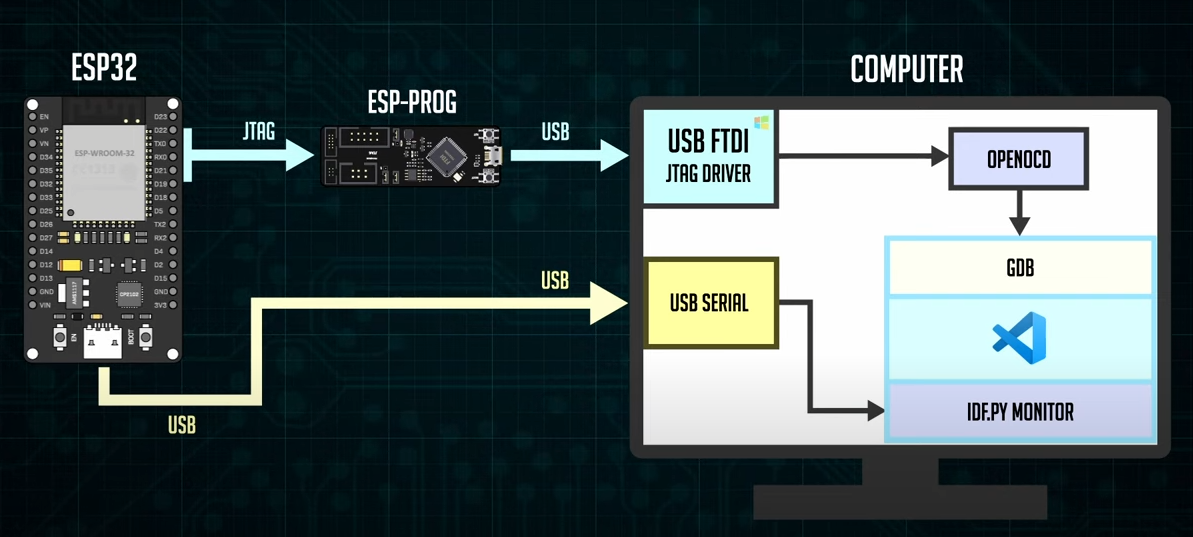
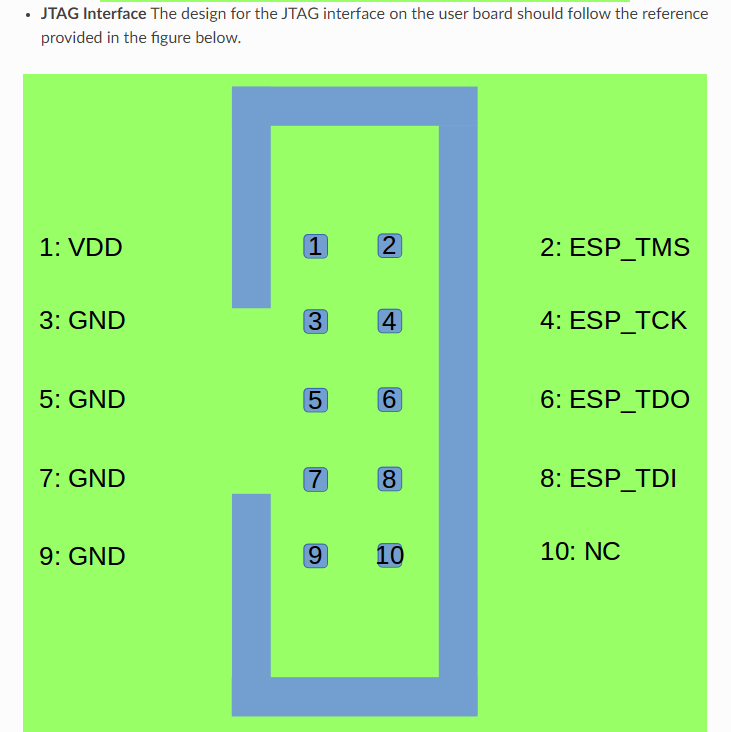
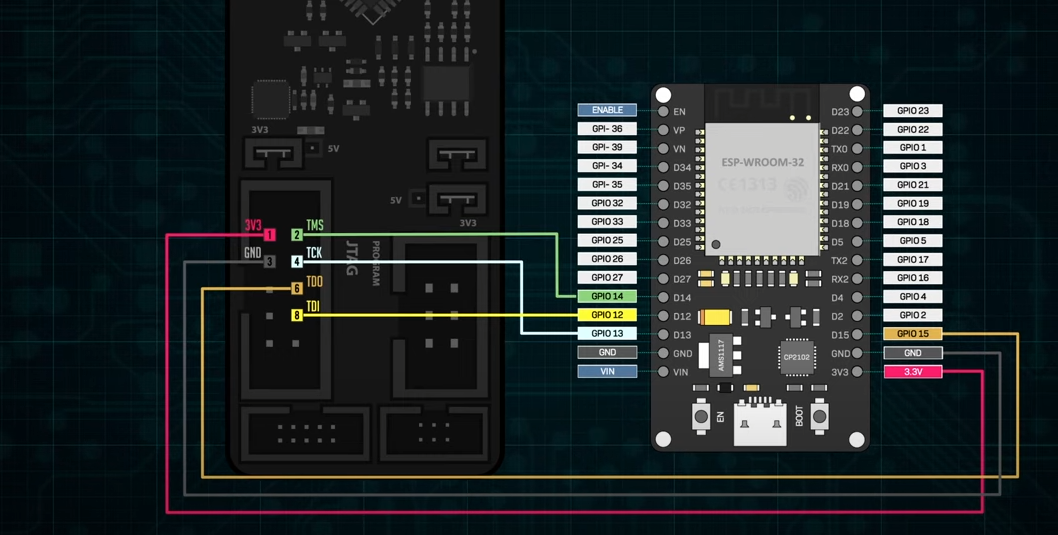
**ESP-Prog JTAG Debugger**

****

[3]



[1]



1. **Download Zadig**

[https://zadig.akeo.ie/#](https://zadig.akeo.ie/)

Abrir el programa y en options seleccionar “List all devices”. Aqui elegir Dual RS232/HS interface 0 y presionar reinstall driver.

Tener en cuenta que si se cambia de Puerto al debugger hay que repetir el proceso.

1. **OCD ( opcional)**

If you have already set up ESP-IDF with CMake build system according to the Getting Started Guide, then OpenOCD is already installed. After setting up the environment in your terminal ESP IDF, you should be able to run OpenOCD. Check this by executing the following command: [2]

openocd --version

The output should be as follows (although the version may be more recent than listed here):

Open On-Chip Debugger v0.10.0-esp32-20190708 (2019-07-08-11:04)

Licensed under GNU GPL v2

For bug reports, read

https://openocd.org/doc/doxygen/bugs.html

1. **Run OpenOCD**

Once target is configured and connected to computer, you are ready to launch OpenOCD.

Open a terminal and set it up for using the ESP-IDF as described in the setting up the environment section of the Getting Started Guide. Then run OpenOCD

* openocd -f board/esp32-wrover-kit-3.3v.cfg

1. **How to confiure GBD**

GDB comes installed with IDF. It is the gnu debugger, an open source debugging software used by VS to communicate with openOCD.

1. **How to configure Visual studio Code**

Open the file Launch.json which is inside of .vscode an replace the code for the one below

|  |
| --- |
| {    "version": "0.2.0",    "configurations": [      {          "name": "ESP32 OpenOCD",          "type": "cppdbg",          "request": "launch",          "cwd": "${workspaceFolder}/build",          "program": "${workspaceFolder}/build/${command:espIdf.getProjectName}.elf",          "miDebuggerPath": "C:\\Users\\eduar\\.espressif\\tools\\xtensa-esp32-elf\\esp-2022r1-11.2.0\\xtensa-esp32-elf\\bin\\xtensa-esp32-elf-gdb.exe",          "setupCommands": [              {"text": "target remote 127.0.0.1:3333" },              {"text": "set remote hardware-watchpoint-limit 2"},              {"text": "monitor reset halt"},              {"text": "flushregs"}          ]      }  ]  } |

En teoría con esto ya se puede. Solo hay que abrir el esp/idf monitor y poner

openocd -c 'set ESP\_RTOS none' -f board/esp32-wrover-kit-3.3v.cfg

Una vez que se este ejecutando el servidor anterior, flash the code y se ejecuta el debugger.

Sin embargo, hay otra manera de automatizar todo. (no funciona)

1. **Automating OPENOCD**

Para esto ir a tasks.json y añadir el siguiente código

|  |
| --- |
| {              "label": "preRun",              "type": "shell",              "windows":{                  "command": "clear ; start openocd -c \"set ESP\_RTOS none\" -f board/esp32-wrover-kit-3.3v.cfg ; exit"              }          } |

Luego ir a tasks.jason y añadir

|  |
| --- |
| {              "label": "preRun",              "type": "shell",              "windows":{                  "command": "clear ; start openocd -c \"set ESP\_RTOS none\" -f board/esp32-wrover-kit-3.3v.cfg ; exit"              }          } |

Y para que el sistema haga el flash desde el debuger añadir el siguiente código en launch.json

|  |
| --- |
| {"text": "flushregs"},                {"text": "mon program\_esp build/bootloader/bootloader.bin 0x1000 verify"},                {"text": "mon program\_esp build/partition\_table/partition-table.bin 0x8000 verify"},                {"text": "mon program\_esp build/blink.bin 0x10000 verify"},                {"text": "monitor reset halt"},                {"text": "flushregs"} |

**Bibliografía**

[1] <https://espressif-docs.readthedocs-hosted.com/projects/espressif-esp-iot-solution/en/latest/hw-reference/ESP-Prog_guide.html>

[2] <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/jtag-debugging/index.html#jtag-debugging-setup-openocd>

[3] https://www.youtube.com/watch?v=uq93H7T7cOQ&ab\_channel=YuriR

Aquí indica como lo hace <https://www.youtube.com/watch?v=NAUV1d53LLY&ab_channel=EasyLearning>