfx.width()-8)/7;
    if(curday >= 1 && curday <= daysinmonth)
    {
        gfx.setCursor(x,y);
        gfx.setTextColor(EPD_BLACK);
        gfx.setFont(&FreeSans9pt7b);
        int16_t fx, fy;
        uint16_t w, h;
        String strday = String(curday);
        if((today->tm_year == pickdate->tm_year) &&
          (today->tm_mon == pickdate->tm_mon) &&
          (curday == pickdate->tm_mday))
        {
          if(currentday != None)
          {
            gfx.setFont(&FreeSansBold9pt7b);
          }
          if(currentday == BlackCircle)
          {
            gfx.setTextColor(EPD_INVERSE);
            gfx.fillCircle(x + (gfx.width()-8)/7/2, y, 16, EPD_BLACK);
          }
          else if(currentday == RedCircle)
          {
            gfx.setTextColor(EPD_RED);
            gfx.fillCircle(x + (gfx.width()-8)/7/2, y, 16, EPD_RED);
          }
        }
        gfx.getTextBounds(strday.c_str(), 0,0, &fx, &fy, &w, &h);
        gfx.setCursor(x+(gfx.width()-8)/7/2-w/2-fx,y+h/2);
        gfx.setColorBuffer(1, true);  // red is inverted
        gfx.print(curday);
```

```
          gfx.setColorBuffer(1, false);  // red is not inverted
      }
      curday++;
    }
    y += 24;
  }
  gfx.display();
  Serial.println("display update completed");
  gfx.powerDown();
  neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
  neopixel.show();
}

#define SERVER "io.adafruit.com"
#define PATH "/api/v2/%s/integrations/time/strftime?x-aio-key=%s"
// our strftime is %Y-%m-%d %H:%M:%S.%L %j %u %z %Z see http://strftime.net/ for
decoding details
// See https://apidock.com/ruby/DateTime/strftime for full options
#define TIME_SERVICE_STRFTIME "&fmt=%25Y-%25m-%25d+%25H%3A%25M%3A%25S.%25L+%25j+
%25u+%25z+%25Z"

struct tm *getDate(bool force = false)
{
  static tm date;
  char buff[500];
  char pathbuff[500];
  sprintf(pathbuff, PATH,aio_username, aio_key);
  String path = String(pathbuff) + String(TIME_SERVICE_STRFTIME);
  //Serial.println(String("path to check: " ) + String(SERVER) + path);
  wget(SERVER, path.c_str(), 443, buff);
  Serial.println("wget got: " + String(buff));
  String datestr = String(buff);
  date.tm_year = atoi(datestr.substring(0,4).c_str());
  date.tm_mon = atoi(datestr.substring(5,7).c_str());
  date.tm_mday = atoi(datestr.substring(8,10).c_str());
  date.tm_wday = atoi(datestr.substring(28,29).c_str());
  date.tm_hour = atoi(datestr.substring(11,13).c_str());
  date.tm_min = atoi(datestr.substring(14,16).c_str());
  date.tm_sec = atoi(datestr.substring(17,19).c_str());
  return &date;
}

void *wget(const char *host, const char *path, int port, char *buff)
{
  neopixel.setPixelColor(0, neopixel.Color(0, 0, 255));
  neopixel.show();
  if (client.connect(host, port)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    // Using HTTP/1.1 to avoid "Transfer-Encoding: chunked" reply
    client.println(String("GET ") + path + String(" HTTP/1.0"));
    client.println("Host: " + String(host));
    client.println("Connection: close");
    client.println();

    uint32_t bytes = 0;
    int capturepos = 0;
    bool capture = false;
    int linelength = 0;
    char lastc = '\0';
    while(true)
    {
      while (client.available()) {
        char c = client.read();
        if((c == '\n') && (lastc == '\r'))
        {
          if(linelength == 0)
          {
            capture = true;
```

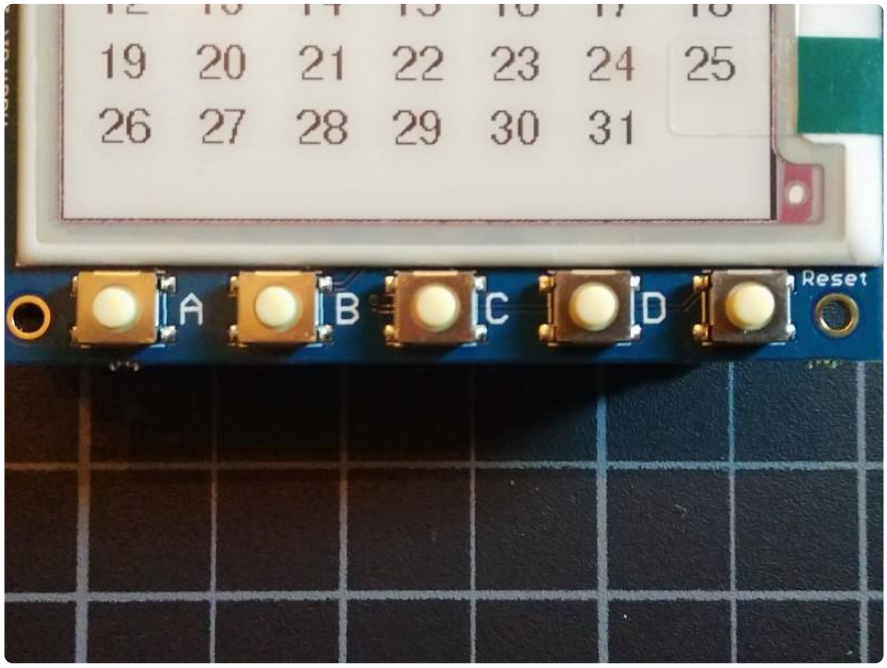
```
          }
          linelength = 0;
        }
        else if(capture)
        {
          buff[capturepos++] = c;
          //Serial.write(c);
        }
        else
        {
          if((c != '\n') && (c != '\r'))
            linelength++;
        }
        lastc = c;
        bytes++;
      }

      // if the server's disconnected, stop the client:
      if (!client.connected()) {
        //Serial.println();
        Serial.println("disconnecting from server.");
        client.stop();
        buff[capturepos] = '\0';
        Serial.println("read " + String(bytes) + " bytes");
        Serial.println("captured " + String(capturepos) + " bytes");
        break;
      }
    }
  }
  neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
  neopixel.show();
}

void setup() {
  Serial.begin(115200);
  //while(!Serial);
  delay(2000);
  Serial.println("Adafruit Airlift ePaper Calendar");

  neopixel.begin();
  neopixel.show();

  Serial.print("Connecting to WiFi ");

  WiFi.setPins(SPIWIFI_SS, SPIWIFI_ACK, ESP32_RESETN, ESP32_GPIO0, &SPIWIFI);

  // check for the WiFi module:
  while (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // slow red blink
    while(1)
    {
      neopixel.setPixelColor(0, neopixel.Color(255, 0, 0));
      neopixel.show();
      delay(900);
      neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
      neopixel.show();
      delay(100);
    }
  }

  String fv = WiFi.firmwareVersion();
  if (fv < "1.0.0") {
    Serial.println("Please upgrade the firmware");
  }

  neopixel.setPixelColor(0, neopixel.Color(0, 0, 255));
  neopixel.show();
  if(WiFi.begin(wifi_ssid, wifi_password) == WL_CONNECT_FAILED)
```

```
  {
    Serial.println("WiFi connection failed!");
    // fast red blink
    while(1)
    {
      neopixel.setPixelColor(0, neopixel.Color(255, 0, 0));
      neopixel.show();
      delay(400);
      neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
      neopixel.show();
      delay(100);
    }
  }

  int wifitimeout = 15;
  while (WiFi.status() != WL_CONNECTED && wifitimeout > 0) {
    delay(900);
    Serial.print(".");
    neopixel.setPixelColor(0, neopixel.Color(0, 0, 80));
    neopixel.show();
    delay(100);
    wifitimeout--;
  }
  if(wifitimeout == 0)
  {
    Serial.println("WiFi connection failed!");
    // fast red blink
    while(1)
    {
      neopixel.setPixelColor(0, neopixel.Color(255, 0, 0));
      neopixel.show();
      delay(400);
      neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
      neopixel.show();
      delay(100);
    }

  }
  neopixel.setPixelColor(0, neopixel.Color(0, 0, 0));
  neopixel.show();
  Serial.println("Connected to wifi");
  gfx.begin();
  Serial.println("ePaper display initialized");
  gfx.clearBuffer();
  gfx.setRotation(2);

  today = getDate();
  Serial.println("today is " + String(today->tm_mon) + "/" + String(today->tm_mday)
+ "/" + String(today->tm_year));
  pickdate->tm_year = today->tm_year;
  pickdate->tm_mon = today->tm_mon;
  pickdate->tm_mday = today->tm_mday;
  pickdate->tm_hour = today->tm_hour;
  pickdate->tm_min = today->tm_min;
  pickdate->tm_sec = today->tm_sec;
  drawCalendar(today,pickdate);

}

void loop() {
  static uint32_t timer = millis();
  static uint8_t lastbutton = 2;
  static uint8_t direction = 1;
  if(millis() > (timer + 1000*60*60))
  {
    timer = millis();
    // update date once an hour
    today = getDate();
    if(lastbutton == 2)
```

```
  {
    // refresh calendar of current month in case date has since changed.
    Serial.println("Refreshing calendar display");
    today = getDate();
    pickdate->tm_year = today->tm_year;
    pickdate->tm_mon = today->tm_mon;
    pickdate->tm_mday = today->tm_mday;
    drawCalendar(today,pickdate);
  }
}
int button = readButtons();
if (button == 0) {
  return;
}
Serial.print("Button "); Serial.print(button); Serial.println(" pressed");
if (button == 1) {
  Serial.println("Previous month");
  direction = -1;
  pickdate->tm_mday = 1;
  pickdate->tm_mon--;
  if(pickdate->tm_mon < 1)
  {
    pickdate->tm_mon = 12;
    pickdate->tm_year--;
  }
  drawCalendar(today,pickdate);
  lastbutton = 1;
}

if (button == 2) {
  Serial.println("Current month");
  direction = 1;
  pickdate->tm_year = today->tm_year;
  pickdate->tm_mon = today->tm_mon;
  pickdate->tm_mday = today->tm_mday;
  pickdate->tm_hour = today->tm_hour;
  pickdate->tm_min = today->tm_min;
  pickdate->tm_sec = today->tm_sec;
  drawCalendar(today,pickdate);
  lastbutton = 2;
}

if (button == 3) {
  Serial.println("Next month");
  direction = 1;
  pickdate->tm_mday = 1;
  pickdate->tm_mon++;
  if(pickdate->tm_mon > 12)
  {
    pickdate->tm_mon = 1;
    pickdate->tm_year++;
  }
  drawCalendar(today,pickdate);
  lastbutton = 3;
}

if (button == 4) {
  if(direction >= 0)
      Serial.println("Next year");
  else
    Serial.println("Previous year");

  // next or previous year, depending on previous button press
  pickdate->tm_year = pickdate->tm_year + direction;
  drawCalendar(today,pickdate);
  lastbutton = 3;
}

// wait until button is released
```

```
  while (readButtons()) {
    delay(10);
  }

}
```

Use



The shield includes 4 programmable buttons, labelled "A" to "D". These buttons are used to change the month of the calendar. When first booting up the device or when pressing the reset button, the display shows the current month with the current day highlighted with a red circle. Pressing buttons "A" or "C" will move to the previous or next months, respectively. Pressing button "B" will switch back to the current month.

The "D" button is used to move the calendar either forward or backward by one year. By default, the button will move to the same month in the next calendar year. However, if the previous month button "A" is pressed, then button "D" will instead move to the previous calendar year. If button "B" or "C" is pressed for the current or next month, then button "D" will again move to the next calendar year. Press button "D" successive times to move forward or backward multiple years.

Status LED

The Metro M4 Express Airlift also comes with a single NeoPixel. This project uses the NeoPixel as a project status indicator.

A blue NeoPixel status means the sketch is currently accessing the Internet, either connecting to the WiFi hotspot or accessing Adafruit.io to grab the date and time.

A green NeoPixel status means the sketch is currently updating the display for the specified month and will turn off when it is completed. This can take a few seconds since ePaper displays are not very speedy at refreshing their screens.

A red NeoPixel status means there is a network communication issue. Pressing the buttons will have no effect if the NeoPixel is showing one of these status colors. Wait for the NeoPixel to turn off before pressing one of the buttons to change the month display.