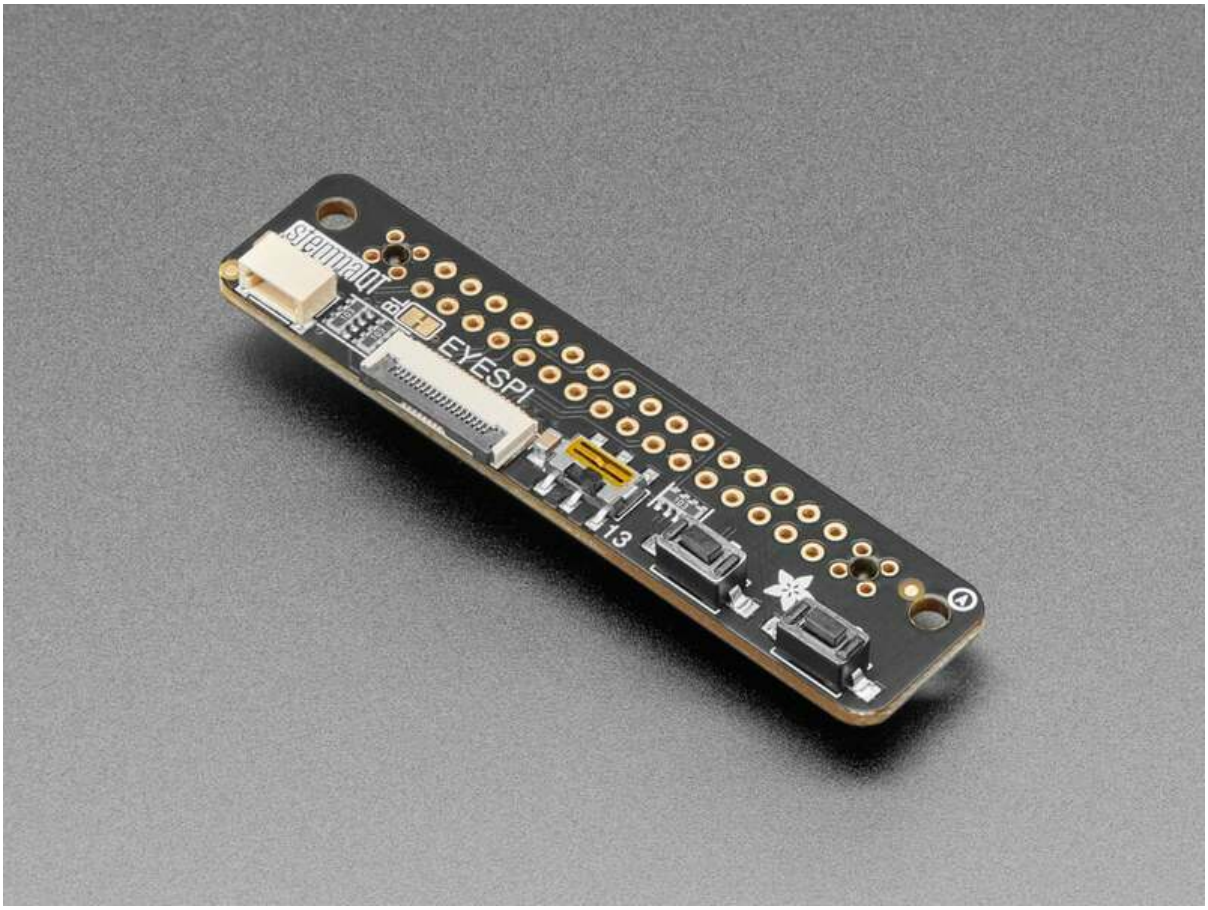
Adafruit EYESPI Pi Beret

Created by Kattni Rembor

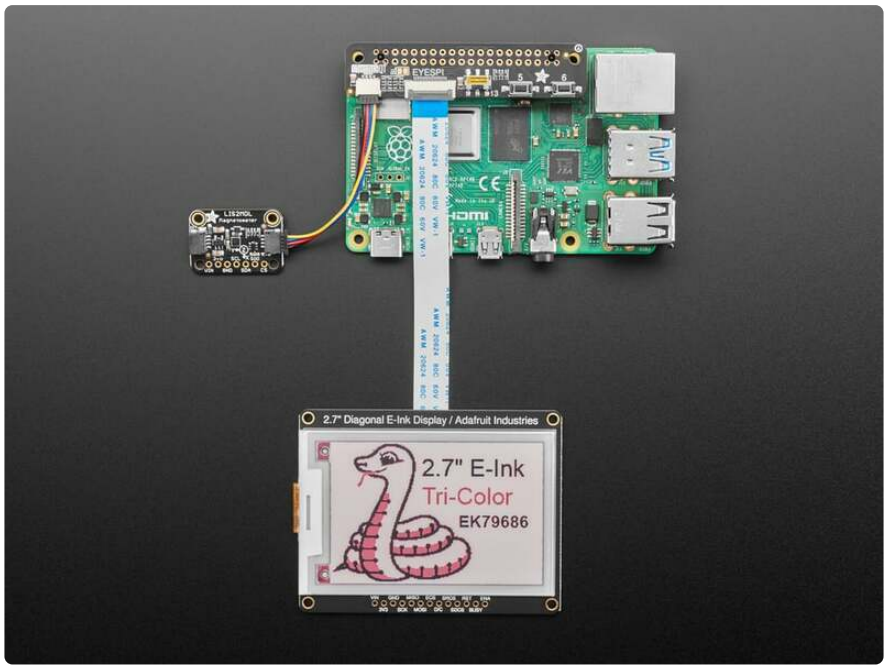


https://learn.adafruit.com/eyespi-pi-beret

Last updated on 2024-06-03 03:54:33 PM EDT

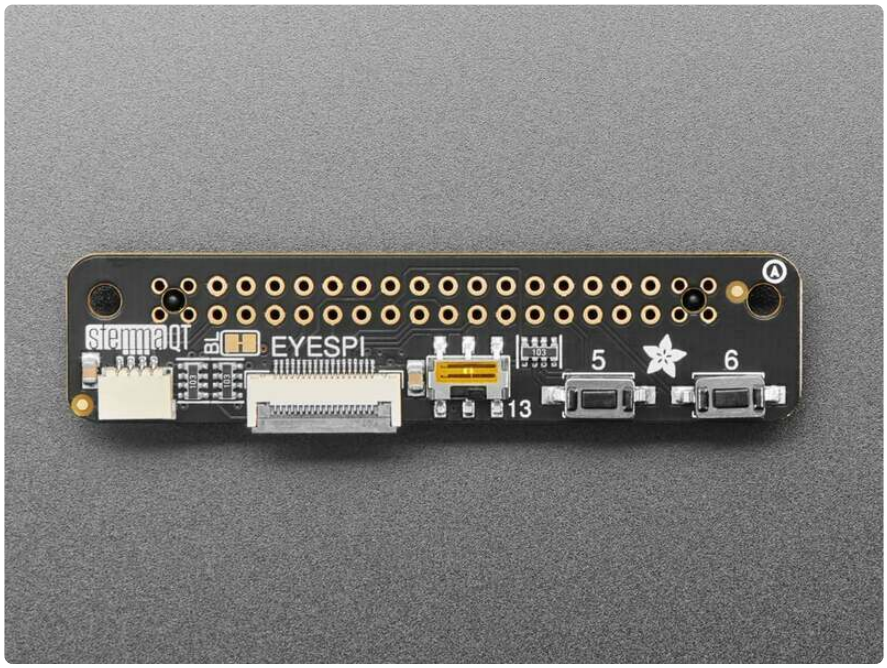
Table of Contents

Overview



Raspberry Pi's make for handy li'l computers, but they're really wonderful when you can connect all sorts of nifty hardware to them: color TFT or E-Ink displays, and sensors are our go-to favorites. Even better is when we make it fast and effortless to wire these up.

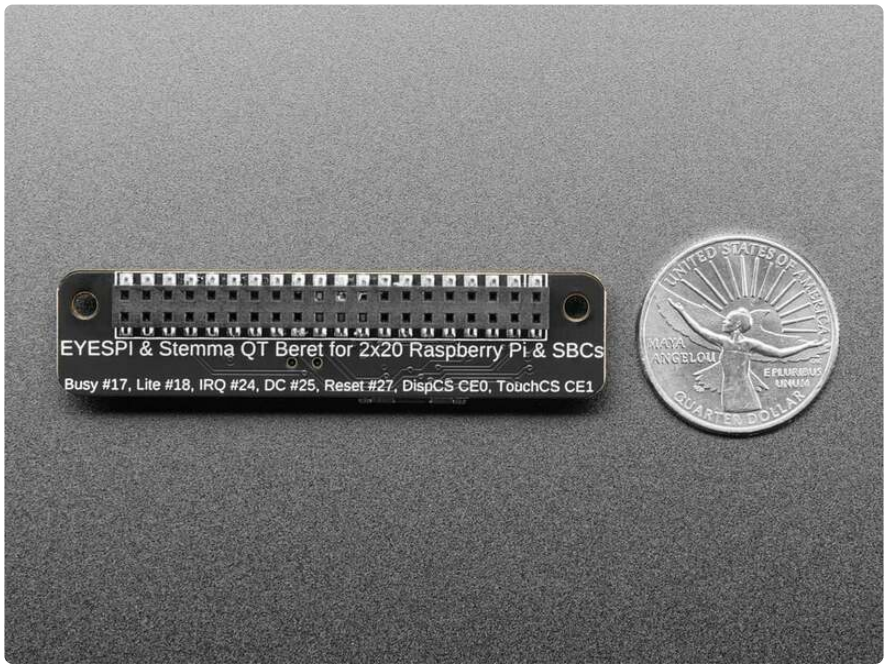
With this new EYESPI Pi Beret, there's no more counting pins or noodling with a breakout board. This slim HAT (https://adafru.it/18XB) plugs in and gives you lots of hardware connection options so that many projects become plug-and-play.



Our most recent display breakouts have come with a new feature: an 18-pin "EYE SPI" standard FPC connector with flip-top connector (https://adafru.it/18fg). This is intended to be a sort-of "STEMMA QT (https://adafru.it/Ft4) for displays" - a way to quickly connect and extend display wiring that uses a lot of SPI pins. In this case, we need a lot of SPI and accessory pins, and we want to be able to use long distances, so we go with an 18-pin 0.5mm pitch FPC.

With such a slim and flexible cable, it's easy to have displays anywhere without them physically attached to the Pi like in our PiTFTs (https://adafru.it/18XC). Accessorize with big bold colorful displays (https://adafru.it/18Kd) or power-sipping E-Ink (https://adafru.it/18Ke).

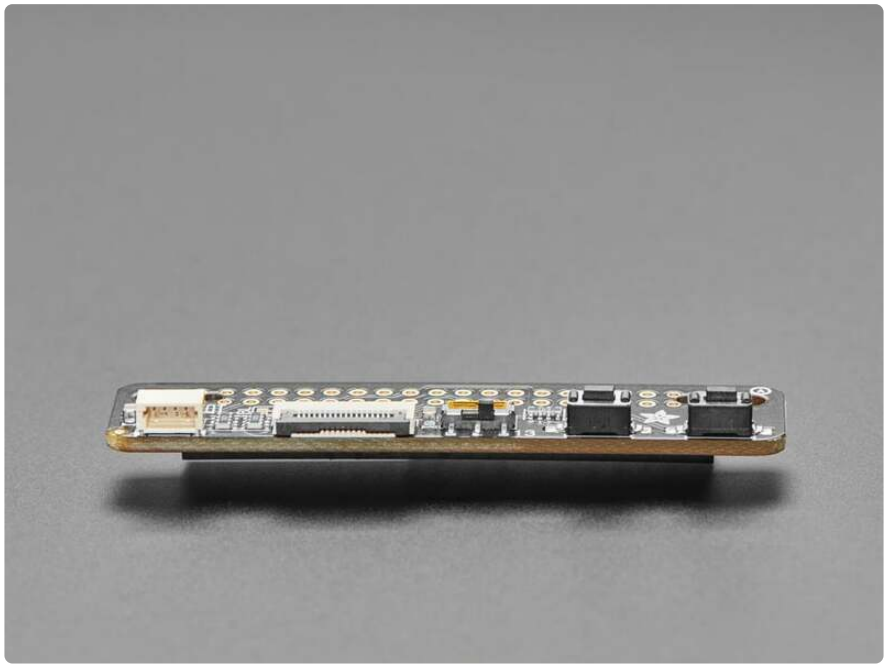
Don't forget you'll also want an 18-pin EYESPI FPC cable **(http://adafru.it/5462). And of course** one of our EYESPI displays **(https://adafru.it/18Kf) too -** look for the EYESPI logo on the back to know you've got one that can clip in.



Pins:

- **MOSI/MISO/SCK** SPI pins are connected to the default **SPI** port on the Raspberry Pi.
- **SDA/SCL** I2C pins are connected to the default **I2C** port on the Pi.
- **Display CS** on Raspberry Pi **CE 0.**
- **Display DC** on Raspberry Pi **GPIO #25.**
- **Display Reset** on Raspberry Pi **GPIO #27.**
- **Display Busy** on Raspberry Pi **GPIO #17** (used for E-Inks).
- **Display Backlight** on Raspberry Pi **GPIO #18** (there's a jumper to cut/disable this if you want to use the PWM output for other uses like NeoPixels).

- **Touch CS** on Raspberry Pi **CE 1.**
- **Touch IRQ** on Raspberry Pi **GPIO #24.**

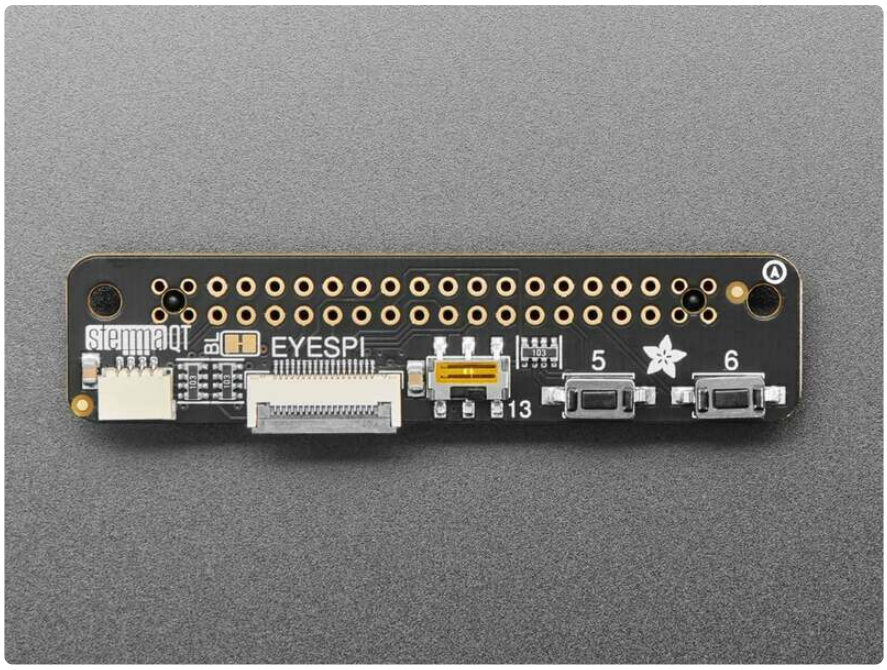


The remaining EYESPI pins are not connected, leaving you with plenty of GPIO for other accessory boards.

We also have a Stemma QT connector for the I2C port, to make connecting all sorts of Qwiic/Stemma QT devices super easy, two tactile switches on GPIO #5 and #6 and a slide switch on GPIO #13 which you can use for any sort of interface project or configuration.

Display and EYESPI cable are not included.

Pinouts



EYESPI Display Connector

- **EYESPI connector** - This connector allows you to connect EYESPI-compatible displays up using an EYESPI cable, with no soldering or jumper wires needed.

Pins needed for using the display:

- **MOSI/MISO/SCK** - SPI pins are connected to the default **SPI** port on the Raspberry Pi.
- **Display CS** on Raspberry Pi **CE 0.**
- **Display DC** on Raspberry Pi **GPIO #25.**
- **Display Reset** on Raspberry Pi **GPIO #27.**
- **Display Busy** on Raspberry Pi **GPIO #17** (This pin is used by E-Ink displays).
- **Display Backlight** on Raspberry Pi **GPIO #18** (There's a jumper to disable this; see below).
- **Display Touch CS** on Raspberry Pi **CE 1** (This pin is not used by all displays).
- **Display Touch IRQ** on Raspberry Pi **GPIO #24** (This pin is not used by all displays).

Backlight Jumper

- **BL jumper** - There is a jumper located next to the EYESPI label on the board silk. If you cut this trace, you will disable the backlight. Do this if you want to use the PWM output for other uses like NeoPixels.

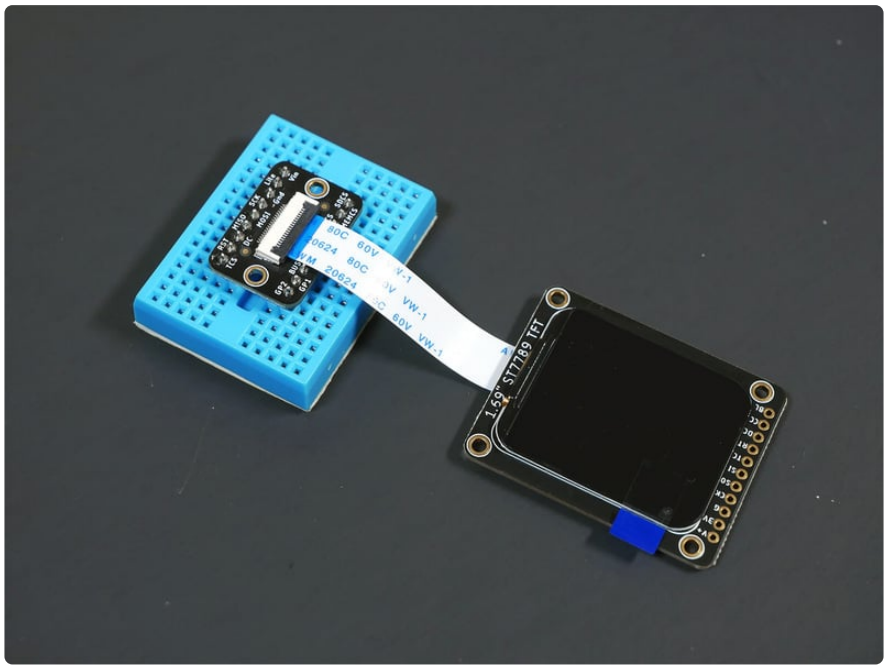
Buttons and Slide Switch

- **Buttons** - There are two buttons on the right side of the Beret, connected to **GPIO #5** and **GPIO #6**. You can use these as inputs.
- **Slide switch** - There is a slide switch towards the center of the Beret connected to **GPIO #13**. This is not a power switch! It is basic slide switch, which is high or low depending on the position of the switch. You can use it as an input.

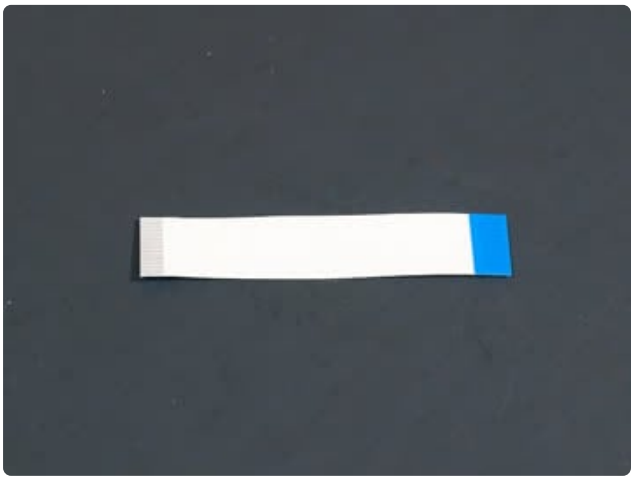
STEMMA QT Connector

- STEMMA QT **(https://adafru.it/Ft4)** - This connector, on the left side of the Beret, allows you to connect to sensors and breakout boards with STEMMA QT / Qwiic connectors or to other things with various associated accessories **(https://adafru.it/JRA)**.
- **SDA/SCL** - The I2C pins for the STEMMA QT connector are connected to the default **I2C** port on the Pi.
- **3.3V/GND** - The power for the STEMMA QT connector is 3.3V. Ground is the common ground for power and logic.

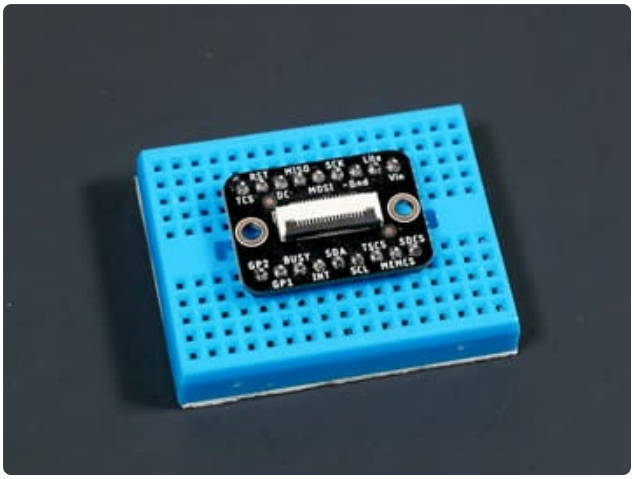
Plugging in an EYESPI Cable



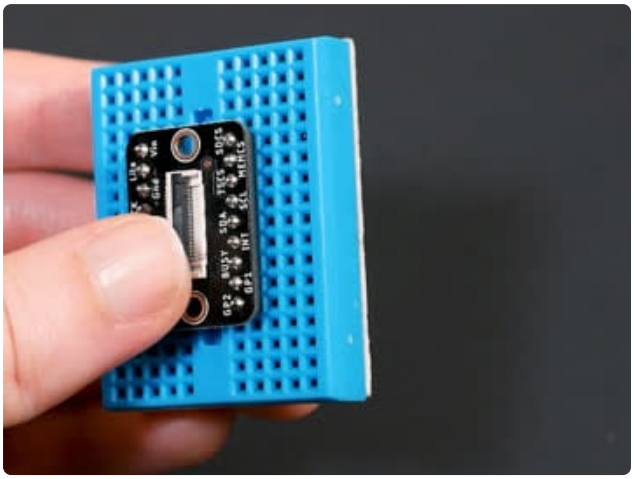
You can connect an EYESPI compatible display to the EYESPI breakout board using an EYESPI cable. An EYESPI cable is an 18 pin flexible PCB (FPC). The FPC can only be connected properly in one orientation, so be sure to follow the steps below to ensure that your display and breakout are plugged in properly.



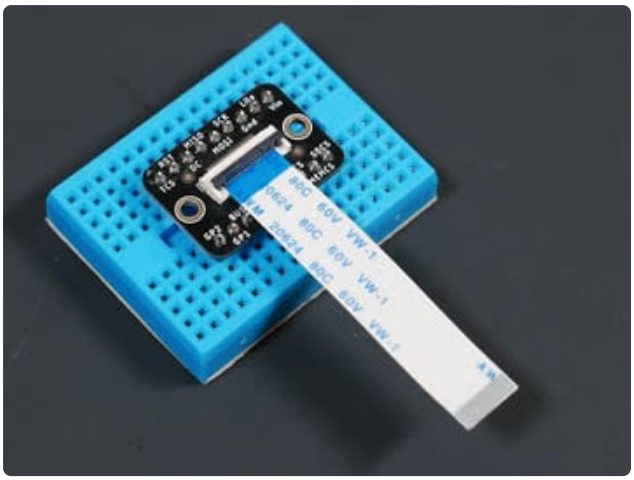
Each EYESPI cable has **blue stripes** on either end. On the other side of the cable, underneath the blue stripe, are the connector pins that make contact with the FPC connector pins on the display or breakout.



To begin inserting an EYESPI cable to an FPC connector, gently lift the FPC connector black latch up.



Then, insert the EYESPI cable into the open FPC connector by sliding the cable into the connector. You want to **see the blue stripe facing up towards you**. This inserts the cable pins into the FPC connector.



To secure the cable, lower the FPC connector latch onto the EYESPI cable.

Repeat this process for the FPC connector on your display. Again, ensure that the **blue stripe** on either end of the cable is facing up.

Python GIF Player Demo

Using the EYESPI Pi Beret is quite straightforward. Attach it to your Pi, plug in a display, and you're ready to go. This demo will test the display and button features of the Beret with an animated GIF player.

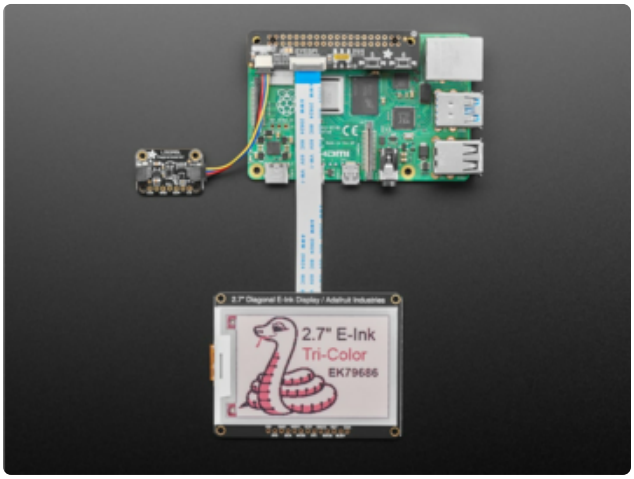
Raspberry Pi Setup

To prepare the Pi for this demo, follow the setup instructions here (https://adafru.it/18XD).

Remember to enable SPI using raspi-config!

Hardware Setup

The first step is attaching the EYESPI Pi Beret to your Raspberry Pi, and then connecting a display using an EYESPI-compatible ribbon cable.



Attach the Beret by pressing the assembled header on the bottom of the Beret onto the GPIO header on the Raspberry Pi.

Then plug an EYESPI cable into the connector on the Beret, and into the connector on a compatible display.

GIF Player Example Code

Copy the following code into a file named **rgb_display_eyespi_beret_animated_gif.py**, into whatever directory on your Raspberry Pi you wish to run it from.

```
# SPDX-FileCopyrightText: 2021 Melissa LeBlanc Williams for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
EYESPI Pi Beret GIF Player Demo

Extracts the frames and other parameters from an animated gif
and then runs the animation on the display.

Save this file as eyespi_beret_gif_player.py to your Raspberry Pi.

Usage:
python3 eyespi_beret_gif_player.py

This example is for use on Raspberry Pi that are using CPython with
Adafruit Blinka to support CircuitPython libraries. CircuitPython does
not support PIL/pillow (python imaging library)!

Author(s): Melissa LeBlanc-Williams for Adafruit Industries
           Mike Mallett <mike@nerdcore.net>
"""
import os
import time
import digitalio
import board
from PIL import Image, ImageOps
import numpy  # pylint: disable=unused-import
from adafruit_rgb_display import ili9341
from adafruit_rgb_display import st7789  # pylint: disable=unused-import
from adafruit_rgb_display import hx8357  # pylint: disable=unused-import
from adafruit_rgb_display import st7735  # pylint: disable=unused-import
from adafruit_rgb_display import ssd1351  # pylint: disable=unused-import
from adafruit_rgb_display import ssd1331  # pylint: disable=unused-import

# Button pins for EYESPI Pi Beret
BUTTON_NEXT = board.D5
BUTTON_PREVIOUS = board.D6

# CS and DC pins for EYEPSPI Pi Beret:
cs_pin = digitalio.DigitalInOut(board.CE0)
dc_pin = digitalio.DigitalInOut(board.D25)

# Reset pin for EYESPI Pi Beret
reset_pin = digitalio.DigitalInOut(board.D27)

# Backlight pin for Pi Beret
backlight = digitalio.DigitalInOut(board.D18)
backlight.switch_to_output()
backlight.value = True

# Config for display baudrate (default max is 64mhz):
BAUDRATE = 64000000

# Setup SPI bus using hardware SPI:
spi = board.SPI()

# pylint: disable=line-too-long
# fmt: off
```

```
# Create the display.
disp = ili9341.ILI9341(spi, rotation=90,                            # 2.2", 2.4",
2.8", 3.2" ILI9341
# disp = st7789.ST7789(spi, rotation=90,                            # 2.0" ST7789
# disp = st7789.ST7789(spi, height=240, y_offset=80, rotation=180,  # 1.3", 1.54"
ST7789
# disp = st7789.ST7789(spi, rotation=90, width=135, height=240, x_offset=53,
y_offset=40, # 1.14" ST7789
# disp = st7789.ST7789(spi, rotation=90, width=172, height=320, x_offset=34, #
1.47" ST7789
# disp = st7789.ST7789(spi, rotation=270, width=170, height=320, x_offset=35, #
1.9" ST7789
# disp = hx8357.HX8357(spi, rotation=180,                           # 3.5" HX8357
# disp = st7735.ST7735R(spi, rotation=90,                           # 1.8" ST7735R
# disp = st7735.ST7735R(spi, rotation=270, height=128, x_offset=2, y_offset=3,   #
1.44" ST7735R
# disp = st7735.ST7735R(spi, rotation=90, bgr=True, width=80,       # 0.96" MiniTFT
Rev A ST7735R
# disp = st7735.ST7735R(spi, rotation=90, invert=True, width=80, x_offset=26,
y_offset=1,  # 0.96" MiniTFT Rev B ST7735R
# disp = ssd1351.SSD1351(spi, rotation=180,                         # 1.5" SSD1351
# disp = ssd1351.SSD1351(spi, height=96, y_offset=32, rotation=180, # 1.27" SSD1351
# disp = ssd1331.SSD1331(spi, rotation=180,                         # 0.96" SSD1331
                       cs=cs_pin,
                       dc=dc_pin,
                       rst=reset_pin,
                       baudrate=BAUDRATE,
                       )
# fmt: on
# pylint: enable=line-too-long


def init_button(pin):
    button = digitalio.DigitalInOut(pin)
    button.switch_to_input()
    button.pull = digitalio.Pull.UP
    return button


class Frame:  # pylint: disable=too-few-public-methods
    def __init__(self, duration=0):
        self.duration = duration
        self.image = None


class AnimatedGif:
    def __init__(self, display, width=None, height=None, folder=None):
        self._frame_count = 0
        self._loop = 0
        self._index = 0
        self._duration = 0
        self._gif_files = []
        self._frames = []

        if width is not None:
            self._width = width
        else:
            self._width = display.width
        if height is not None:
            self._height = height
        else:
            self._height = display.height
        self.display = display
        self.advance_button = init_button(BUTTON_NEXT)
        self.back_button = init_button(BUTTON_PREVIOUS)
        if folder is not None:
            self.load_files(folder)
            self.run()
```

```
    def advance(self):
        self._index = (self._index + 1) % len(self._gif_files)

    def back(self):
        self._index = (self._index - 1 + len(self._gif_files)) %
len(self._gif_files)

    def load_files(self, folder):
        gif_files = [f for f in os.listdir(folder) if f.endswith(".gif")]
        for gif_file in gif_files:
            gif_file = os.path.join(folder, gif_file)
            image = Image.open(gif_file)
            # Only add animated Gifs
            if image.is_animated:
                self._gif_files.append(gif_file)

        print("Found", self._gif_files)
        if not self._gif_files:
            print("No Gif files found in current folder")
            exit()  # pylint: disable=consider-using-sys-exit

    def preload(self):
        image = Image.open(self._gif_files[self._index])
        print("Loading {}...".format(self._gif_files[self._index]))
        if "duration" in image.info:
            self._duration = image.info["duration"]
        else:
            self._duration = 0
        if "loop" in image.info:
            self._loop = image.info["loop"]
        else:
            self._loop = 1
        self._frame_count = image.n_frames
        self._frames.clear()
        for frame in range(self._frame_count):
            image.seek(frame)
            # Create blank image for drawing.
            # Make sure to create image with mode 'RGB' for full color.
            frame_object = Frame(duration=self._duration)
            if "duration" in image.info:
                frame_object.duration = image.info["duration"]
            frame_object.image = ImageOps.pad(  # pylint: disable=no-member
                image.convert("RGB"),
                (self._width, self._height),
                method=Image.NEAREST,
                color=(0, 0, 0),
                centering=(0.5, 0.5),
            )
            self._frames.append(frame_object)

    def play(self):
        self.preload()

        _prev_advance_btn_val = self.advance_button.value
        _prev_back_btn_val = self.back_button.value
        # Check if we have loaded any files first
        if not self._gif_files:
            print("There are no Gif Images loaded to Play")
            return False
        while True:
            for frame_object in self._frames:
                start_time = time.monotonic()
                self.display.image(frame_object.image)
                _cur_advance_btn_val = self.advance_button.value
                _cur_back_btn_val = self.back_button.value
                if not _cur_advance_btn_val and _prev_advance_btn_val:
                    self.advance()
                    return False
                if not _cur_back_btn_val and _prev_back_btn_val:
```

```
                    self.back()
                    return False

                _prev_back_btn_val = _cur_back_btn_val
                _prev_advance_btn_val = _cur_advance_btn_val
                while time.monotonic() < (start_time + frame_object.duration /
1000):
                    pass

            if self._loop == 1:
                return True
            if self._loop > 0:
                self._loop -= 1

    def run(self):
        while True:
            auto_advance = self.play()
            if auto_advance:
                self.advance()


if disp.rotation % 180 == 90:
    disp_height = disp.width  # we swap height/width to rotate it to landscape!
    disp_width = disp.height
else:
    disp_width = disp.width
    disp_height = disp.height

gif_player = AnimatedGif(disp, width=disp_width, height=disp_height, folder=".")
```

This demo is explained in detail here (https://adafru.it/18XE). The explanation is for a slightly different version of the code, but most of it is identical. This demo code was rearranged a bit, and the pins are different. Everything else is the same.

The default displays for this demo are the 2.2", 2.4", 2.8", and 3.2" ILI9341. If you're using a different display, you'll need to alter the code.

If you are using a display other than the 2.2", 2.4", 2.8", or 3.2" ILI9341 display, you'll need to comment out the first line under `# Create the display`, and uncomment the line below it that applies to your display. Check out the demo explanation (https://adafru.it/18XE) for details.

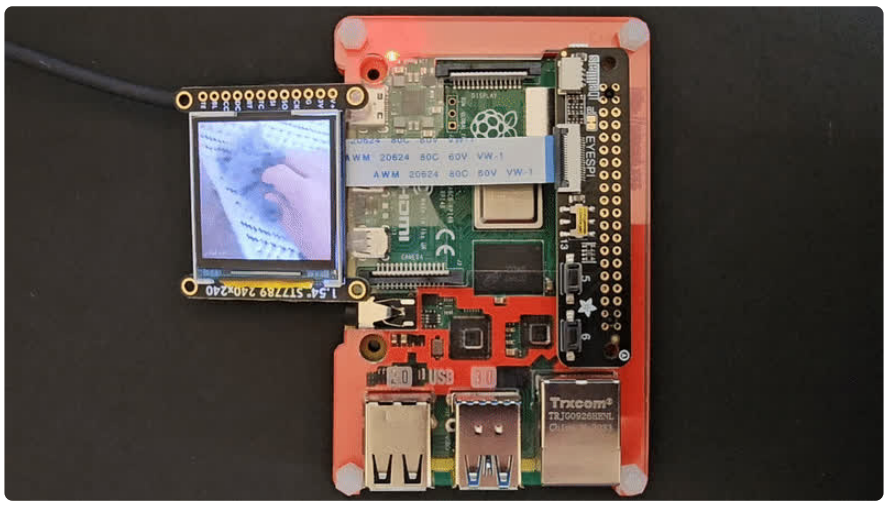
Usage

Place two or more GIF files in the same directory as you saved the code file. You can find animated GIFs all over the internet. A great website for finding a large collection of them is giphy.com (https://adafru.it/GeU). Once you find them, you'll need to get them over to your Raspberry Pi. The easiest way would be to just use FTP or downloading them directly from the internet on your Pi if that's what you would prefer. Just make sure they're in the same folder as the script.

Then, run the code by typing the following into your command line.

```
python3 rgb_display_eyespi_beret_animated_gif.py
```

The first GIF should begin playing on the display.



To cycle through the other GIFs, press the buttons on the Pi Beret. The button marked 6 will advance to the next one, and the button marked 5 will go back to the previous one.

Downloads

Files

- EYESPI Pi Beret Fritzing object in the Adafruit Fritzing Library (https://adafru.it/18XF)
- EYESPI Pi Beret with EYESPI cable Fritzing object in the Adafruit Fritzing Library (https://adafru.it/18Yb)
- EagleCAD PCB Files on GitHub (https://adafru.it/18Yc)

Schematic and Fab Print