

1) Установка Программного Обеспечения

Ваш МАС обязательно должен иметь доступ в интернет!!!

Здесь нам понадобится *Terminal*, который находится в папке «Утилиты» на вашем Mac.

Запускаем *Terminal*:

```
Last login: Thu Apr 30 13:10:09 on console
belg21s-mbp:~ belg21$ █
```

Установка pip.

Вводим команду:

```
sudo easy_install pip
```

Нажимаем enter, далее вводим пароль от системы и опять нажимаем enter.

```
Last login: Thu Apr 30 13:10:09 on console
belg21s-mbp:~ belg21$ sudo easy_install pip
Password:
Searching for pip
Best match: pip 20.1
Processing pip-20.1-py2.7.egg
pip 20.1 is already the active version in easy-install.pth
Installing pip script to /usr/local/bin
Installing pip2.7 script to /usr/local/bin
Installing pip2 script to /usr/local/bin

Using /Library/Python/2.7/site-packages/pip-20.1-py2.7.egg
Processing dependencies for pip
Finished processing dependencies for pip
belg21s-mbp:~ belg21$ █
```

Установка pyserial.

Вводим команду:

```
pip install --user pyserial
```

```
Downloading/unpacking pyserial
  Downloading pyserial-2.7.tar.gz (122kB): 122kB downloaded
  Running setup.py (path:/private/var/folders/18/dkysrtmn0c75xwng3q1dx2qc0000gp/T/pip_build_vixtor/pyserial/setup.py) egg_info for package pyserial

Installing collected packages: pyserial
  Running setup.py install for pyserial
    changing mode of build/scripts-2.7/miniterm.py from 644 to 755

    changing mode of /Library/Frameworks/Python.framework/Versions/2.7/bin/miniterm.py to 755
Successfully installed pyserial
Cleaning up...
```

Система управления пакетами Pip скачается и установится.

Установка MACPorts.

Переходим по ссылке: <https://www.macports.org/install.php>

macOS Package (.pkg) Installer

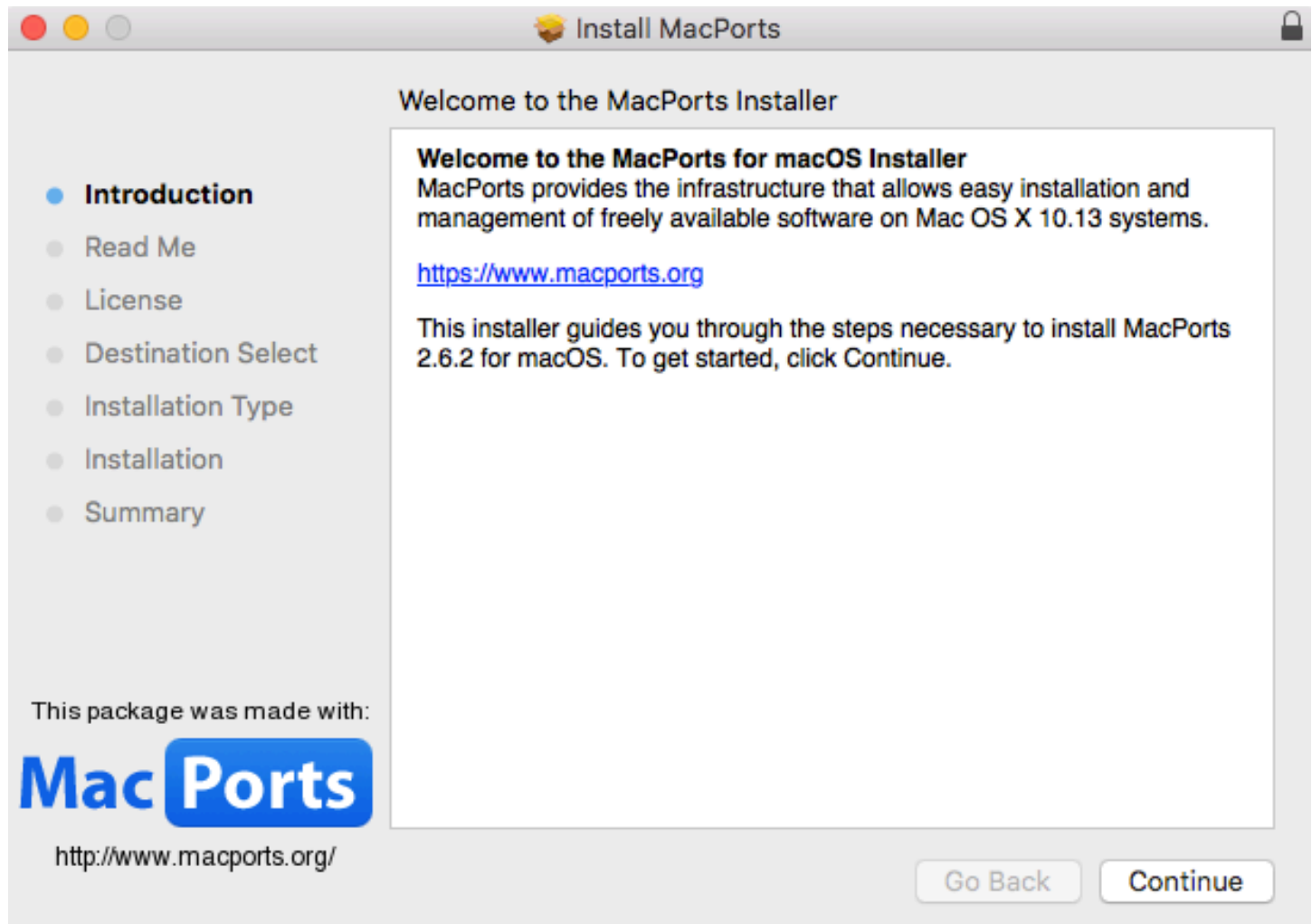
The easiest way to install MacPorts on a Mac is by downloading the pkg or dmg for [Catalina](#), [Mojave](#), [High Sierra](#), [Sierra](#), [El Capitan](#), [Yosemite](#), [Mavericks](#), [Mountain Lion](#), [Lion](#), [Snow Leopard](#), [Leopard](#) or [Tiger](#) and running the system's Installer by double-clicking on the pkg contained therein, following the on-screen instructions until completion.

This procedure will place a fully-functional and default MacPorts installation on your host system, ready for usage. If needed your shell configuration files will be [adapted by the installer](#) to include the necessary settings to run MacPorts and the programs it installs, but you may need to open a new shell for these changes to take effect.

The MacPorts "selfupdate" command will also be run for you by the installer to ensure you have our latest available release and the latest revisions to the "Portfiles" that contain the instructions employed in the building and installation of ports. After installation is done, it is recommended that you run this step manually on a regular basis to to keep your MacPorts system always current:

Выбираем вашу операционную систему и скачиваем .dmg для пакетной установки.

После скачивания установить.



Установка CMake & Ninja.

Возвращаемся в *Terminal* и вводим команду:

```
sudo port install cmake ninja
```

Начнется загрузка и установка пакетов.

```
belg21s-mbp:~ belg21$ sudo port install cmake ninja
---> Computing dependencies for cmake
---> Fetching archive for cmake
---> Attempting to fetch cmake-3.17.1_0.darwin_17.x86_64.tbz2 from https://pek.
cn.packages.macports.org/macports/packages/cmake
---> Attempting to fetch cmake-3.17.1_0.darwin_17.x86_64.tbz2 from http://mse.u
k.packages.macports.org/sites/packages.macports.org/cmake
---> Attempting to fetch cmake-3.17.1_0.darwin_17.x86_64.tbz2.rmd160 from http:
//mse.uk.packages.macports.org/sites/packages.macports.org/cmake
---> Installing cmake @3.17.1_0
---> Activating cmake @3.17.1_0
---> Cleaning cmake
---> Fetching archive for ninja
---> Attempting to fetch ninja-1.10.0_0.darwin_17.x86_64.tbz2 from https://pek.
cn.packages.macports.org/macports/packages/ninja
---> Attempting to fetch ninja-1.10.0_0.darwin_17.x86_64.tbz2 from http://mse.u
k.packages.macports.org/sites/packages.macports.org/ninja
---> Attempting to fetch ninja-1.10.0_0.darwin_17.x86_64.tbz2.rmd160 from http:
//mse.uk.packages.macports.org/sites/packages.macports.org/ninja
---> Installing ninja @1.10.0_0
---> Activating ninja @1.10.0_0
---> Cleaning ninja
---> Scanning binaries for linking errors
```

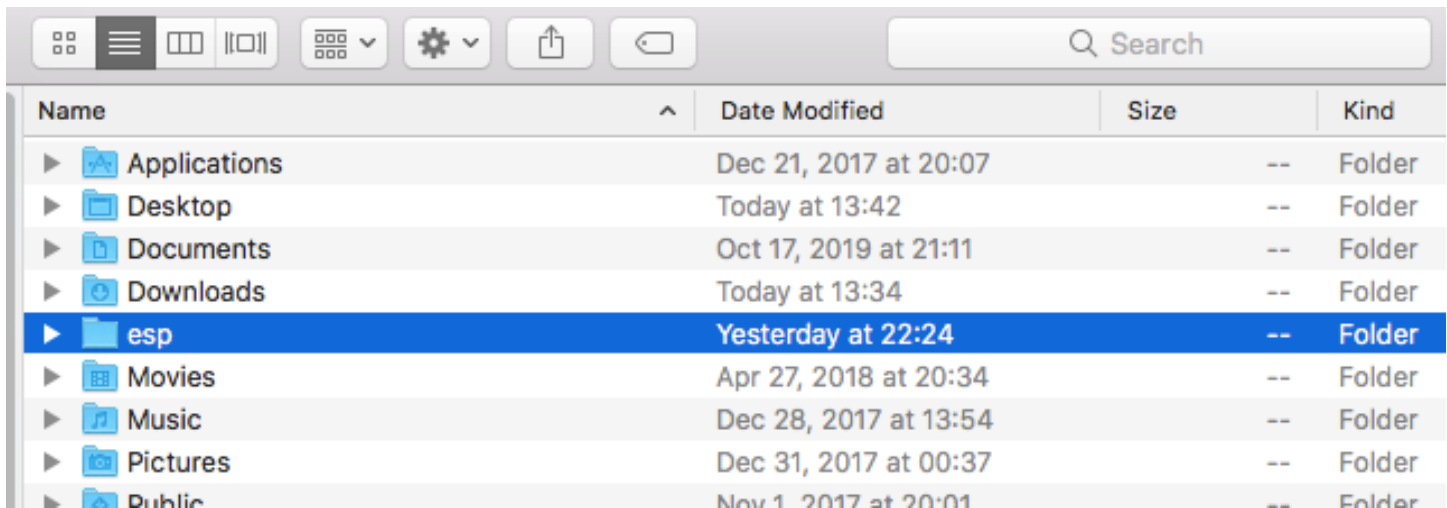
На этом же этапе если у вас отсутствует *XCode*, то *Terminal* предложит скачать и установить дополнительное ПО.

Установка библиотек программного обеспечения Espressif в хранилище ESP-IDF.

Вводим команды нажимая Enter после каждой:

```
cd ~/esp
git clone --recursive https://github.com/espressif/esp-idf.git
```

После загрузки (~ 1.2 gb) по адресу *MAC/users/«ваше имя пользователя»/* появится папка *esp*.



И так мы практически готовы начать прошивать, но еще надо установить драйвера на USB, а также скачать проект *PICKLE* от **Искандера**.

Установка драйверов USB CP210x.

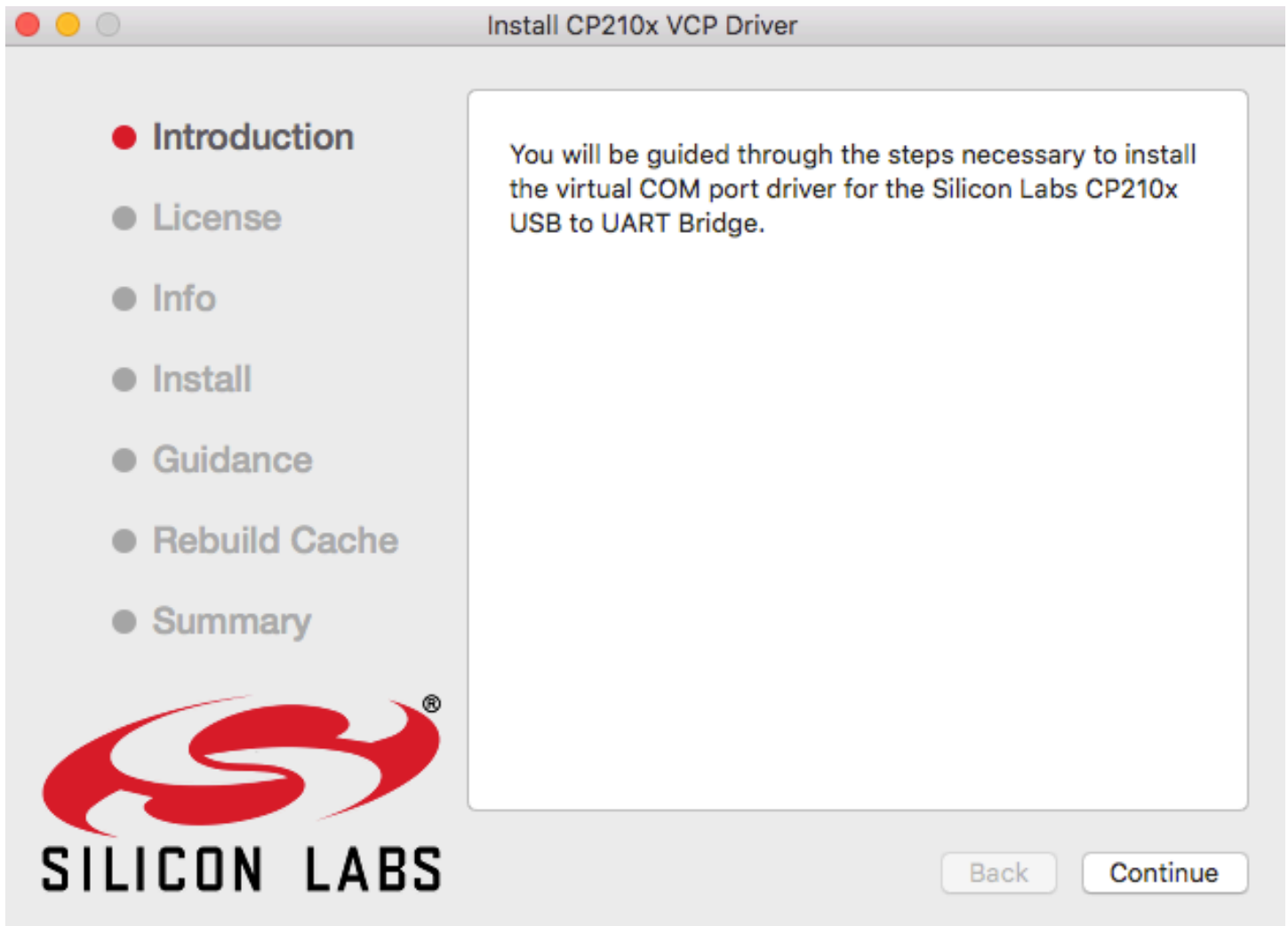
Переходим по ссылке и скачиваем .dmg архив:

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

Download for Macintosh OSX (v5.2.4)

Platform	Software	Release Notes
Mac OS X	Download VCP (832 KB)	Download Mac VCP Revision History

Запускаем .dmg архив.

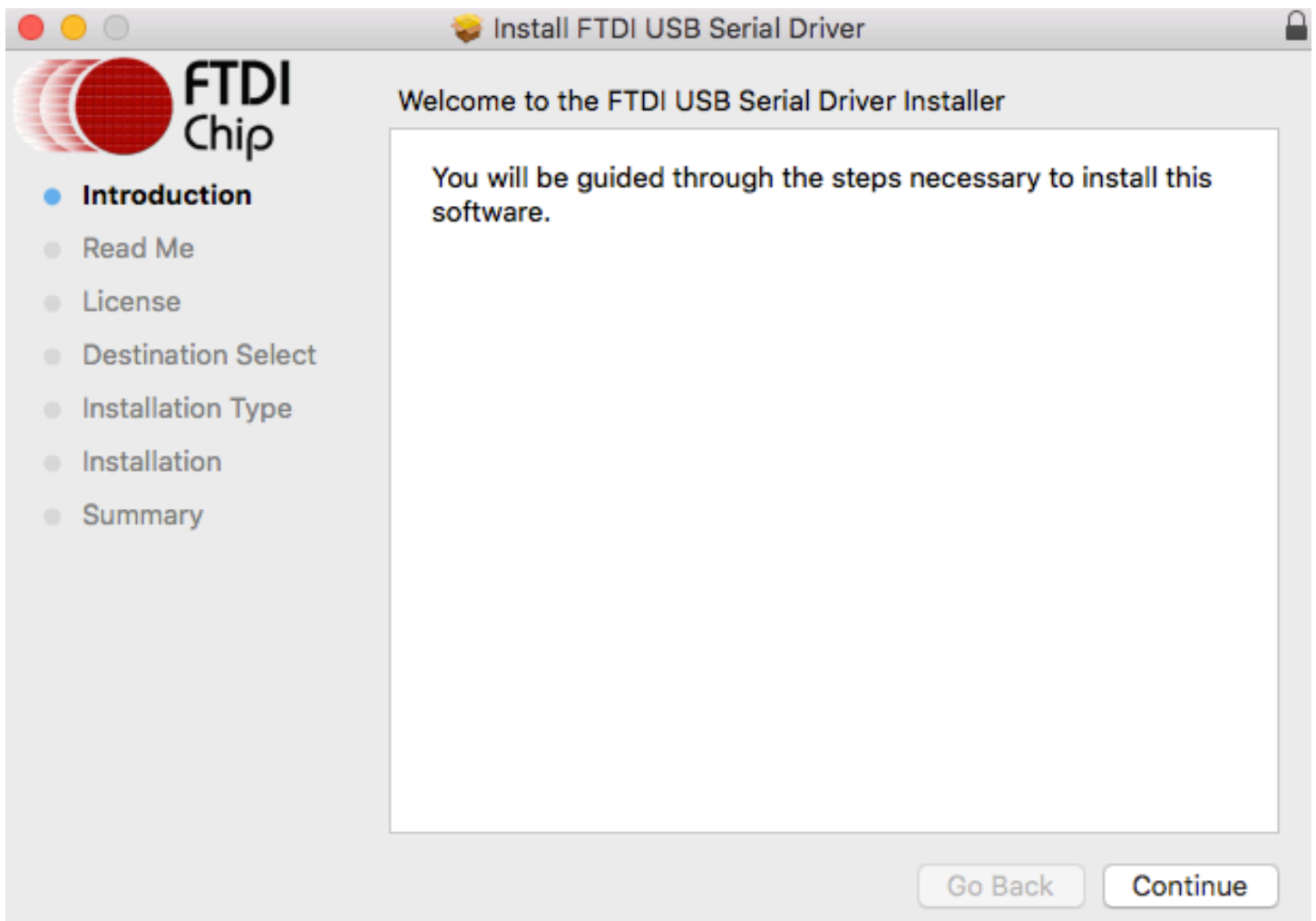


Установка FTDI USB Serial Driver.

Переходим по ссылке и скачиваем .dmg архив: <https://www.ftdichip.com/Drivers/VCP.htm>

Operating System	Release Date	Processor Architecture							Comments
		x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	
Windows*	2017-08-30	2.12.28	2.12.28	-	-	-	-	-	WHQL Certified. Includes VCP and D2XX. Available as a setup executable Please read the Release Notes and Installation Guides .
Linux	-	-	-	-	-	-	-	-	All FTDI devices now supported in Ubuntu 11.10, kernel 3.0.0-19 Refer to TN-101 if you need a custom VCP VID/PID in Linux VCP drivers are integrated into the kernel.
Mac OS X 10.3 to 10.8	2012-08-10	2.2.18	2.2.18	2.2.18	-	-	-	-	Refer to TN-105 if you need a custom VCP VID/PID in MAC OS
Mac OS X 10.9 and above	2019-12-24	-	2.4.2	-	-	-	-	-	This driver is signed by Apple

Запускаем .dmg архив.



2) Установка прошивки Pickle от Искандера на плату «TTGO ESP32 868/915 MHz LoRa OLED module» или же «868/915 MHz SX1276 ESP32 LoRa 0,96 OLED module»

Скачиваем проект Pickle от Искандера: <https://github.com/d3xr/pickle>

Копируем проект в папку *esp*.

MAC/users/«ваше имя пользователя»/esp/esp-idf/

Name	Date Modified	Size	Kind
add_path.sh	Yesterday at 22:11	723 bytes	Plain Text
CMakeLists.txt	Yesterday at 22:11	5 KB	Plain Text
components	Yesterday at 22:12	--	Folder
CONTRIBUTING.rst	Yesterday at 22:11	2 KB	Document
docs	Yesterday at 22:12	--	Folder
examples	Yesterday at 23:39	--	Folder
export.bat	Yesterday at 22:12	2 KB	Document
export.ps1	Yesterday at 22:12	2 KB	Document
export.sh	Yesterday at 22:12	3 KB	Plain Text
gnarl	Yesterday at 23:40	--	Folder
install.bat	Yesterday at 22:12	529 bytes	Document
install.ps1	Yesterday at 22:12	657 bytes	Document
install.sh	Yesterday at 22:12	357 bytes	Plain Text
Kconfig	Yesterday at 22:11	15 KB	TextEdit
LICENSE	Yesterday at 22:11	11 KB	TextEdit
make	Yesterday at 22:12	--	Folder
pickle	Today at 00:03	--	Folder
README_CN.md	Yesterday at 22:11	6 KB	Document
README.md	Yesterday at 22:11	6 KB	Document
requirements.txt	Yesterday at 22:12	971 bytes	Plain Text
sdkconfig.rename	Yesterday at 22:12	2 KB	Document
SUPPORT_POLICY_CN.md	Yesterday at 22:11	3 KB	Document
SUPPORT_POLICY.md	Yesterday at 22:11	3 KB	Document
tools	Yesterday at 22:12	--	Folder

Все готово для начала этапа прошивки вашей «TTGO ESP32 868/915 MHz LoRa OLED module» или же «868/915 MHz SX1276 ESP32 LoRa 0,96 OLED module». Подключите плату к компьютеру.

Возвращаемся в *Terminal* и вводим команду (не пропустите этот шаг, без этой команды не произойдет подгрузка библиотек, и машина просто не будет вас понимать):

```
└─$HOME/esp/esp-idf/export.sh
```



```
belg21s-mbp:~ belg21$ . $HOME/esp/esp-idf/export.sh
Adding ESP-IDF tools to PATH...
Checking if Python packages are up to date...
Python requirements from /Users/belg21/esp/esp-idf/requirements.txt are satisfied.
Added the following directories to PATH:
/Users/belg21/esp/esp-idf/components/esptool_py/esptool
/Users/belg21/esp/esp-idf/components/espcoredump
/Users/belg21/esp/esp-idf/components/partition_table/
/Users/belg21/.espressif/tools/xtensa-esp32-elf/esp-2019r2-8.2.0/xtensa-esp32-elf/bin
/Users/belg21/.espressif/tools/xtensa-esp32s2-elf/esp-2019r2-8.2.0/xtensa-esp32s2-elf/bin
/Users/belg21/.espressif/tools/esp32ulp-elf/2.28.51-esp-20191205/esp32ulp-elf-binutils/bin
/Users/belg21/.espressif/tools/esp32s2ulp-elf/2.28.51-esp-20191205/esp32s2ulp-elf-binutils/bin
/Users/belg21/.espressif/tools/openocd-esp32/v0.10.0-esp32-20191114/openocd-esp32/bin
/Users/belg21/.espressif/python_env/idf4.1_py2.7_env/bin
/Users/belg21/esp/esp-idf/tools
Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

idf.py build
```

Далее переходим в папку проекта *pickle*.

Вводим команду:

```
cd ~/esp/esp-idf/pickle
```

```
belg21s-mbp:~ belg21$ cd ~/esp/esp-idf/pickle
belg21s-mbp:pickle belg21$
```

Красным отмечен удачный переход в папку *pickle* через Terminal.

Подготовка сборки.

Вводим команду:

```
make
```

```
belg21s-mbp:pickle belg21$ make
Initial setup of pickle project is complete.
Now change to the "project" subdirectory and run "make" or "idf.py build".
belg21s-mbp:pickle belg21$
```

Переходим в подпапку *projects*.

Вводим команду:

```
cd ~/esp/esp-idf/pickle/project
```

```
belg21s-mbp:pickle belg21$ cd ~/esp/esp-idf/pickle/project
belg21s-mbp:project belg21$ █
```

Красным отмечен удачный переход в подпапку project через Terminal.

Запуск сборки.

Вводим команду:

```
make -j
```

```
belg21s-mbp:project belg21$ make -j
Toolchain path: /Users/belg21/.espressif/tools/xtensa-esp32-elf/esp-2019r2-8.2.0/xtensa-esp32-elf/bin/x
tensa-esp32-elf-gcc
Toolchain version: esp-2019r2
Compiler version: 8.2.0
Python requirements from /Users/belg21/esp/esp-idf/requirements.txt are satisfied.
GENCONFIG
App "pickle" version: v4.1-beta1-168-g69c8d9211
CC build/app_trace/app_trace.o
CC build/app_trace/app_trace_util.o
CC build/bootloader_support/src/bootloader_clock.o
CXX build/asio/asio/asio/src/asio.o
CC build/app_trace/heap_trace_tohost.o
CC build/app_trace/host_file_io.o
CC build/bootloader_support/src/bootloader_common.o
CXX build/cxx/cxx_exception_stubs.o
CC build/bootloader_support/src/bootloader_efuse_esp32.o
CC build/cbor/tinycbor/src/cborencoder.o
CXX build/cxx/cxx_guards.o
```

Будет очень много длинного кода. Терпеливо ждем завершения.

```
AR build/lwip/liblwip.a
esptool.py v2.9-dev
AR build/nvs_flash/libnvs_flash.a
AR build/nghttp/libnghttp.a
AR build/wpa_supplicant/libwpa_supplicant.a
AR build/expat/libexpat.a
AR build/mdns/libmdns.a
AR build/mbedtls/libmbedtls.a
AR build/asio/libasio.a
Generating esp32.project.ld
LD build/pickle.elf
esptool.py v2.9-dev
To flash all build output, run 'make flash' or:
python /Users/belg21/esp/esp-idf/components/esptool_py/esptool/esptool.py --chip esp32 --port /dev/ttyUSB0 -
-baud 115200 --before default_reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --fl
ash_size detect 0x1000 /Users/belg21/esp/esp-idf/pickle/project/build/bootloader/bootloader.bin 0x10000 /Use
rs/belg21/esp/esp-idf/pickle/project/build/pickle.bin 0x8000 /Users/belg21/esp/esp-idf/pickle/project/build/
partitions.bin
belg21s-mbp:project belg21$ █
```



```
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Compressed 3072 bytes to 104...
Wrote 3072 bytes (104 compressed) at 0x00008000 in 0.0 seconds (effective 1671.8 kbit/s)...
Hash of data verified.
Compressed 24640 bytes to 15453...
Wrote 24640 bytes (15453 compressed) at 0x00001000 in 0.4 seconds (effective 528.2 kbit/s)...
Hash of data verified.
Compressed 481552 bytes to 290279...
Wrote 481552 bytes (290279 compressed) at 0x00010000 in 6.9 seconds (effective 558.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done
```

Процесс завершен.

Проверка записи.

Вводим команду:

```
idf.py -p /dev/cu.SLAB_USBtoUART monitor
```

```
belg21s-mbp:project belg21$ idf.py -p /dev/cu.SLAB_USBtoUART monitor
Executing action: monitor
Running idf_monitor in directory /Users/belg21/esp/esp-idf/pickle/project
Executing "/Users/belg21/.espressif/python_env/idf4.1_py2.7_env/bin/python /Users/belg21/esp/esp-idf/tools/idf_monitor.py -p /dev/
.SLAB_USBtoUART -b 115200 --toolchain-prefix xtensa-esp32-elf- /Users/belg21/esp/esp-idf/pickle/project/build/pickle.elf -m '/Use
belg21/.espressif/python_env/idf4.1_py2.7_env/bin/python' '/Users/belg21/esp/esp-idf/tools/idf.py' '-p' '/dev/cu.SLAB_USBtoUART'"
--- idf_monitor on /dev/cu.SLAB_USBtoUART 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
??UUPJ?B?u?
?U?*PK???J???UUVX?U?U???DUu??(UZ??U??TUU?QTU?V?Z?UX??i?UP??-???U??TUU?QTU?V?Z?UX?????U?TU?U??TU*?u?J?UZ?UX?????U?
??T?T?P?P?ZU??-Q??Ue??UUPRUB??Z?-??UTU?UT???
?UT?UU?TU?U??TU*?u??-??UTU+UP???
I?UUU?TU?U??T?*??Z?-??UT??Z?
??UJU?TU?U??TU*?u?
??RUV?U5P*Z?Pj??X?UZ?VZ??UT???UUP?TB?u?
Q?????(*RUZ??UE??UU??U??Z?
?T??(??????UP?????.?U?MUZJP?UV????UE??UU??U*?u?
???-??????R??????Q?Q?Q?Q?U?TU?U??TE*??Z
?
?bd
???)??????Q
QQUQ?QQ??U?TU?U??T?*?u?
??URkZ????R?ZE,P??Q?QQQ?QQUQQ?U?TU?U??TU*?u?
?+??P?K????ZTT?UU?TU?U??T??ZJ?UZ?UX????V?U?U??XTU?????)UX?????
?U?TU?U??T?*U-?UV??VU?ZQ
??T?Q(Q?TQ??T?Q?TQU??Z?QRUQ?U??U?X??Ue??UUPJU?l??V??VU?Z?
??T?Q(QEUX??T?Q??T?U??
?,??UU?(UUU?TU?U??T?UB?X??V??VU?ZU
??T?Q(QU+e????Q?TQQQ????Q(??Q(?UU(u?UUU?TU?U??T??B?X??V??VU?ZU
??T?Q(QU?????Q?TQE*?U??,?U*Q?EUY??T?U?U?UUPJU?l??V??VU?Z?
??T?Q(QETQ??T?QTQETQ?U??\VT??+?U?UX??Ue??UUP??U-?U?V??
?ZU-??T?Q(Q??R??T?QTQ?U????Q?U?BUUU(u?UUU????UUP?JU??Z??TJPXUK??T?????)UZJP??Q??U??TUU?QU??u?
I??J??ZSU?TU+?V??UZ?VZ??UT??U?P??*??Z?(??*?JP??J??U??ZUZiT?PK??U?U+iT
?p?PUV
Q?T??EVU?B??V?ZUP??
U+iT
?n?PU?V
```

Остановить проверку CTRL+]

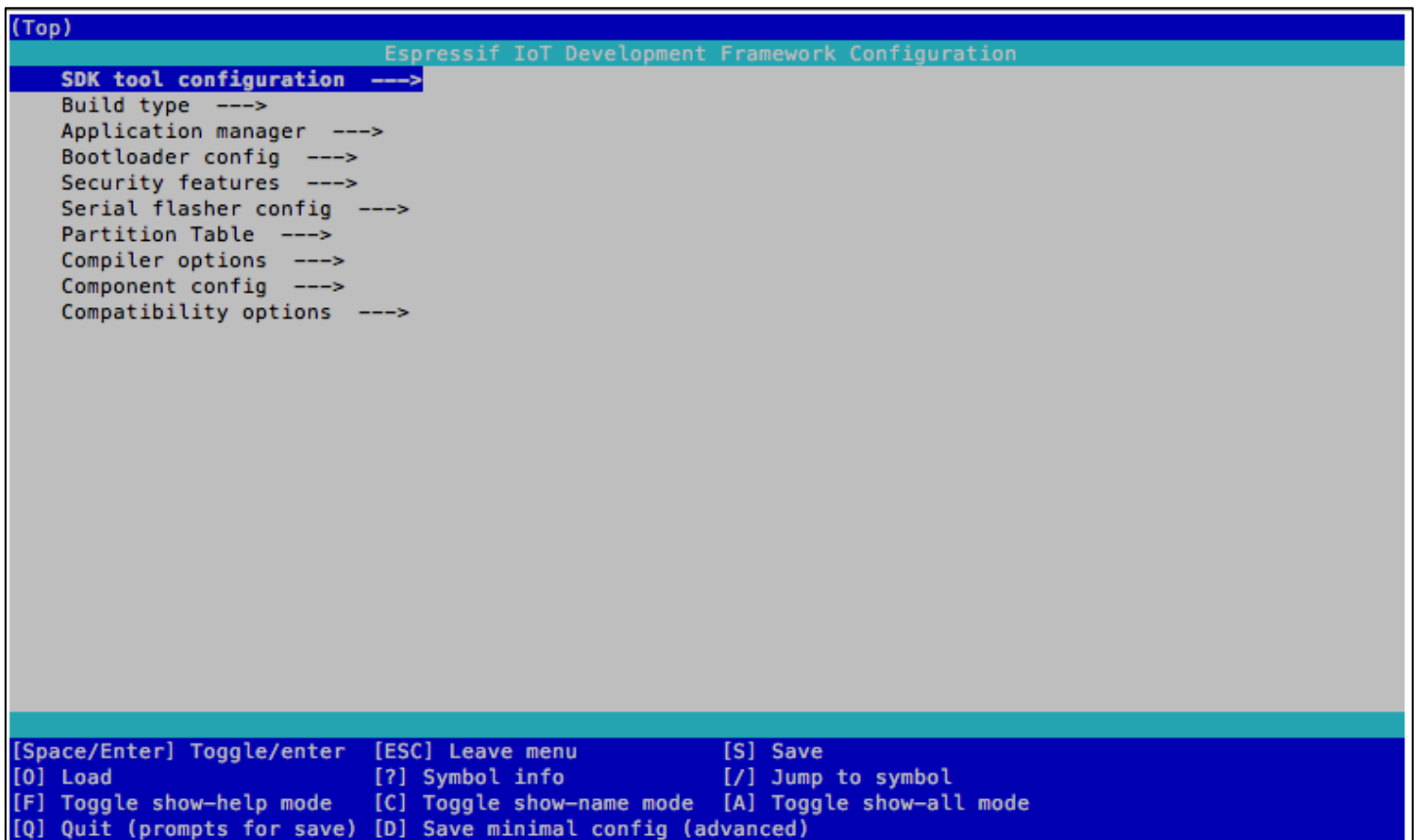
Если вы видите этот мусор у себя в *Terminal* значит ваша плата использует 26 MHz crystal, нам нужно поменять на 40 MHz. (Если у вас после проверки нет этого мусора, то сразу перемещайтесь в конец инструкции.)

Доступ в `menuconfig`.

Вводим команду:

```
idf.py menuconfig
```

Попадаем в это меню.



```
(Top)
Espressif IoT Development Framework Configuration
SDK tool configuration ---->
Build type ---->
Application manager ---->
Bootloader config ---->
Security features ---->
Serial flasher config ---->
Partition Table ---->
Compiler options ---->
Component config ---->
Compatibility options ---->

[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)
```

Далее переходим в *Component config*.

Далее в *Main XTAL frequency* и меняем значение на 40 MHz.

```
(Top) > Component config > ESP32-specific
Espressif IoT Development Framework Configuration
Minimum Supported ESP32 Revision (Rev 0) --->
CPU frequency (80 MHz) --->
[ ] Support for external, SPI-connected RAM
[ ] Use TRAX tracing feature
Number of universally administered (by IEEE) MAC address (Four) --->
[ ] Enable Ultra Low Power (ULP) Coprocessor
Panic handler behaviour (Print registers and reboot) --->
[*] Make exception and panic handlers JTAG/OCD aware
[*] Hardware brownout detect & reset
Brownout voltage level (2.43V +/- 0.05) --->
[*] Reduce PHY TX power when brownout reset
Timers used for gettimeofday function (RTC and high-resolution timer) --->
RTC clock source (Internal 150kHz RC oscillator) --->
(1024) Number of cycles for RTC_SLOW_CLK calibration
(2000) Extra delay in deep sleep wake stub (in us)
Main XTAL frequency (40 MHz) --->
[ ] Permanently disable BASIC ROM Console
[ ] App compatible with bootloaders before IDF v2.1
[ ] Use fixed static RAM size
(5) Disable the interrupt level for the DPORT workarounds

[Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
[0] 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)
```

После смены на 40MHz нажимаем клавишу «S» далее клавишу «ESP» и тем самым выходим обратно в *Terminal*.

Вводим вновь команду для прошивки платы:

```
idf.py -p /dev/cu.SLAB_USBtoUART flash
```

```
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Compressed 3072 bytes to 104...
Wrote 3072 bytes (104 compressed) at 0x00008000 in 0.0 seconds (effective 1671.8 kbit/s)...
Hash of data verified.
Compressed 24640 bytes to 15453...
Wrote 24640 bytes (15453 compressed) at 0x00001000 in 0.4 seconds (effective 528.2 kbit/s)...
Hash of data verified.
Compressed 481552 bytes to 290279...
Wrote 481552 bytes (290279 compressed) at 0x00010000 in 6.9 seconds (effective 558.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done
```

Вновь видим, что процесс завершен.

Опять запускаем проверку записи.

Вводим команду:

```
idf.py -p /dev/cu.SLAB_USBtoUART monitor
```

```
belg21s-mbp:project belg21$ idf.py -p /dev/cu.SLAB_USBtoUART monitor
Executing action: monitor
Running idf_monitor in directory /Users/belg21/esp/esp-idf/pickle/project
Executing "/Users/belg21/.espressif/python_env/idf4.1_py2.7_env/bin/python /Users/belg21/esp/esp-idf/tools/idf_monitor.py -p
.SLAB_USBtoUART -b 115200 --toolchain-prefix xtensa-esp32-elf- /Users/belg21/esp/esp-idf/pickle/project/build/pickle.elf -m '
belg21/.espressif/python_env/idf4.1_py2.7_env/bin/python' '/Users/belg21/esp/esp-idf/tools/idf.py' '-p' '/dev/cu.SLAB_USBtoUA
--- idf_monitor on /dev/cu.SLAB_USBtoUART 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
I (12) boot: ESP-IDF v4.1-beta1-168-g69c8d9211 2nd stage bootloader
I (12) boot: compile time 15:24:24
I (12) boot: chip revision: 1
I (15) boot_comm: chip revision: 1, min. bootloader chip revision: 0
I (32) boot.esp32: SPI Speed      : 40MHz
I (32) boot.esp32: SPI Mode      : DIO
I (32) boot.esp32: SPI Flash Size : 4MB
I (36) boot: Enabling RNG early entropy source...
I (42) boot: Partition Table:
I (45) boot: ## Label                Usage              Type ST Offset   Length
I (52) boot:  0 nvs                   WiFi data          01 02 00009000 00006000
I (60) boot:  1 phy_init              RF data           01 01 0000f000 00001000
I (67) boot:  2 factory               factory app       00 00 00010000 00200000
I (75) boot: End of partition table
I (79) boot_comm: chip revision: 1, min. application chip revision: 0
I (86) esp_image: segment 0: paddr=0x00010020 vaddr=0x3f400020 size=0x10280 ( 66176) map
I (120) esp_image: segment 1: paddr=0x000202a8 vaddr=0x3ffbdb60 size=0x03528 ( 13608) load
I (126) esp_image: segment 2: paddr=0x000237d8 vaddr=0x40080000 size=0x00404 ( 1028) load
0x40080000: _WindowOverflow4 at /Users/belg21/esp/esp-idf/components/freertos/xtensa_vectors.S:1778
I (127) esp_image: segment 3: paddr=0x00023be4 vaddr=0x40080404 size=0x0c434 ( 50228) load
I (158) esp_image: segment 4: paddr=0x00030020 vaddr=0x400d0020 size=0x4d9b4 (317876) map
0x400d0020: _stext at ??:?
I (279) esp_image: segment 5: paddr=0x0007d9dc vaddr=0x4008c838 size=0x07f10 ( 32528) load
0x4008c838: set_chan_dig_gain at /home/aiqin/git_tree/chip7.1_phy/chip_7.1/board_code/app_test/pp/phy/phy_chip_v7_cal.c:1882
I (304) boot: Loaded app from partition at offset 0x10000
```

Результатом проверки становится красивый программный код без какого-либо мусора.

Я вас поздравляю. Плата готова к дальнейшей настройке в FreeAps или Loop.

Огромное спасибо **ИСКАНДЕРУ**.

Так же более подробные инструкции установки ПО и подготовки вашего MAC есть по ссылке: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html#get-started-get-esp-idf>