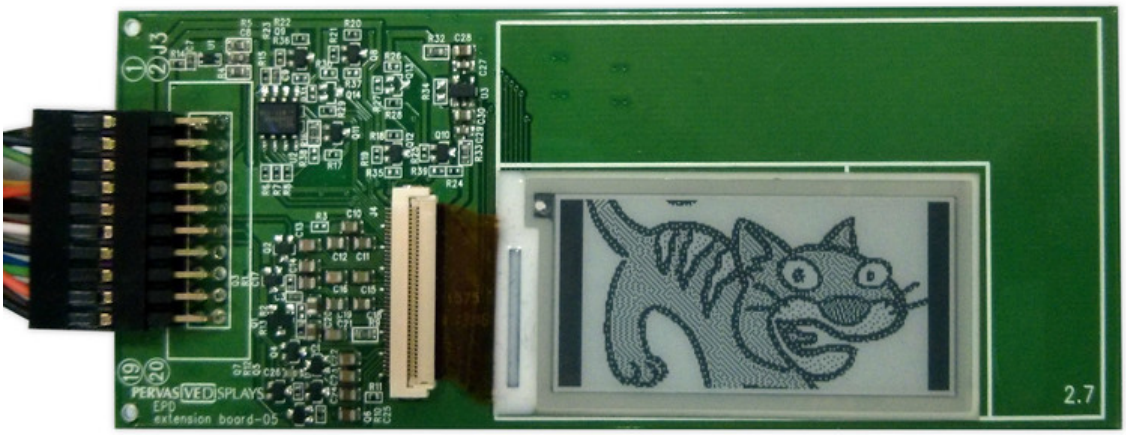
RePaper eInk Development Board for ARM + GNU/Linux

Created by Sean Moss-Pultz



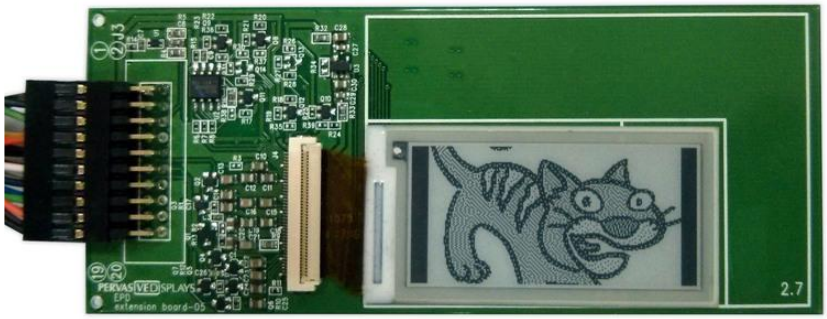
https://learn.adafruit.com/repaper-eink-development-board-arm-linux-raspberry-pi-beagle-bone-black

Last updated on 2024-06-03 01:25:24 PM EDT

Table of Contents

Overview

The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts



Now you can add eInk displays to your favorite single-board Linux computers! This tutorial will show you how to wire and run the RePaper eInk displays in the Adafruit shop on a Raspberry Pi or BBB

The RePaper development boards from Pervasive Displays come with a driver board that is powered from 3V and has level shifting on all the I/O pins so it can be used with 5V microcontrollers such as the Arduino. The PCB also has a lot of driver circuitry required to keep the display running smoothly such a temperature sensor, FLASH memory and ZIF socket. All signals are broken out to a 20 male socket header on the left. A 20 pin socket/socket cable is included to make wiring easier and there's also some extra-long header so you can plug these wires into a header or a breadboard.

The displays are available in 1.44", 2" and 2.7" diagonal sizes with resolutions of 128x96, 200x96 and 264x176 pixels. These are intended for use as small dynamic signage in grocery stores since a barcode displayed on it can be scanned by a laser barcode-reader. The display does not require any power to keep the image and will stay 'on' without any power connection for many days before slowly fading. Of course, its also daylight readable and is very high contrast. This makes it excellent for data-logging applications, outdoor displays, or any other ultra-low power usages

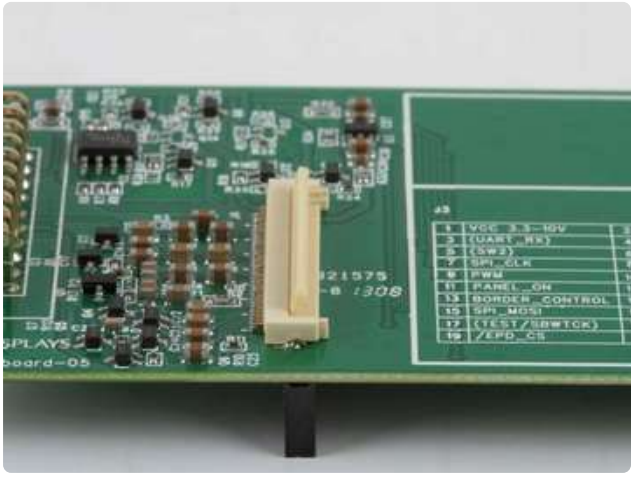
RePaper/PDI have provided a suite of example code for Raspberry Pi and BeagleBone Black.

Assembly and Wiring

The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts

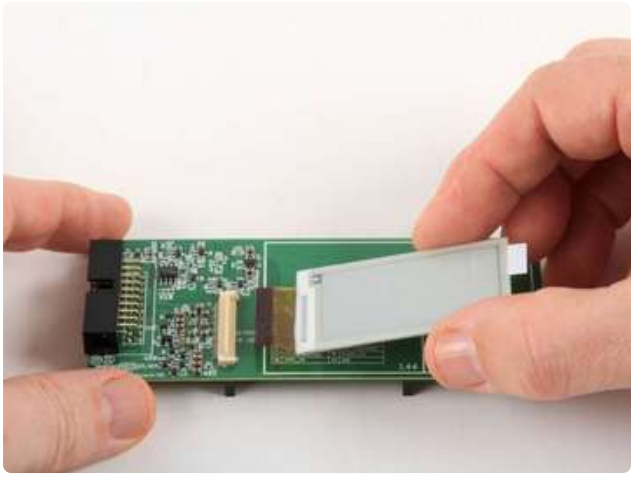
The eInk development board comes completely assembled. All you need to do is attach the eInk display and the breakout cable.

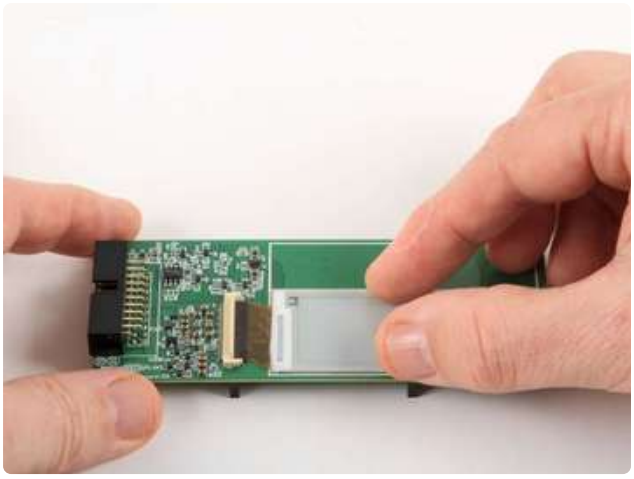
Attach the Display

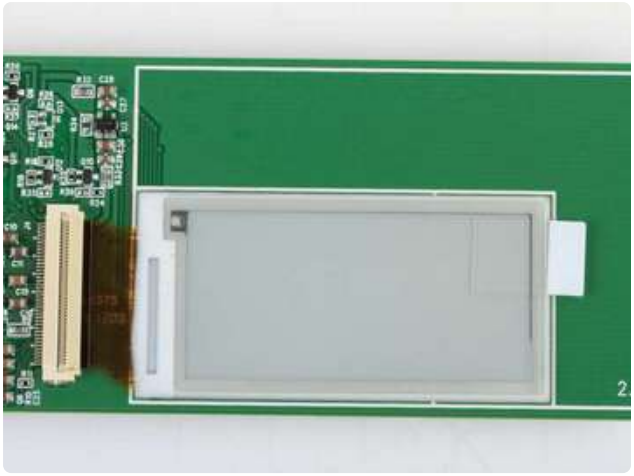


Open the Socket

Lift the front edge of the flex connector socket and raise it to the vertical position as shown on the left.



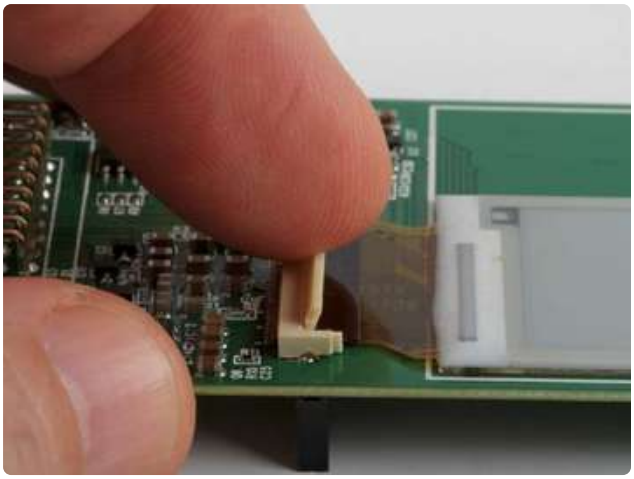


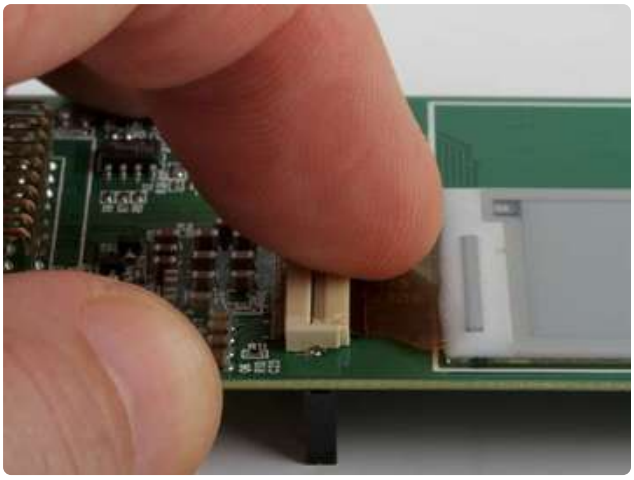


Insert the Flex Connector

Gently slide the flex connector (contact side down) into the socket.

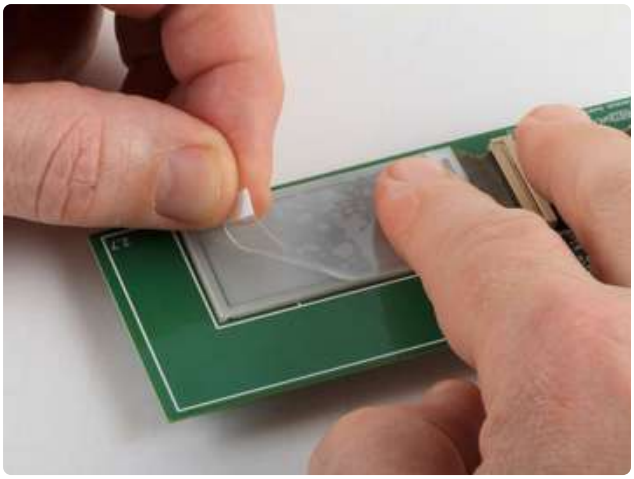
Make sure that the display is aligned with the outline on the silkscreen.

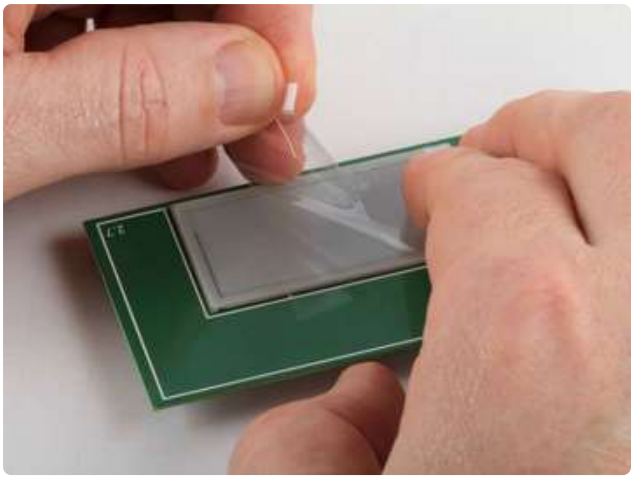




Close the Socket

Gently push the socket closed to lock the flex connector in place.





Remove the protective film

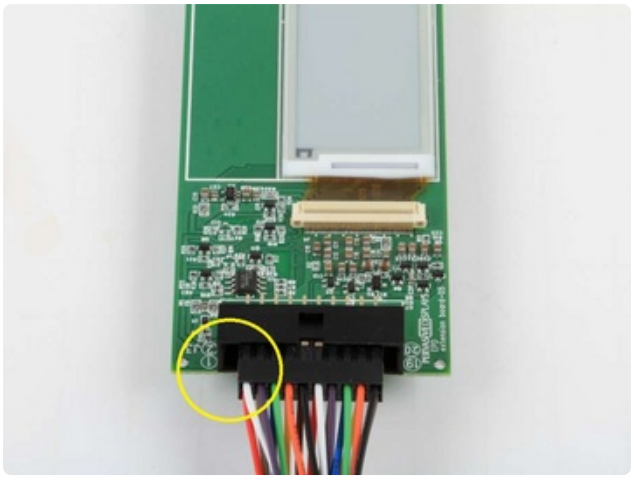
Lift the white tab and gently peel back the protective film from the display.

Use care when handling the assembled development board. The display is attached only by the flex connector, and this can be damaged by excessive force. If desired, a small piece of tape can be used to stabilize the display on the board.

Attach the Breakout Cable

Locate Pin 1

Pin 1 is marked with a white dot on the connector. There will be a red wire connected to that pin.



Insert the cable

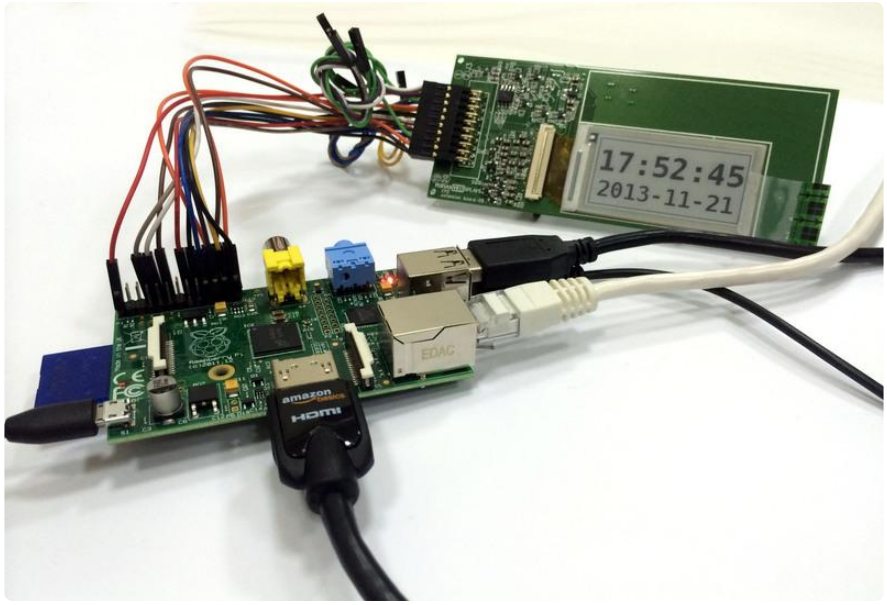
Flip the cable over so that the white dot is facing down.

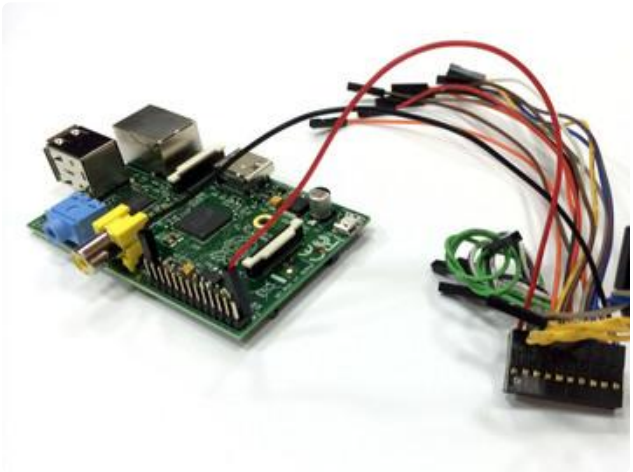
Center the connector in the socket with Pin 1 on the side marked with a circled "1" on the board.

Be sure to center the connector in the socket. The socket is wider than the connector and the display will not work if you plug it in off-center.

Wiring the Raspberry Pi

The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts





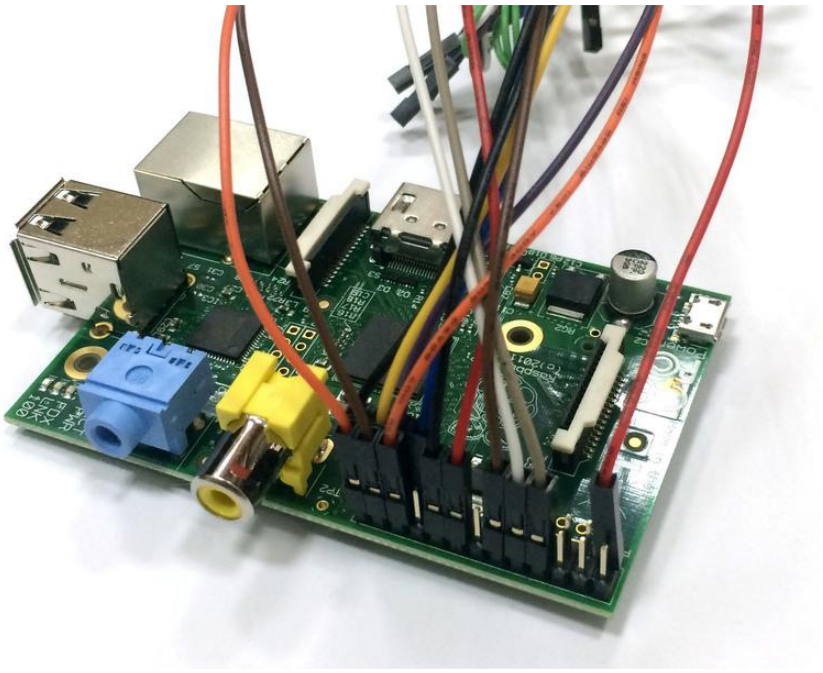
Connections:

Start by connecting Pin 1 of the breakout cable (red wire) to the header pin for 3.3v.

Continue making connections in the order listed below. There are duplicate wire colors, so don't mix them up!

Make connections to:

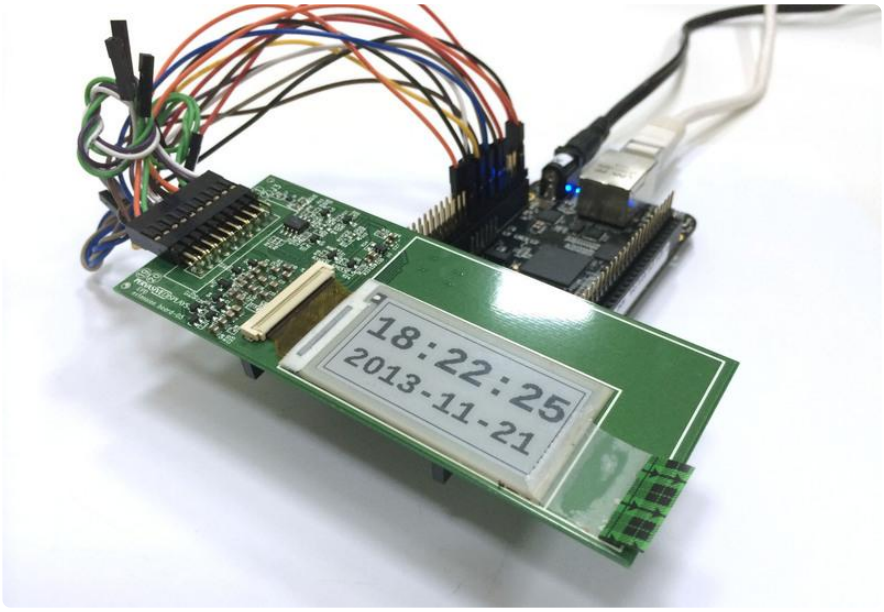
(Red #1) -> P1-01
(Green #6) -> not used (ext ADC required)
(Yellow #7) -> P1-23
(Orange #8) -> P1-22
(Brown #9) -> P1-12
(Black #10) -> P1-18
(Red #11) -> P1-16
(White #12) -> P1-10
(Grey #13) -> P1-08
(Purple #14) -> P1-21
(Blue #15) -> P1-19
(Orange #18) -> P1-26
(Brown #19) -> P1-24
(Black #20) -> P1-25

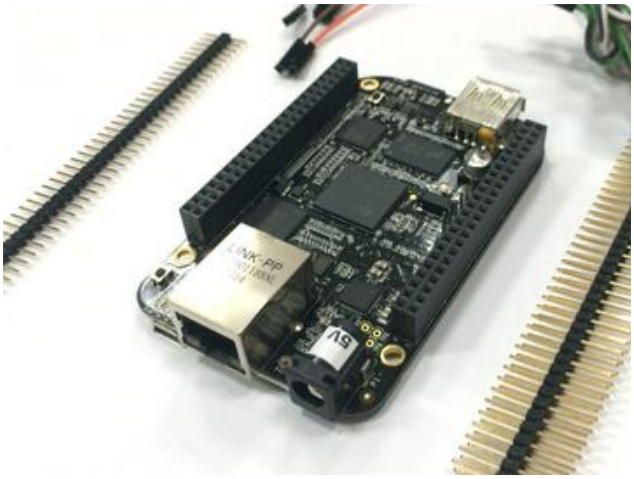


Now you are done wiring and ready to test the display!

Wiring the BeagleBone Black

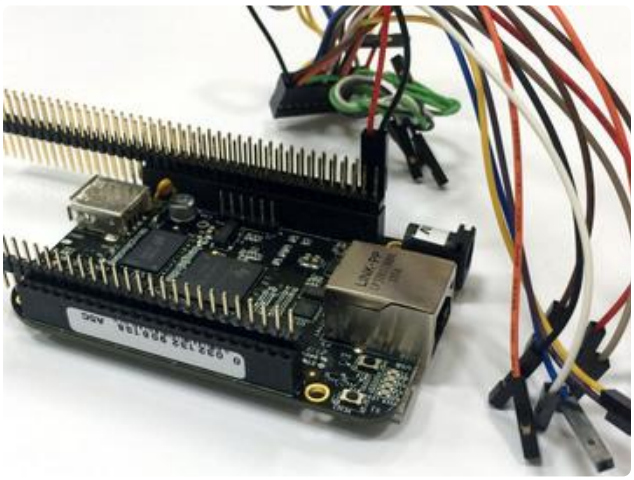
The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts

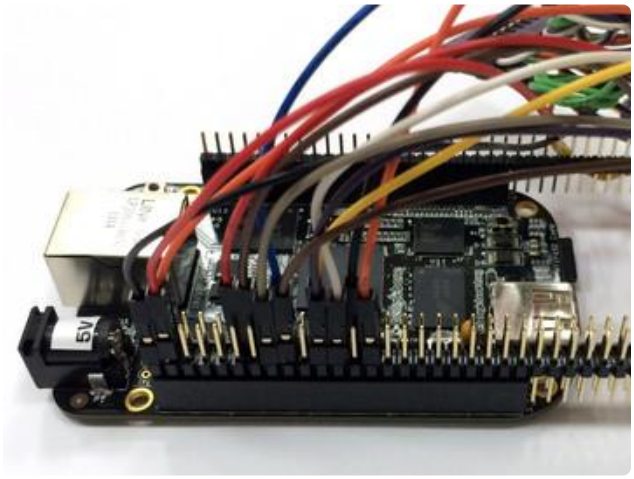




Insert the header segments

Insert the double-sided male header sections into the BeagleBone's headers. You can use both sides to create a nice stand for the display.





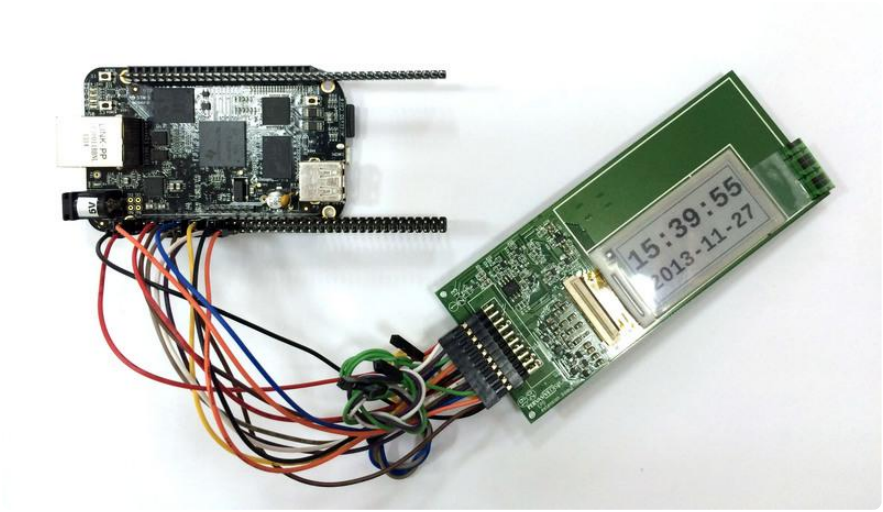
Connections:

Start by connecting Pin 1 of the breakout cable (red wire) to the header pin for 3.3v.

Continue making connections in the order listed below. There are duplicate wire colors, so don't mix them up!

Make connections to:

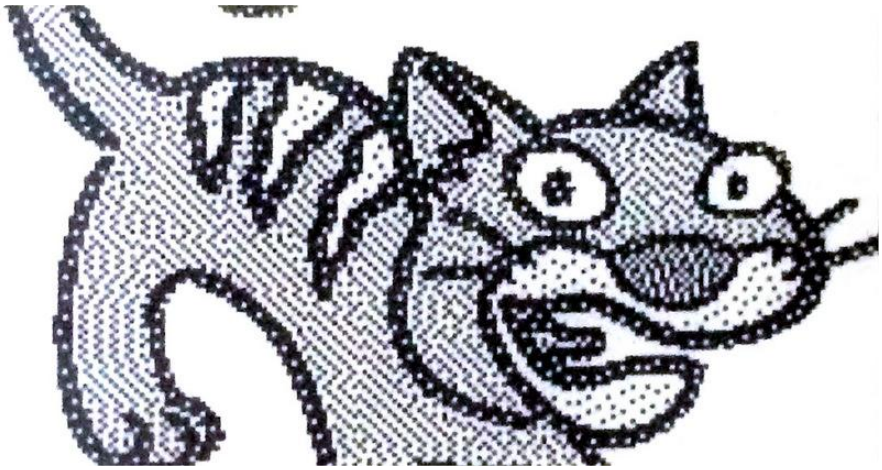
(Red #1) -> P9-03
(Green #6) -> -?- P9-39 (not yet)
(Yellow #7) -> P9-22
(Orange #8) -> P9-27
(Brown #9) -> P9-14
(Black #10) -> P9-26
(Red #11) -> P9-12
(White #12) -> P9-23
(Grey #13) -> P9-15
(Purple #14) -> P9-21
(Blue #15) -> P9-18
(Orange #18) -> - Vcc P9-04
(Brown #19) -> P9-17
(Black #20) -> P9-01



Now you are done wiring and ready to test the display!

Driver and Examples

The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts



Compiling

These test programs should compile with no additional libraries, but the EPD driver needs the fuse development library installed. And a few examples make us of the Python Imaging Library (PIL)

Raspberry Pi

```
sudo apt-get install libfuse-dev python-imaging

# For the Twitter Demo
sudo apt-get install python-setuptools
sudo easy_install pip

git clone https://github.com/repaper/gratis.git
cd gratis/PlatformWithOS
make rpi
sudo make rpi-install
```

BeagleBone Black

```
sudo opkg install libfuse-dev python-imaging

# For the Twitter demo
opkg install python-pip python-setuptools

# Extra item for BeagleBone
# The program uses device tree files to access the GPIOs
git clone https://github.com/nomel/beaglebone.git
cp -p beaglebone/gpio-header/generated/gpio-P9.* /lib/firmware

git clone https://github.com/repaper/gratis.git
cd gratis/PlatformWithOS
make bb
sudo make bb-install
```

EPD fuse

This allows the display to be represented as a virtual directory of files, which are:

```
File         Read/Write   Description
--------     -----------  --------------------------------
version      Read Only    The driver version number
panel        Read Only    String describing the panel and giving
                          its pixel width and height
current      Read Only    Binary image that  matches the currently
                          displayed image (big endian)
display      Read Write   Image being assembled for next display
                          (big endian)
temperature  Read Write   Set this to the current temperature in Celsius
command      Write Only   Execute display operation
BE           Directory    Big endian version of current and display
LE           Directory    Little endian version of current and display

Command   Byte   Description
--------  -----  ------------------------------
'C'       0x43   Clear the EPD, set `current` to all zeros, `display`
                 is not affected
'U'       0x5A   Erase `current` from EPD, output `display` to EPD,
                 copy display to `current`
```

Notes:

- The default bit ordering for the display is big endian i.e. the top left pixel is the value 0x80 in the first byte.
- The `BE` directory is the same as the root `current` and `display`.
- The `LE` directory `current` and `display` reference the top left pixel as 0x01in the first byte.
- The `current_inverse` and `display_inverse` represent black as zero (0) and white as one (1) while those item without the suffix represent the display's natural coding (0=>white, 1=>black)
- The particular combination of `BE/display_inverse` is used in the Python EPD demo since it fits better with the Imaging library used.

Set Panel Size in FUSE Configuration

```
cat /etc/default/epd-fuse
```

Expected Output:

```
# Default settings for epd-fuse file is sourced by /bin/sh from
# /etc/init.d/epd-fuse

# Options to pass to epd_fuse
#EPD_MOUNTPOINT=/dev/epd
#EPD_SIZE=2.0
#EPD_OPTS=
```

Note: All the configurations are commented out and represent the default settings. If your panel is NOT the 2.0", then you must change this configuration.

How to edit the configuration file:

```
sudo nano /etc/default/epd-fuse
```

Uncomment the **EPD_SIZE** and change the value to your panel size (either **2.7** or **1.44**). Save and exit.

Start the Driver

```
sudo service epd-fuse start
cat /dev/epd/panel
```

Expected Output for a 2.7" panel:

```
EPD 2.7 264x176
```

Python Demo Programs - directory "demo"

Drawing Demo

Draw some lines, graphics and text:

```
python demo/DrawDemo.py
```

Nothing happening on the screen? You may have a "COG 2" version of the display. See the FAQ page!

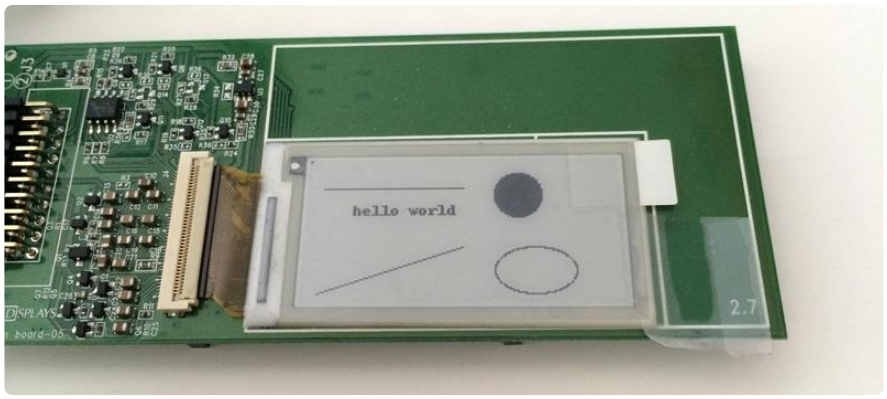
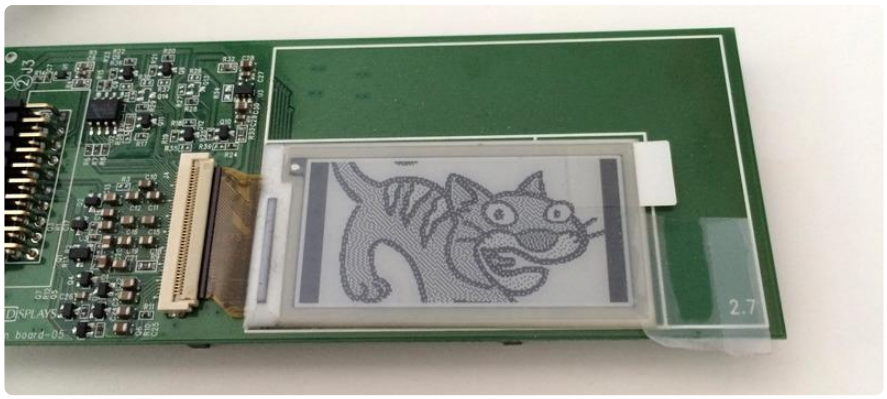


Image demo

- Accepts a lists of image files on the command line.
- Converts them to grey scale to ensure formats like PNG, JPEG and GIF will work.
- Inverts the image since the E-Ink panel is reverse (i.e. black on white).
- Converts image to single bit (PIL "1" mode).
- Display the middle of the image (using crop function).
- Delay.
- Display the re-sized image.
- Delay before fetching next file.

Note: if scratch is installed on the system, the following commands will show some cartoon animals. The images when re-sized will be distorted if the aspect ration of the original image is not the same as the display.

```
python demo/ImageDemo.py /usr/share/scratch/Media/Costumes/Animals/cat*
python demo/ImageDemo.py /usr/share/scratch/Media/Costumes/Animals/d*.png
```



Twitter Demo

Setup easy_install and use it to get pip, then use pip to get tweepy. Copy the sample configuration an edit to include your authentication information. Rather than using the basic authentication it is better to set up a Twitter App and generate a token for this. The token generation is quick at https://dev.twitter.com/ (https://adafru.it/d2C).

After creating the App, just click the button to create an access token.

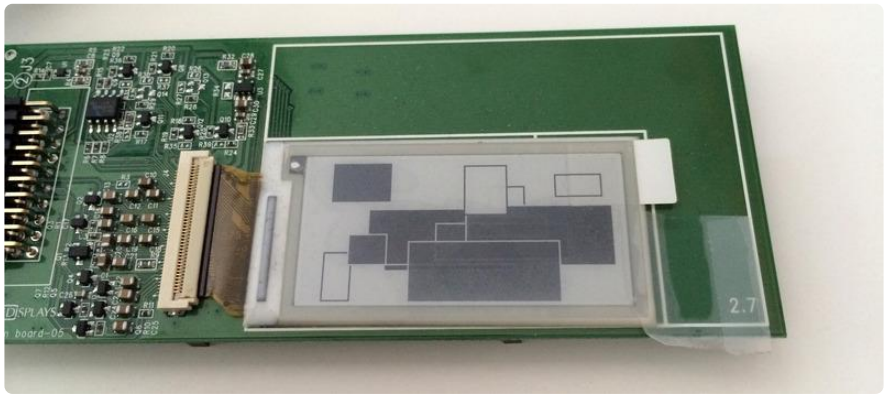
Use **Ctrl-C** to stop this program.

```
sudo pip install tweepy

# setup the config
cp demo/tweepy_auth.py-SAMPLE demo/tweepy_auth.py

# *** edit the config
nano demo/tweepy_auth.py

# run the demo (this watches for linux)
python demo/TwitterDemo.py linux
```

Partial Demo

Display random overlapping rectangles using partial update. First argument is number of rectangle to generate before updating the EPD, second number is the number of frames to display before the program exits.

```
python demo/PartialDemo.py 3 20
```

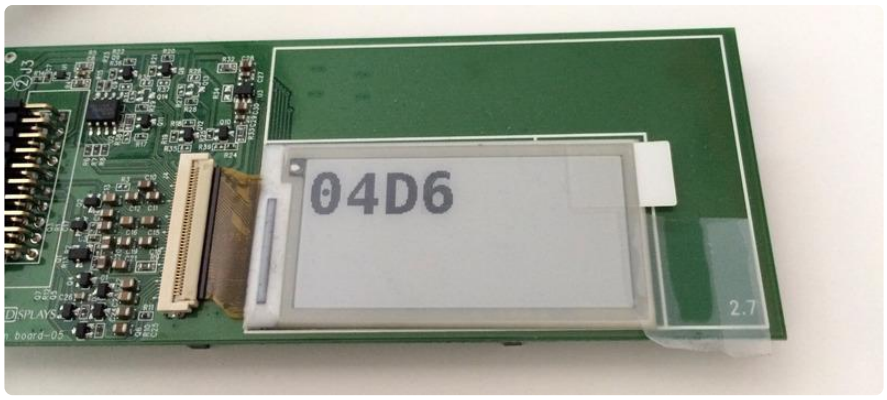


Counter Demo

Display a 4 digit hex counter uses partial update to only change the updated digits. This will look somewhat strange as the display inversion will make the counter appear to go through a sequence like: 0000 0001 0000 0001 ...delay... 0001 0002 0001 0002

Use **Ctrl-C** to stop this program.

```
python demo/CounterDemo.py 3 20
```



FAQ

The examples in this guide are no longer supported. Please check out the Adafruit eInk Display Breakouts guide for CircuitPython and Python usage: https://learn.adafruit.com/adafruit-eink-display-breakouts

Everything compiled correctly, but I get no activity on the screen

The default driver compile is for a COG 1 (Computer On Glass, version 1). Your RePaper screen may be a COG 2.
You can check this my carefully lifting the display and looking on the back of it. Look for a barcode starting with "TEM" or "VEM".
If the seventh character of the barcode is 'B', then you have a COG 2. Follow these instructions for building the driver:

Raspberry Pi

```
make rpi-clean
make COG_VERSION=V2 rpi
sudo make COG_VERSION=V2 rpi-install
sudo reboot
```

After reboot make sure it is showing a "COG 2"

```
% cat /dev/epd/panel
EPD 2.7 264x176 COG 2
% echo C &gt; /dev/epd/command
```

BeagleBone Black

```
make bb-clean
make COG_VERSION=V2 bb
make COG_VERSION=V2 bb-install
reboot
```

I can't get the eInk Display service to start on my Pi

See https://forums.adafruit.com/viewtopic.php?f=22&t=105673 (https://adafru.it/djF) for a possible fix!