Capturing Camera Images with CircuitPython

Created by Jeff Epler


Last updated on 2022-03-18 10:20:55 AM EDT
# Table of Contents

## Overview
- Parts 3
- Parts 3

## Cameras and Pinouts
- Pins 7

## Install TinyUF2 on Espressif Kaluga
- Method 1: WebSerial ESPTool / esptool 9
- Step 1. Download the tinyuf2 combined.bin file here 9
- Step 2. Place your board in bootloader mode 9
- Step 3 Option A. Use the Web Serial ESPTool to upload 11
- Step 3. Option B. Use esptool.py to upload (for advanced users) 15
- Step 4. Reset the board 16
- Method 2: Flash an Arduino Sketch 17
- Arduino IDE Setup 17
- Load the Blink Sketch 18

## Espressif Kaluga Setup
- Kaluga 1.3 with OV2640 and ili9341 display 20
- Kaluga 1.3 with OV2640 and st7789 display 22
- Kaluga 1.3 with OV7670 and ili9341 display 24
- Adapting to other ESP32-S2 boards 26

## Raspberry Pi Pico Wiring
- Wiring 28
- For the OV2640 camera 30
- For the OV7670 camera 32
- Adapting to other RP2040 boards 35

## Grand Central M4 Wiring
- Adapting to other SAM D5x/E5x boards 38

## Working with Image Data
- RGB Data 40
- YUV Data 43
- Test modes 45

## Capturing JPEG data (OV2640)
- Code 47
- Parts 49

Documentation: adafruit_ov2640 50

Documentation: adafruit_ov7670 50

Documentation: adafruit_ov5640 51
Overview

CircuitPython 7 adds support for capturing images from "parallel cameras" on select boards, and libraries are available to configure the popular OV7670 and OV2640 cameras.

While not up to standards we’re used to from a current smartphone or laptop, it’s nicely balanced to the capabilities of recent 32-bit microcontrollers.

The Arduino library for OV7670 cameras on the Grand Central M4 has its own dedicated guide (https://adafru.it/TAG).

Parts

Items needed

• Compatible microcontroller board: any ESP32-S2 including Kaluga supported by CircuitPython with enough pins exposed, any RP2040 supported by
CircuitPython including Pico and other boards with enough pins, or Grand Central M4

- SPI TFT module
- OV7670 or OV2640 camera module — the board pinout must match exactly
- Two 2.2k resistors, if pull-up resistors are necessary
- Soldering iron and supplies
- Appropriate USB data + power cable
- Prototyping supplies such as breadboards and jumper wires

The Espressif Kaluga v1.3 development kit has all you need (including an OV2640 camera module and 320x240 LCD), and no soldering is required, so it’s probably the best way to get started! Be aware that you must use the USB Breakout Cable with this board.

It is not possible to access the CIRCUITPY drive or the REPL using the USB Micro B connections on the Kaluga board so you will also need a USB Plug Breakout cable (see below). Also, you must have v1.3 of the Kaluga kit, v1.2 has some incompatible wiring that makes it unusable in this project!

OV7670 and OV2640 camera modules with the 18 pin, 2-row header can be found on Amazon, eBay, etc. Make sure the pinout matches the camera shown above, as occasionally there are incompatible variants. The cameras are sometimes sold in sets which is a good idea, as they’re easily damaged with the wrong voltage or rough handling, especially if you need to modify it for use with the Grand Central M4.

Other sensor models (such as OV3660 and OV5640) exist, but they require different initialization code and cannot currently be used.

Parts

ESP32-S2 Kaluga Dev Kit featuring ESP32-S2 WROVER
The ESP32-S2-Kaluga-1 kit is a full featured development kit by Espressif for the ESP32-S2 that comes with everything but the kitchen sink! From TFTs to touch panels,...
https://www.adafruit.com/product/4729
USB Type A Plug Breakout Cable with Premium Female Jumpers
If you'd like to connect a USB-capable chip to your USB host, this cable will make the task very simple. There is no converter chip in this cable! It's basically a...
https://www.adafruit.com/product/4448

Raspberry Pi Pico RP2040
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/4864

Adafruit Grand Central M4 Express featuring the SAMD51
Are you ready? Really ready? Cause here comes the Adafruit Grand Central featuring the Microchip ATSAMD51. This dev board is so big, it's not...
https://www.adafruit.com/product/4064

USB C to Micro B Cable - 3 ft 1 meter
As technology changes and adapts, so does Adafruit! Rather than the regular USB A, this cable has USB C to Micro B plugs! USB C is the latest...
https://www.adafruit.com/product/3878
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

1 x OV7670 Camera Boards
Lot of 5 camera boards
https://www.amazon.com/AITRIP-OV7670-640x480-0-3Mega-Arduino/dp/B09126R875

1 x OV2640 Camera Board
Camera module, 2 Megapixel
https://www.waveshare.com/OV2640-Camera-Board.htm

Cameras and Pinouts

The ESP-LyraP-CAM (https://adafruit.it/TAH) v1.1 module included with the Kaluga development kit is ready to go: it includes the pull-up resistors on the I2C communication lines, here labeled SIOC and SIOD (equivalent to SCL and SDA).

The typical OV7670 module doesn't include the I2C pull-up resistors, so you may have to add them. This one was also modified so that it would fit directly on the header of a Grand Central M4 board.
Pins

See the specific board pages for details on how to connect the pins to a microcontroller.

- **3.3V**: Connect to a regulated 3.3V supply
- **GND (sometimes labeled DGND)**: Connect to GND
- **SDA, SCL (sometimes labeled SIOC, SIOD)**: I2C bus used to configure the camera.
- **XLK (sometimes labeled XCLK)**: Clock signal from the microcontroller to the camera.
- **VS, HS, PLK (sometimes labeled SYNC, HREF, PCLK)**: Vertical & Horizontal synchronization pulses, and pixel clock from the camera to the microcontroller
- **D0..D7**: Pixel data from the camera to the microcontroller
- **RET, PWDN**: Reset and Power Down pins from the microcontroller are used during initialization to reset the camera into a known state.
Install TinyUF2 on Espressif Kaluga

There are two versions of the Kaluga board, v1.2 and v1.3. Check which version you have, and install the correct build of CircuitPython. The board revisions change the pinout of the camera connector slightly.

Now, use the breakout USB connection in lieu of either of the built-in USB Micro B ports to install and use CircuitPython.

Start by connecting the USB Breakout Cable to the Kaluga board.

Black: Use a Male/Female Extension Jumper Wire to connect to GND
White: Connect to IO19
Green: Connect to IO20
Red: Use a Male/Female Extension Jumper Wire to connect to 5V

Do not connect the Red wire to 3V3, it will irreversibly damage the Kaluga.

If you’re familiar with our other products and chipsets you may be familiar with our drag-n-drop bootloader, a.k.a UF2. We have a UF2 bootloader for the ESP32-S2, that will let you drag firmware on/off a USB disk drive.

Unlike the M0 (SAMD21) and M4 (SAMD51) boards, there is no bootloader protection for the UF2 bootloader. That means it is possible to erase or damage the bootloader, especially if you upload Arduino sketches to ESP32S2 boards that don't "know" there's a bootloader it should not overwrite!

However, thanks to the ROM bootloader, you don’t have to worry about it if the UF2 bootloader is damaged. The ROM bootloader can never be disabled or erased, so its always there if you need it! You can simply re-load the UF2 bootloader (USB-disk-style) with the ROM bootloader (non-USB-drive)

You can use the TinyUF2 bootloader to load code directly, say CircuitPython or the binary output of an Arduino compilation or you can use it to load a second bootloader on, like UF2 which has a drag-n-drop interface.
Method 1: WebSerial ESPTool / esptool

This section outlines using WebSerial ESPTool or esptool to flash the UF2 bootloader onto your ESP32-S2 board.

Step 1. Download the tinyuf2 combined.bin file here

Note that this file is 3MB but that's because the bootloader is near the end of the available flash. It's not actually 3MB large, most of the file is empty but its easier to program if we give you one combined 'swiss cheese' file. Save this file to your desktop or wherever you plan to run esptool from

combined.bin

https://adafru.it/TAI

Step 2. Place your board in bootloader mode

Entering the bootloader is easy. Complete the following steps.

1. Make sure your ESP32-S2 is plugged into USB port to your computer using a data/sync cable. Charge-only cables will not work!
2. Turn on the On/Off switch - If your board has a power switch, check that you see the OK light on so you know the board is powered, a prerequisite!
3. Press and hold the DFU / Boot0 button down. Don't let go of it yet!
4. Press and release the Reset button. You should have the DFU/Boot0 button pressed while you do this.
5. Now you can release the DFU / Boot0 button
Because there are several incompatible versions of the Kaluga TFT display, the bootloader's screen may appear incorrectly or not at all. This does not affect its operation.

Check for a new serial / COM port

On Windows check the Device manager - you will see a COM port, for example here its COM88. You may also see another "Other device" called ESP32-S2

It's best to do this with no other dev boards plugged in so you don't get confused about which COM port is the ESP32-S2
On Mac/Linux you will need to find the tty name which lives under /dev

On Linux, try `ls /dev/ttyS*` for example, to find the matching serial port name. In this case it shows up as `/dev/ttyS87`. If you don't see it listed try `ls /dev/ttyA*` on some Linux systems it might show up like `/dev/ttyACM0`

On Mac, try `ls /dev/cu.usbmodem*` for example, to find the matching serial port name. In this case, it shows up as `/dev/cu.usbmodem01`

It's best to do this with no other dev boards plugged in so you don't get confused about which serial port is the ESP32-S2

Step 3 Option A. Use the Web Serial ESPTool to upload

The WebSerial ESPTool was designed to be a web-capable option for programming ESP32-S2 boards. It allows you to erase the contents of the microcontroller and program up to 4 files at different offsets.

You will have to use the Chrome browser for this to work, Safari and Firefox, etc are not supported because we need Web Serial and only Chrome is supporting it to the level needed.

Enable Web Serial (For older chrome)

As of chrome 89, Web Serial is already enabled, so this step is only necessary on older browsers.
Visit `chrome://flags` from within Chrome.
Find and enable the Experimental Web Platform features

Restart Chrome

Connecting

In the Chrome browser visit https://adafruit.github.io/Adafruit_WebSerial_ESPTool/ (https://adafruit.it/PMB) it should look like the image to the left

Press the Connect button in the top right of the web browser. You will get a pop up asking you to select the COM or Serial port. You may want to remove all other USB devices so only the ESP32-S2 board is attached, that way there's no confusion over multiple ports!
The Javascript code will now try to connect to the ROM bootloader. It may timeout for a bit until it succeeds. On success, you will see that it is Connected and will print out a unique MAC address identifying the board.

Once you have successfully connected, the command toolbar will appear.

Erasing the Contents

If you would like to erase the entire flash area so that you can start with a clean slate, you can use the erase feature. We recommend doing this if you are having issues.

To erase the contents, click the Erase button. You will be prompted whether you want to continue. Click OK to continue or if you changed your mind, just click cancel.
Programming the Microcontroller

Programming the microcontroller can be done with up to 4 files at different locations, but with the tinyuf2combo BIN file, which you should have downloaded under Step 1 on this page, you only need to use 1 file.

You can click on Choose a file... from any of the available buttons. It will only attempt to program buttons with a file and a unique location. Then select the Adafruit CircuitPython BIN files (not the UF2 file!)

Verify that the Offset box next to the file location you used is 0x0.

Once you choose a file, the button text will change to match your filename. You can then select the Program button to start flashing.

A progress bar will appear and after a minute or two, you will have written the firmware.
Step 3. Option B. Use esptool.py to upload (for advanced users)

Once you have entered ROM bootloader mode, you can then use Espressif's esptool program (https://adafru.it/E9p) to communicate with the chip! esptool is the 'official' programming tool and is the most common/complete way to program an ESP chip.

Install ESPTool.py

You will need to use the command line / Terminal to install and run esptool.

You will also need to have pip and Python installed (any version!)

Install the latest version using pip (you may be able to run pip without the 3 depending on your setup):

```
pip3 install --upgrade esptool
```

Then, you can run:

```
esptool.py
```

Test the Installation

Run esptool.py in a new terminal/command line and verify you get something like the below:

Make sure you are running esptool v3.0 or higher, which adds ESP32-S2 support

Run the following command, replacing the identifier after --port with the COMxx, /dev/cu.usbmodemxx or /dev/ttySxx you found above.
esptool.py --port COM88 chip_id

You should get a notice that it connected over that port and found an ESP32-S2

Installing the Bootloader

Run this command and replace the serial port name with your matching port and the file you just downloaded

esptool.py --port COM88 write_flash 0x0 tinyuf2_combo.bin

Don't forget to change the **--port** name to match.

There might be a bit of a ‘wait’ when programming, where it doesn't seem like it's working. Give it a minute, it has to erase the old flash code which can cause it to seem like it's not running.

You'll finally get an output like this:

Step 4. Reset the board

Click the RESET button to launch the bootloader. You'll see a new disk drive on your computer with the name KALUGA1BOOT.
You're now ready to copy the CircuitPython UF2 on to the drive which will set up CircuitPython!

Method 2: Flash an Arduino Sketch

This section outlines flashing an Arduino sketch onto your ESP32-S2 board, which automatically installs the UF2 bootloader as well.

Arduino IDE Setup

If you don't already have the Arduino IDE installed, the first thing you will need to do is to download the latest release of the Arduino IDE. ESP32-S2 requires version 1.8 or higher. Click the link to download the latest.

Arduino IDE Download

https://adafruit.it/Pd5

After you have downloaded and installed the latest version of Arduino IDE, you will need to start the IDE and navigate to the Preferences menu. You can access it from the File > Preferences menu in Windows or Linux, or the Arduino > Preferences menu on OS X.

The Preferences window will open.

In the Additional Boards Manager URLs field, you'll want to add a new URL. The list of URLs is comma separated, and you will only have to add each URL once. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

Copy the following URL.


Add the URL to the the Additional Boards Manager URLs field (highlighted in red below).
Click OK to save and close Preferences.

In the Tools > Boards menu you should see the ESP32 Arduino menu. In the expanded menu, it should contain the ESP32 boards along with all the latest ESP32-S2 boards.

Now that your IDE is setup, you can continue on to loading the sketch.

**Load the Blink Sketch**

In the Tools > Boards menu you should see the ESP32 Arduino menu. In the expanded menu, look for the menu option for the ESP32S2 Dev Module, and click on it to choose it.

Open the Blink sketch by clicking through File > Examples > 01.Basics > Blink.

Once open, click Upload from the sketch window.
Once successfully uploaded, the little red LED will begin blinking once every second. At that point, you can now enter the bootloader.

**Espressif Kaluga Setup**

The Kaluga board v1.2's reset button will not work when the camera is attached. Instead, the board must be power cycled. This hardware bug was corrected for v1.3 of the board.

On the Kaluga, the camera connector shares pins with the JTAG debugging facility. It is not possible to use a JTAG debugger together with the camera on this board.

The Kaluga development kit from Espressif includes almost everything you need: The microcontroller, a camera, and an LCD.

Take the assembled Kaluga board stack (all three boards) and attach the camera at the dedicated header, making sure the pins are inserted properly.

You do not need to add any pull-up resistors; they are already provided on the Kaluga's audio daughterboard.

There are at least 3 variants of the LCD board that ship with the Kaluga:

- st7789
- ili9341
- ili9341 with rotation=90
There are no markings to distinguish the three, so you will need to try each variant until you find the one that works.

First, make sure you can see the Kaluga's CIRCUITPY drive and connect to the REPL. Open the REPL and double check that `import imagecapture` works without showing an error. Then, copy the correct bundle to your device. It will automatically reload and start displaying the image from the camera on the built-in LCD.

### Kaluga 1.3 with OV2640 and ili9341 display

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire `lib` folder and the code.py file to your CIRCUITPY drive.

---

The Kaluga development kit comes in two versions (v1.2 and v1.3); this demo is tested on v1.3. It probably won't work on v1.2 without modification.

The v1.3 development kit's LCD can have one of two chips, the ili9341 or st7789. Furthermore, there are at least 2 ILI9341 variants, one of which needs rotation=90! This demo is for the ili9341. If the display is garbled, try adding rotation=90, or try modifying it to use ST7799.

The audio board must be mounted between the Kaluga and the LCD, it provides the I2C pull-ups(!)

```python
import board
import busio
import displayio
from adafruit_ili9341 import ILI9341
import adafruit_ov2640

# Pylint is unable to see that the "size" property of OV2640_GrandCentral exists
# pylint: disable=attribute-defined-outside-init
```
# Release any resources currently in use for the displays

displayio.release_displays()

spi = busio.SPI(MOSI=board.LCD_MOSI, clock=board.LCD_CLK)
display_bus = displayio.FourWire(
    spi, command=board.LCD_D_C, chip_select=board.LCD_CS, reset=board.LCD_RST
)
display = ILI9341(display_bus, width=320, height=240, rotation=90)

bus = busio.I2C(scl=board.CAMERA_SIOC, sda=board.CAMERA_SIOD)
cam = adafruit_ov2640.OV2640(
    bus,
    data_pins=board.CAMERA_DATA,
    clock=board.CAMERA_PCLK,
    vsync=board.CAMERA_VSYNC,
    href=board.CAMERA_HREF,
    mclk=board.CAMERA_XCLK,
    mclk_frequency=20_000_000,
    size=adafruit_ov2640.OV2640_SIZE_QVGA,
)

cam.flip_x = False
cam.flip_y = True
pid = cam.product_id
ver = cam.product_version

print(f"Detected pid={pid:x} ver={ver:x}"

# cam.test_pattern = True

g = displayio.Group(scale=1)
bitmap = displayio.Bitmap(320, 240, 65536)
tg = displayio.TileGrid(
    bitmap,
    pixel_shader=displayio.ColorConverter(
        input_colorspace=displayio.Colorspace.BGR565_SWAPPED
    ),
)
g.append(tg)
display.show(g)

display.auto_refresh = False

while True:
    cam.capture(bitmap)
    bitmap.dirty()
    display.refresh(minimum_frames_per_second=0)

cam.deinit()

Your CIRCUITPY drive should resemble the screenshot below

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov2640.mpy
- adafruit_ili9341.mpy

©Adafruit Industries
CircuitPython will automatically reload and begin showing the image from the camera on the LCD. If it doesn’t, you can open up the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.4 or newer.

If the image does not fill the whole display, try removing `rotation=90` from the line beginning `display = ILI9341`. If it does not appear at all or is in reverse video, try the example for the st7789 display.

Kaluga 1.3 with OV2640 and st7789 display

If you have a Kaluga 1.3 board with an ili9341 LCD display, use the project below.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov2640.mpy
- adafruit_st7789.mpy

CircuitPython will automatically reload and begin showing the image from the camera on the LCD. If it doesn’t, you can open up the REPL to diagnose what went wrong.
Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.4 or newer.

If the image does not appear at all or is in reverse video, try the example for the ili9341. display.

The author did not have a Kaluga with an st7789 display, so this example is untested.

```python
import board
import busio
import displayio
from adafruit_st7789 import ST7789
import adafruit_ov2640

# Pylint is unable to see that the "size" property of OV2640_GrandCentral exists
# pylint: disable=attribute-defined-outside-init

# Release any resources currently in use for the displays
displayio.release_displays()

spi = busio.SPI(MOSI=board.LCD_MOSI, clock=board.LCD_CLK)
display_bus = displayio.FourWire(
    spi, command=board.LCD_D_C, chip_select=board.LCD_CS, reset=board.LCD_RST
)
display = ST7789(
    display_bus, width=320, height=240, rotation=90, reverse_bytes_in_word=True
)

bus = busio.I2C(scl=board.CAMERA_SIOC, sda=board.CAMERA_SIOD)
cam = adafruit_ov2640.OV2640(
    bus,
```

The Kaluga development kit comes in two versions (v1.2 and v1.3); this demo is tested on v1.3.

The v1.3 development kit’s LCD can have one of two chips, the ili9341 or st7789. This demo is for the ili9341. There is no marking to distinguish the two chips. If the visible portion of the display’s flexible cable has a bunch of straight lines, it may be an ili9341. If it has a bunch of wiggly traces, it may be an st7789. If in doubt, try both demos.

The audio board must be mounted between the Kaluga and the LCD, it provides the I2C pull-ups(!)
Kaluga 1.3 with OV7670 and ili9341 display

Take the assembled Kaluga board stack (all three boards) and attach the camera at the dedicated header, making sure the pins are inserted properly.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
CircuitPython will automatically reload and begin showing the image from the camera on the LCD. If it doesn't, you can open up the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.4 or newer.

If the image does not fill the whole display, try removing `rotation=90` from the line beginning `display = ILI9341`. If it does not appear at all or is in reverse video, try the example for the st7789 display.

```python
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

# The Kaluga development kit comes in two versions (v1.2 and v1.3); this demo is
tested on v1.3. It probably won't work on v1.2 without modification.

The v1.3 development kit's LCD can have one of two chips, the ili9341 or
st7789. Furthermore, there are at least 2 ILI9341 variants, one of which needs
rotation=90! This demo is for the ili9341. If the display is garbled, try adding
rotation=90, or try modifying it to use ST7799.

The camera included with the Kaluga development kit is the incompatible OV2640,
it won't work.

The audio board must be mounted between the Kaluga and the LCD, it provides the
I2C pull-ups(!)

```
Adapting to other ESP32-S2 boards

By selecting appropriate pins, you can adapt the example to work on other RP2040 boards:

- mclk, pclk, vsync, href: Free choice of any pin
- reset, shutdown: Free choice of any pin. Can omit one or both, but the initialization sequence is less reliable.
- d0...d7: Free choice of any pin

**ESP32-S2 Kaluga Dev Kit featuring ESP32-S2 WROVER**
The ESP32-S2-Kaluga-1 kit is a full featured development kit by Espressif for the ESP32-S2 that comes with everything but the kitchen sink! From TFTs to touch panels,...
https://www.adafruit.com/product/4729

**USB Type A Plug Breakout Cable with Premium Female Jumpers**
If you'd like to connect a USB-capable chip to your USB host, this cable will make the task very simple. There is no converter chip in this cable! Its basically a...
https://www.adafruit.com/product/4448

**USB Extension Cable - 3 meters / 10 ft long**
This handy USB extension cable will make it easy for you to extend your USB cable when it won't reach. The connectors are gold plated for years of reliability. We use these handy...
https://www.adafruit.com/product/993
Raspberry Pi Pico Wiring

Wiring

There's no ready-made breakout board for the OV cameras and the Raspberry Pi, so get ready to do some wiring.

This diagram shows the many connections needed. Continue below for a list.

Because the camera has two rows of connections that are 0.100 apart, a standard solderless breadboard doesn't work very well. Use a solderable breadboard or perfboard. If your solderable perfboard has a gap down the middle, you can use an IDC Breakout Helper, but these do not work well with solderless breadboards.
You can also use jumper wires (M-F to go from breadboard to camera, or F-F to go from pico pin header to camera), which is also nice because it gives you some flexibility to orient the camera differently than the LCD.

**Power & Ground**

- Connect GND of LCD, Pico, and Camera
- Connect 3V3 from Pico to Camera
- Connect 3V3 from Pico to I2C pull-up resistors (×2)
- Connect VSYS from Pico to LCD VIN

**LCD Connections**

- Connect Pico GP2 to LCD SCK
- Connect Pico GP3 to LCD MOSI
- Connect Pico GP0 to LCD D/C
- Connect Pico GP1 to LCD CS

**I2C Connections**

- Connect one I2C pull-up resistor to Pico GP8
- Connect the other I2C pull-up resistor to Pico GP9
- Connect Pico GP8 to Camera SDA
- Connect Pico GP9 to Camera SCL

**Camera Control Connections**

- Connect Pico GP7 to Camera VSYNC
- Connect Pico GP10 to Camera RESET
- Connect Pico GP11 to Camera CLOCK
- Connect Pico GP20 to Camera MCLK
- Connect Pico GP21 to Camera HREF

**Camera Data Connections**

- Connect Pico GP12 to Camera D0
- Connect Pico GP13 to Camera D1
- Connect Pico GP14 to Camera D2
- Connect Pico GP15 to Camera D3
- Connect Pico GP16 to Camera D4
• Connect Pico GP17 to Camera D5
• Connect Pico GP18 to Camera D6
• Connect Pico GP19 to Camera D7

If your camera board includes pull-up resistors, you can omit them from the breadboard. Most OV2640 camera modules include them, while most OV7670 camera modules exclude them.

Make sure you can see the Pico's CIRCUITPY drive and connect to the REPL. Open the REPL and double check that `import imagecapture` works without showing an error. Then, copy the correct bundle to your device. It will automatically reload and start displaying the image from the camera on the built-in LCD.

For the OV2640 camera

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

• code.py

And in the lib folder on your CIRCUITPY drive:

• adafruit_bus_device
• adafruit_ov2640.mpy
• adafruit_st7789.mpy
Capture an image from the camera and display it on a supported LCD.

```python
import time
from displayio import (Bitmap, Group, TileGrid, FourWire, release_displays, ColorConverter, Colorspace,
)

from adafruit_st7789 import ST7789
import board
import digitalio
import adafruit_ov2640

release_displays()

# Set up the display (You must customize this block for your display!)
spi = busio.SPI(clock=board.GP2, MOSI=board.GP3)
display_bus = FourWire(spi, command=board.GP0, chip_select=board.GP1, reset=None)
display = ST7789(display_bus, width=320, height=240, rotation=270)
display.auto_refresh = False

# Ensure the camera is shut down, so that it releases the SDA/SCL lines,
# then create the configuration I2C bus

with digitalio.DigitalInOut(board.GP10) as reset:
    reset.switch_to_output(False)
    time.sleep(0.001)
bus = busio.I2C(board.GP9, board.GP8)

# Set up the camera (you must customize this for your board!)
cam = adafruit_ov2640.OV2640(
    bus,
    data_pins=[
        board.GP12,
        board.GP13,
        board.GP14,
        board.GP15,
        board.GP16,
        board.GP17,
        board.GP18,
        board.GP19,
],
    clock=board.GP11,
    vsync=board.GP7,
    href=board.GP21,
    mclk=board.GP20,
    shutdown=None,
    reset=board.GP10,
)

width = display.width
height = display.height

cam_size = adafruit_ov2640.OV2640_SIZE QQVGA
# cam.test_pattern = True
bitmap = Bitmap(cam.width, cam.height, 65536)
```
```python
print(width, height, cam.width, cam.height)
if bitmap is None:
    raise SystemExit("Could not allocate a bitmap")

g = Group(scale=1, x=(width - cam.width) // 2, y=(height - cam.height) // 2)
tg = TileGrid(
    bitmap, pixel_shader=ColorConverter(input_colorspace=Colorspace.BGR565_SWAPPED)
)
g.append(tg)
display.show(g)

display.auto_refresh = False
while True:
    cam.capture(bitmap)
    bitmap.dirty()
    display.refresh(minimum_frames_per_second=0)
```

CircuitPython will automatically reload and begin showing the image from the camera on the LCD. If it doesn't, you can open up the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.0 or newer.

For the OV7670 camera

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov7670.mpy
- adafruit_st7789.mpy
Capture an image from the camera and display it on a supported LCD.

```python
import time
from displayio import (Bitmap, Group, TileGrid, FourWire, release_displays, ColorConverter, Colorspace,
)
from adafruit_st7789 import ST7789
import board
import busio
import digitalio
from adafruit_ov7670 import (OV7670,
    OV7670_SIZE_DIV1,
    OV7670_SIZE_DIV16,
)

# Set up the display (You must customize this block for your display!)
release_displays()
spi = busio.SPI(clock=board.GP2, MOSI=board.GP3)
display_bus = FourWire(spi, command=board.GPIO, chip_select=board.GPIO, reset=None)
display = ST7789(display_bus, width=320, height=240, rotation=270)

# Ensure the camera is shut down, so that it releases the SDA/SCL lines,
# then create the configuration I2C bus
with digitalio.DigitalInOut(board.GPIO) as reset:
    reset.switch_to_output(False)
    time.sleep(0.001)
    bus = busio.I2C(board.GPIO, board.GPIO)

# Set up the camera (you must customize this for your board!)
cam = OV7670(
    bus,
data_pins=[
        board.GPIO12,
        board.GPIO13,
        board.GPIO14,
        board.GPIO15,
        board.GPIO16,
        board.GPIO17,
```
board.GP18,
board.GP19,
], # [16] [org] etc
clock=board.GP11, # [15] [blk]
vsync=board.GP7, # [10] [brn]
href=board.GP21, # [27/014] [red]
mclk=board.GP20, # [16/015]
shutdown=None,
reset=board.GP10,
) # [14]

width = display.width
height = display.height

# cam.test_pattern = OV7670_TEST_PATTERN_COLOR_BAR

bitmap = None
# Select the biggest size for which we can allocate a bitmap successfully, and
# which is not bigger than the display
for size in range(OV7670_SIZE_DIV1, OV7670_SIZE_DIV16 + 1):
    cam.size = size
    if cam.width > width:
        continue
    if cam.height > height:
        continue
    try:
        bitmap = Bitmap(cam.width, cam.height, 65536)
        break
    except MemoryError:
        continue

print(width, height, cam.width, cam.height)
if bitmap is None:
    raise SystemExit("Could not allocate a bitmap")

g = Group(scale=1, x=(width - cam.width) // 2, y=(height - cam.height) // 2)
tg = TileGrid(
    bitmap, pixel_shader=ColorConverter(input_colorspace=Colorspace.RGB565_SWAPPED)
)
g.append(tg)
display.show(g)

t0 = time.monotonic_ns()
display.auto_refresh = False
while True:
    cam.capture(bitmap)
    bitmap.dirty()
    display.refresh(minimum_frames_per_second=0)
    t1 = time.monotonic_ns()
    print("fps", 1e9 / (t1 - t0))
t0 = t1

CircuitPython will automatically reload and begin showing the image from the camera
on the LCD. If it doesn’t, you can open up the REPL to diagnose what went wrong.
Double check that you copied all the files from the bundle, and that you have a
compatible build of CircuitPython installed, 7.0.0-beta.0 or newer.
Adapting to other RP2040 boards

By selecting appropriate pins, you can adapt the example to work on other RP2040 boards which are supported by CircuitPython and have enough pins exposed for the connections below:

- mclk, pclk, vsync, href: Free choice of any pin
- reset, shutdown: Free choice of any pin. Can omit one or both, but the initialization sequence is less reliable.
- d0...d7: These 8 pins must be consecutive in the "IO##" ordering, so you could use IO3...IO10, IO9...IO16, etc.

Raspberry Pi Pico RP2040
The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...
https://www.adafruit.com/product/4864

IDC Breakout Helper - 2x13 (26 pin)
The "poor woman's" Raspberry Pi cobbler! This combo of 2x13 pin (0.1 spaced) header and socket is for our popular GPIO...
https://www.adafruit.com/product/2101
Premium Female/Female Jumper Wires - 40 x 6"
Handy for making wire harnesses or jumpering between headers on PCB's. These premium jumper wires approximately 6" (150mm) long and come in a 'strip' of 40 (4 pieces... https://www.adafruit.com/product/266

Premium Female/Male 'Extension' Jumper Wires - 40 x 6" (150mm)
Handy for making wire harnesses or jumpering between headers on PCB's. These premium jumper wires are 6" (150mm) long and come in a 'strip' of 40 (4 pieces of each of... https://www.adafruit.com/product/826

Through-Hole Resistors - 2.2K ohm 5% 1/4W - Pack of 25
ΩMG! You're not going to be able to resist these handy resistor packs! Well, axially, they do all of the resisting for you! This is a 25 Pack of 2.2K... https://www.adafruit.com/product/2782
Grand Central M4 Wiring

Before you can use an OV7670 camera with the Grand Central M4, you have to perform some hardware modifications which are detailed on their own page. (https://adafruit.it/TAJ) These modifications are not easily reversible and will make it difficult to use the camera modules on other boards.

Then, position the camera so that it sticks off the right side of the Grand Central PCB and the SDA/SCL pins insert into the third row of the header and insert it carefully.

Align the pins of the TFT Shield and insert it into the Grand Central too.

Connect GND and 3V3 from the TFT Shield to the modified pins of the OV7670.

Now, make sure you can see the Grand Central's CIRCUITPY drive and connect to the REPL. Open the REPL and double check that `import imagecapture` works without showing an error. Then, copy the correct bundle to your device. It will automatically reload and start displaying the image from the camera on the built-in LCD.

We've found the ParallelImageCapture function to be very unstable under CircuitPython. It works sometimes with OV7670 and never or almost never with OV2640. If you need to use the SAM D5x/E5x microcontroller, consider using the Arduino version instead of CircuitPython.
Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov7670.mpy
- adafruit_st7789.mpy

CircuitPython will automatically reload and begin showing the image on the camera on the LCD. If it doesn't, you can open up the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.0 or newer.

Adapting to other SAM D5x/E5x boards

By selecting appropriate pins, you can adapt the example to work on other SAM D5x/E5x boards supported by CircuitPython which expose enough pins for the connections below:

- mclk: Free choice of any PWM pin
- pclk, vsync, href: Only the specific PCC pins may be used.
reset, shutdown: Free choice of any pin. Can omit one or both, but the initialization sequence is less reliable.

d0…d7: Only the specific PCC pins may be used.

Adafruit Grand Central M4 Express featuring the SAMD51
Are you ready? Really ready? Cause here comes the Adafruit Grand Central featuring the Microchip ATSAMD51. This dev board is so big, it's not...
https://www.adafruit.com/product/4064

Adafruit 1.8" Color TFT Shield w/microSD and Joystick
This lovely little shield is the best way to add a small, colorful and bright display to any project. We took our popular 1.8" TFT breakout board and remixed it into an Arduino...
https://www.adafruit.com/product/802

Through-Hole Resistors - 2.2K ohm 5% 1/4W - Pack of 25
ΩMG! You're not going to be able to resist these handy resistor packs! Well, axially, they do all of the resisting for you!This is a 25 Pack of 2.2K...
https://www.adafruit.com/product/2782
Working with Image Data

RGB Data

When the OV7670 is working in RGB mode (the default), it is in a 16-bit format called "RGB565-swapped". This means that every pixel is treated as a 16-bit number, with the left and right 8 bits "swapped":

![RGB565-swapped pixel diagram]

The OV2640 uses a 16-bit format called "BGR565-swapped", which switches the positions of the red and blue values within the pixel.

Happily, displayio's ColorConverter is able to deal with this format natively, but it is tough to write efficient Python code to work with it. In some circumstances, the CircuitPython version of the `ulab` library can help.

In the examples before now, the "capture" operation worked with a bitmap object, but it can also work with a `ulab` array:

```python
from ulab import numpy as np
arr = np.zeros((80, 60), dtype=np.uint16)
camera.capture(arr)
arr.byteswap(inplace=True)
```

After using byteswap, the order of the values within the pixel is now:

![BGR565-swapped pixel diagram]
making it possible to extract an individual red, green, or blue value with bit shifts (this code is for RGB565):

```python
def R(pixel):
    return pixel >> 11

def G(pixel):
    return (pixel & 0b11111100000) >> 5

def B(pixel):
    return pixel & 0b11111

print("The green value of the pixel at (0,0) is", G(arr[0,0]))
```

Wholesale modifications of the pixel data can be done with ulab. For instance, to invert colors in the whole image,

```python
arr[:] = ~arr
```

That's exactly what the following program does on the Espressif Kaluga with OV2640.

Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in / of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov2640.mpy
- adafruit_ili9341.mpy

CircuitPython will automatically reload and begin showing the image from the camera on the LCD. If it doesn't, you can open up the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.0 or newer.
The Kaluga development kit comes in two versions (v1.2 and v1.3); this demo is
tested on v1.3. It probably won't work on v1.2 without modification.

The v1.3 development kit's LCD can have one of two chips, the ili9341 or
st7789. Furthermore, there are at least 2 ILI9341 variants, one of which needs
rotation=90! This demo is for the ili9341. If the display is garbled, try adding
rotation=90, or try modifying it to use ST7799.

The camera included with the Kaluga development kit is the incompatible OV2640,
it won't work.

The audio board must be mounted between the Kaluga and the LCD, it provides the
I2C pull-ups(!)
YUV Data

YUV is another representation of image data. Y represents the luminance (brightness) of a pixel, while U and V represent the color information. [Wikipedia has an article about YUV](https://adafruit.it/TAK), if you'd like to know more.

The most useful thing about YUV data is that it allows easily extracting a greyscale image, by using the data from every other byte. That is how this "image in a terminal window" example works on the Espressif Kaluga with OV2640.

The OV2640 can be placed in YUV mode by assigning a property:

```python
cam.colorspace = adafruit_ov2640.OV2640_COLOR_YUV
```

The OV7670 is similar:

```python
cam.colorspace = adafruit_ov7670.OV7670_COLOR_YUV
```

To demonstrate the YUV mode, the simplest demo converts a low resolution camera image into lo-fi ASCII art.
Click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, and copy the entire lib folder and the code.py file to your CIRCUITPY drive.

Your CIRCUITPY drive should resemble the image.

You should have in `/` of the CIRCUITPY drive:

- code.py

And in the lib folder on your CIRCUITPY drive:

- adafruit_bus_device
- adafruit_ov2640.mpy

CircuitPython will automatically reload. Connect to the REPL and the image will be shown in the finest 3-bit ASCII art. If it doesn’t, use the REPL to diagnose what went wrong. Double check that you copied all the files from the bundle, and that you have a compatible build of CircuitPython installed, 7.0.0-beta.0 or newer.

If the REPL updates very slowly, one trick is to reset the Kaluga so that it forgets about the LCD display if you ran one of the demos that uses the LCD. Updating the LCD display is much slower than sending data over the USB CDC connection.
"""Capture an image from the camera and display it as ASCII art.

This demo is designed to run on the Kaluga, but you can adapt it to
other boards by changing the constructors for `bus` and `cam`
appropriately.

The camera is placed in YUV mode, so the top 8 bits of each color
value can be treated as "greyscale".

It's important that you use a terminal program that can interpret
"ANSI" escape sequences. The demo uses them to "paint" each frame
on top of the previous one, rather than scrolling.

Remember to take the lens cap off, or un-comment the line setting
the test pattern!
"""

```python
import sys
import time
import busio
import board

import adafruit_ov2640

bus = busio.I2C(scl=board.CAMERA_SIOC, sda=board.CAMERA_SIOD)
cam = adafruit_ov2640.OV2640(bus,
data_pins=board.CAMERA_DATA,
clock=board.CAMERA_PCLK,
vsync=board.CAMERA_VSYNC,
href=board.CAMERA_HREF,
mclk=board.CAMERA_XCLK,
mclk_frequency=20_000_000,
size=adafruit_ov2640.OV2640_SIZE_QQVGA)
cam.colorspace = adafruit_ov2640.OV2640_COLOR_YUV
cam.flip_y = True
# cam.test_pattern = True

buf = bytearray(2 * cam.width * cam.height)
chars = b" .:-=+*#%@"
remap = [chars[i * (len(chars) - 1) // 255] for i in range(256)]

width = cam.width
row = bytearray(2 * width)
sys.stdout.write(\033[2J"

while True:
cam.capture(buf)
    for j in range(cam.height // 2):
        sys.stdout.write(f"\033[{j}H")
        for i in range(cam.width // 2):
            row[i * 2] = row[i * 2 + 1] = remap[buf[4 * (width * j + i)]]
        sys.stdout.write(row)
sys.stdout.write(\033[\K")
sys.stdout.write("\033[J")
time.sleep(0.05)
```

Test modes

These cameras have a test mode which shows color bars.
You can activate the test pattern mode on the OV2640 camera like so:

```python
cam.test_pattern = True
```

It's a little different on the OV7670:

```python
cam.test_pattern = adafruit_ov7670.OV7670_TEST_PATTERN_COLOR_BAR
```

---

**Capturing JPEG data (OV2640)**

The OV2640 camera module can also capture JPEG images up to 2 megapixels (1600x1200 pixels). This still requires a large buffer (of width×height÷5 bytes, or 384,000 bytes for a 2-megapixel image), so the demo below is coded for the Kaluga development board together with the MicroSD card breakout board+

CircuitPython doesn't encode or decode the JPEG image itself, it just uses a JPEG-encoded file produced by the camera and stores it on the SD card.

Make the following connections for the SD card breakout:

- IO18 to CLK
- IO14 to DI
- IO17 to DO
- IO12 to CS
- GND to GND
While the demo runs, it will show a live image on the LCD. When you hold the REC button, it will save the picture as a jpeg image to the inserted SD card. Note that because the REC button is only polled when the screen is not updating, you have to hold it, not just quickly press it.

The exposure in JPEG mode doesn't match what's shown on the LCD. We hope a future enhancement of the OV2640 library will improve this.

Code

```python
# SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
# SPDX-FileCopyrightText: Copyright (c) 2021 Jeff Epler for Adafruit Industries
#
# SPDX-License-Identifier: Unlicense

""
Display an image on the LCD, then record an image when the REC button is pressed/ held.

The Kaluga development kit comes in two versions (v1.2 and v1.3); this demo is tested on v1.3.

The audio board must be mounted between the Kaluga and the LCD, it provides the I2C pull-ups(!)

The v1.3 development kit's LCD can have one of two chips, the ili9341 or st7789. Furthermore, there are at least 2 ILI9341 variants, one of which needs rotation=90! This demo is for the ili9341. If the display is garbled, try adding rotation=90, or try modifying it to use ST7799.

This example also requires an SD card breakout wired as follows:
* IO18: SD Clock Input
* IO17: SD Serial Output (MISO)
* IO14: SD Serial Input (MOSI)
* IO12: SD Chip Select

Insert a CircuitPython-compatible SD card before powering on the Kaluga. Press the "Record" button on the audio daughterboard to take a photo.
""

```import os
import analogio
import board
import busio
import displayio
import sdcardio
import storage
from adafruit_ili9341 import ILI9341
import adafruit_ov2640
V_MODE = 1.98
V_RECORD = 2.41
a = analogio.AnalogIn(board.IO6)

# Release any resources currently in use for the displays
displayio.release_displays()

SPI = busio.SPI(MOSI=board.LCD_MOSI, clock=board.LCD_CLK)
display_bus = displayio.FourWire(
    SPI, command=board.LCD_D_C, chip_select=board.LCD_CS, reset=board.LCD_RST
)
display = ILI9341(display_bus, width=320, height=240, rotation=90)

bus = busio.I2C(scl=board.CAMERA_SIOC, sda=board.CAMERA_SIOD)
cam = adafruit_ov2640.OV2640(bus,
    data_pins=board.CAMERA_DATA,
    clock=board.CAMERA_PCLK,
    vsync=board.CAMERA_VSYNC,
    href=board.CAMERA_HREF,
    mclk=board.CAMERA_XCLK,
    mclk_frequency=20_000_000,
    size=adafruit_ov2640.OV2640_SIZE_QVGA,
)

cam.flip_x = False
cam.flip_y = True
pid = cam.product_id
ver = cam.product_version
print(f"Detected pid={pid:x} ver={ver:x}"

# cam.test_pattern = True

g = displayio.Group(scale=1)
bitmap = displayio.Bitmap(320, 240, 65536)
tg = displayio.TileGrid(bitmap,
    pixel_shader=displayio.ColorConverter(input_colorspace=displayio.Colorspace.BGR565_SWAPPED,
)
)
g.append(tg)
display.show(g)

display.auto_refresh = False

sd_spi = busio.SPI(clock=board.IO18, MOSI=board.IO14, MISO=board.IO17)
sd_cs = board.IO12
sdcard = sdcardio.SDCard(sd_spi, sd_cs)
vfs = storage.VfsFat(sdcard)
storage.mount(vfs, "/sd"

def exists(filename):
    try:
        os.stat(filename)
        return True
    except OSError:
        return False

_image_counter = 0

def open_next_image():
    global _image_counter
    # pylint: disable=global-statement
    while True:
        filename = f"/sd/img{image_counter:04d}.jpg"
        _image_counter += 1
        if exists(filename):
            continue
        print("#", filename)
        return open(filename, "wb")
    # pylint: disable=consider-using-with
def capture_image():
    old_size = cam.size
    old_colorspace = cam.colorspace

    try:
        cam.size = adafruit_ov2640.OV2640_SIZE_UXGA
        cam.colorspace = adafruit_ov2640.OV2640_COLOR_JPEG
        b = bytearray(cam.capture_buffer_size)
        jpeg = cam.capture(b)

        print(f"Captured {len(jpeg)} bytes of jpeg data")
        with open_next_image() as f:
            f.write(jpeg)
    finally:
        cam.size = old_size
        cam.colorspace = old_colorspace

display.auto_refresh = False
while True:
    a_voltage = a.value * a.reference_voltage / 65535  # pylint: disable=no-member
    record_pressed = abs(a_voltage - V_RECORD) < 0.05
    if record_pressed:
        capture_image()
        cam.capture(bitmap)
        bitmap.dirty()
        display.refresh(minimum_frames_per_second=0)

Parts

ESP32-S2 Kaluga Dev Kit featuring ESP32-S2 WROVER
The ESP32-S2-Kaluga-1 kit is a full featured development kit by Espressif for the ESP32-S2 that comes with everything but the kitchen sink! From TFTs to touch panels,...
https://www.adafruit.com/product/4729

MicroSD card breakout board+
Not just a simple breakout board, this microSD adapter goes the extra mile - designed for ease of use. Onboard 5v-&gt; 3v regulator provides 150mA for power-hungry...
https://www.adafruit.com/product/254
8GB Class 10 SD/MicroSD Memory Card - SD Adapter Included
Add mega-storage in a jiffy using this 8 GB micro-SD card. It comes with a SD adapter so you can use it with any of our shields or adapters! Preformatted to FAT so it works out of the...
https://www.adafruit.com/product/2692

USB Type A Plug Breakout Cable with Premium Female Jumpers
If you'd like to connect a USB-capable chip to your USB host, this cable will make the task very simple. There is no converter chip in this cable! Its basically a...
https://www.adafruit.com/product/4448

USB Extension Cable - 3 meters / 10 ft long
This handy USB extension cable will make it easy for you to extend your USB cable when it won't reach. The connectors are gold plated for years of reliability. We use these handy...
https://www.adafruit.com/product/993

Documentation: adafruit_ov2640
Documentation: adafruit_ov2640 (https://adafruit.it/TnE)

Documentation: adafruit_ov7670
Documentation: adafruit_ov7670 (https://adafruit.it/Toa)
Documentation: adafruit_ov5640

Documentation: adafruit_ov5640 (https://adafruit.it/VfR)