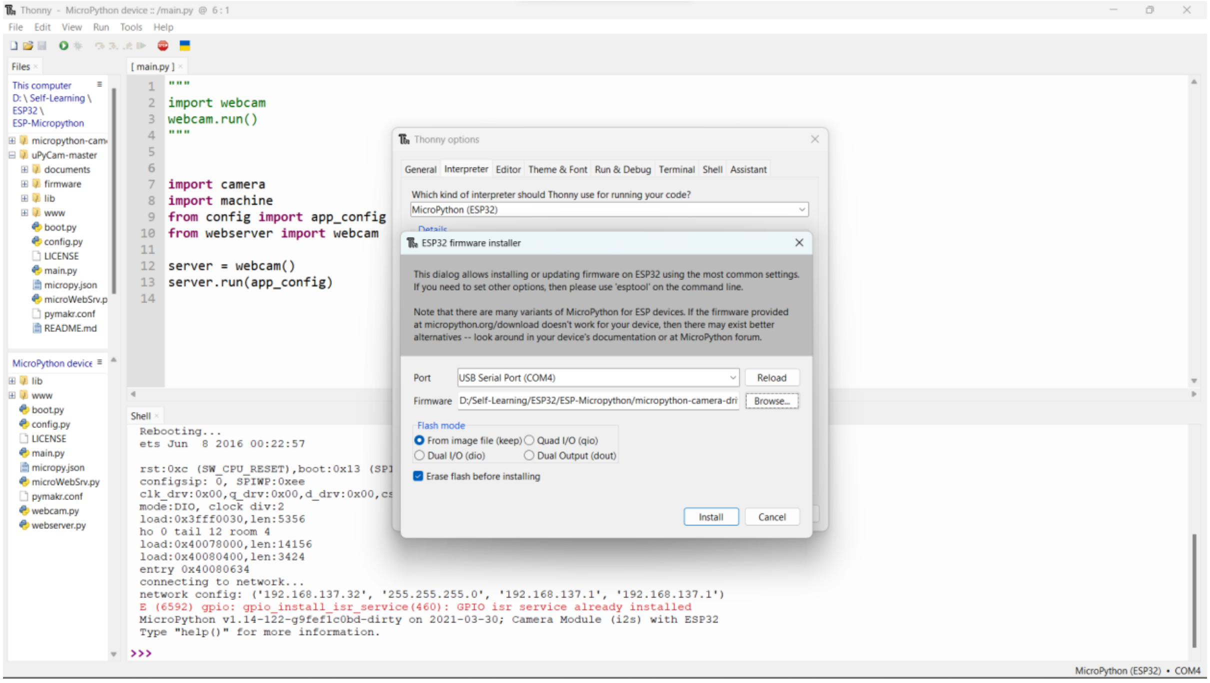
Micropython ESP32 Camera

1. **Install custom firmware** ([lemariva/micropython-camera-driver: add camera support to MicroPython](#))



```python
"""
import webcam
webcam.run()
"""


import camera
import machine
from config import app_config
from webserver import webcam

server = webcam()
server.run(app_config)

```

```
Rebooting...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs
mode:DIO, clock div:2
load:0x3fff0030,len:5356
ho 0 tail 12 room 4
load:0x40078000,len:14156
load:0x40080400,len:3424
entry 0x40080634
connecting to network...
network config: ('192.168.137.32', '255.255.255.0', '192.168.137.1', '192.168.137.1')
E (6592) gpio: gpio_install_isr_service(460): GPIO isr service already installed
MicroPython v1.14-122-g9fef1c0bd-dirty on 2021-03-30; Camera Module (i2s) with ESP32
Type "help()" for more information.
>>>
```

Which kind of interpreter should Thonny use for running your code?
MicroPython (ESP32)

This dialog allows installing or updating firmware on ESP32 using the most common settings. If you need to set other options, then please use 'esptool' on the command line.

Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download doesn't work for your device, then there may exist better alternatives -- look around in your device's documentation or at MicroPython forum.

Port: USB Serial Port (COM4)

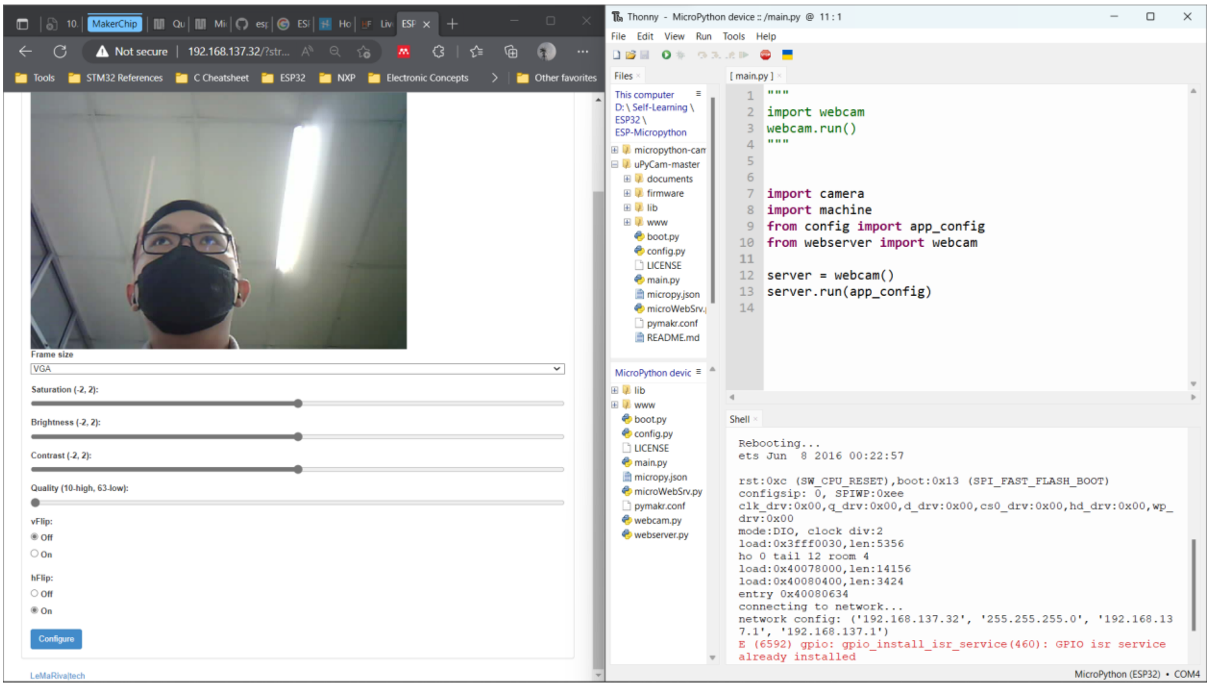
Firmware: D:/Self-Learning/ESP32/ESP-Micropython/micropython-camera-dri

Flash mode
● From image file (keep) ○ Quad I/O (qio)
○ Dual I/O (dio) ○ Dual Output (dout)

☑ Erase flash before installing

While installing firmware, connect GPIO0 to GND and hold reset button at the same time for several seconds then release.

2. **Upload the necessary program files to the device**
3. **Press Run & Copy the Network ip to your browser**



Frame size

VGA

Saturation (-2, 2):

Brightness (-2, 2):

Contrast (-2, 2):

Quality (10-high, 63-low):

vFlip:

● Off

○ On

hFlip:

○ Off

● On

```
"""
import webcam
webcam.run()
"""


import camera
import machine
from config import app_config
from webserver import webcam

server = webcam()
server.run(app_config)
```

```
Rebooting...
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_
drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:5356
ho 0 tail 12 room 4
load:0x40078000,len:14156
load:0x40080400,len:3424
entry 0x40080634
connecting to network...
network config: ('192.168.137.32', '255.255.255.0', '192.168.13
7.1', '192.168.137.1')
E (6592) gpio: gpio_install_isr_service(460): GPIO isr service
already installed
```

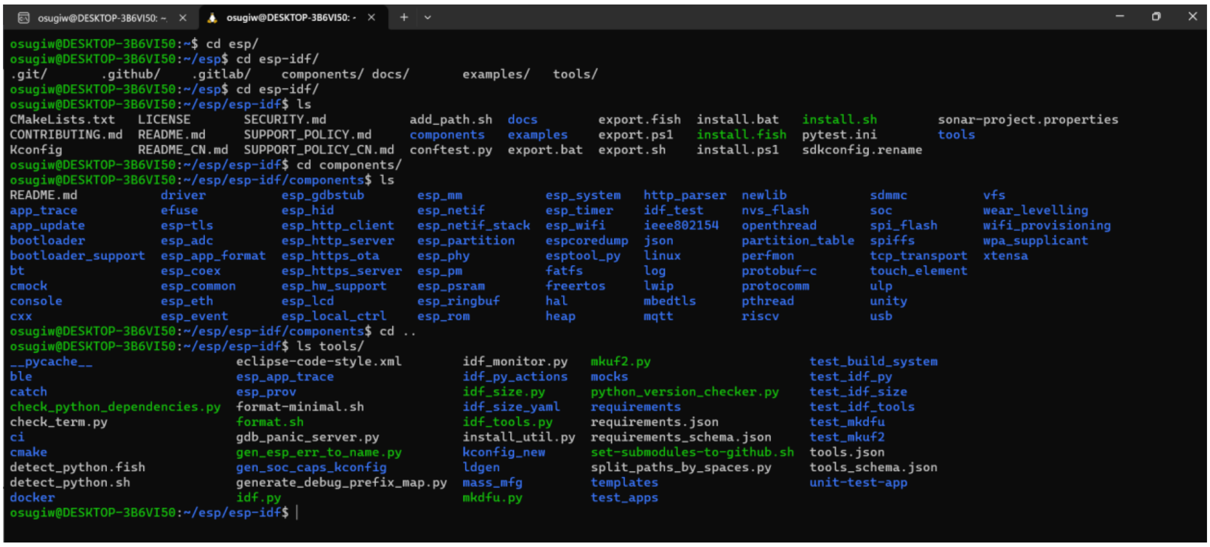
Experiment Micropython & ESP-IDF Custom Firmware

(Only works with ESP-IDF v.4.4.4)

ESP Tools for Build Firmware

Follow these steps:

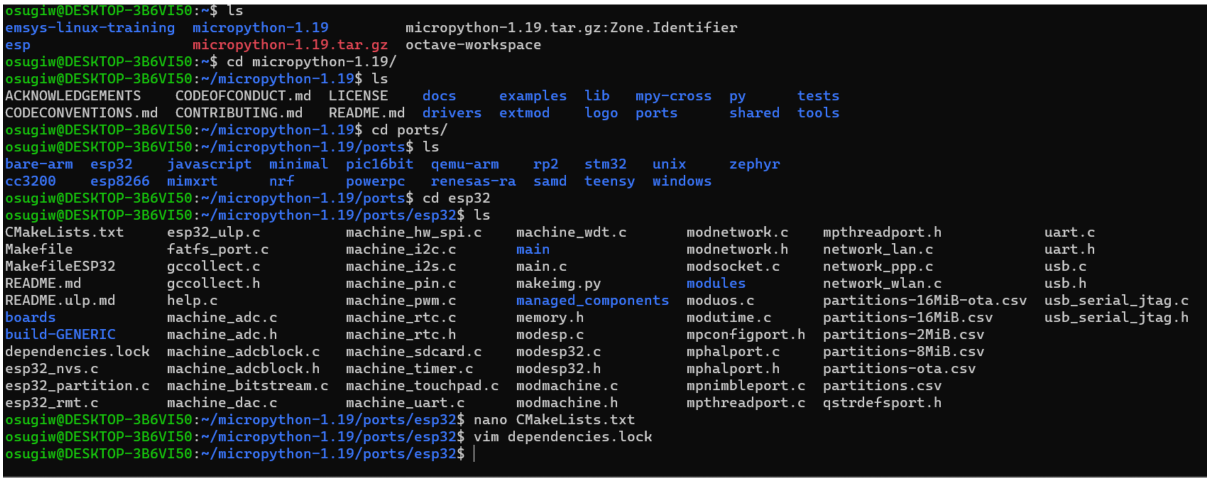
- Standard Toolchain Setup for Linux and macOS - ESP32 - — ESP-IDF Programming Guide latest documentation (espressif.com) or;
- micropython/ports/esp32 at master · micropython/micropython (github.com)



```
osugiw@DESKTOP-3B6VI50:~$ cd esp/
osugiw@DESKTOP-3B6VI50:~/esp$ cd esp-idf/
.git/       .github/    .gitlab/    components/ docs/       examples/   tools/
osugiw@DESKTOP-3B6VI50:~/esp$ cd esp-idf/
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf$ ls
CMakeLists.txt   LICENSE       SECURITY.md           add_path.sh  docs         export.fish  install.bat   install.sh        sonar-project.properties
CONTRIBUTING.md  README.md     SUPPORT_POLICY.md     components   examples     export.ps1   install.fish  pytest.ini        tools
Kconfig          README_CN.md  SUPPORT_POLICY_CN.md  conftest.py  export.bat   export.sh    install.ps1   sdkconfig.rename
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf$ cd components/
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf/components$ ls
README.md           driver          esp_gdbstub       esp_mm           esp_system   http_parser  newlib           sdmmc          vfs
app_trace           efuse           esp_hid           esp_netif        esp_timer    idf_test     nvs_flash        soc            wear_levelling
app_update          esp-tls         esp_http_client   esp_netif_stack  esp_wifi     ieee802154   openthread       spi_flash      wifi_provisioning
bootloader          esp_adc         esp_http_server   esp_partition    espcoredump  json         partition_table  spiffs         wpa_supplicant
bootloader_support  esp_app_format  esp_https_ota     esp_phy          esptool_py   linux        perfmon          tcp_transport  xtensa
bt                  esp_coex        esp_https_server  esp_pm           fatfs        log          protobuf-c       touch_element
cmock               esp_common      esp_hw_support    esp_psram        freertos     lwip         protocomm        ulp
console             esp_eth         esp_lcd           esp_ringbuf      hal          mbedtls      pthread          unity
cxx                 esp_event       esp_local_ctrl    esp_rom          heap         mqtt         riscv            usb
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf/components$ cd ..
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf$ ls tools/
__pycache__                   eclipse-code-style.xml        idf_monitor.py   mkuf2.py                     test_build_system
ble                           esp_app_trace                 idf_py_actions   mocks                        test_idf_py
catch                         esp_prov                      idf_size.py      python_version_checker.py    test_idf_size
check_python_dependencies.py  format-minimal.sh             idf_size_yaml    requirements                 test_idf_tools
check_term.py                 format.sh                     idf_tools.py     requirements.json            test_mkdfu
ci                            gdb_panic_server.py           install_util.py  requirements_schema.json     test_mkuf2
cmake                         gen_esp_err_to_name.py        kconfig_new      set-submodules-to-github.sh  tools.json
detect_python.fish            gen_soc_caps_kconfig          ldgen            split_paths_by_spaces.py     tools_schema.json
detect_python.sh              generate_debug_prefix_map.py  mass_mfg         templates                    unit-test-app
docker                        idf.py                        mkdfu.py         test_apps
osugiw@DESKTOP-3B6VI50:~/esp/esp-idf$ |
```

Micropython File Structures

Download Micropython: https://github.com/micropython/micropython/releases



```
osugiw@DESKTOP-3B6VI50:~$ ls
emsys-linux-training  micropython-1.19         micropython-1.19.tar.gz:Zone.Identifier
esp                   micropython-1.19.tar.gz  octave-workspace
osugiw@DESKTOP-3B6VI50:~$ cd micropython-1.19/
osugiw@DESKTOP-3B6VI50:~/micropython-1.19$ ls
ACKNOWLEDGEMENTS    CODEOFCONDUCT.md  LICENSE    docs     examples  lib   mpy-cross  py      tests
CODECONVENTIONS.md  CONTRIBUTING.md   README.md  drivers  extmod    logo  ports      shared  tools
osugiw@DESKTOP-3B6VI50:~/micropython-1.19$ cd ports/
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports$ ls
bare-arm  esp32    javascript  minimal  pic16bit  qemu-arm    rp2   stm32   unix     zephyr
cc3200    esp8266  mimxrt      nrf      powerpc   renesas-ra  samd  teensy  windows
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports$ cd esp32
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports/esp32$ ls
CMakeLists.txt     esp32_ulp.c          machine_hw_spi.c    machine_wdt.c       modnetwork.c    mpthreadport.h             uart.c
Makefile           fatfs_port.c         machine_i2c.c       main                modnetwork.h    network_lan.c              uart.h
MakefileESP32      gccollect.c          machine_i2s.c       main.c              modsocket.c     network_ppp.c              usb.c
README.md          gccollect.h          machine_pin.c       makeimg.py          modules         network_wlan.c             usb.h
README.ulp.md      help.c               machine_pwm.c       managed_components  moduos.c        partitions-16MiB-ota.csv   usb_serial_jtag.c
boards             machine_adc.c        machine_rtc.c       memory.h            modutime.c      partitions-16MiB.csv       usb_serial_jtag.h
build-GENERIC      machine_adc.h        machine_rtc.h       modesp.c            mpconfigport.h  partitions-2MiB.csv
dependencies.lock  machine_adcblock.c   machine_sdcard.c    modesp32.c          mphalport.c     partitions-8MiB.csv
esp32_nvs.c        machine_adcblock.h   machine_timer.c     modesp32.h          mphalport.h     partitions-ota.csv
esp32_partition.c  machine_bitstream.c  machine_touchpad.c  modmachine.c        mpnimbleport.c  partitions.csv
esp32_rmt.c        machine_dac.c        machine_uart.c      modmachine.h        mpthreadport.c  qstrdefsport.h
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports/esp32$ nano CMakeLists.txt
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports/esp32$ vim dependencies.lock
osugiw@DESKTOP-3B6VI50:~/micropython-1.19/ports/esp32$ |
```

Steps:

- Unzip micropython file:

tar -xvf tar -xvf micropython-1.11.tar.gz

- Change directory to your micropython folder “micropython/”
- Compile the micropython cross-compiler folder

make -C mpy-cross

- Edit makefile for our board and port

cd micropython/ports/esp32/

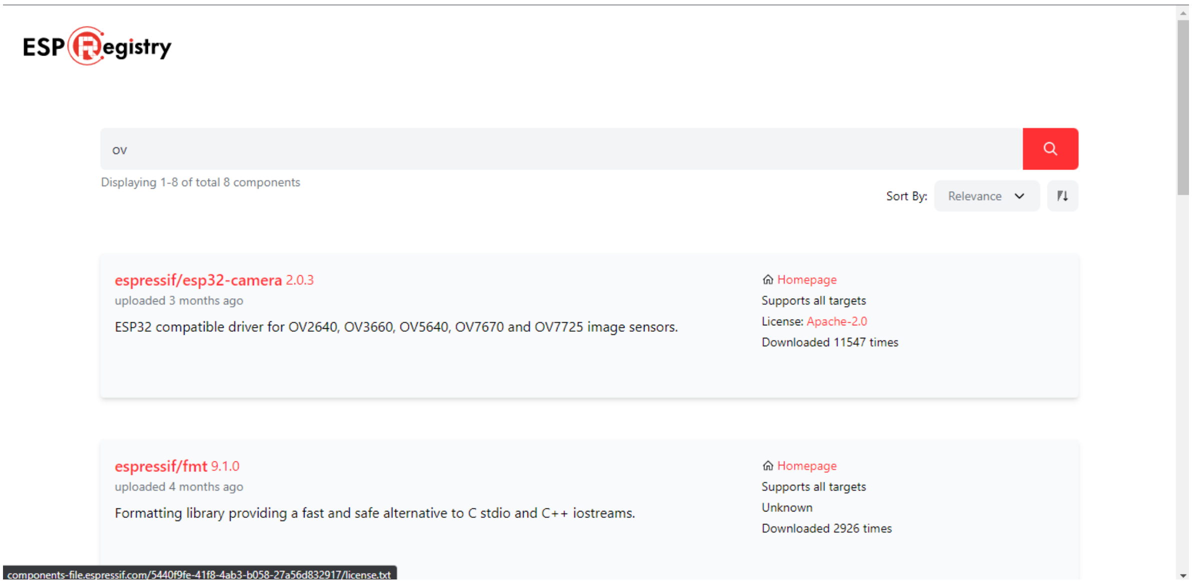
nano Makefile

ESPIDF = /home/(USER)/esp/esp-idf

PORT = /dev/ttyUSB0

- From console add desired dependency (i.e. Camera, ADC) from ESP Registry

idf.py add-dependency "espressif/esp32-camera^2.0.3"

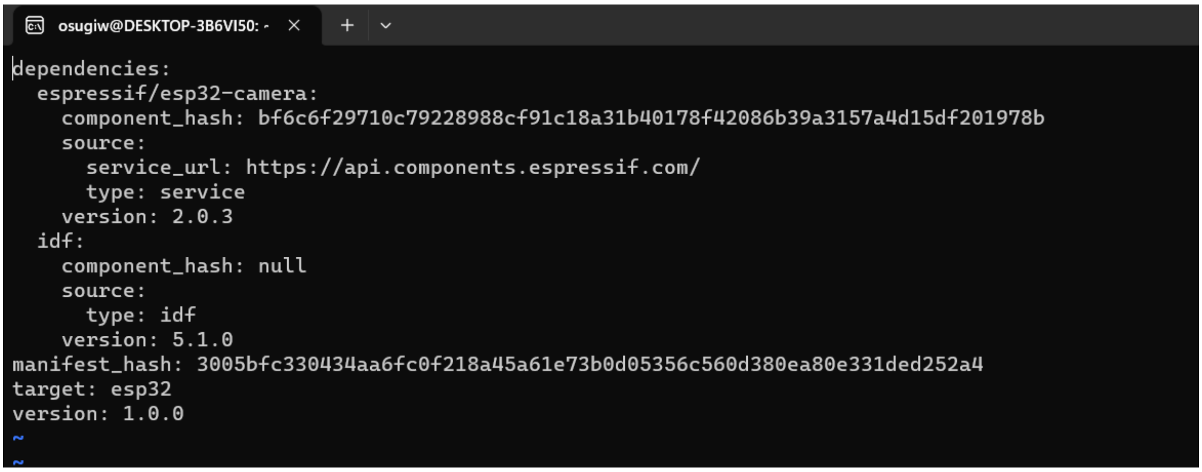


Displaying 1-8 of total 8 components

espressif/esp32-camera 2.0.3
uploaded 3 months ago
ESP32 compatible driver for OV2640, OV3660, OV5640, OV7670 and OV7725 image sensors.
Homepage
Supports all targets
License: Apache-2.0
Downloaded 11547 times

espressif/fmt 9.1.0
uploaded 4 months ago
Formatting library providing a fast and safe alternative to C stdio and C++ iostreams.
Homepage
Supports all targets
Unknown
Downloaded 2926 times

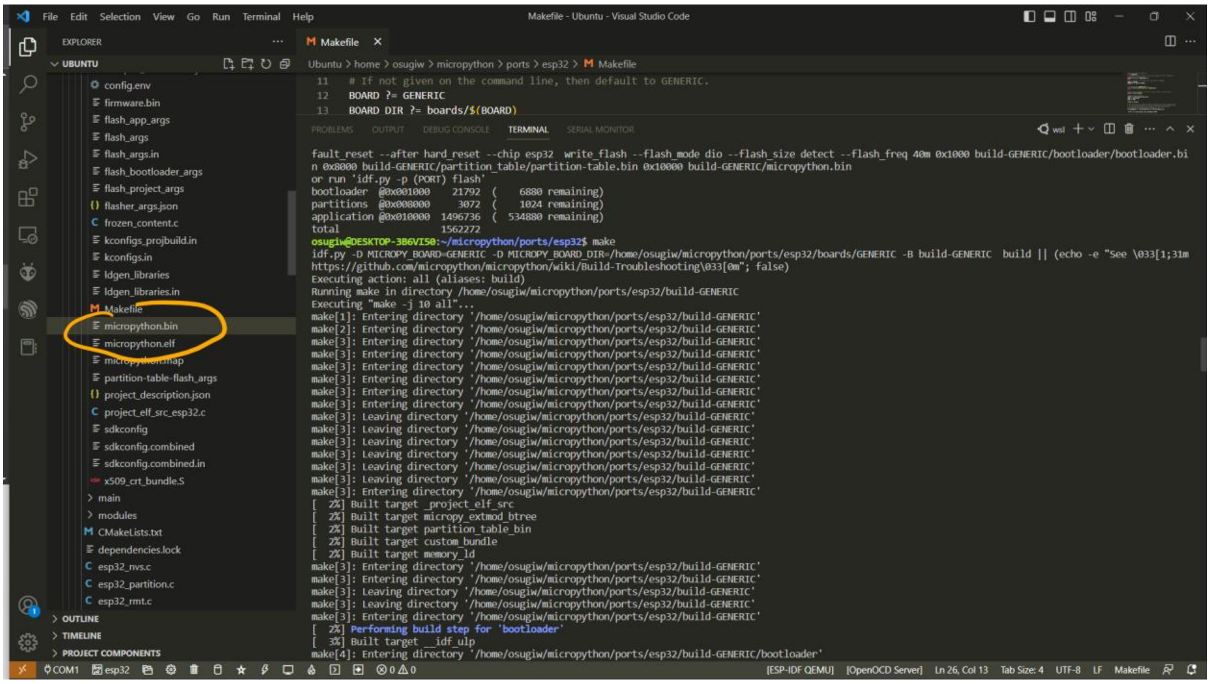
components-file.espressif.com/5440f9fe-41f8-4ab3-b058-27a56d832917/license.txt



```
dependencies:
  espressif/esp32-camera:
    component_hash: bf6c6f29710c79228988cf91c18a31b40178f42086b39a3157a4d15df201978b
    source:
      service_url: https://api.components.espressif.com/
      type: service
    version: 2.0.3
  idf:
    component_hash: null
    source:
      type: idf
    version: 5.1.0
manifest_hash: 3005bfc330434aa6fc0f218a45a61e73b0d05356c560d380ea80e331ded252a4
target: esp32
version: 1.0.0
~
~
```

- Compile Makefile

```
make
```



```
# If not given on the command line, then default to GENERIC.
BOARD ?= GENERIC
BOARD_DIR ?= boards/$(BOARD)
```

```
fault_reset --after hard_reset --chip esp32  write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x1000 build-GENERIC/bootloader/bootloader.bi
n 0x8000 build-GENERIC/partition_table/partition-table.bin 0x10000 build-GENERIC/micropython.bin
or run 'idf.py -p (PORT) flash'
bootloader  @0x001000    21792  (    6880 remaining)
partitions  @0x008000     3072  (    1024 remaining)
application @0x010000  1496736  (  534880 remaining)
total                  1562272
osugiw@DESKTOP-3B6VI50:~/micropython/ports/esp32$ make
idf.py -D MICROPY_BOARD=GENERIC -D MICROPY_BOARD_DIR=/home/osugiw/micropython/ports/esp32/boards/GENERIC -B build-GENERIC  build || (echo -e "See \033[1;31m
https://github.com/micropython/micropython/wiki/Build-Troubleshooting\033[0m"; false)
Executing action: all (aliases: build)
Running make in directory /home/osugiw/micropython/ports/esp32/build-GENERIC
Executing "make -j 10 all"...
make[1]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[2]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
[  2%] Built target _project_elf_src
[  2%] Built target micropy_extmod_btree
[  2%] Built target partition_table_bin
[  2%] Built target custom_bundle
[  2%] Built target memory_ld
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Leaving directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
make[3]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC'
[  2%] Performing build step for 'bootloader'
[  3%] Built target __idf_ulp
make[4]: Entering directory '/home/osugiw/micropython/ports/esp32/build-GENERIC/bootloader'
```

Connecting USB through WSL

Reference: [You can now connect USB devices in Windows Subsystem for Linux under Windows 11 (xda-developers.com)]

- Install the latest usbipd-win (.msi) from *https://github.com/dorssel/usbipd-win/releases/latest*
- From wsl console install the user space tools for USB/IP and a database of USB hardware identifiers:

```
sudo apt install linux-tools-5.4.0-77-generic hwdata
```

- Open windows command prompt and list all attached devices

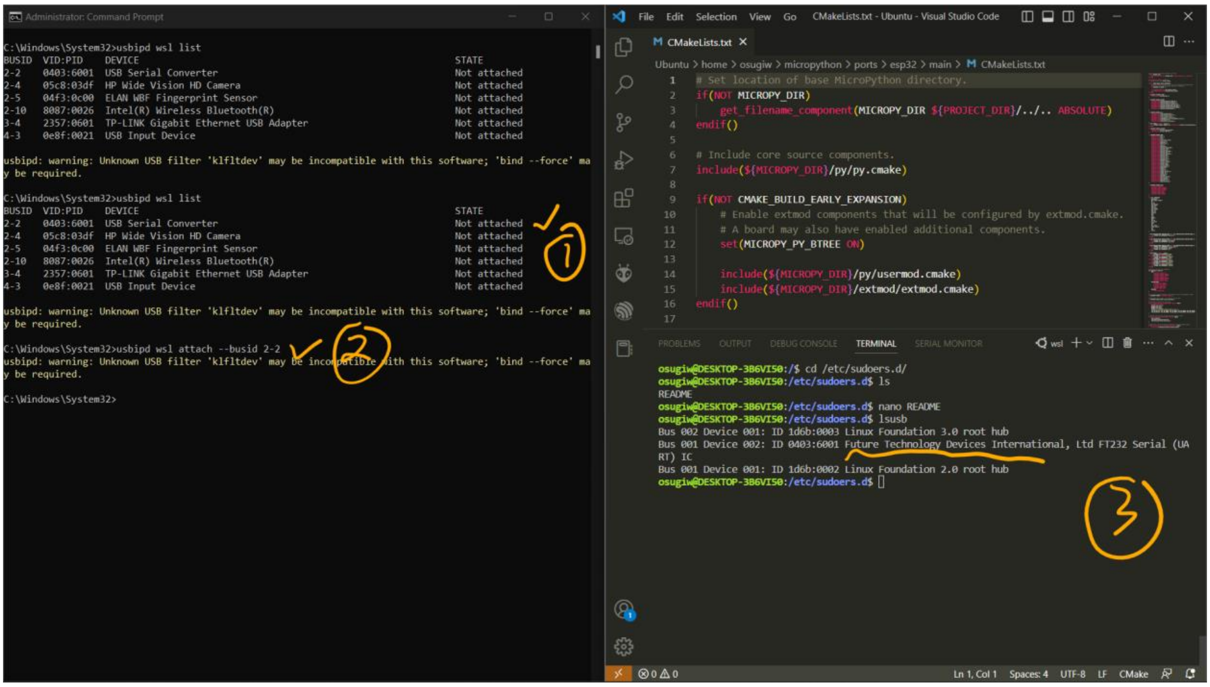
```
usbipd wsl list
```

- Attach ports that we need to WSL

```
usbipd wsl attach --busid <busid>
```

- Open WSL console again and list the attached USB in WSL

```
lsusb
```



Administrator: Command Prompt
File Edit Selection View Go CMakeLists.txt - Ubuntu - Visual Studio Code
CMakeLists.txt
Ubuntu > home > osugiw > micropython > ports > esp32 > main > CMakeLists.txt
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SERIAL MONITOR
wsl
Ln 1, Col 1 Spaces: 4 UTF-8 LF CMake
1
2
3

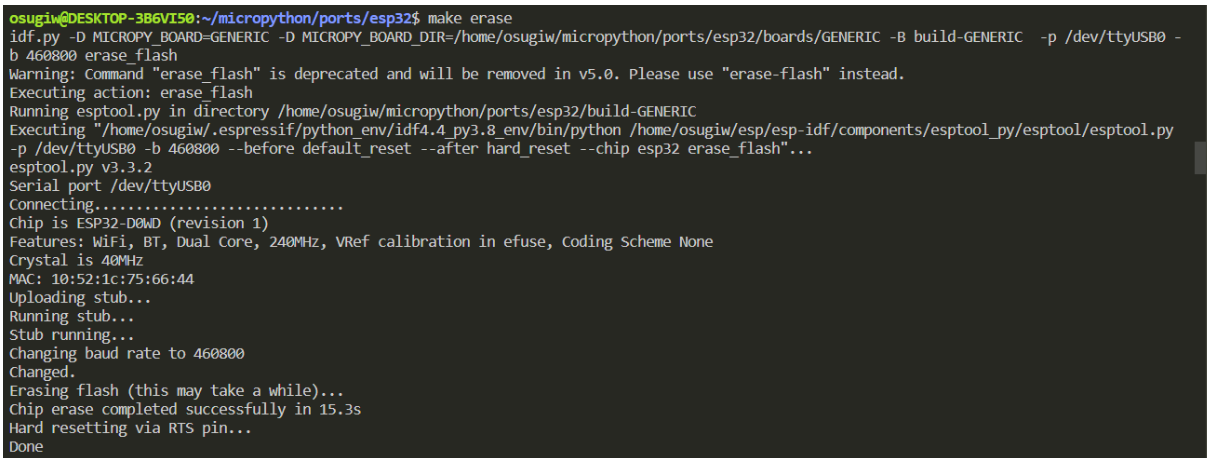
```
C:\Windows\System32>usbipd wsl list
BUSID  VID:PID    DEVICE                                                        STATE
2-2    0403:6001  USB Serial Converter                                          Not attached
2-4    05c8:03df  HP Wide Vision HD Camera                                      Not attached
2-5    04f3:0c00  ELAN WBF Fingerprint Sensor                                   Not attached
2-10   8087:0026  Intel(R) Wireless Bluetooth(R)                                Not attached
3-4    2357:0601  TP-LINK Gigabit Ethernet USB Adapter                          Not attached
4-3    0e8f:0021  USB Input Device                                              Not attached

usbipd: warning: Unknown USB filter 'klfltdev' may be incompatible with this software; 'bind --force' ma
y be required.

C:\Windows\System32>usbipd wsl list
BUSID  VID:PID    DEVICE                                                        STATE
2-2    0403:6001  USB Serial Converter                                          Not attached
2-4    05c8:03df  HP Wide Vision HD Camera                                      Not attached
2-5    04f3:0c00  ELAN WBF Fingerprint Sensor                                   Not attached
2-10   8087:0026  Intel(R) Wireless Bluetooth(R)                                Not attached
3-4    2357:0601  TP-LINK Gigabit Ethernet USB Adapter                          Not attached
4-3    0e8f:0021  USB Input Device                                              Not attached

usbipd: warning: Unknown USB filter 'klfltdev' may be incompatible with this software; 'bind --force' ma
y be required.

C:\Windows\System32>usbipd wsl attach --busid 2-2
usbipd: warning: Unknown USB filter 'klfltdev' may be incompatible with this software; 'bind --force' ma
y be required.

C:\Windows\System32>
```

```
# Set location of base MicroPython directory.
if(NOT MICROPY_DIR)
    get_filename_component(MICROPY_DIR ${PROJECT_DIR}/../.. ABSOLUTE)
endif()

# Include core source components.
include(${MICROPY_DIR}/py/py.cmake)

if(NOT CMAKE_BUILD_EARLY_EXPANSION)
    # Enable extmod components that will be configured by extmod.cmake.
    # A board may also have enabled additional components.
    set(MICROPY_PY_BTREE ON)

    include(${MICROPY_DIR}/py/usermod.cmake)
    include(${MICROPY_DIR}/extmod/extmod.cmake)
endif()
```

```
osugiw@DESKTOP-3B6VI50:/$ cd /etc/sudoers.d/
osugiw@DESKTOP-3B6VI50:/etc/sudoers.d$ ls
README
osugiw@DESKTOP-3B6VI50:/etc/sudoers.d$ nano README
osugiw@DESKTOP-3B6VI50:/etc/sudoers.d$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 0403:6001 Future Technology Devices International, Ltd FT232 Serial (UA
RT) IC
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
osugiw@DESKTOP-3B6VI50:/etc/sudoers.d$ 
```

- Erase the existing firmware on ESP32

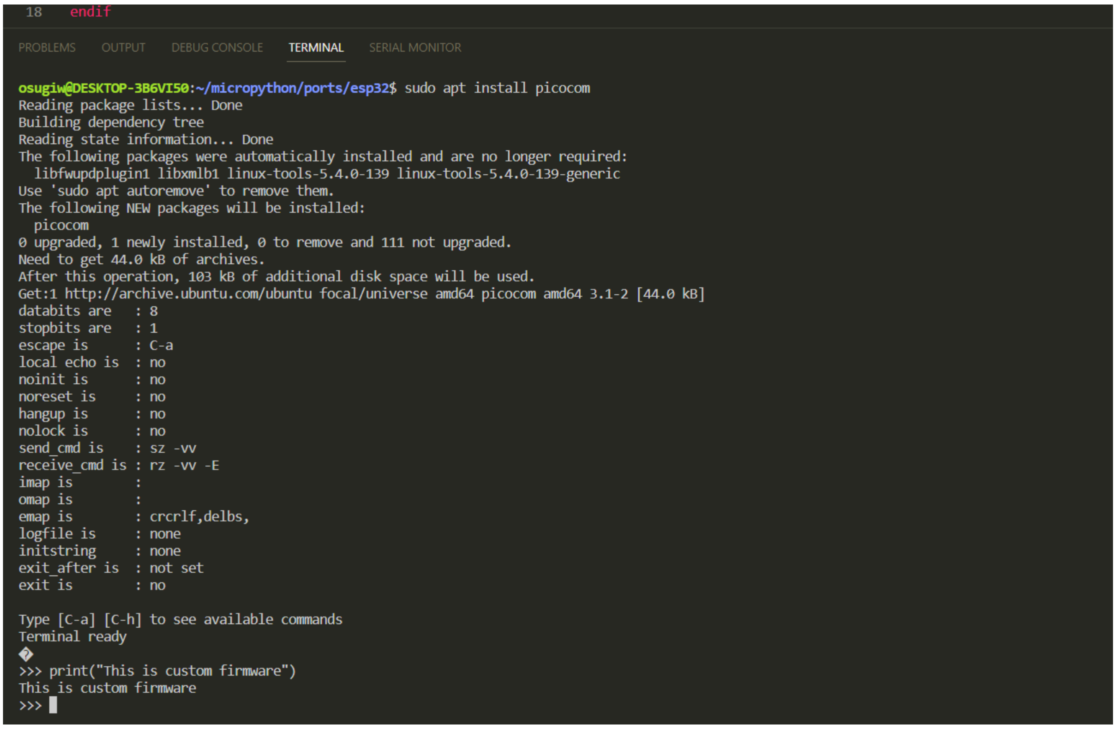
Make erase



```
osugiw@DESKTOP-3B6VI50:~/micropython/ports/esp32$ make erase
idf.py -D MICROPY_BOARD=GENERIC -D MICROPY_BOARD_DIR=/home/osugiw/micropython/ports/esp32/boards/GENERIC -B build-GENERIC  -p /dev/ttyUSB0 -
b 460800 erase_flash
Warning: Command "erase_flash" is deprecated and will be removed in v5.0. Please use "erase-flash" instead.
Executing action: erase_flash
Running esptool.py in directory /home/osugiw/micropython/ports/esp32/build-GENERIC
Executing "/home/osugiw/.espressif/python_env/idf4.4_py3.8_env/bin/python /home/osugiw/esp/esp-idf/components/esptool_py/esptool/esptool.py
-p /dev/ttyUSB0 -b 460800 --before default_reset --after hard_reset --chip esp32 erase_flash"...
esptool.py v3.3.2
Serial port /dev/ttyUSB0
Connecting.............................
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 10:52:1c:75:66:44
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Erasing flash (this may take a while)...
Chip erase completed successfully in 15.3s
Hard resetting via RTS pin...
Done
```

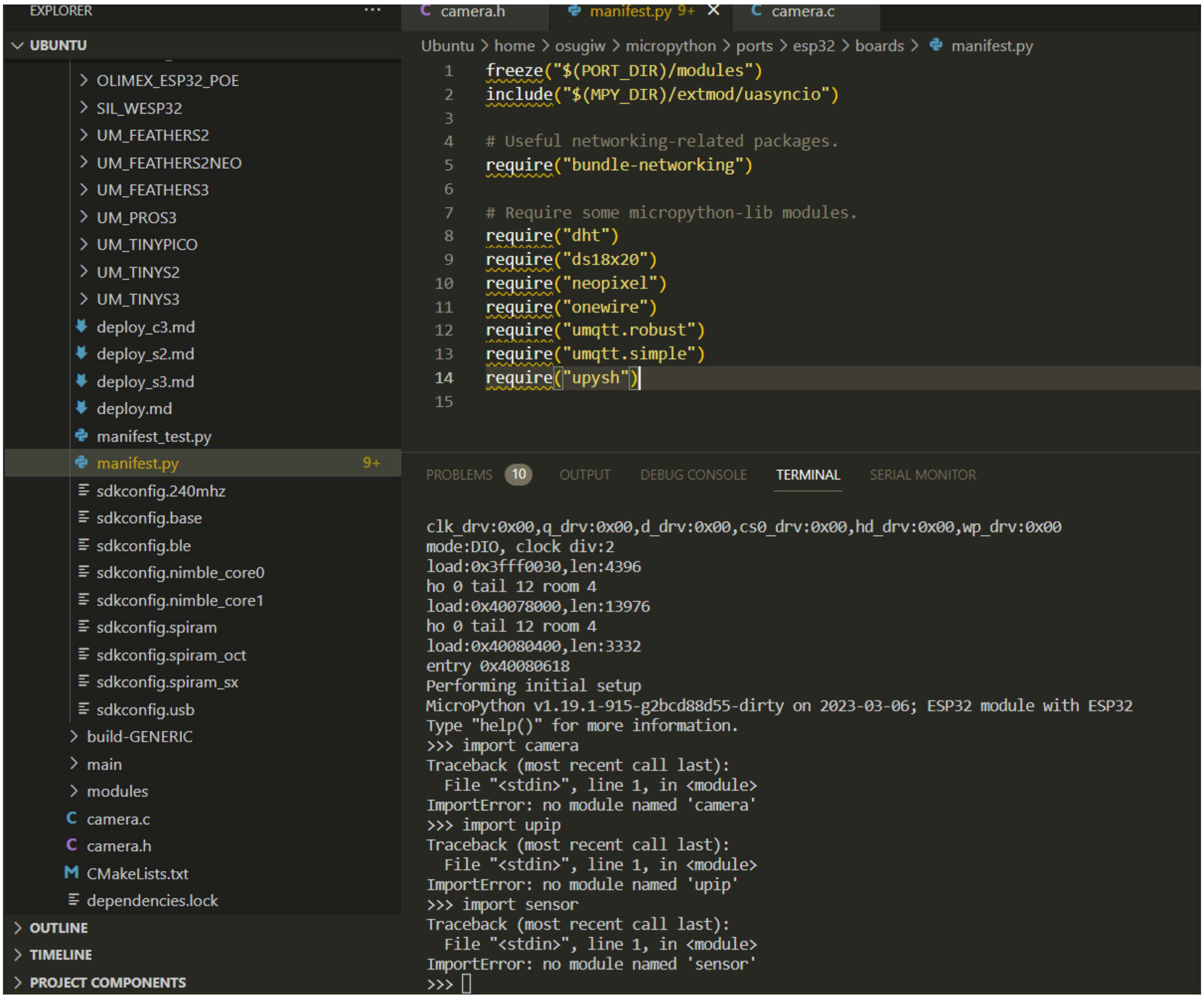
- Deploy to the ESP32 Board

Make deploy



```
osugiw@DESKTOP-3B6VI50:~/micropython/ports/esp32$ sudo apt install picocom
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfwupdplugin1 libxmlb1 linux-tools-5.4.0-139 linux-tools-5.4.0-139-generic
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  picocom
0 upgraded, 1 newly installed, 0 to remove and 111 not upgraded.
Need to get 44.0 kB of archives.
After this operation, 103 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 picocom amd64 3.1-2 [44.0 kB]
databits are   : 8
stopbits are   : 1
escape is      : C-a
local echo is  : no
noinit is      : no
noreset is     : no
hangup is      : no
nolock is      : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crcrlf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready
�
>>> print("This is custom firmware")
This is custom firmware
>>> 
```

Add Python library to device before build the firmware

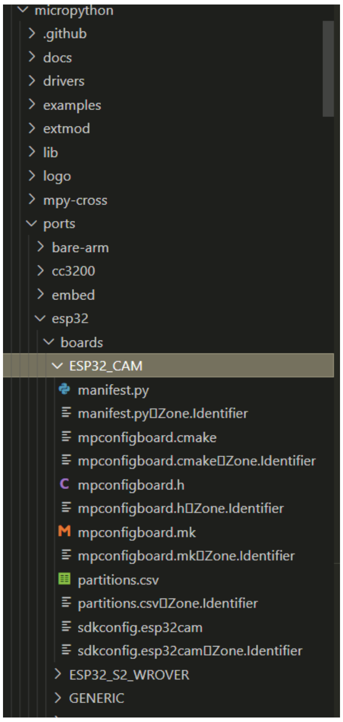


```python
freeze("$(PORT_DIR)/modules")
include("$(MPY_DIR)/extmod/uasyncio")

# Useful networking-related packages.
require("bundle-networking")

# Require some micropython-lib modules.
require("dht")
require("ds18x20")
require("neopixel")
require("onewire")
require("umqtt.robust")
require("umqtt.simple")
require("upysh")
```

```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4396
ho 0 tail 12 room 4
load:0x40078000,len:13976
ho 0 tail 12 room 4
load:0x40080400,len:3332
entry 0x40080618
Performing initial setup
MicroPython v1.19.1-915-g2bcd88d55-dirty on 2023-03-06; ESP32 module with ESP32
Type "help()" for more information.
>>> import camera
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: no module named 'camera'
>>> import upip
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: no module named 'upip'
>>> import sensor
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: no module named 'sensor'
>>> 
```

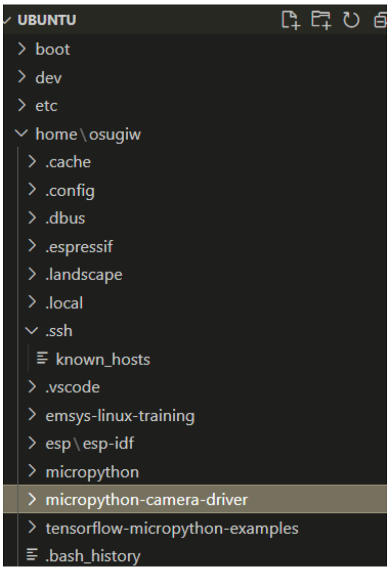
Add Camera Driver to Micropython

- Copy files inside "/board" to "micropython/ports/esp32/boards" by cloning file from lemariva/micropython-camera-driver: add camera support to MicroPython (github.com)



```
micropython
  .github
  docs
  drivers
  examples
  extmod
  lib
  logo
  mpy-cross
  ports
    bare-arm
    cc3200
    embed
    esp32
      boards
        ESP32_CAM
          manifest.py
          manifest.py□Zone.Identifier
          mpconfigboard.cmake
          mpconfigboard.cmake□Zone.Identifier
          mpconfigboard.h
          mpconfigboard.h□Zone.Identifier
          mpconfigboard.mk
          mpconfigboard.mk□Zone.Identifier
          partitions.csv
          partitions.csv□Zone.Identifier
          sdkconfig.esp32cam
          sdkconfig.esp32cam□Zone.Identifier
        ESP32_S2_WROVER
        GENERIC
```

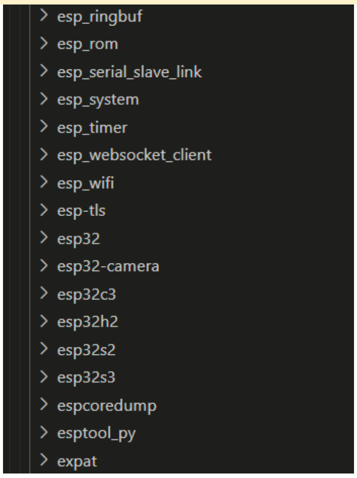
- Clone the camera driver (micropython-camera-driver) with the same level of micropython folder (lemariva/micropython-camera-driver: add camera support to MicroPython (github.com))



```
UBUNTU
  boot
  dev
  etc
  home\osugiw
    .cache
    .config
    .dbus
    .espressif
    .landscape
    .local
    .ssh
      known_hosts
    .vscode
    emsys-linux-training
    esp\esp-idf
    micropython
    micropython-camera-driver
    tensorflow-micropython-examples
    .bash_history
```

- Copy esp32-camera components to “~/esp/esp-idf/components”

```
cd ~/esp/esp-idf/components
git clone https://github.com/espressif/esp32-camera
```



```
> esp_ringbuf
> esp_rom
> esp_serial_slave_link
> esp_system
> esp_timer
> esp_websocket_client
> esp_wifi
> esp-tls
> esp32
> esp32-camera
> esp32c3
> esp32h2
> esp32s2
> esp32s3
> espcoredump
> esptool_py
> expat
```

- Compile the firmware:

```
cd micropython/ports/esp32
make USER_C_MODULES=../../../../micropython-camera-driver/src/micropython.cmake BOARD=ESP32_CAM all
```

- Erase the previous firmware inside the device and deploy the new firmware

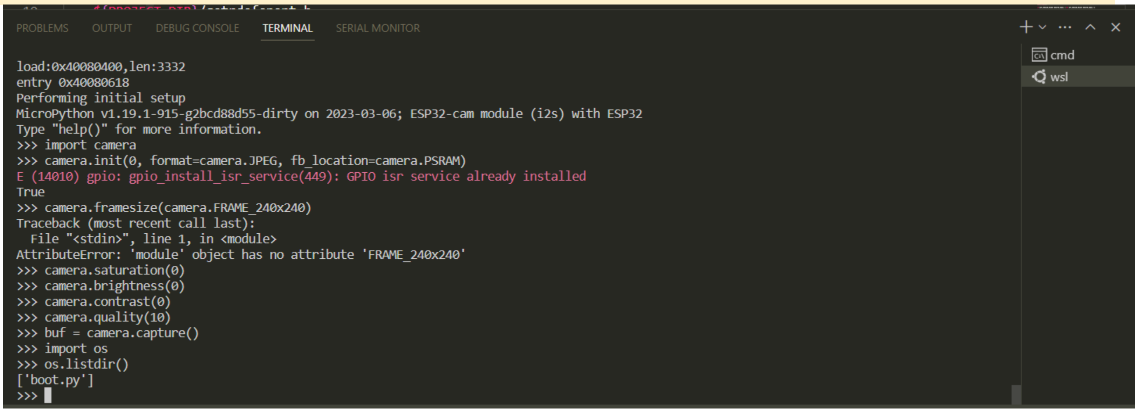
```
Make erase
esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000 build-ESP32_CAM/firmware.bin
```

- Connect to ESP32-Cam board, then press reset button

```
picocom -b 115200 /dev/ttyUSB0
```

- Try to import camera and initialize it

```
Import camera
Camera.init(0, format=camera.JPEG, fb_location=camera.PSRAM)
```



```
load:0x40080400,len:3332
entry 0x40080618
Performing initial setup
MicroPython v1.19.1-915-g2bcd88d55-dirty on 2023-03-06; ESP32-cam module (i2s) with ESP32
Type "help()" for more information.
>>> import camera
>>> camera.init(0, format=camera.JPEG, fb_location=camera.PSRAM)
E (14010) gpio: gpio_install_isr_service(449): GPIO isr service already installed
True
>>> camera.framesize(camera.FRAME_240x240)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'FRAME_240x240'
>>> camera.saturation(0)
>>> camera.brightness(0)
>>> camera.contrast(0)
>>> camera.quality(10)
>>> buf = camera.capture()
>>> import os
>>> os.listdir()
['boot.py']
>>> 
```

Tensorflow Lite in ESP32 (C/C++ Version)

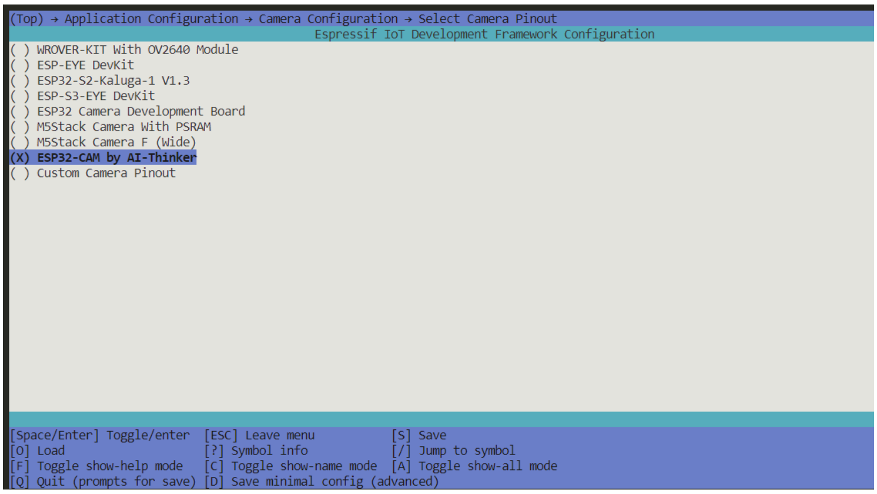
- Clone the tflite-micro-esp-examples (https://github.com/espressif/tflite-micro-esp-examples.git)
- Enter to tflite-micro-esp-examples folder, specifically in examples/person_detection/

```
Cd  tflite-micro-esp-examples/examples/person_detection
```

- Setting the camera type to ESP32-CAM by AI-Thinker by entering menuconfig first

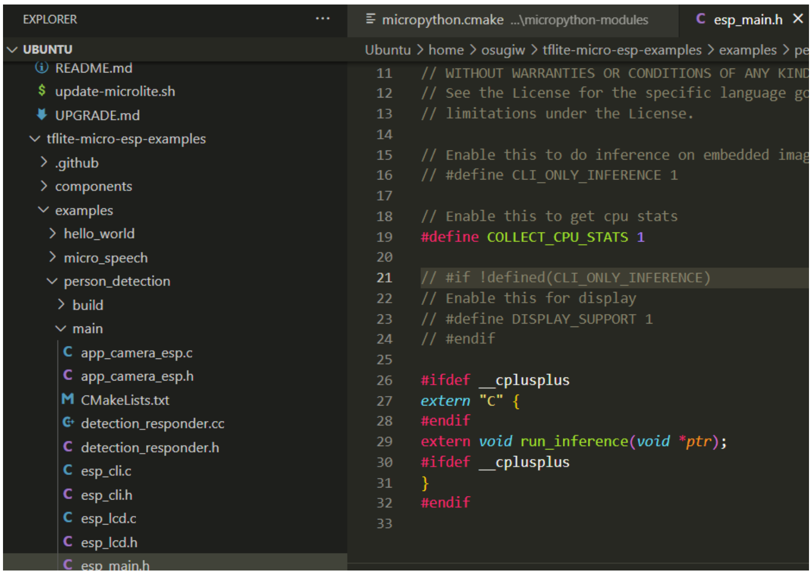
```
Idf.py menuconfig
```

- Under application configuration -> Camera Configuration -> Select Camera -> ESP32-CAM by AI-Thinker. After that, type S to save the configuration and ESC to escape



```
(Top) → Application Configuration → Camera Configuration → Select Camera Pinout
                        Espressif IoT Development Framework Configuration
( ) WROVER-KIT With OV2640 Module
( ) ESP-EYE DevKit
( ) ESP32-S2-Kaluga-1 V1.3
( ) ESP-S3-EYE DevKit
( ) ESP32 Camera Development Board
( ) M5Stack Camera With PSRAM
( ) M5Stack Camera F (Wide)
(X) ESP32-CAM by AI-Thinker
( ) Custom Camera Pinout

[Space/Enter] Toggle/enter  [ESC] Leave menu           [S] Save
[O] Load                    [?] Symbol info            [/] Jump to symbol
[F] Toggle show-help mode   [C] Toggle show-name mode  [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

- To inference using camera please comment #define CLI_ONLY_INFERENCE 1 inside /examples/person_detection/main/esp_main.h or Uncomment that MACROS to inference on embedded image



```
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND
// See the License for the specific language go
// limitations under the License.

// Enable this to do inference on embedded imag
// #define CLI_ONLY_INFERENCE 1

// Enable this to get cpu stats
#define COLLECT_CPU_STATS 1

// #if !defined(CLI_ONLY_INFERENCE)
// Enable this for display
// #define DISPLAY_SUPPORT 1
// #endif

#ifdef __cplusplus
extern "C" {
#endif
extern void run_inference(void *ptr);
#ifdef __cplusplus
}
#endif
```

- Set the target board and begin to build the firmware

```
Idf.py set-target esp32

Idf.py clean build
```

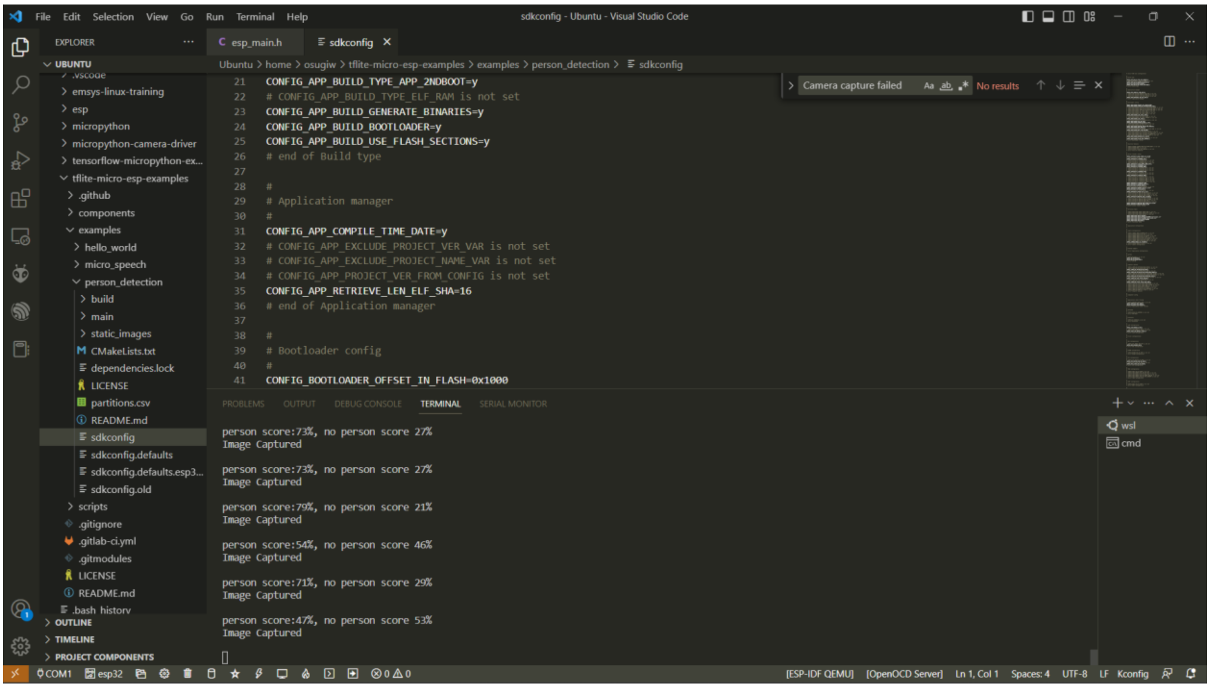
- Give permission to the USB port and flash the image

```
Sudo chmod 777 /dev/ttyUSB0

Idf.py -p /dev/ttyUSB0 flash
```

- Inference to detect person using picocom in linux or Thonny from windows

```
picocom -b 115200 /dev/ttyUSB0
```

- The model will run automatically to detect person from camera



```
CONFIG_APP_BUILD_TYPE_APP_2NDBOOT=y
# CONFIG_APP_BUILD_TYPE_ELF_RAM is not set
CONFIG_APP_BUILD_GENERATE_BINARIES=y
CONFIG_APP_BUILD_BOOTLOADER=y
CONFIG_APP_BUILD_USE_FLASH_SECTIONS=y
# end of Build type

#
# Application manager
#
CONFIG_APP_COMPILE_TIME_DATE=y
# CONFIG_APP_EXCLUDE_PROJECT_VER_VAR is not set
# CONFIG_APP_EXCLUDE_PROJECT_NAME_VAR is not set
# CONFIG_APP_PROJECT_VER_FROM_CONFIG is not set
CONFIG_APP_RETRIEVE_LEN_ELF_SHA=16
# end of Application manager

#
# Bootloader config
#
CONFIG_BOOTLOADER_OFFSET_IN_FLASH=0x1000
```

```
person score:73%, no person score 27%
Image Captured

person score:73%, no person score 27%
Image Captured

person score:79%, no person score 21%
Image Captured

person score:54%, no person score 46%
Image Captured

person score:71%, no person score 29%
Image Captured

person score:47%, no person score 53%
Image Captured
```

Tensorflow Lite in ESP32 (Micropython Version)

- Clone repository from https://github.com/mocleiri/tensorflow-micropython-examples.git
- Install python libraries for the virtual environment requirement

```
Pip3 install wave
Pip3 install pillow
```

- Enter to tensorflow-micropython-examples folder and download all submodules

```
Cd tensorflow-micropython-examples
Git submodule init
Git submodule update --recursive
```

- Regenerate the microlite/tflm directory

```
Cd tensorflow
../micropython-modules/microlite/prepare-tflm-esp.sh
```

- Setup micropython libraries

```
Cd ../micropython
Git init submodule
Git submodule update --recursive
```

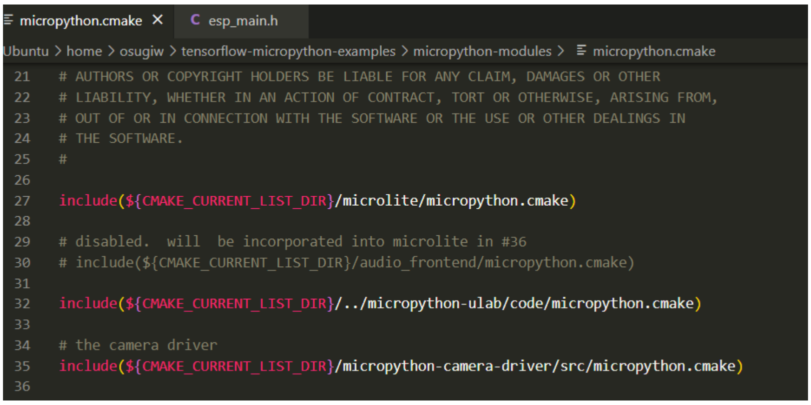
- Compile the cross compiler in micropython

```
Cd mpy-cross
make
```

- Clone the camera-driver-examples to replace the old one

```
Cd micropython-modules
Rm -rf camera-driver-examples
Git clone https://github.com/lemariva/micropython-camera-driver
```

- Edit micropython-modules/micropython.cmake on camera-driver directory path to resemble like this


micropython.cmake ×
esp_main.h
Ubuntu > home > osugiw > tensorflow-micropython-examples > micropython-modules > micropython.cmake

```
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.
#

include(${CMAKE_CURRENT_LIST_DIR}/microlite/micropython.cmake)

# disabled.  will  be incorporated into microlite in #36
# include(${CMAKE_CURRENT_LIST_DIR}/audio_frontend/micropython.cmake)

include(${CMAKE_CURRENT_LIST_DIR}/../micropython-ulab/code/micropython.cmake)

# the camera driver
include(${CMAKE_CURRENT_LIST_DIR}/micropython-camera-driver/src/micropython.cmake)

```

- Edit some code in micropython-modules/microlite/tensorflow/tensorflow-microlite.c

Line 215 from

```
const mp_obj_type_t microlite_tensor_type = {
    { &mp_type_type },
    .name = MP_QSTR_tensor,
    .print = tensor_print,
    .locals_dict = (mp_obj_dict_t*)&tensor_locals_dict,
};
```

to

```
MP_DEFINE_CONST_OBJ_TYPE(
    microlite_tensor_type,
    MP_QSTR_tensor,
    MP_TYPE_FLAG_NONE,
    print, tensor_print,
    locals_dict, (mp_obj_dict_t*)&tensor_locals_dict
);
```

Line 264 from

```
const mp_obj_type_t microlite_audio_frontend_type = {
    { &mp_type_type },
    .name = MP_QSTR_audio_frontend,
    .make_new = af_make_new,
    .print = af_print,
    .locals_dict = (mp_obj_dict_t*)&audio_frontend_locals_dict,
};
```

To

```
MP_DEFINE_CONST_OBJ_TYPE(
    microlite_audio_frontend_type,
    MP_QSTR_audio_frontend,
    MP_TYPE_FLAG_NONE,
    make_new, af_make_new,
    print, af_print,
    locals_dict, (mp_obj_dict_t*)&audio_frontend_locals_dict
);
```

Line 414 from

```
const mp_obj_type_t microlite_interpreter_type = {
   { &mp_type_type },
   .name = MP_QSTR_interpreter,
   .print = interpreter_print,
   .make_new = interpreter_make_new,
   .locals_dict = (mp_obj_dict_t*)&interpreter_locals_dict,
};
```

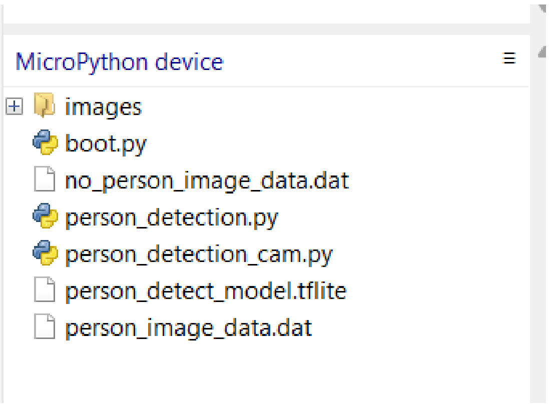
To

```
MP_DEFINE_CONST_OBJ_TYPE(
   microlite_interpreter_type,
   MP_QSTR_interpreter,
   MP_TYPE_FLAG_NONE,
   print, interpreter_print,
   make_new, interpreter_make_new,
   locals_dict, (mp_obj_dict_t*)&interpreter_locals_dict
);
```

- Begin to build the firmware. Change directory to tensorflow-micropython examples\boards\esp32\MICROLITE_SPIRAM_CAM

```
Idf.py set-target esp32
Idf.py clean build
Idf.py flash
```

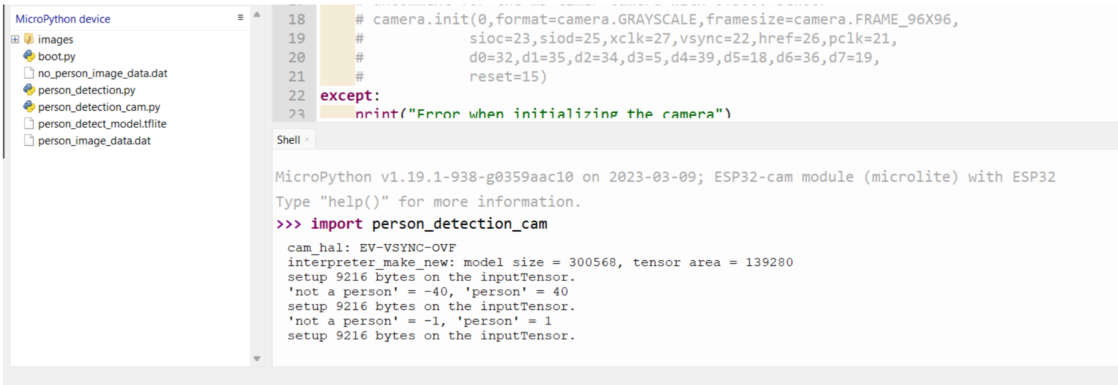
- After successfully flashing the firmware, upload these necessary files to the device. In my case, I was using Thonny to upload files. (Ps. These files are from /examples/person_detection)



MicroPython device

images
boot.py
no_person_image_data.dat
person_detection.py
person_detection_cam.py
person_detect_model.tflite
person_image_data.dat

- Finally, do inference by typing “import person_detection_cam” in the shell. The LED will flash if the model detect person in the camera, morover the confidence threshold in the default setting is 10 and could be adjusted depend on the needs.



```
    # camera.init(0,format=camera.GRAYSCALE,framesize=camera.FRAME_96X96,
    #             sioc=23,siod=25,xclk=27,vsync=22,href=26,pclk=21,
    #             d0=32,d1=35,d2=34,d3=5,d4=39,d5=18,d6=36,d7=19,
    #             reset=15)
except:
    print("Error when initializing the camera")
```

```
MicroPython v1.19.1-938-g0359aac10 on 2023-03-09; ESP32-cam module (microlite) with ESP32
Type "help()" for more information.
>>> import person_detection_cam
 cam_hal: EV-VSYNC-OVF
 interpreter_make_new: model size = 300568, tensor area = 139280
 setup 9216 bytes on the inputTensor.
 'not a person' = -40, 'person' = 40
 setup 9216 bytes on the inputTensor.
 'not a person' = -1, 'person' = 1
 setup 9216 bytes on the inputTensor.
```