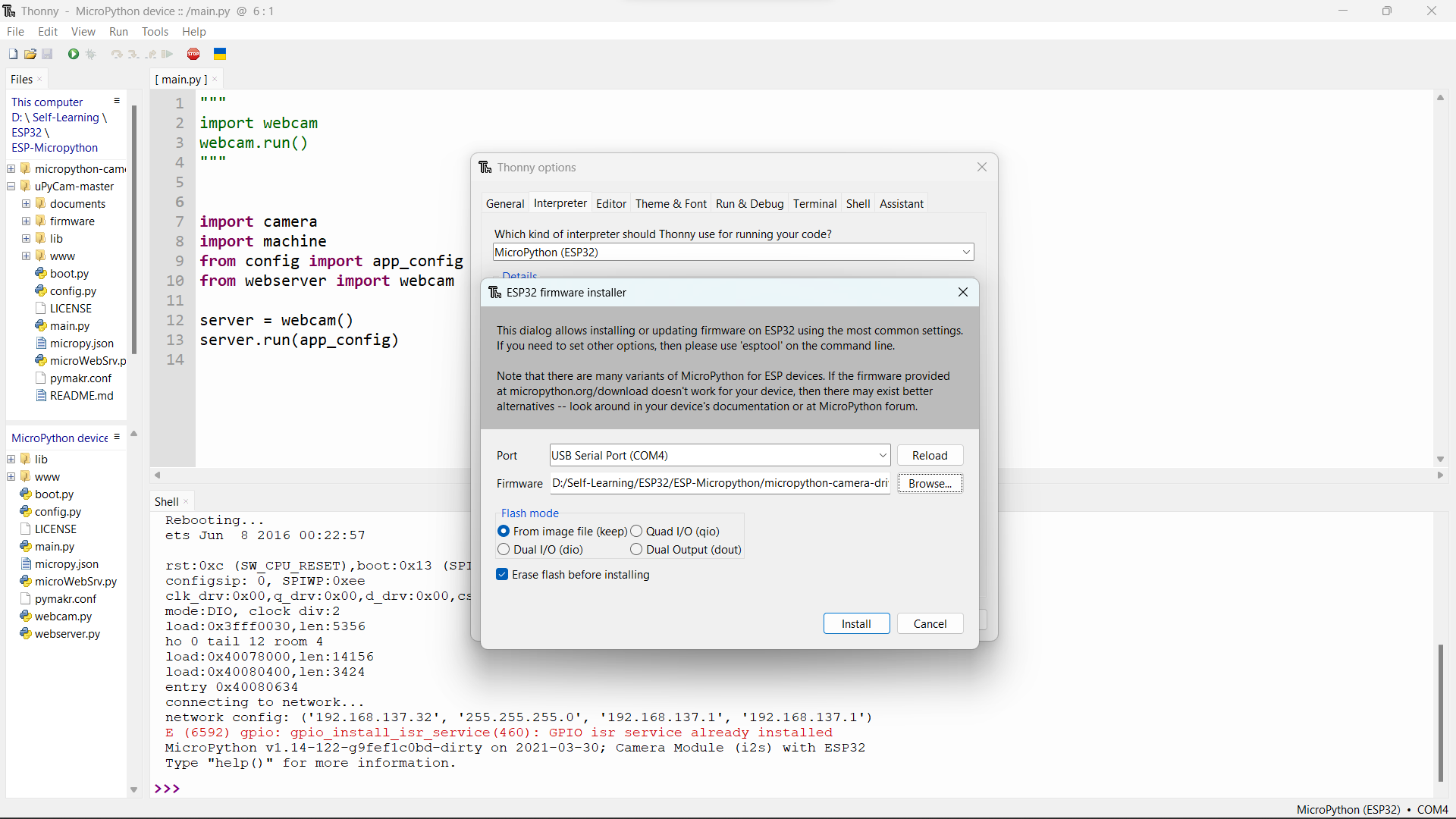
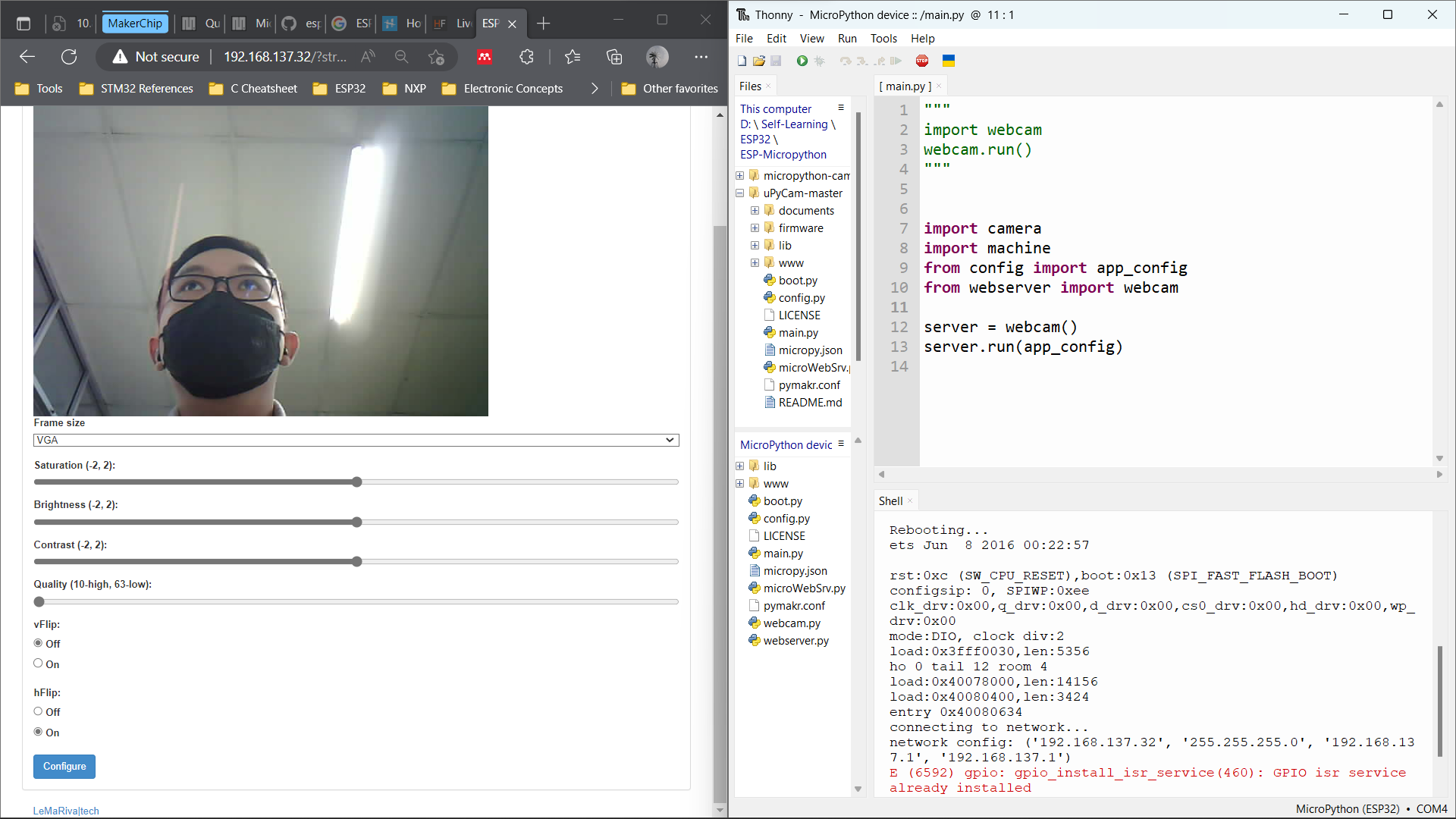
**Micropython ESP32 Camera**

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

****

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

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



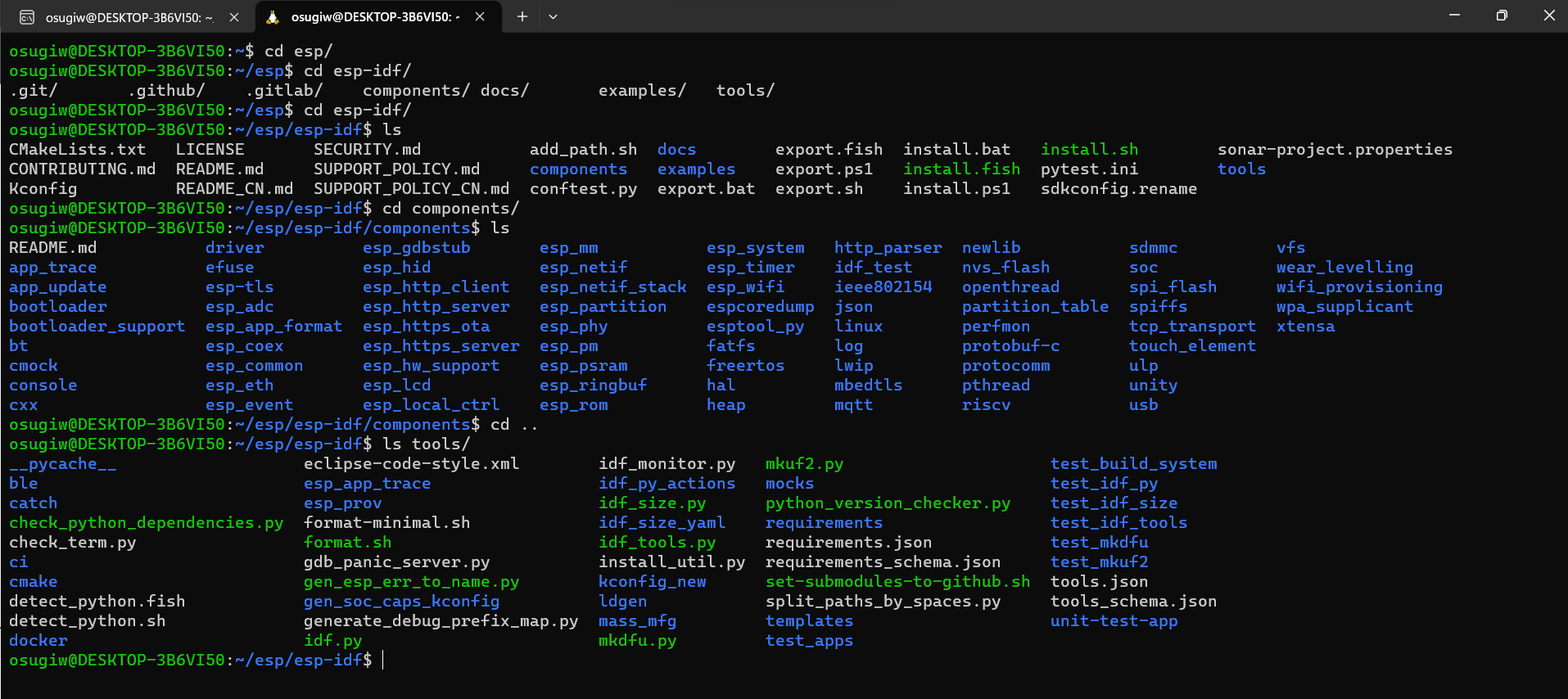
**Experiment Micropython & ESP-IDF Custom Firmware**

*(Only works with ESP-IDF v.4.4.4)*

**ESP Tools for Build Firmware**

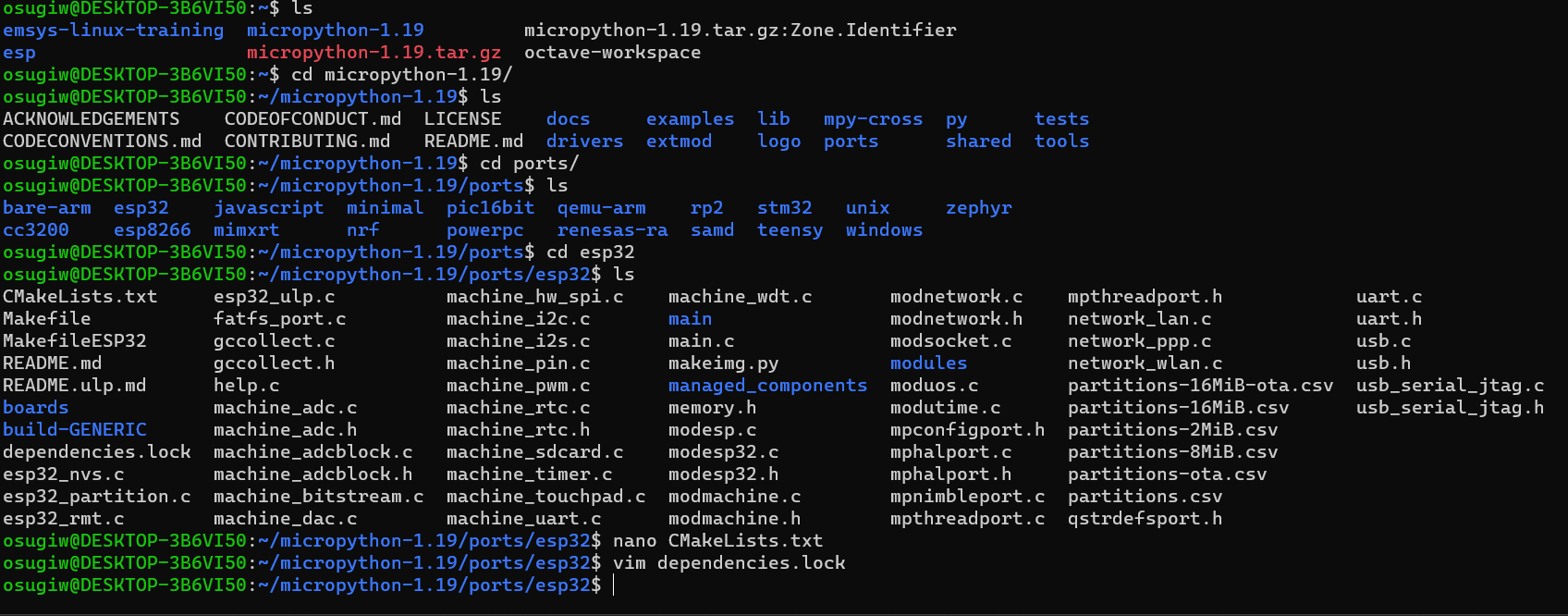
*Follow these steps:*

* [Standard Toolchain Setup for Linux and macOS - ESP32 - — ESP-IDF Programming Guide latest documentation (espressif.com)](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/linux-macos-setup.html) or;
* [micropython/ports/esp32 at master · micropython/micropython (github.com)](https://github.com/micropython/micropython/tree/master/ports/esp32)

****

**Micropython File Structures**

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

****

***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/**

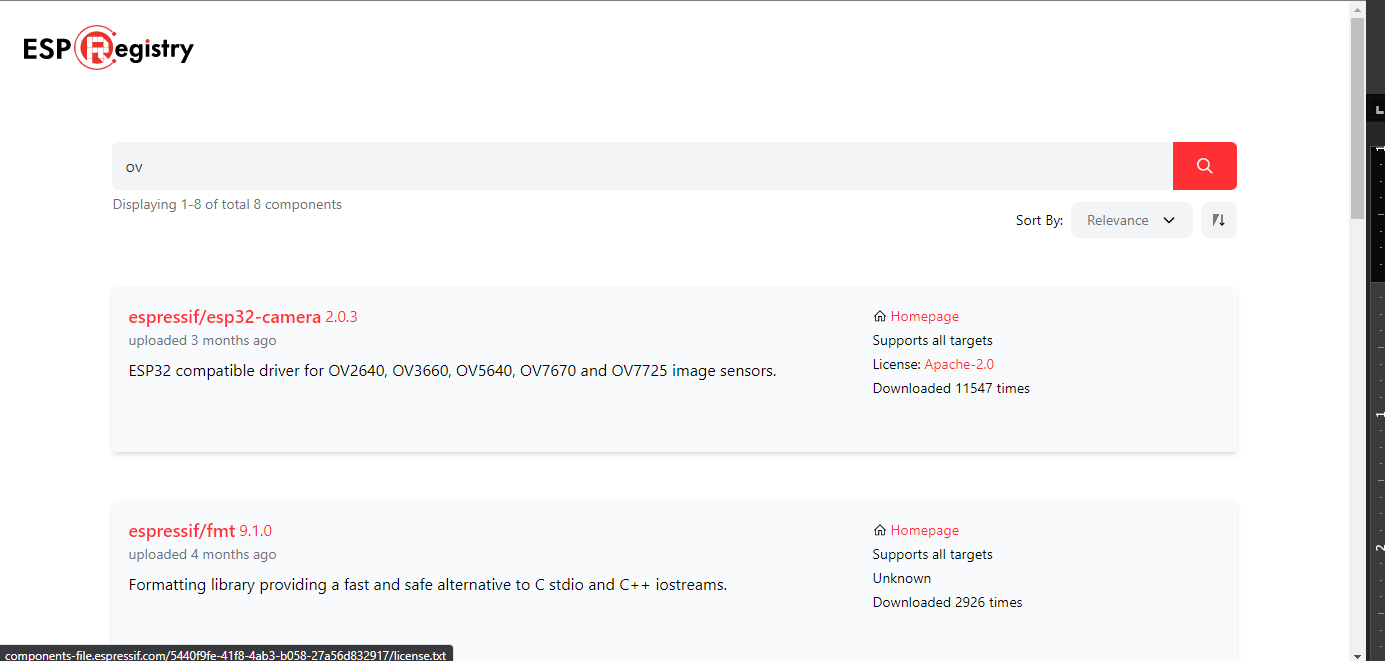
**nano Makefile**

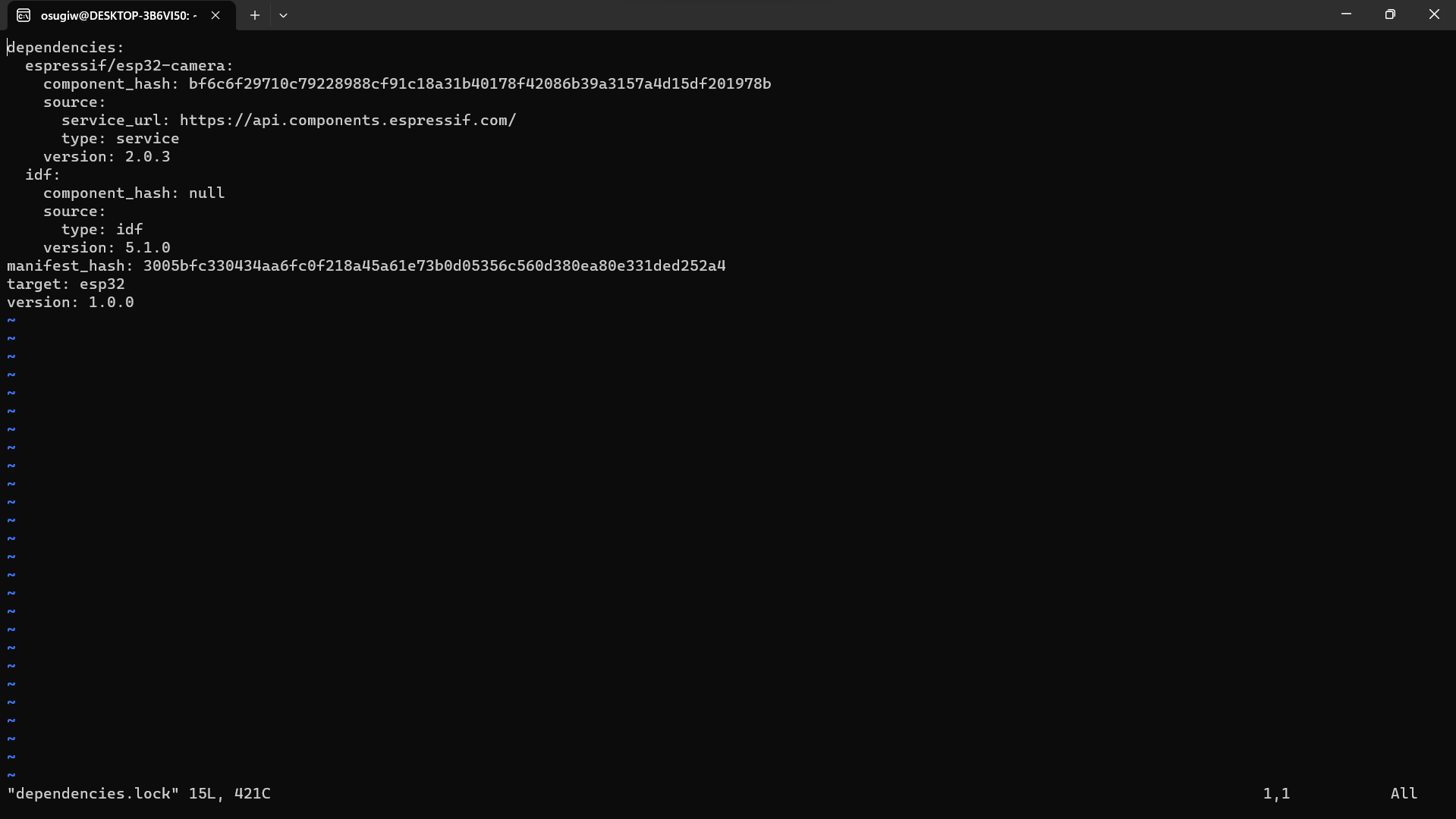
**ESPIDF = /home/(USER)/esp/esp-idf**

**PORT = /dev/ttyUSB0**

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

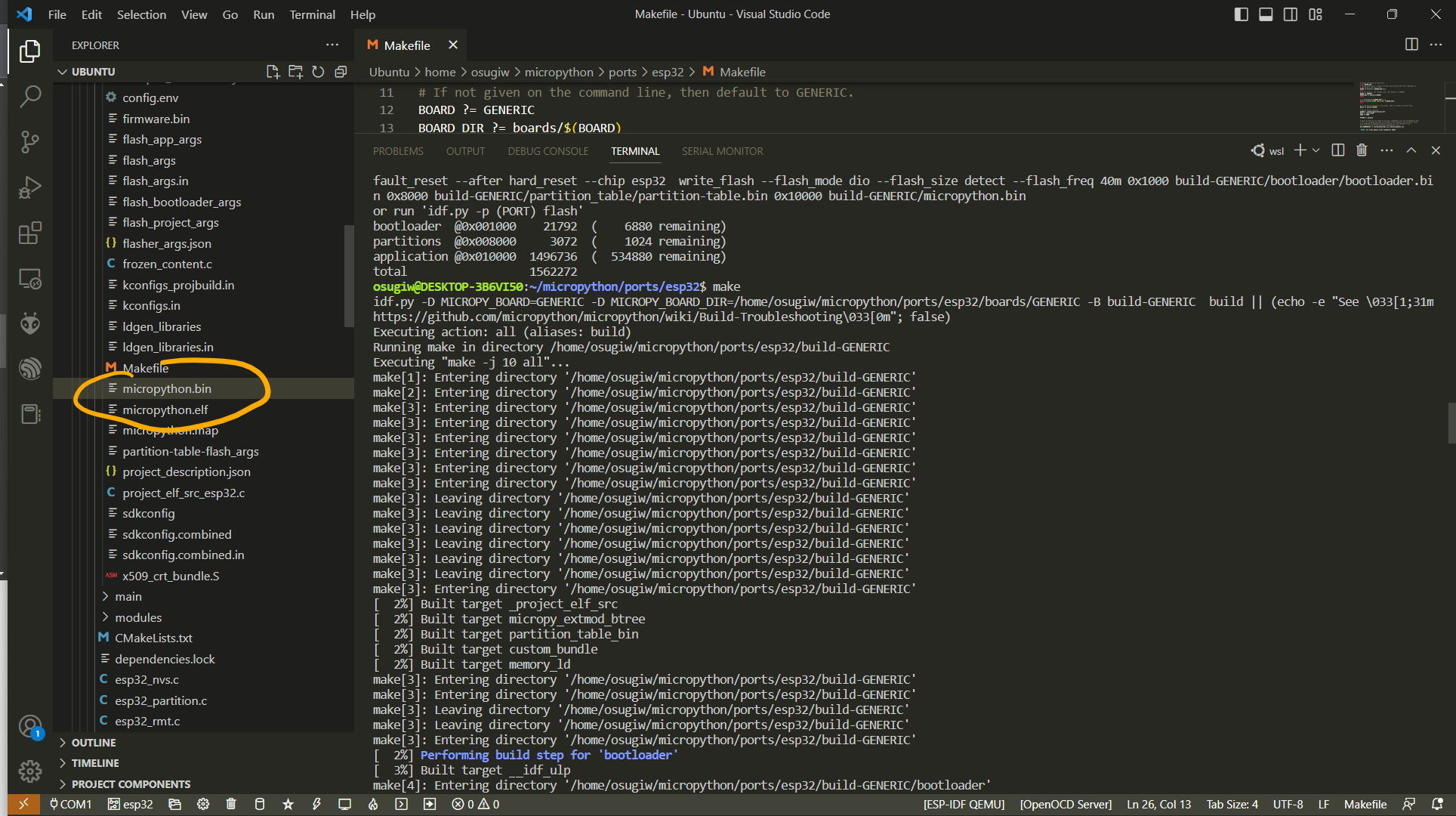
**idf.py add-dependency "espressif/esp32-camera^2.0.3"**

****



* Compile Makefile

**make**

****

**Connecting USB through WSL**

*Reference:* [You can now connect USB devices in Windows Subsystem for Linux under Windows 11 (xda-developers.com)](https://www.xda-developers.com/wsl-connect-usb-devices-windows-11/)

* Install the latest usbipd-win (.msi) from[*https://github.com/dorssel/usbipd-win/releases/latest*](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-generic hwdata**

**sudo update-alternatives --install /usr/local/bin/usbip usbip /usr/lib/linux-tools/\*-generic/usbip 20**

* Open windows command prompt and list all attached devices

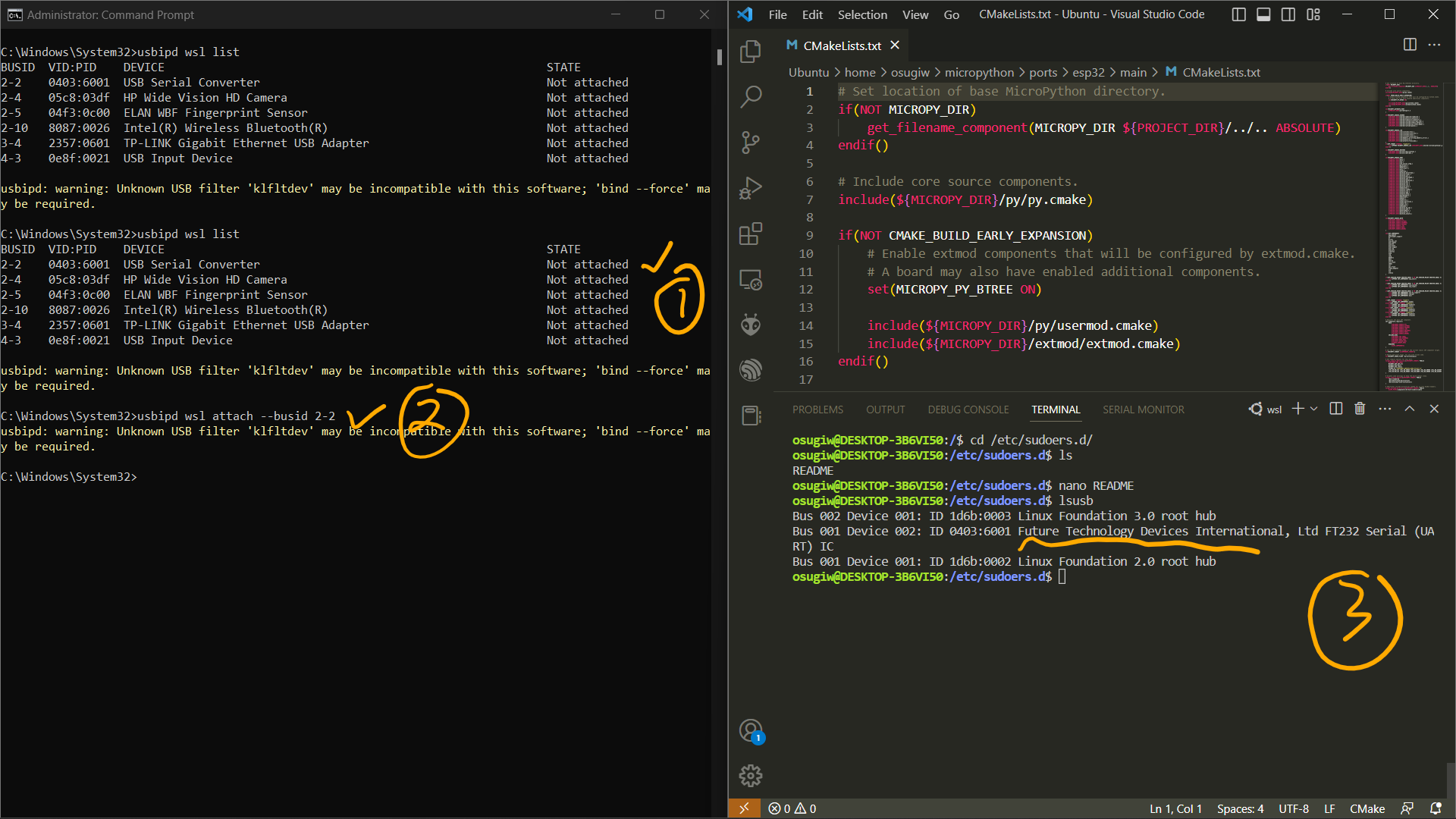
**usbipd wsl list**

* Attach ports that we need to WSL

**usbipd wsl attach --busid <busid>**

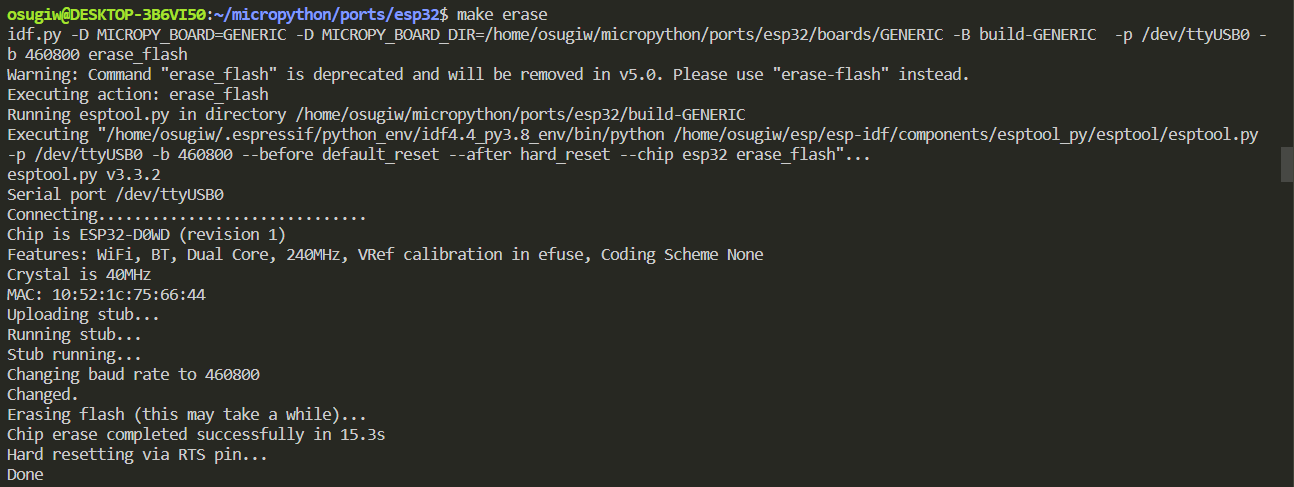
* Open WSL console again and list the attached USB in WSL

**lsusb**

****

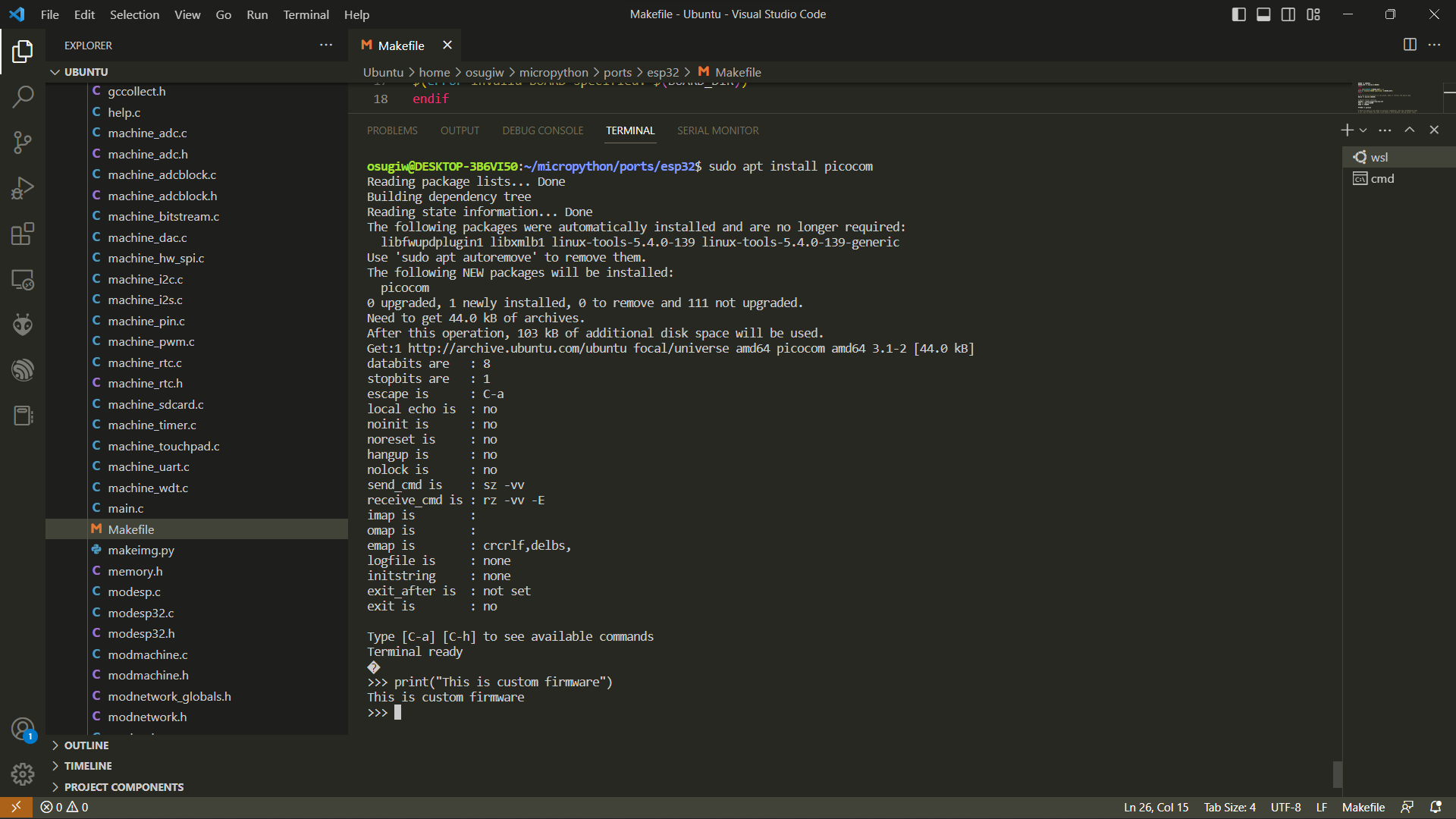
* Erase the existing firmware on ESP32

**Make erase**

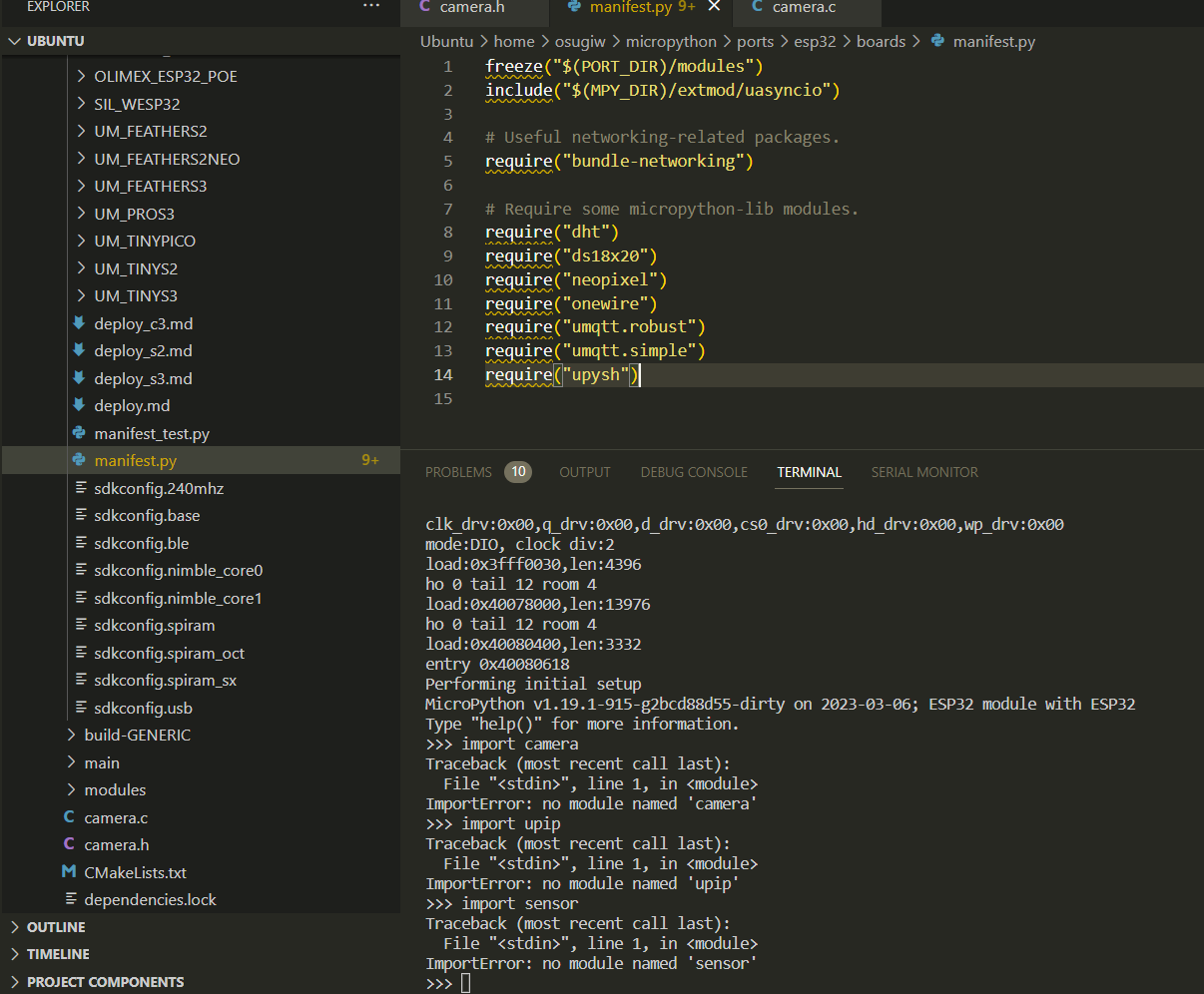
****

* Deploy to the ESP32 Board

**Make deploy**

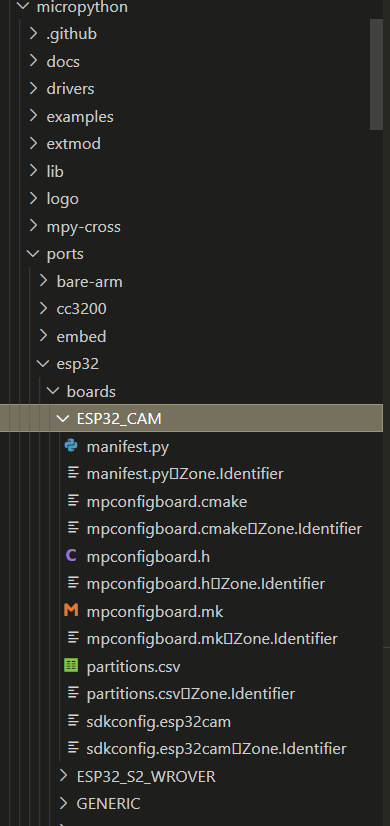
****

**Add Python library to device before build the firmware**

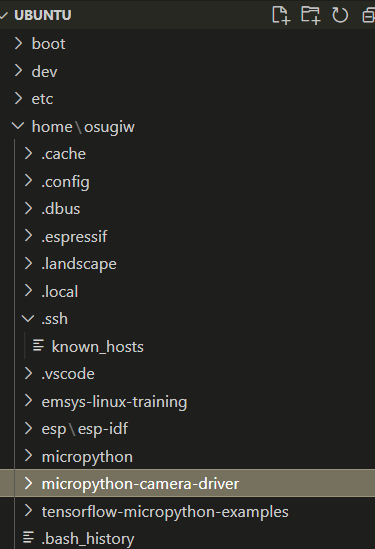


**Add Camera Driver to Micropython**

* Copy files inside “/board” to “micropython/ports/esp32/boards” by cloning file from [lemariva/micropython-camera-driver: add camera support to MicroPython (github.com)](https://github.com/lemariva/micropython-camera-driver)

****

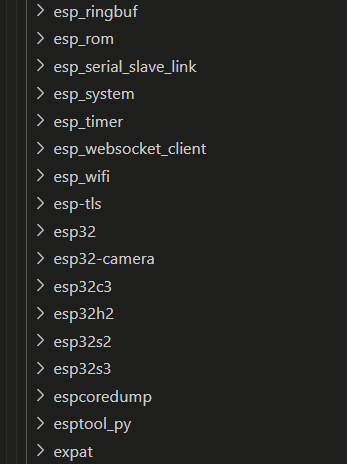
* Clone the camera driver (micropython-camera-driver) with the same level of micropython folder ([lemariva/micropython-camera-driver: add camera support to MicroPython (github.com)](https://github.com/lemariva/micropython-camera-driver))

****

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

**cd ~/esp/esp-idf/components**

**git clone https://github.com/espressif/esp32-camera**

****

* 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**

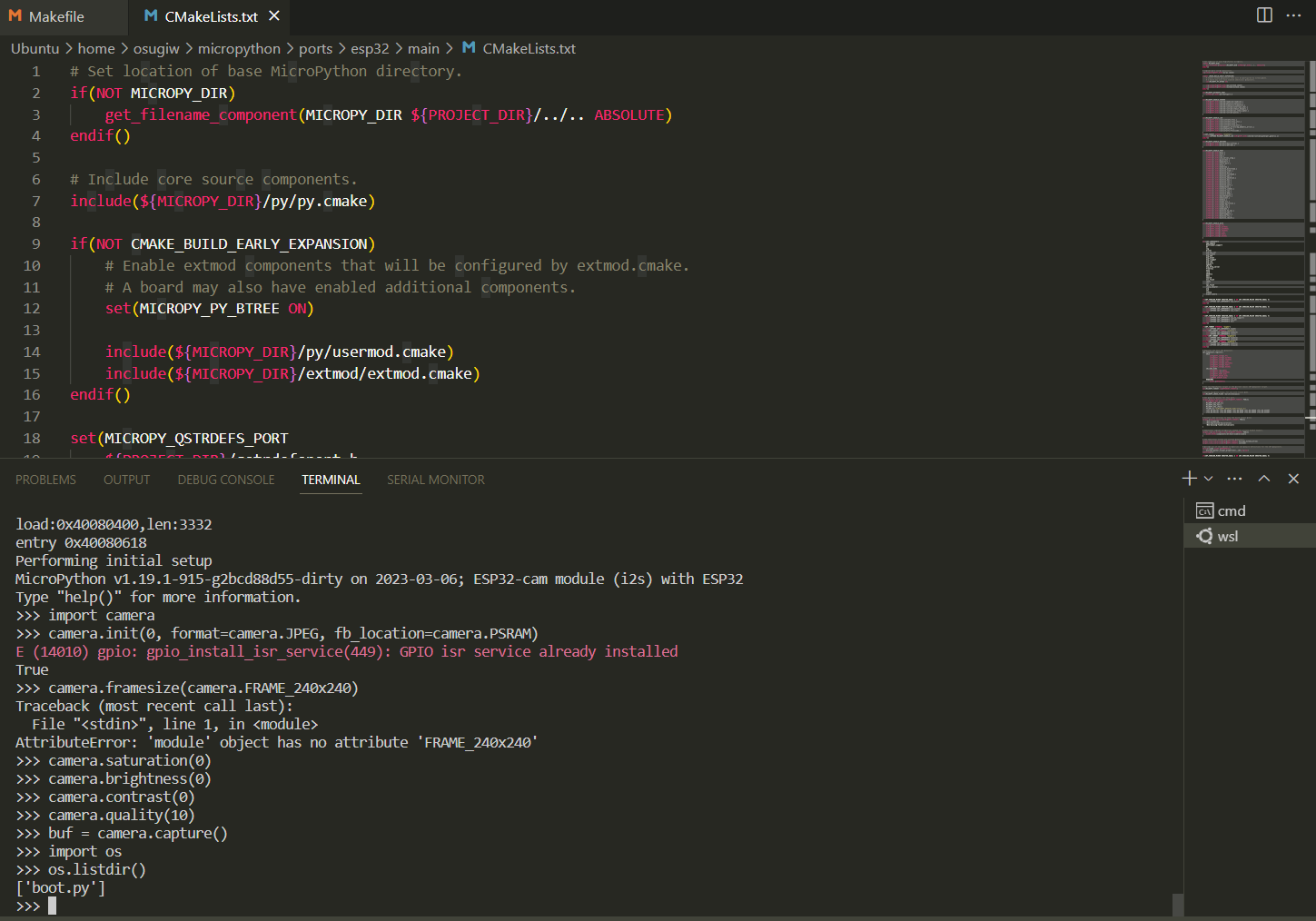
* Connect to ESP32-Cam board, then press reset button

**picocom -b 115200 /dev/ttyUSB0**

* Try to import camera and initialize it

**Import camera**

**Camera.init(0, format=camera.JPEG, fb\_location=camera.PSRAM)**

****

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

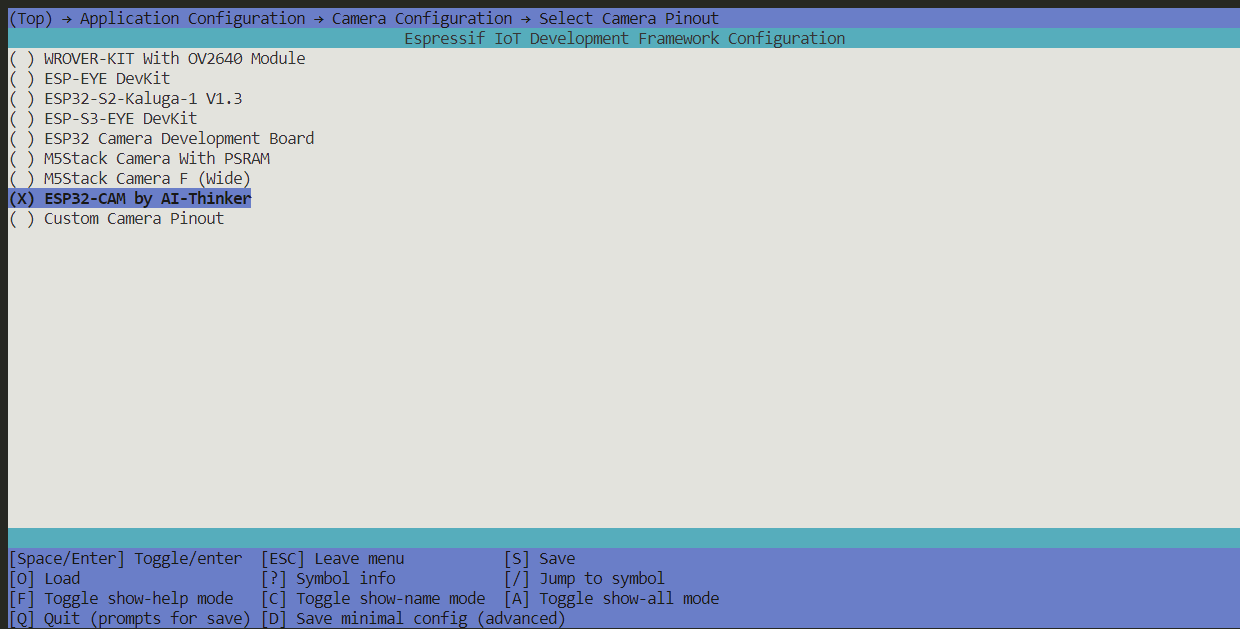
* Clone the tflite-micro-esp-examples (<https://github.com/espressif/tflite-micro-esp-examples.git>)
* Enter to tflite-micro-esp-examples folder, specifically in examples/person\_detection/

**Cd tflite-micro-esp-examples/examples/person\_detection**

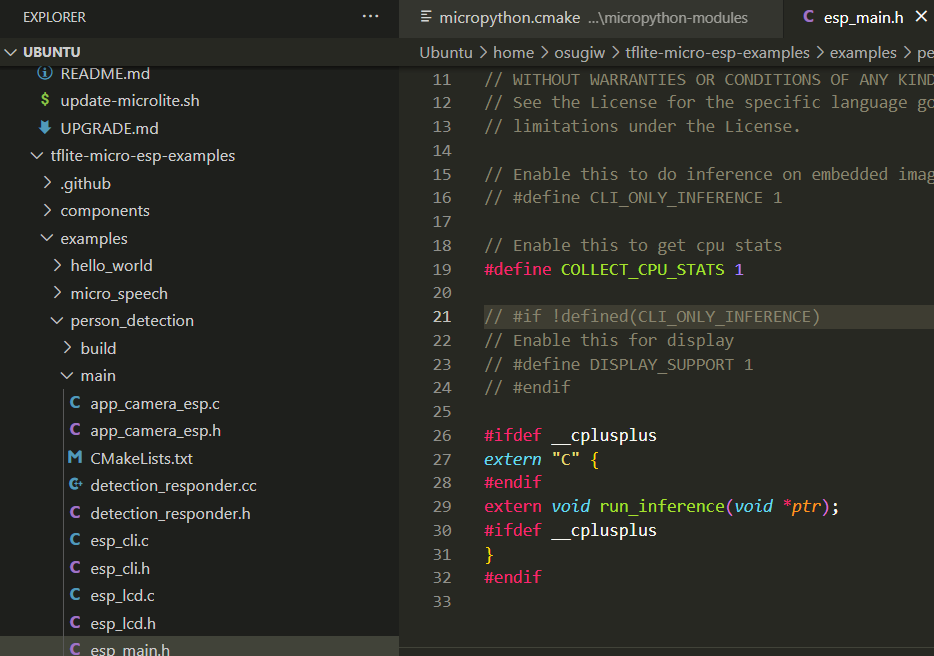
* Setting the camera type to ESP32-CAM by AI-Thinker by entering menuconfig first

**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



* 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

****

* 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

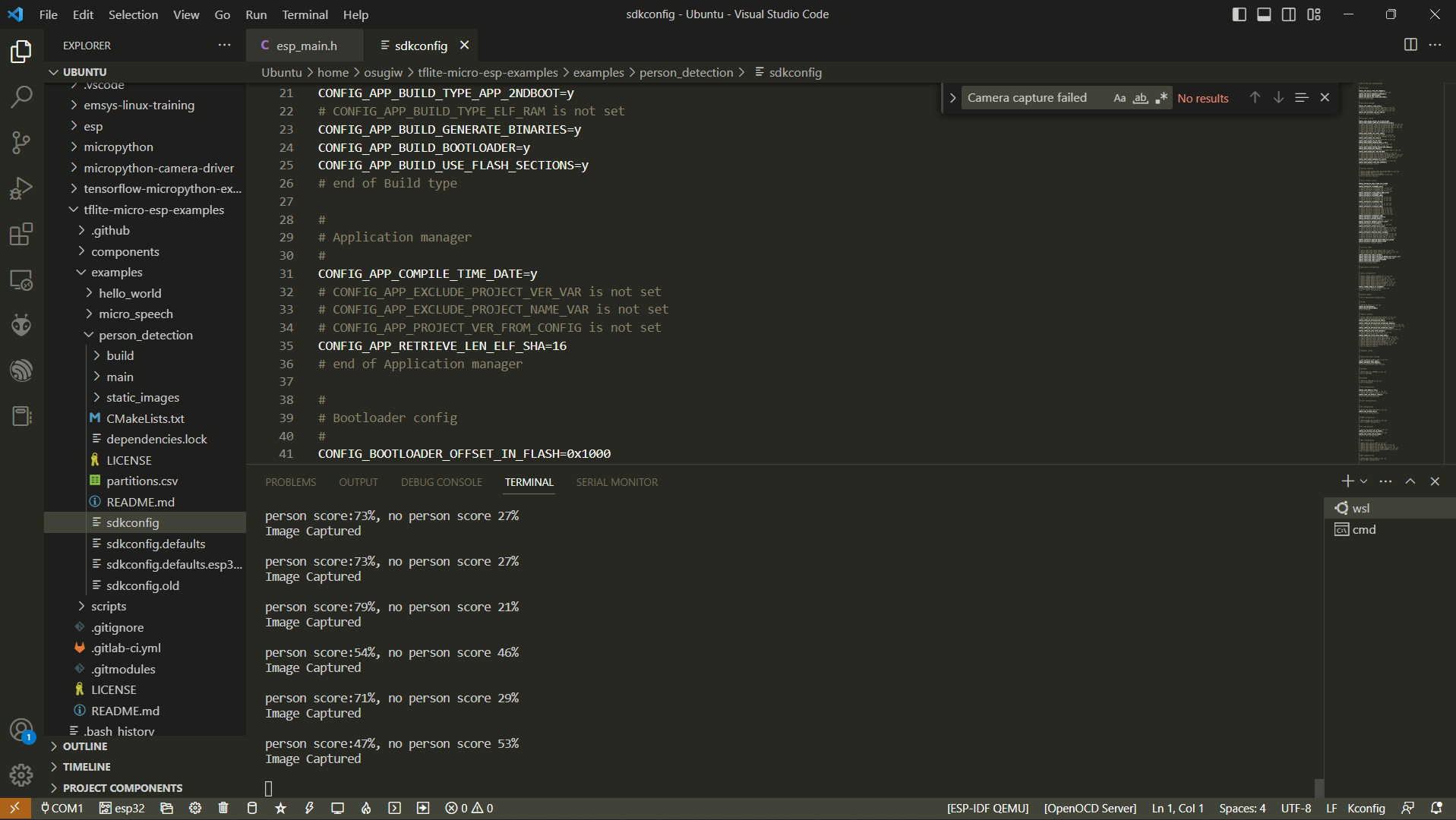
**Sudo chmod 777 /dev/ttyUSB0**

**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



**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**

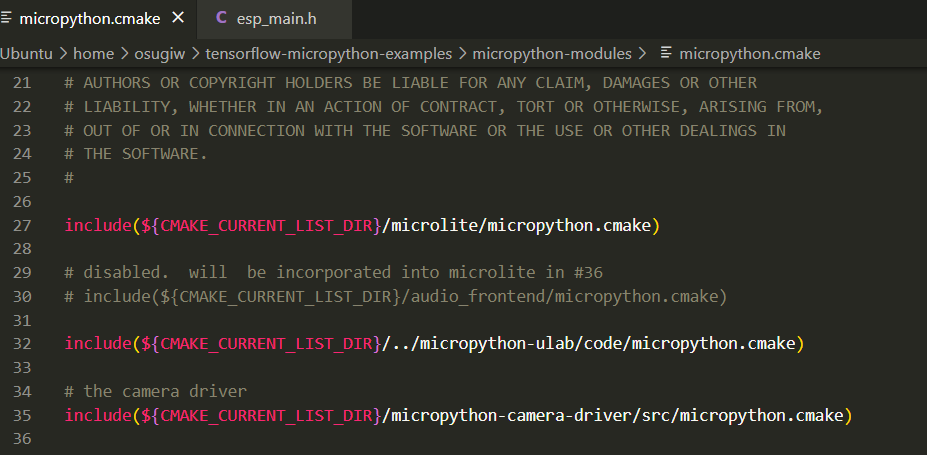
* Clone the camera-driver-examples to replace the old one

**Cd micropython-modules**

**Rm -rf camera-driver-examples**

**Git clone** [**https://github.com/lemariva/micropython-camera-driver**](https://github.com/lemariva/micropython-camera-driver)

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



* 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**

**);**

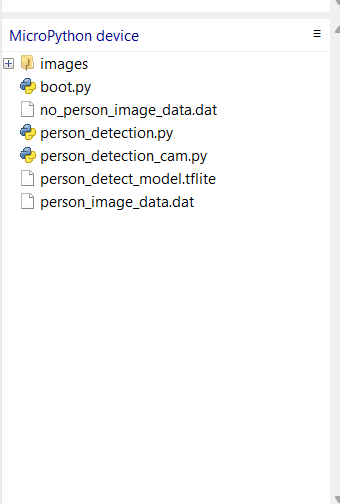
* Begin to build the firmware. Change directory to tensorflow-micropython examples\boards\esp32\MICROLITE\_SPIRAM\_CAM

**Idf.py set-target esp32**

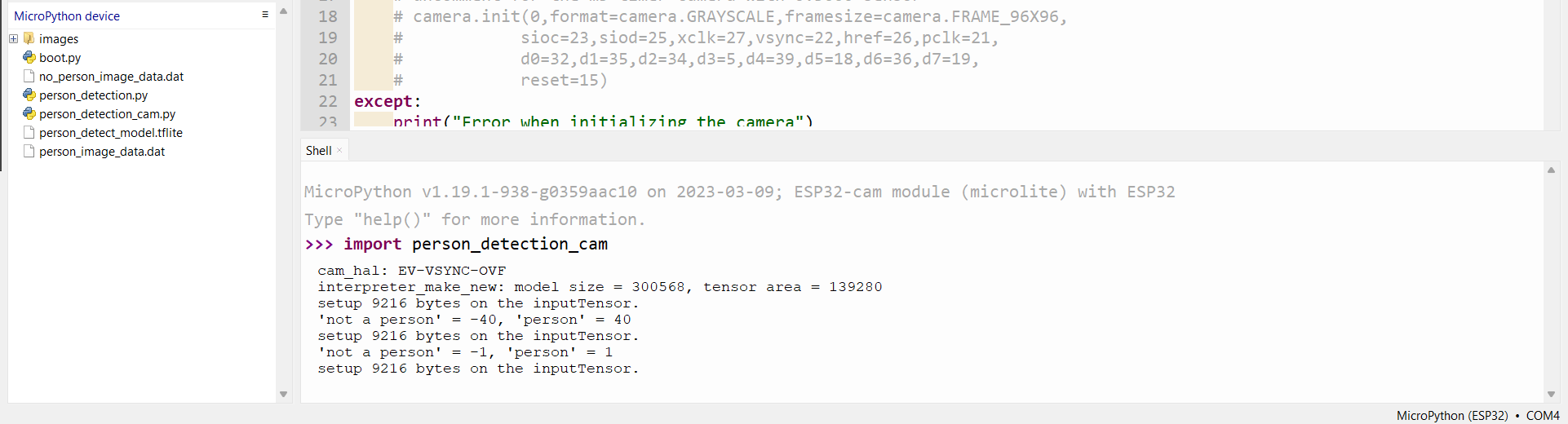
**Idf.py clean build**

**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)

****

* 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.

****

