Get started with the Dev Board Mini

This page shows you how to set up the Coral Dev Board Mini, a single-board computer that accelerates machine learning models with the on-board Edge TPU.

The board in your box is already flashed with Mendel Linux, so the setup simply requires connecting to the board's shell console and updating some software. Then we'll show you how to run a TensorFlow Lite model on the board. From there, you can try any of our other examples—perhaps connect a camera and run an object detection model, pose detection model, keyphrase detector, and more.
Warning: Use caution when handling the board to avoid electrostatic discharge or contact with conductive materials (metals). Failure to properly handle the board can result in a short circuit, electric shock, serious injury, death, fire, or damage to your board and other property.

For a "choose your own adventure" walk-through with the Dev Board Mini, check out the following video.
https://youtu.be/uF5FEzwOgzM

1: Gather requirements

To get started, be sure you have the following:

- A host computer running Linux (recommended), Mac, or Windows 10
    - Python 3 installed
- One USB-C power supply (2 A / 5 V), such as a phone charger
- One USB-C to USB-A cable (to connect to your computer)
- An available Wi-Fi internet connection

Notice that this board is intended as a headless device. That is, you typically do not connect a keyboard and monitor, and use it as a desktop computer. So, this guide is focused on connecting to the board remotely with a secure shell terminal.

However, the Mendel OS does include a simple desktop with a terminal application. So if you prefer, you can connect a monitor to the micro-HDMI port, and connect a keyboard and mouse to the USB OTG port.

Other Windows requirements

For compatibility with our command-line instructions and the shell scripts in our example code, we recommend you use the Git Bash terminal on Windows, which is included with Git for Windows. You should install it now and open the Git Bash terminal for all the command-line instructions below.

However, as of Git for Windows v2.28, the Git Bash terminal cannot directly access Python. So after you install Git Bash, open it and run the following commands to make Python accessible:

```
echo "alias python3='winpty python3.exe'" >> ~/.bash_profile

source ~/.bash_profile
```

2: Install MDT

MDT is a command line tool for your host computer that helps you interact with the Dev Board Mini. For example, MDT can list connected devices, install Debian packages on the board, and open a shell terminal on the board.

You can install MDT on your host computer follows:

```
python3 -m pip install --user mendel-development-tool
```

You might see a warning that `mdt` was installed somewhere that's not in your `PATH` environment variable. If so, be sure you add the given location to your `PATH`, as appropriate for your operating system. If you're on Linux, you can add it like this:

```
echo 'export PATH="$PATH:$HOME/.local/bin"' >> ~/.bash_profile

source ~/.bash_profile
```

Windows users: If you're using Git Bash, you'll need an alias to run MDT. Run this in your Git Bash terminal:

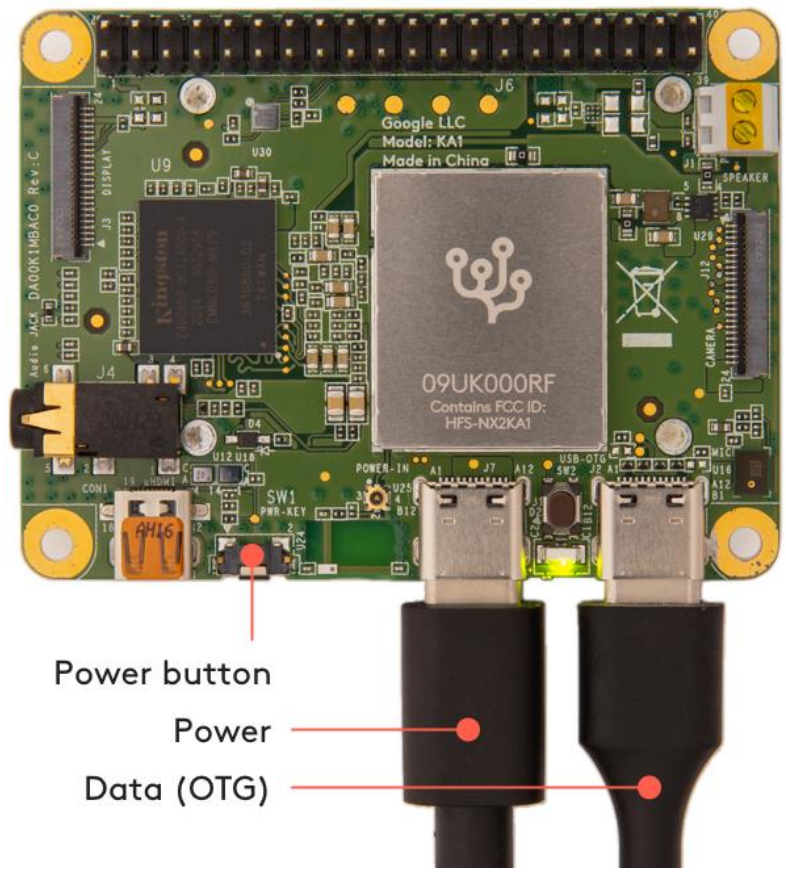
```
echo "alias mdt='winpty mdt'" >> ~/.bash_profile

source ~/.bash_profile
```

3: Plug in the board

1. Connect your power supply to the board's USB power plug (the left plug, as shown in figure 1) and connect it to an outlet.
2. Connect your USB data cable to the other USB plug and to your host computer.

When you connect the USB cable to your computer, the board automatically boots up, and the board's LED turns green. It then takes 20-30 seconds for the system to boot up.

Note: We've released a system update that instead turns the LED orange during boot-up, and then green when the system is booted and ready for login. (Further below, you'll find instruction to install this update.)



About the USB ports

Both USB ports can power the board, but they behave differently:

- **USB power port (left):** This port supports power input only (no data). If you connect only this port, you must firmly press the power button to boot the board. Once the board is booted, you can press the power button to safely shut it down.
- **USB OTG port (right):** This port supports power input and USB data (as host or device). If you connect power here, the board boots up automatically. If you press the power button, the board reboots—you cannot fully shut down the board when the OTG port is connected to power (even if you shut down the operating system, it will reboot).

So to safely power-off the board, you must deliver power with the power port only, and then you can toggle power with the power button. Whereas, if you want the board always on, deliver power with the OTG port, but beware that you then cannot safely power off the board unless you transition power to the power port.

4: Connect to the board's shell via MDT

MDT provides an easy way to establish an SSH connection with the board over USB, as follows.

Mac users: Beginning with macOS Catalina (10.15), you cannot create an SSH connection over USB, so the following MDT commands won't work until you get the board online and install your SSH key. See the instructions to use MDT on macOS, and then return here to complete the setup.

1. Make sure MDT can see your device by running this command from your host computer:
   ```
   mdt devices
   ```
   You should see output showing your board hostname and IP address (your name will be different):
   ```
   fun-zebra        (192.168.100.2)
   ```
   If you don't see your device, it might be because the system is still booting up. Wait a moment and try again.

2. Now to open the device shell, run this command:
   ```
   mdt shell
   ```
   After a moment, you should see the board's shell prompt, which looks like this:
   ```
   mendel@fun-zebra:~$
   ```

Now you're in the board!

Note: Your board name will be different. The default username is always "mendel" but the board's hostname is randomly generated the first time it boots (after a new flashing). We do this to ensure that each device within a local fleet is likely to have a unique name. Of course, you can change this name using standard Linux hostname tooling (such as `hostname`).

For details about how the shell connection works and what else you can do with MDT, see the MDT documentation.

5: Connect to the internet

You'll need the board online to download system updates, TensorFlow models, and code examples.

1. Select a Wi-Fi network on the board. You can use one of two option:

   - Enter your network name and password with this command:
     ```
     nmcli dev wifi connect <NETWORK_NAME> password <PASSWORD> ifname wlan0
     ```
   - Or use an interactive menu to select the network (this does not work with Git Bash on Windows):

     1. Launch the network manager menu:

```
nmtui
```

2. Use your keyboard to select **Activate a connection**.
3. Select a network from the list, enter a password if required, and then quit the menu.

2. Verify your connection with this command:

```
nmcli connection show
```

You should see your selected network listed in the output. For example:

```
NAME               UUID                                  TYPE              DEVICE
MyNetworkName      61f5d6b2-5f52-4256-83ae-7f148546575a  wifi              wlan0
```

You might see other connections listed, such as the Ethernet-over-USB connection you're using for MDT.

Tip: Next time you boot up the board, you can connect to the shell over Wi-Fi instead of connecting the USB cable. Just power the board and it will log onto your Wi-Fi automatically, and then you can connect wirelessly with `mdt shell`.

6: Update the Mendel software

Some of our software updates are delivered with Debian packages separate from the system image, so make sure you have the latest software by running the following commands in the board's shell terminal:

```
sudo apt-get update

sudo apt-get dist-upgrade

sudo reboot now
```

Reboot is required for some kernel changes to take effect.

Help! If you see an error that certificate verification failed, it's probably because the system has not fetched the correct date yet. You can manually set the date with a command like this: `sudo date +%Y%m%d -s "20210121"`. Then try to upgrade again.

When the board's LED turns green after reboot, it should be ready for login, so you can reconnect from your computer with MDT:

```
mdt shell
```

Now you're ready to run a TensorFlow Lite model on the Edge TPU!

Note: The `dist-upgrade` command updates all system packages for your current Mendel version. If you want to upgrade to a newer version of Mendel, you need to flash a new system image.

7: Run a model using the PyCoral API

To get you started, we've created some simple examples that demonstrate how to perform an inference on the Edge TPU using the TensorFlow Lite API (assisted by the PyCoral API).

Execute the following commands from the Dev Board Mini shell to run our image classification example:

1. Download the example code from GitHub:

```
2. mkdir coral && cd coral
3.
4. git clone https://github.com/google-coral/pycoral.git
5.
   cd pycoral
```

6. Download the model, labels, and bird photo:

```
bash examples/install_requirements.sh classify_image.py
```

7. Run the image classifier with the bird photo (shown in figure 2):

```
8.  python3 examples/classify_image.py \
9.  --model test_data/mobilenet_v2_1.0_224_inat_bird_quant_edgetpu.tflite \
10. --labels test_data/inat_bird_labels.txt \
    --input test_data/parrot.jpg
```

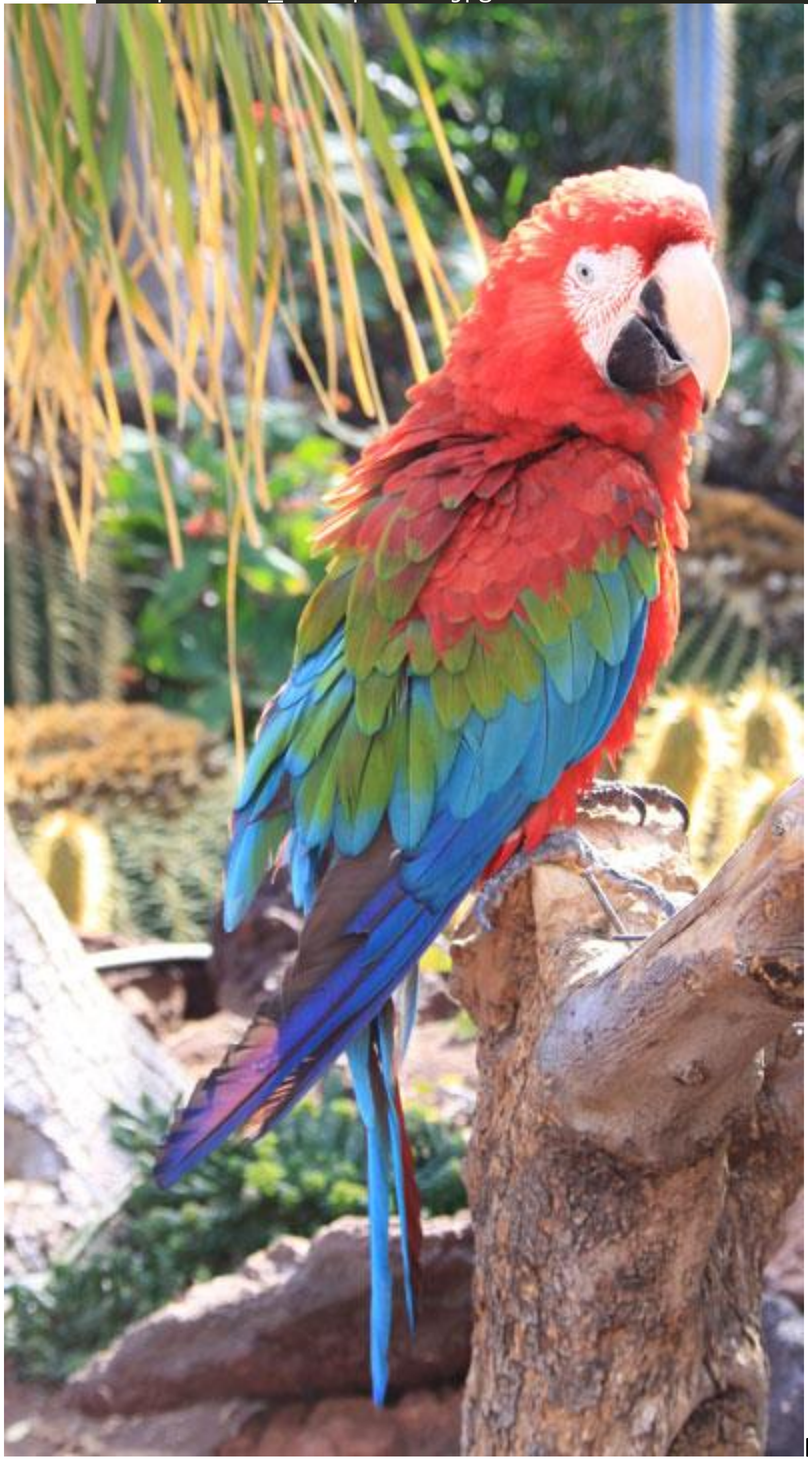


Figure 2. parrot.jpg

You should see results like this:

```
----INFERENCE TIME----
Note: The first inference on Edge TPU is slow because it includes loading the model into Edge TPU
memory.
158.3ms
```

```
14.3ms
14.3ms
14.5ms
14.3ms
-------RESULTS--------
Ara macao (Scarlet Macaw): 0.75781
```

Congrats! You just performed an inference on the Edge TPU using TensorFlow Lite.

To demonstrate varying inference speeds, the example repeats the same inference five times. It prints the time to perform each inference and then the top classification result (the label name and the confidence score, from 0 to 1.0).

To learn more about how the code works, take a look at the classify_image.py source code and read our guide about how to run inference with TensorFlow Lite.

8: Safely shut down the board

Caution: Do not unplug the board while it is running. Doing so could corrupt the system image.

You should shut down the Dev Board Mini as follows:

1. Be sure the board is connected to power through the USB power port (see figure 1).
2. Unplug the USB OTG cable (if connected). This is important, because if the OTG port is connected, the board will immediately reboot upon shutdown.
3. Either press the board power button or run `sudo shutdown now` from the board terminal.
4. When the board's LED turns off, you can unplug the power supply.

Next steps

To run some other types of neural networks, check out our example projects, including examples that perform real-time object detection, pose estimation, keyphrase detection, on-device transfer learning, and more.

To run vision models using the Coral Camera, check out the camera setup guide, and our other camera examples.

If you want to train your own TensorFlow model for the Edge TPU, try these tutorials:

- Retrain an image classification model (MobileNet) (runs in Google Colab)
- Retrain an object detection model (EfficientDet) (runs in Google Colab)
- More model retraining tutorials are available on GitHub.
- Or to build your own model that's compatible with the Edge TPU, read TensorFlow Models on the Edge TPU

If you'd like to browse the Mendel source code and build it yourself for the Dev Board Mini, take a look at the Mendel Getting Started guide.

Connect to the Dev Board Mini I/O pins

The Dev Board Mini provides access to several peripheral interfaces through the 40-pin expansion header, including GPIO, I2C, UART, and SPI. This page describes how you can interact with devices connected to these pins.

Because the Dev Board Mini runs Mendel Linux, you can interact with the pins from user space using Linux interfaces such as device files (`/dev`) and sysfs files (`/sys`). There are also several API libraries you can use to program the peripherals connected to these pins. This page describes a few API options, including python-periphery, Adafruit Blinka, and libgpiod.

Note: If most of your GPIO experience comes from boards like the Raspberry Pi, beware that—although the pin layout on this board might be the same—the vast majority of GPIO APIs out there are board-specific or must be updated for new boards. So the APIs you've used before probably won't work with this board.

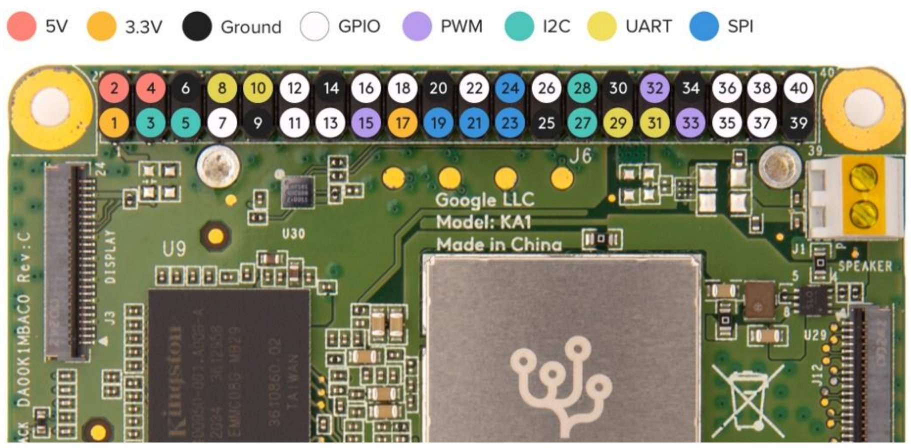


Figure 1. Default pin functions on the 40-pin header

Warning: When handling the GPIO pins, be cautious to avoid electrostatic discharge or contact with conductive materials (metals). Failure to properly handle the board can result in a short circuit, electric shock, serious injury, death, fire, or damage to your board and other property.

Header pinout

Table 1 shows the header pinout, including the device or sysfs file for each pin, plus the character device numbers. If you'd like to see the SoC pin names instead, refer to section 4.9 in the Dev Board Mini datasheet. You can also see the pinout by typing `pinout` from the board's shell terminal.

All pins are powered by the 3.3 V power rail, with a max current of ~16 mA on most pins (although the default configuration is 2-4 mA max for most pins).

Note: Pins 8, 10, 29, 31, and 37 should not be used to drive resistive loads directly, due to weak drive strength.

Table 1. Pinout for the Dev Board Mini 40-pin header, with device file names and character device IDs (chip_number, line_number)

Chip, line	Device path	Pin function	Pin	Pin	Pin function	Device path	Chip, line
		+3.3 V	1	2	+5 V		
	/dev/i2c-3	I2C1_SDA	3	4	+5 V		
	/dev/i2c-3	I2C1_SCL	5	6	Ground		
0, 22	/sys/class/gpio/gpio409	GPIO22	7	8	UART0_TX	/dev/ttyS0	
		Ground	9	10	UART0_RX	/dev/ttyS0	
0, 9	/sys/class/gpio/gpio396	GPIO9	11	12	GPIO36	/sys/class/gpio/gpio423	0, 36
0, 10	/sys/class/gpio/gpio397	GPIO10	13	14	Ground		
0, 2	/sys/class/pwm/pwmchip0/pwm2	PWM_C	15	16	GPIO0	/sys/class/gpio/gpio387	0, 0

Table 1. Pinout for the Dev Board Mini 40-pin header, with device file names and character device IDs (chip_number, line_number)

Chip, line	Device path	Pin function	Pin	Pin	Pin function	Device path	Chip, line
		+3.3 V	17	18	GPIO1	/sys/class/gpio/gpio388	0, 1
	/dev/spidev0	SPI_MO	19	20	Ground		
	/dev/spidev0	SPI_MI	21	22	GPIO7	/sys/class/gpio/gpio394	0, 7
	/dev/spidev0	SPI_CLK	23	24	SPI_CSB	/dev/spidev0.0	
		Ground	25	26	GPIO8	/sys/class/gpio/gpio395	0, 8
	/dev/i2c-0	I2C2_SDA	27	28	I2C2_SCL	/dev/i2c-0	
	/dev/ttyS1	UART1_TX	29	30	Ground		
	/dev/ttyS1	UART1_RX	31	32	PWM_A	/sys/class/pwm/pwmchip0/pwm0	0, 0
0, 1	/sys/class/pwm/pwmchip0/pwm1	PWM_B	33	34	Ground		
0, 37	/sys/class/gpio/gpio424	GPIO37	35	36	GPIO13	/sys/class/gpio/gpio400	0, 13
0, 45	/sys/class/gpio/gpio432	GPIO45	37	38	GPIO38	/sys/class/gpio/gpio425	0, 38
		Ground	39	40	GPIO39	/sys/class/gpio/gpio426	0, 39

Program with python-periphery

The python-periphery library provides a generic Linux interface that's built atop the sysfs and character device interface, providing APIs to control GPIO, PWM, I2C, SPI, and UART pins.
By default, the python-periphery package is included with the Mendel system image on the Dev Board Mini. So no installation is required.

The following sections show how to instantiate an object for each pin on the Dev Board Mini header.

GPIO

You can instantiate a GPIO object using either the sysfs path (deprecated) or the character device numbers.

The following code instantiates each GPIO pin as input using the character devices:

```
gpio22 = GPIO("/dev/gpiochip0", 22, "in")   # pin 7
gpio9 = GPIO("/dev/gpiochip0", 9, "in")     # pin 11
gpio36 = GPIO("/dev/gpiochip0", 36, "in")   # pin 12
gpio10 = GPIO("/dev/gpiochip0", 10, "in")   # pin 13
gpio0 = GPIO("/dev/gpiochip0", 0, "in")     # pin 16
gpio1 = GPIO("/dev/gpiochip0", 1, "in")     # pin 18
gpio7 = GPIO("/dev/gpiochip0", 7, "in")     # pin 22
gpio8 = GPIO("/dev/gpiochip0", 8, "in")     # pin 26
gpio37 = GPIO("/dev/gpiochip0", 37, "in")   # pin 35
gpio13 = GPIO("/dev/gpiochip0", 13, "in")   # pin 36
gpio45 = GPIO("/dev/gpiochip0", 45, "in")   # pin 37
gpio38 = GPIO("/dev/gpiochip0", 38, "in")   # pin 38
gpio39 = GPIO("/dev/gpiochip0", 39, "in")   # pin 40
```

Note: Do not use pin 37 (gpio432) to drive resistive loads directly, due to weak drive strength.

For example, here's how to turn on an LED when you push a button:

```
from periphery import GPIO

led = GPIO("/dev/gpiochip0", 39, "out")   # pin 40
button = GPIO("/dev/gpiochip0", 13, "in")   # pin 36

try:
  while True:
    led.write(button.read())
finally:
  led.write(False)
  led.close()
  button.close()
```

For more examples, see the periphery GPIO documentation.

PWM

The following code instantiates each of the PWM pins:

```
pwm_a = PWM(0, 0)   # pin 32
pwm_b = PWM(0, 1)   # pin 33
pwm_c = PWM(0, 2)   # pin 15
```

For usage examples, see the periphery PWM documentation.

I2C

The following code instantiates each of the I2C ports:

```
i2c1 = I2C("/dev/i2c-3")   # pins 3/5
i2c2 = I2C("/dev/i2c-0")   # pins 27/28
```

For usage examples, see the periphery I2C documentation.

SPI

The following code instantiates the SPI port:

```
spi0 = SPI("/dev/spidev0.0", 0, 10000000)   # pins 19/21/23/14 (Mode 0, 10MHz)
```

For usage examples, see the periphery SPI documentation.

UART

The following code instantiates each of the UART ports:

```
uart0 = Serial("/dev/ttyS0", 115200)   # pins 8/10 (115200 baud)
uart1 = Serial("/dev/ttyS1", 9600)     # pins 29/31 (9600 baud)
```

For usage examples, see the periphery Serial documentation.

Program with Adafruit Blinka

The Blinka library not only offers a simple API for GPIO, PWM, I2C, and SPI, but also provides compatibility with a long list of sensor libraries built for CircuitPython. That means you can reuse CircuitPython code for peripherals that was originally used on microcontrollers or other boards such as Raspberry Pi.

To get started, install Blinka and libgpiod on your Dev Board Mini as follows:

```
sudo apt-get install python3-libgpiod

python3 -m pip install adafruit-blinka
```

Then you can turn on an LED when you push a button as follows (notice this uses pin names from the pinout above):

```
import board
import digitalio

led = digitalio.DigitalInOut(board.GPIO39)   # pin 40
led.direction = digitalio.Direction.OUTPUT

button = digitalio.DigitalInOut(board.GPIO13)   # pin 36
button.direction = digitalio.Direction.INPUT

try:
  while True:
    led.value = button.value
finally:
  led.value = False
  led.deinit()
  button.deinit()
```

For more information, including example code using I2C and SPI, see the Adafruit guide for CircuitPython libraries on Coral. But we suggest you skip their setup guide and install Blinka as shown above. And beware that their guide was written for the Coral Dev Board, so some pin names are different on the Dev Board Mini. Also check out the Blinka API reference.

Program GPIOs with libgpiod

You can also interact with the GPIO pins using the libgpiod library, which provides both C++ and Python API bindings. But libgpiod is for GPIOs only, not any digital protocols. (The Blinka library uses libgpiod as its implementation for GPIOs.)

There's currently no online API docs for libgpiod, but the source code is fully documented. If you clone the repo, you can build C++ docs with Doxygen. For Python, you can install the libgpiod package and print the API docs as follows:

```
sudo apt-get install python3-libgpiod

python3 -c 'import gpiod; help(gpiod)'
```

Then you can turn on an LED when you push a button as follows:

```
import gpiod

```

```
CONSUMER = "led-demo"
chip = gpiod.Chip("0", gpiod.Chip.OPEN_BY_NUMBER)

led = chip.get_line(39)  # pin 40
led.request(consumer=CONSUMER, type=gpiod.LINE_REQ_DIR_OUT, default_vals=[0])
button = chip.get_line(13)  # pin 36
button.request(consumer=CONSUMER, type=gpiod.LINE_REQ_DIR_IN)

try:
  while True:
    led.set_value(button.get_value())
finally:
  led.set_value(0)
  led.release()
  button.release()
```

Connect a camera to the Dev Board Mini

To perform real-time inferencing with a vision model, you can connect the Dev Board Mini to the Coral Camera.
Once you connect your camera, try the demo scripts below.
Note:

- USB cameras are currently not supported with the Dev Board Mini. Because the board's USB port supports USB 2.0 only, the limited bandwidth makes it difficult to sustain both a high frame-rate and high image quality. We're working to support USB cameras, but for optimal performance, we recommend using the CSI camera interface.
- The CSI cable connector on the Dev Board Mini is designed to be compatible with the Coral Camera only.

Connect the Coral Camera

The Coral Camera connects to the CSI connector on the top of the Dev Board Mini.

You can connect the camera to the Dev Board Mini as follows:

1. Make sure the board is powered off and unplugged.
2. On the top of the Dev Board Mini, locate the "Camera" connector and flip the small black latch so it's facing upward, as shown in figure 1.

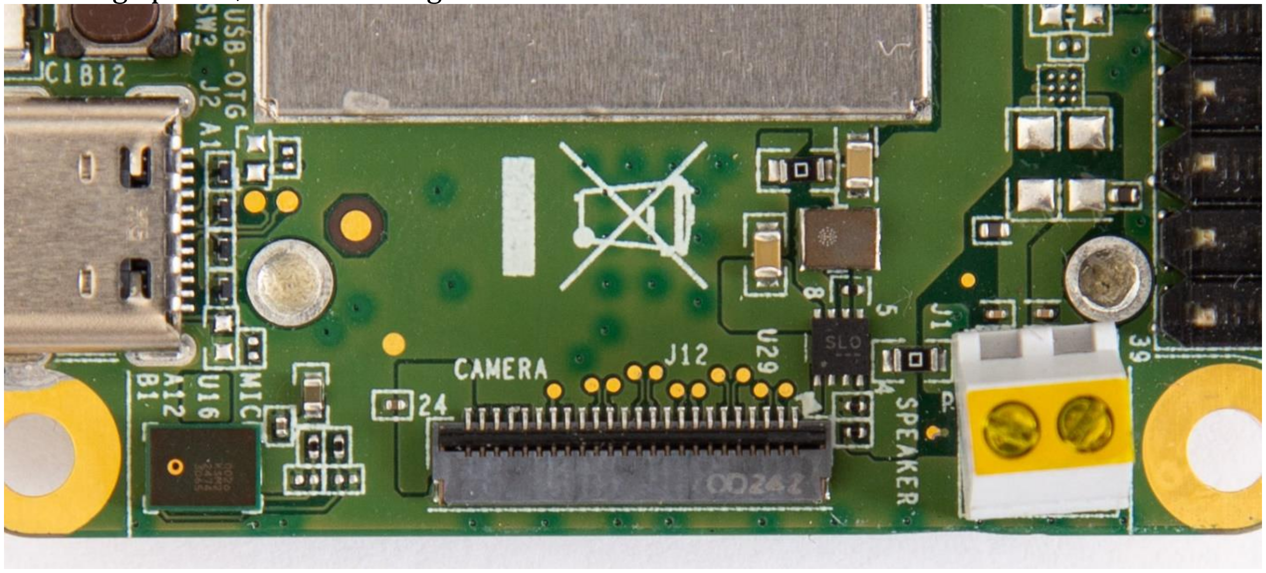


Figure 1. The board's camera connector with the latch open

3. Slide the flex cable into the connector with the contact pins facing toward the board (the blue strip is facing away from the board), as shown in figure 2.
4. Close the black latch.

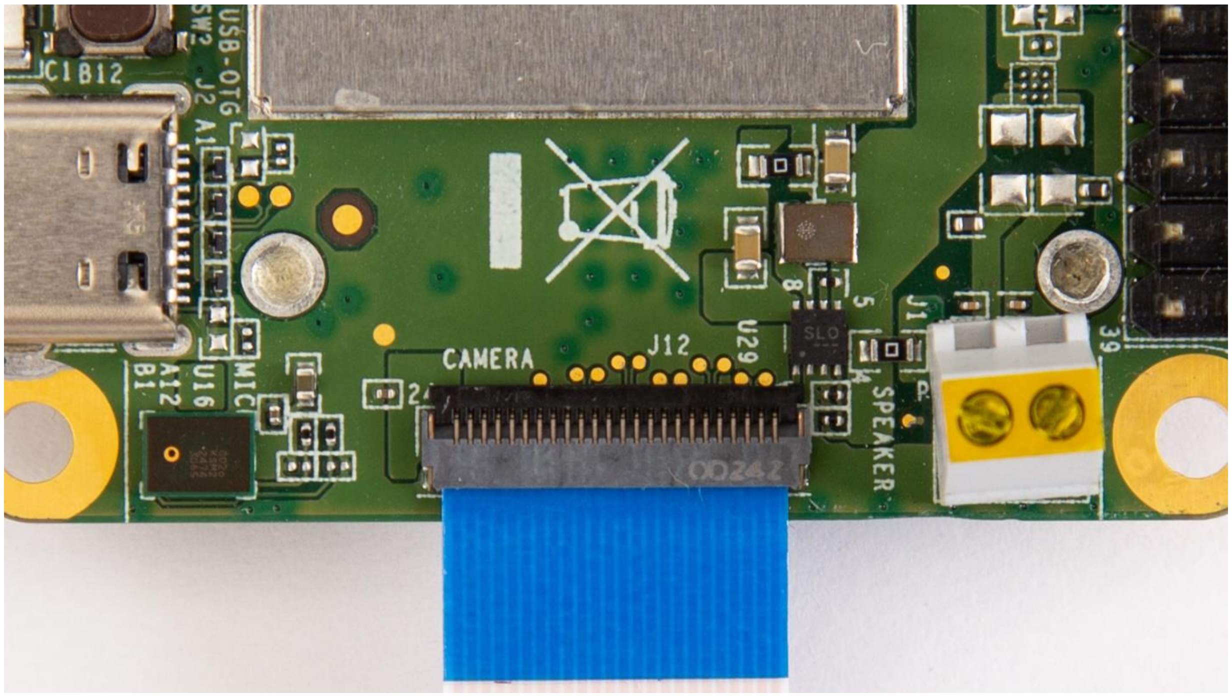


Figure 2. The cable inserted to the board and the latch closed

5. Likewise, ensure that the other end is secured on the camera module.

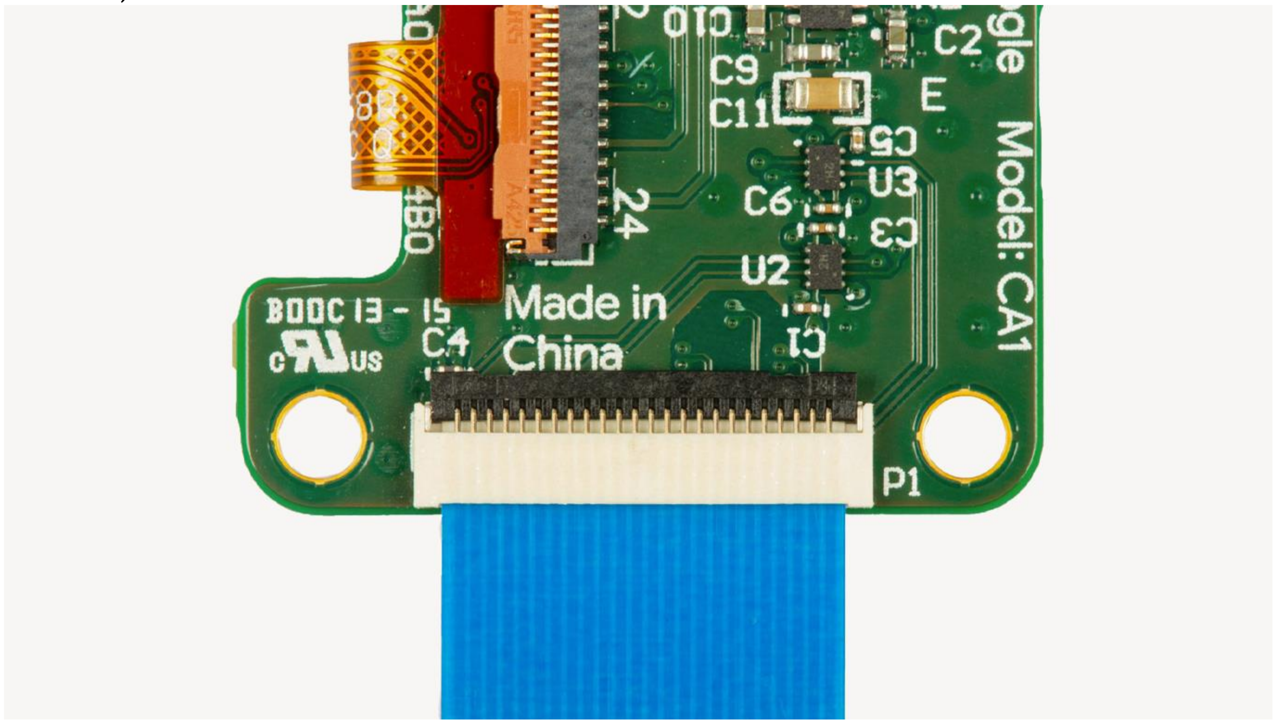


Figure 3. The camera cable connected to the camera module

6. Power on the board and verify it detects the camera by running this command:

```
v4l2-ctl --list-devices
```

You should see the camera listed as `/dev/video1`:

```
platform:mt8167 (platform:mt8167):
        /dev/video0
        /dev/video1

14001000.rdma (platform:mt8173):
        /dev/video2
```

Note: Even when the camera is not connected, you will see `video0` and `video2` listed, because they represent the MediaTek SoC's video decoder and scaler/converter, respectively.

For a quick camera test, connect to the board's shell terminal and run the snapshot tool:

```
snapshot
```

If you have a monitor attached to the board, you'll see the live camera feed.

You can press Spacebar to save an image to the home directory.

Run a demo with the camera

The Mendel system image on the Dev Board Mini includes two demos that perform real-time image classification and object detection.

First, connect a monitor to the board's micro HDMI port so you can see the video results.
Then log on to the board (using MDT or the serial console) and run these commands to be sure you have the latest software:

```
sudo apt-get update

sudo apt-get dist-upgrade
```

Note: The following demo code is optimized for performance on the Dev Board Mini, and you can get the source code from the edgetpuvision Git repo. If you'd like to see other examples using a camera, which are more broadly applicable for other devices (not just the Dev Board Mini), see the examples-camera GitHub repo.

Download the model files

Before you run either demo, you'll need to download the model files on your board.

First, set this environment variable:

```
export DEMO_FILES="$HOME/demo_files"
```

Then download the following files on your board (be sure you're connected to the internet):

```
# The image classification model and labels file
wget -P ${DEMO_FILES}/ https://github.com/google-
coral/test_data/raw/master/mobilenet_v2_1.0_224_quant_edgetpu.tflite

wget -P ${DEMO_FILES}/ https://raw.githubusercontent.com/google-coral/test_data/release-
frogfish/imagenet_labels.txt

# The face detection model (does not require a labels file)
wget -P ${DEMO_FILES}/ https://github.com/google-
coral/test_data/raw/master/ssd_mobilenet_v2_face_quant_postprocess_edgetpu.tflite
```

Run a classification model

This demo classifies 1,000 different objects shown to the camera:

```
edgetpu_classify \
--model ${DEMO_FILES}/mobilenet_v2_1.0_224_quant_edgetpu.tflite \
--labels ${DEMO_FILES}/imagenet_labels.txt
```

Run a face detection model

This demo draws a box around any detected human faces:

```
edgetpu_detect \
--model ${DEMO_FILES}/ssd_mobilenet_v2_face_quant_postprocess_edgetpu.tflite
```

Try other example code

We have several other examples that are compatible with almost any camera and any Coral device with an Edge TPU (including the Dev Board). They each show how to stream images from a camera and run classification or detection models . Each example uses a different camera library, such as GStreamer, OpenCV, and PyGame.
To explore the code and run them, see the instructions at github.com/google-coral/examples-camera.

Connect to the Dev Board Mini's serial console

Although we recommend that you access the shell terminal with MDT, some situations require direct connection with the serial console. So this page shows you how to do that with the Dev Board Mini.

You'll need the following items:

- Coral Dev Board Mini
- Linux, Windows 10, or macOS computer
- USB-to-TTL serial cable (it must support 3.3 V logic levels, such as this one by Adafruit)
- 2 A / 5 V USB-C power supply

Note: If you're on Linux or Mac and your board is fully booted, you can instead open the serial console using the USB OTG port.

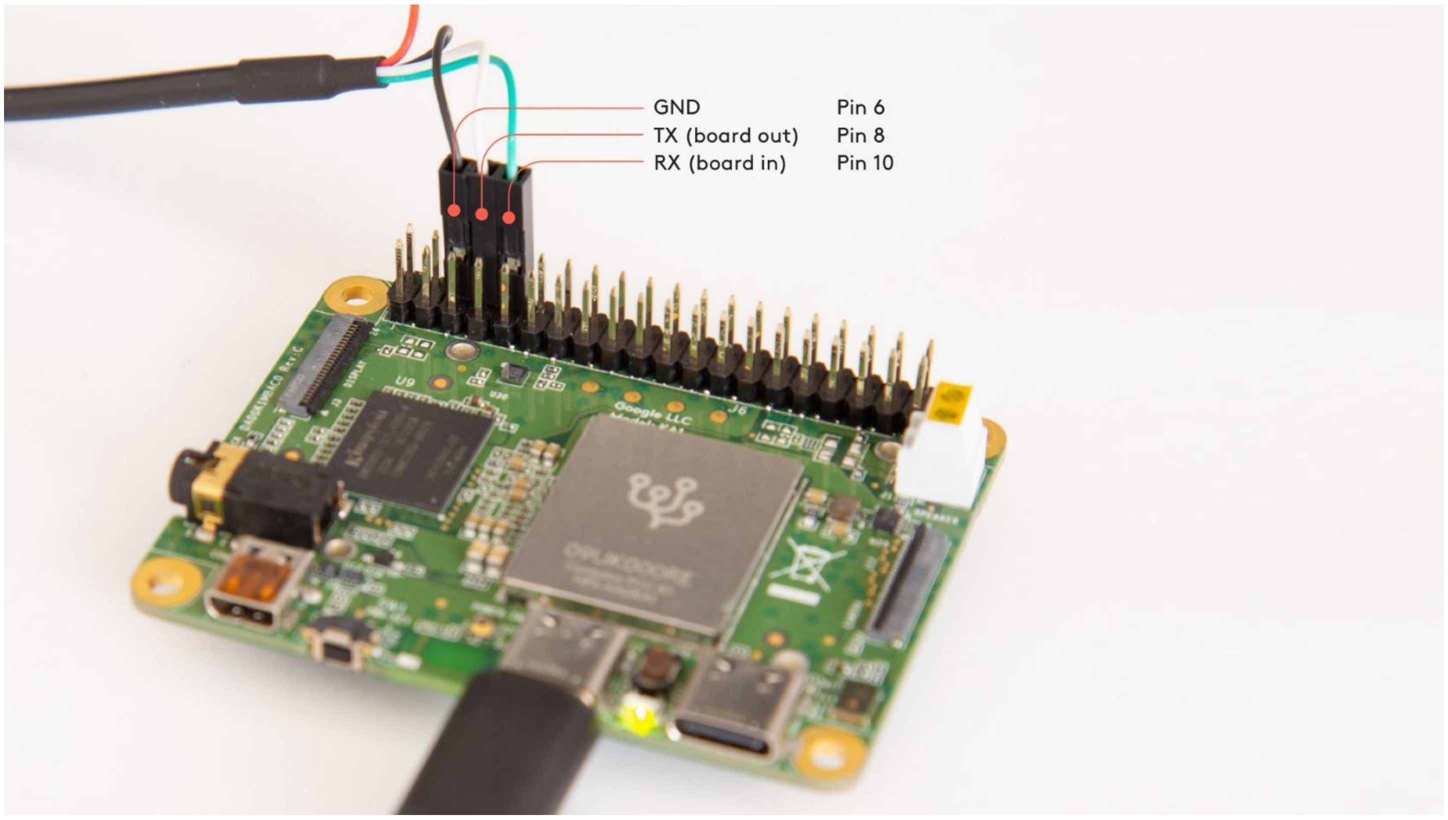


Figure 1. The ground and UART pins connected to 40-pin header

Connect with Linux

You can connect to the Dev Board Mini's serial console from Linux as follows:

1. First make sure your Linux user account is in the `plugdev` and `dialout` system groups by running this command:
   ```
   sudo usermod -aG plugdev,dialout <username>
   ```
   Then reboot your computer for the new groups to take effect.

2. Connect the USB-to-TTL serial cable to your computer and the board, as shown in figure 1:
   - Pin 6 is ground
   - Pin 8 is UART TX
   - Pin 10 is UART RX

   Warning: Do not connect the power wire (if provided). Doing so allows power to flow between the USB power and your computer, which degrades the USB power supply's ability to power the board, and can potentially damage to your hardware.

3. Connect the board to power using the USB power port, and firmly press the power button to boot the board.

4. Determine the USB-to-TTL cable's device name by running this command on your Linux computer:
   ```
   dmesg | grep ttyUSB
   ```
   You should see results such as this:
   ```
   [ 6437.706335] usb 2-13.1: cp210x converter now attached to ttyUSB0
   ```
   If you don't see anything like this, double-check your USB cable is connected.

5. Then connect with this command (using the name of the device listed in the previous step):
   ```
   screen /dev/ttyUSB0 115200
   ```
   Help! If `screen` prints `Cannot access line '/dev/ttyUSB0'`, then your Linux user account is not in the `plugdev` and/or `dialout` system group. Ask your system admin to add your account to both groups, and then restart your computer for it to take effect.

   If you see `[screen is terminating]`, it might also be due to the system groups, or there's something else wrong with `screen`—ensure all `screen` sessions are closed (type `screen -ls` to see open sessions), unplug the USB cable from the Dev Board, and then try again.

6. If the screen terminal is blank, press Enter and then you should see the login prompt appear.

   The default username and password are both "mendel".

When you're done, exit the `screen` session by pressing Control+A, D.

Connect with Windows

You can connect to the Dev Board Mini's serial console from a Windows 10 computer as follows:

1. Connect the USB-to-TTL serial cable to your computer and the board, as shown in figure 1:
   - Pin 6 is ground
   - Pin 8 is UART TX
   - Pin 10 is UART RX

   Warning: Do not connect the power wire (if provided). Doing so allows power to flow between the USB power and your computer, which degrades the USB power supply's ability to power the board, and can potentially damage to your hardware.

2. Connect the board to power using the USB power port, and firmly press the power button to boot the board.
3. On your Windows computer, open **Device Manager** and find the cable's COM port.
   Within a minute of connecting the USB cable, Windows should automatically install the necessary driver. So if you expand **Ports (COM & LPT)**, you should see a device corresponding to your serial cable (for example, "Silicon Labs CP210x USB to UART Bridge" or "Prolific USB-to-Serial Comm Port").

   Take note of the COM port for the serial cable (such as "COM3"). You'll use it in the next step.

   If Windows cannot identify the USB cable, it should instead be listed under **Other devices**. Right-click on the device and select **Update driver** to find the appropriate device driver.

4. Open PuTTY or another serial console app and start a serial console connection with the above COM port, using a baud rate of `115200`. For example, if using PuTTY:
   - Select **Session** in the left pane.
   - For the **Connection type**, select **Serial**.
   - Enter the COM port ("COM3") for **Serial line**, and enter "115200" for **Speed**.
   - Then click **Open**.

5. If the screen terminal is blank, press Enter and then you should see the login prompt appear.

   The default username and password are both "mendel".

Connect with macOS

You can connect to the Dev Board Mini's serial console from a macOS computer as follows:

1. Connect the USB-to-TTL serial cable to your computer and the board, as shown in figure 1:

   - Pin 6 is ground

   - Pin 8 is UART TX

   - Pin 10 is UART RX

   Warning: Do not connect the power wire (if provided). Doing so allows power to flow between the USB power and your computer, which degrades the USB power supply's ability to power the board, and can potentially damage to your hardware.

2. Connect the board to power using the USB power port, and firmly press the power button to boot the board.

3. Determine the USB-to-TTL cable's device name by running this command from your Mac terminal:
   ```
   ls /dev/cu*
   ```
   You should see a device with the name `usbserial` (such as `/dev/cu.usbserial-0001`). If not, you might need to install a driver provided by the cable manufacturer.

   You'll use this `usbserial` device name in the next step. However, if you've already installed a driver for the chipset in your serial cable (such as the CP210x VCP driver required for the Dev Board), then you should use a different device name that appears (such as `cu.SLAB_USBtoUART` for CP210x devices/cables).

4. Connect to the serial console using `screen` and a baud rate of 115200. For example:
   ```
   screen /dev/cu.usbserial-0001 115200
   ```

   Your device name might be different, but the baud rate is always 115200.

5. If the screen terminal is blank, press Enter and then you should see the login prompt appear.

   The default username and password are both "mendel".

When you're done, exit the `screen` session by pressing Control+A, D.

Connect over USB OTG

You can also connect to the serial console without the USB-to-TTL cable, but only if your host computer is Linux or Mac, and your board is fully booted up.

Note: The USB OTG port provides serial console access only when Mendel is booted up. As such, this approach does not provide access to the kernel boot logs or the u-boot command prompt. If you want those things or your board fails to boot Mendel, you must use the USB-to-TTL cable as described above.

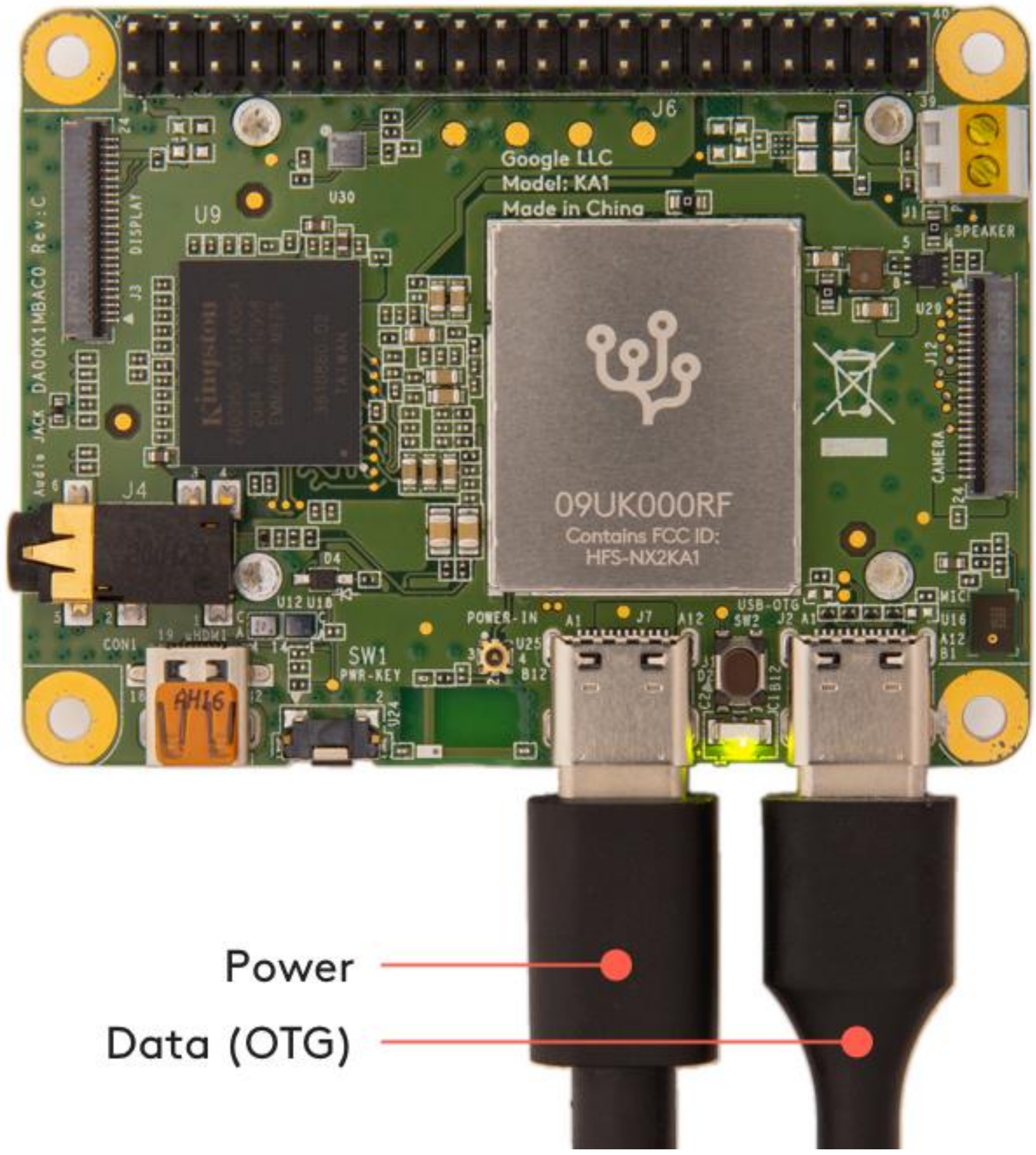


Figure 2. The USB data line can open the serial console

To connect to the serial console using the USB OTG port, follow these steps:

1. Make sure your board is fully booted up.
2. Connect a USB cable from your computer to the board's OTG port, and then run the following command in a terminal:

    - On Linux:
      ```
      ls /dev/ttyACM*
      ```
    - On Mac:
      ```
      ls /dev/cu.usbmodem*
      ```
3. Connect to the device shown using a serial console program such as `screen` as follows:
   ```
   screen /dev/ttyACM0 115200
   ```
4. If the screen terminal is blank, press Enter and then you should see the login prompt.

   The default username and password are both "mendel".

When you're done, exit the `screen` session by pressing Control+A, D.

Update or flash the Dev Board Mini

Now and then, we'll release updates for our Coral software or a new version of Mendel Linux for the Dev Board Mini. This page describes how to install these updates on your board.

Update your board with apt-get

To update the software packages in your current Mendel version—such as when we release a new Edge TPU Runtime and other API libraries—connect to the board (with MDT or the serial console) and run the following commands (first, make sure your board is online).

Note: To upgrade the Mendel version, you must flash a new system image.

```
sudo apt-get update

sudo apt-get dist-upgrade

sudo reboot now
```

Rebooting isn't always required, but recommended in case there are any kernel updates.

Flash a new system image

Flashing your board is necessary if you want to upgrade to the latest Mendel version (you can retain all user data in the Home directory) or if your board is in a failed state and you want to start clean. If you just want to update your existing system with new packages, instead update with apt-get.

You can see your Mendel version if you connect to the board and run `cat /etc/mendel_version`. Then see what Mendel versions are available on the Software page.

Note: The flashing procedure below works with Linux and Mac only.

First-time setup

If it's your first time flashing a Dev Board Mini, start with the following one-time setup:

1. Install the fastboot tool.

   On Linux, you can install as follows:
   ```
   sudo apt-get install fastboot
   ```
   For Mac, it's available from the Android SDK platform tools. This package has many tools, but you only need fastboot. So move that to a location in your `PATH` environment variable, such as `/Users/yourname/bin`.

2. If you're on Linux, reload the udev rules installed by fastboot above (actually from the (`android-sdk-platform-tools-common` (package dependency):
   ```
   sudo udevadm control --reload-rules && sudo udevadm trigger
   ```
   Also make sure your Linux user account is in the `plugdev` and `dialout` system groups by running this command:

```
sudo usermod -aG plugdev,dialout <username>
```

Then reboot your computer for the new groups to take effect.

Flash the board

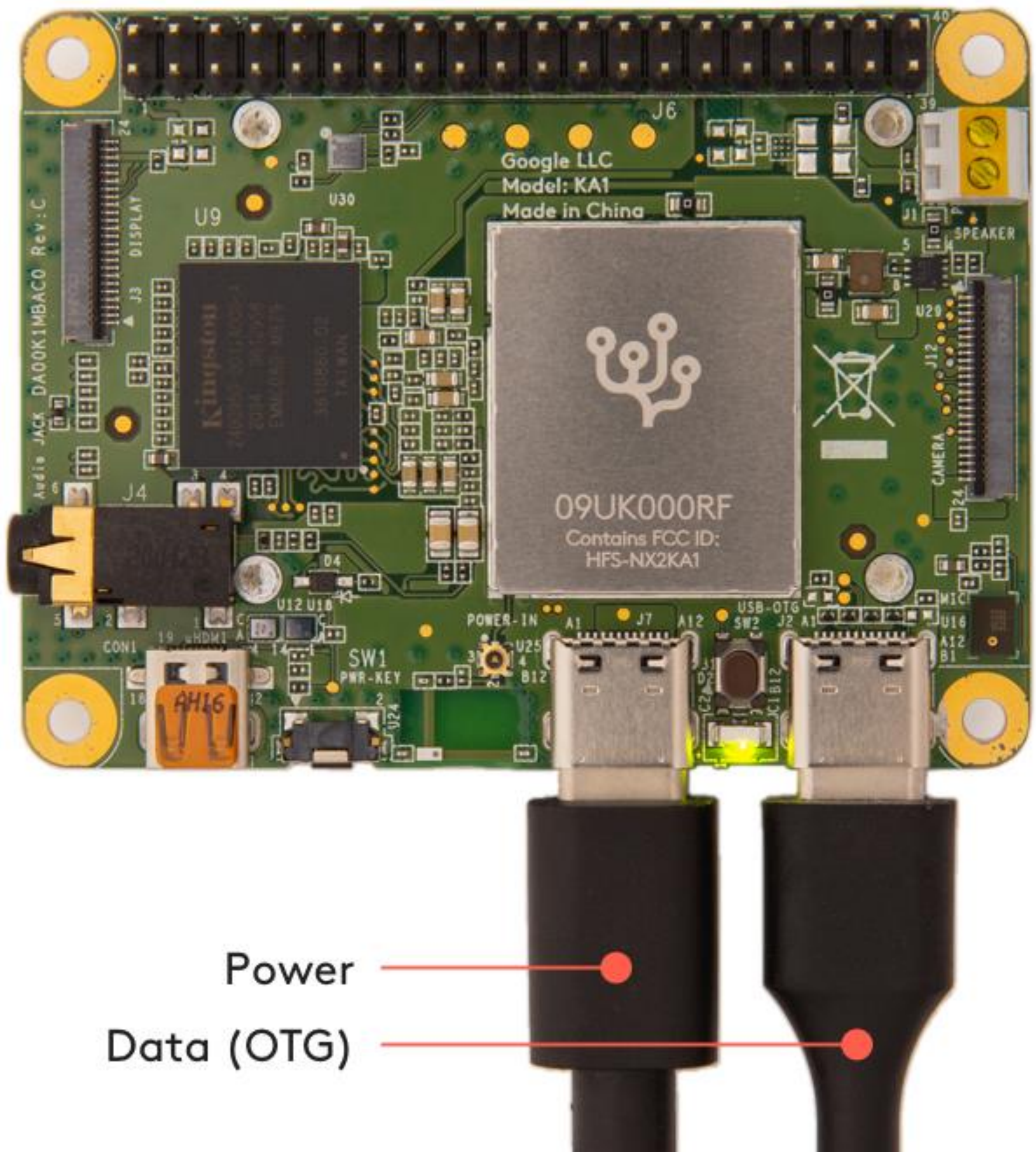
Now you can flash the Dev Board Mini as follows:

1. Download the latest system image on your host computer:

```
cd $HOME/Downloads

curl -O https://mendel-linux.org/images/excelsior/eagle/excelsior-eagle-20201210233645.zip

unzip excelsior-eagle-20201210233645.zip
```

To get a different Mendel version, see the software downloads.

6. Connect the board to power and to your computer via USB.



1. **Figure 1.** The USB power and data cables connected

2. Enable fastboot mode on the board:

    1. If you're already in the board's shell, run this command:
    ```
    sudo reboot-bootloader
    ```
    2. Otherwise, run this from your host computer:
    ```
    mdt reboot-bootloader
    ```
3. If you can't access the board with MDT, then connect to the serial console and run `sudo reboot-bootloader`.
4. If your board will not boot at all, then you can force-boot into fastboot mode.
5. Wait a moment for the board to reboot and enter fastboot mode.

    You'll know it's in fastboot mode when the board's LED turns red.

6. Run the flash script:
```
cd excelsior-eagle-20201210233645

bash flash.sh
```
   If you want to also wipe the Home directory, add the `-H` flag.

Flashing should finish within 5 minutes, and then the system reboots. It takes another few minutes for the the board to boot up for the first time (later boots are much faster).

When the board is finished rebooting, you can connect with MDT. To avoid guessing when it's done, run the following command and MDT will automatically connect when the board is ready:

```
mdt wait-for-device && mdt shell
```

Mac users: If you're on macOS Catalina (10.15) or higher, MDT won't work right now (even if you preserved the board's `/home` directory), because flashing the board resets the known networks. So if you preserved the Home directory, you just need to get the board back your Wi-Fi using the serial console. If you wiped the Home directory, then see how to connect MDT on macOS.

Note: Your board's hostname is randomly generated the first time it boots from a new flashing. We do this to ensure that each device within a local fleet is likely to have a unique name. Of course, you can change this name using standard Linux hostname tooling (such as `hostname`).

If you connect with the serial console, the login and password are reset to the defaults: both are **`mendel`**.

Force-boot into fastboot mode

If you can't even boot your board to execute `fastboot-reboot`, then you can force the system into fastboot mode by grounding a test-point pin on the top of the Dev Board Mini and then run a special boot script, as follows.

1. Complete the first-time setup above, if necessary.
2. **Do not power the board.** If connected, unplug all cables from the board now.

    However, connect your USB data cable to your computer, and have it ready to connect to the board later.

3. On your host computer, install these Python packages required by the boot script:
```
python3 -m pip install pyftdi pyserial
```
4. Download the latest system image on your host computer:
```
cd $HOME/Downloads

curl -O https://mendel-linux.org/images/excelsior/eagle/excelsior-eagle-20201210233645.zip

unzip excelsior-eagle-20201210233645.zip
```
   To get a different Mendel version, see the software downloads.

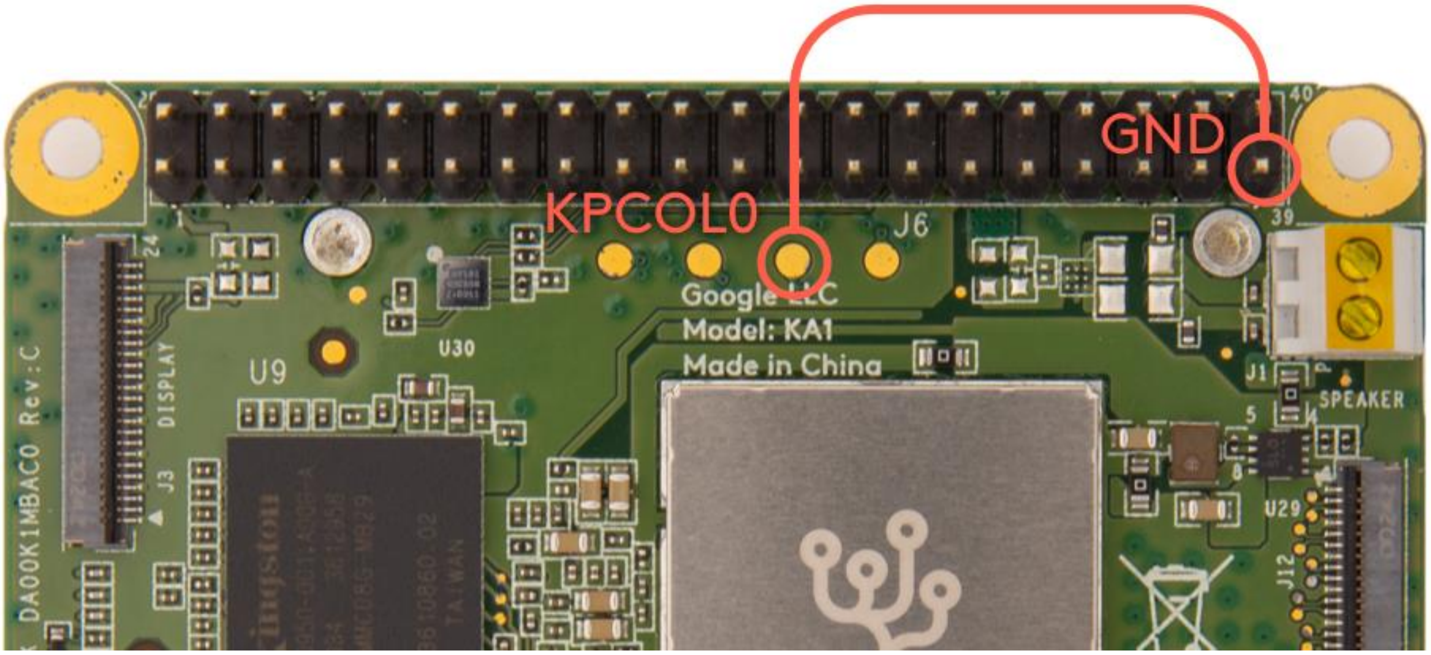
9. Run `enable_lk_fastboot.sh` to enable fastboot via the LK bootloader:
```
cd excelsior-eagle-20201210233645

```

```
bash enable_lk_fastboot.sh
```

By design, this has no immediate effect; the script waits for the device to boot. So the console should print several lines, ending with this one:

```
INFO: Waiting to connect platform...
```

12. Use a wire to connect the `KPCOL0` test point to ground, as shown in figure 2. Ideally, use a female-to-male jumper wire, with the female end attached to pin 39 on the GPIO header (ground), and then hold the male end firmly on the test point.



1. **Figure 2.** The KPCOL0 test point and an available ground pin
2. While maintaining a firm connection with the jumper wire, carefully connect the USB data cable to the board's USB OTG port.
   As the board boots, the `enable_lk_fastboot.sh` script you ran above identifies the board, enables the LK bootloader, and initiates fastboot mode. That prompt will return to you after it prints this:

```
INFO: Loading file: lk.bin

INFO: Send lk.bin
INFO: Jump da
```

   If you don't see this, then your wire connection failed—power off the board and try again from step 5.

3. When the prompt returns to you, you can release the jumper wire.
4. Unlike the `reboot-bootloader` command, the LED does not turn red to indicate it's in fastboot mode. So you can verify the board is fastboot mode with this command:

```
fastboot devices
```

   You should see your device listed, like this:

```
0123456789ABCDEF   fastboot
```

5. Now you can flash the board:

```
bash flash.sh
```

   To also wipe the Home directory, add the `-H` flag.