

Other IDEs

ESP32 Open IoT and IIoT Gateways (P01 & P02)

The device can be programmed also in other development environments. Programming gateways is supported on every popular operating systems like Windows, Linux or MAC OS. This document was prepared in reference to Windows.

ESP-IDF framework

The main tool is ESP-IDF framework provided by Espressif. To get more information about installation, visit manufacturer's

website: [Get Started - ESP32 - — ESP-IDF Programming Guide latest documentation.](#)

Manual Installation of ESP-IDF: [Standard Setup of Toolchain for Windows - ESP32 - — ESP-IDF Programming Guide latest documentation.](#)

During the ESP-IDF installation you might be asked for install Eclipse additionally.

After successful process it is necessary to add new environment variables.

Variables names:

- IDF_PATH - paste the path of the directory with ESP-IDF framework
- IDF_TOOLS_PATH - paste the path of the directory with ESP-IDF tools

Name	Date modified	Type	Size
.git	09/09/2022 11:04	File folder	
.github	08/09/2022 14:41	File folder	
.gitlab	08/09/2022 14:41	File folder	
components	08/09/2022 14:41	File folder	
docs	08/09/2022 14:41	File folder	
examples	08/09/2022 14:41	File folder	
make	08/09/2022 14:41	File folder	
tools	08/09/2022 14:42	File folder	
.editorconfig	08/09/2022 14:41	Editor Config Sour...	1 KB
.flake8	08/09/2022 14:41	FLAKE8 File	9 KB
.gitignore	08/09/2022 14:41	Text Document	2 KB
.gitlab-ci	08/09/2022 14:41	Yaml Source File	6 KB
.gitmodules	08/09/2022 14:41	Text Document	4 KB
.mypy	08/09/2022 14:41	Configuration sett...	1 KB
.pre-commit-config	08/09/2022 14:41	Yaml Source File	5 KB
.pylintrc	08/09/2022 14:41	PYLINTRC File	19 KB
.readthedocs	08/09/2022 14:41	Yaml Source File	1 KB

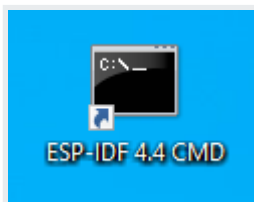
Edytowanie zmiennej systemowej

Nazwa zmiennej:

Wartość zmiennej:

Przeglądaj katalog... Przeglądaj plik... OK Anuluj

Finally you can run ESP-IDF CMD and start to manage your project. There should be an icon on the desktop or easy access to ESP-IDF.



Create new project

```
idf.py create-project -p <name>
```

Build project

```
idf.py -p <port> build
```

Flash project

```
idf.py -p <port> flash
```

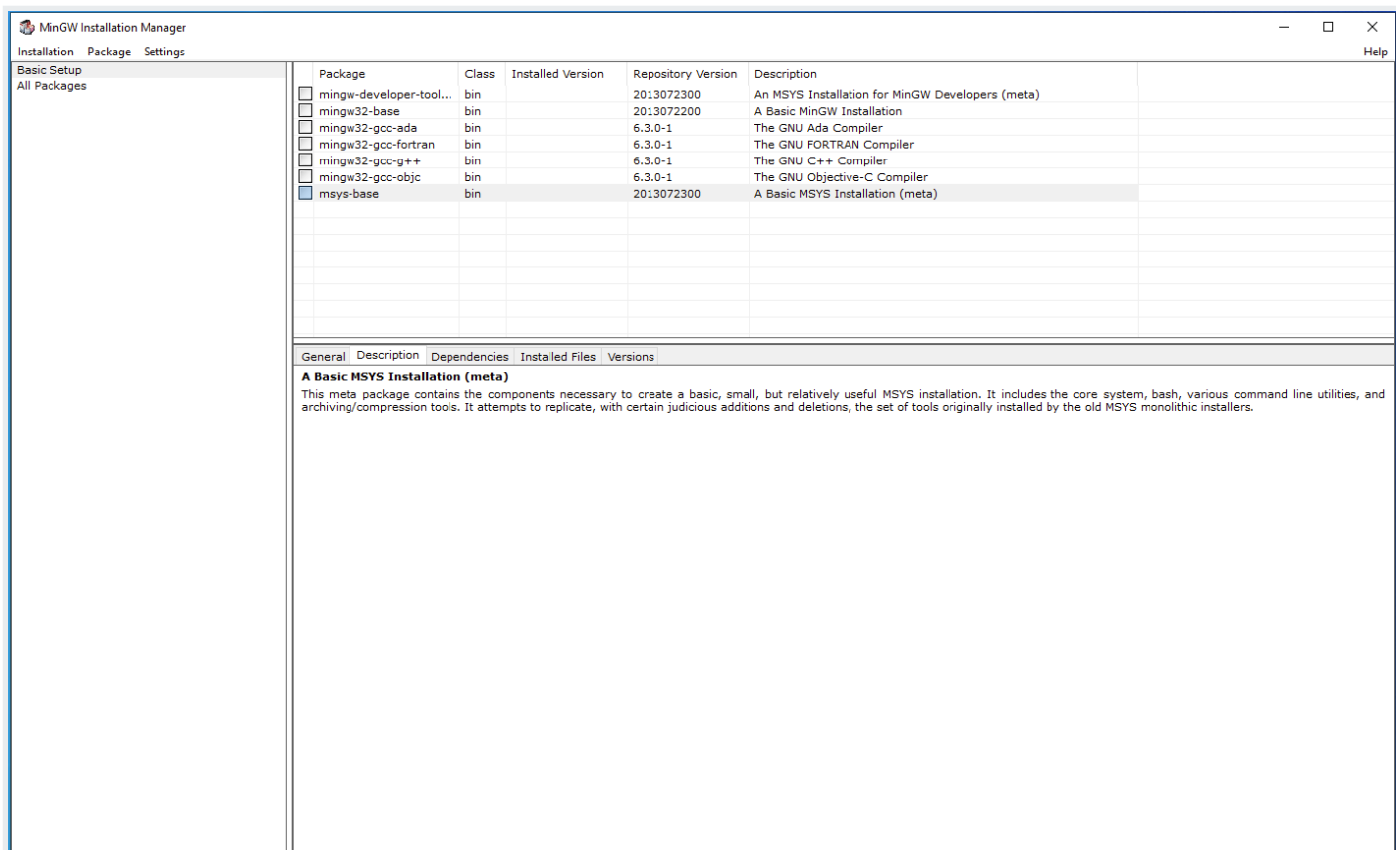
```
idf.py -p <port> flash monitor
```

Erase flash

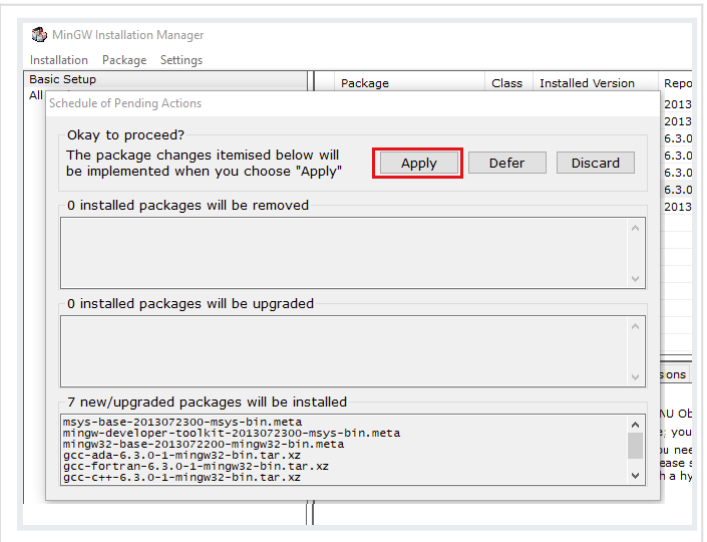
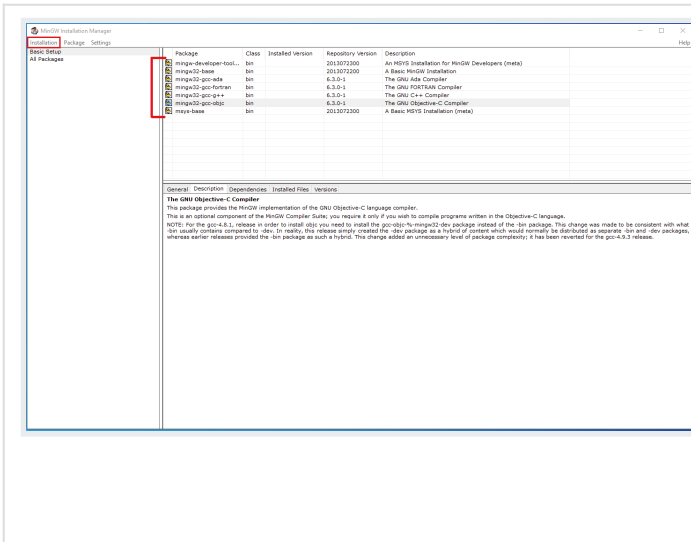
```
esptool.py --port <port> erase_flash
```

MinGW

Download MinGW with GUI from [MinGW - Minimalist GNU for Windows](#) and install on your PC. After a successful installation run MinGW Installation Manager (GUI).

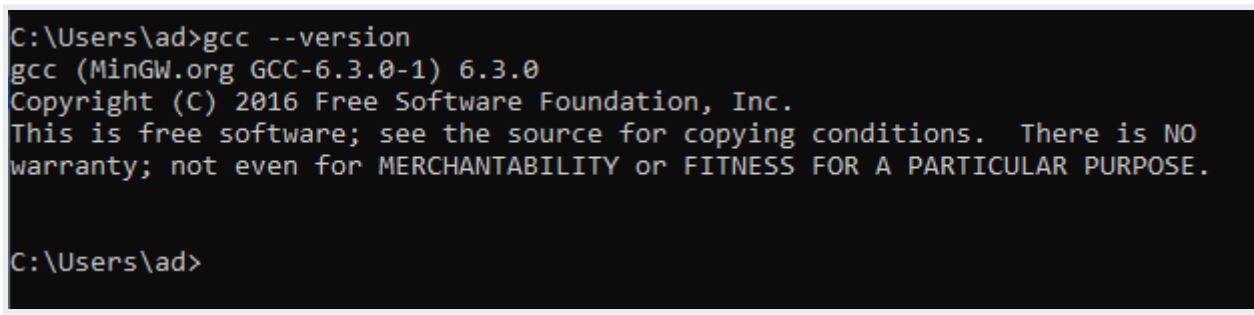


Now we need to install Basic Setup. Right click on every square fields in "Package" tab, then "Mark for Installation". Next, in "Installation" tab click on "Apply Changes" and then "Apply".



To check if the installation is successful, open Command line and type:

```
gcc --version
```



IDE

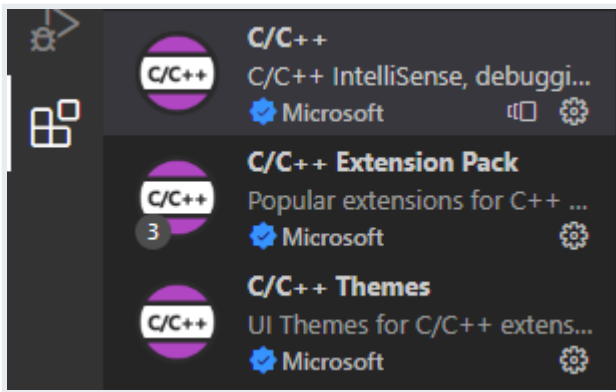
You can edit code in your preferred IDE as ESP-IDF is handling the final build and flash.

Recommended IDEs:

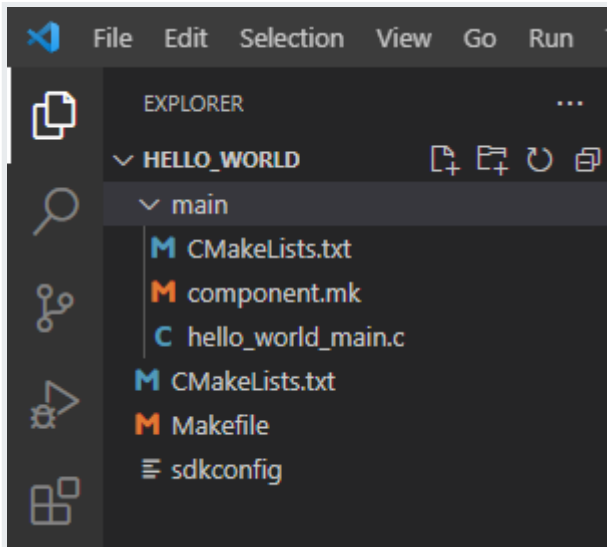
- [CodeBlocks](#)
- [Visual Studio Code](#)
- [Eclipse](#)
- [Thonny IDE](#)

Visual Studio Code

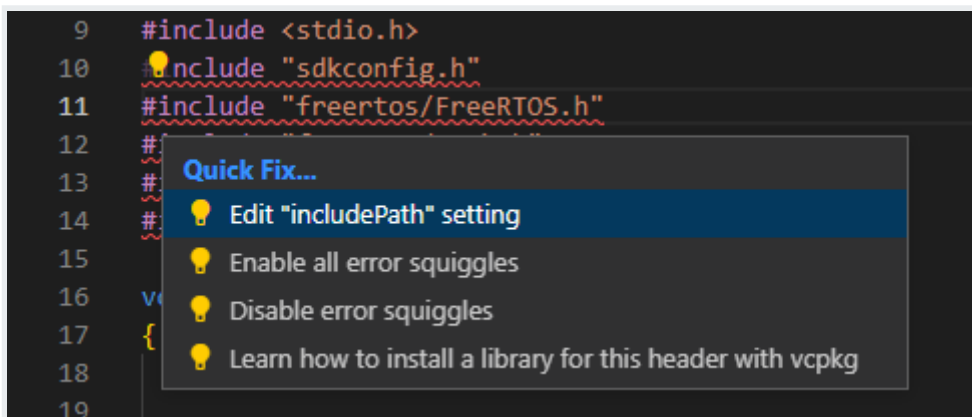
- Download the package with example projects for IoT Gateway.
- Select one of the demos and copy it to a new directory.
- Open Visual Studio Code and click on the extension tab.
- Install and configure C/C++ extensions.



- Open directory in VS Code.



- If errors occur, edit "includePath" settings.



- Add the line `"${env:IDF_PATH}/**"`.

Include path

An include path is a folder that contains header files (such as `#include "myHeaderFile.h"`) that are included in a source file. Specify a list of paths for the IntelliSense engine to use while searching for included header files. Searching on these paths is not recursive. Specify `**` to indicate recursive search. For example, `${workspaceFolder}/**` will search through all subdirectories while `${workspaceFolder}` will not. If on Windows with Visual Studio installed, or if a compiler is specified in the `compilerPath` setting, it is not necessary to list the system include paths in this list.

One include path per line.

```
${workspaceFolder}/**  
${env:IDF_PATH}/**
```

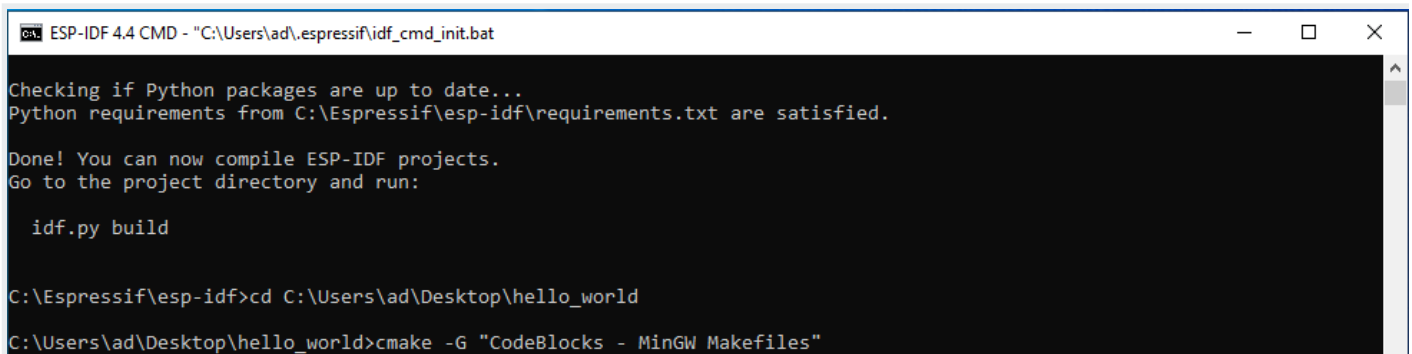
- Now code is ready to be modified.
- To build or flash project, use ESP-IDF CMD.

Visual Studio Code provides an extension “Espressif IDF” which has some issues at this moment. However it is not essential for editing code.

CodeBlocks

- Download the package with example projects for IoT Gateway.
- Select one of the demos and copy it to a new directory.
- Run ESP-IDF CMD, set the path to your project and then generate project for CodeBlocks:

```
cmake -G "CodeBlocks - MinGW Makefiles"
```



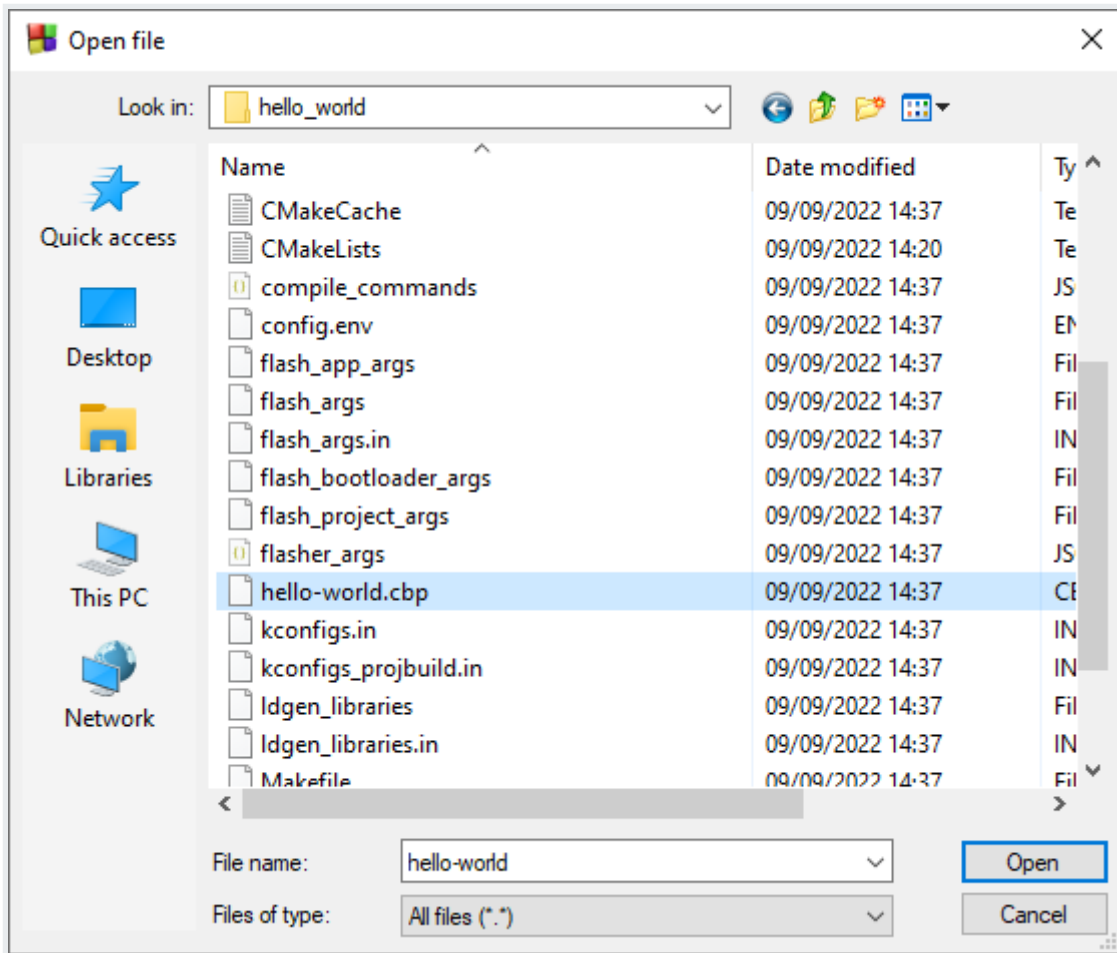
```
ESP-IDF 4.4 CMD - "C:\Users\ad\esp-idf_cmd_init.bat"
Checking if Python packages are up to date...
Python requirements from C:\Espressif\esp-idf\requirements.txt are satisfied.
Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

idf.py build

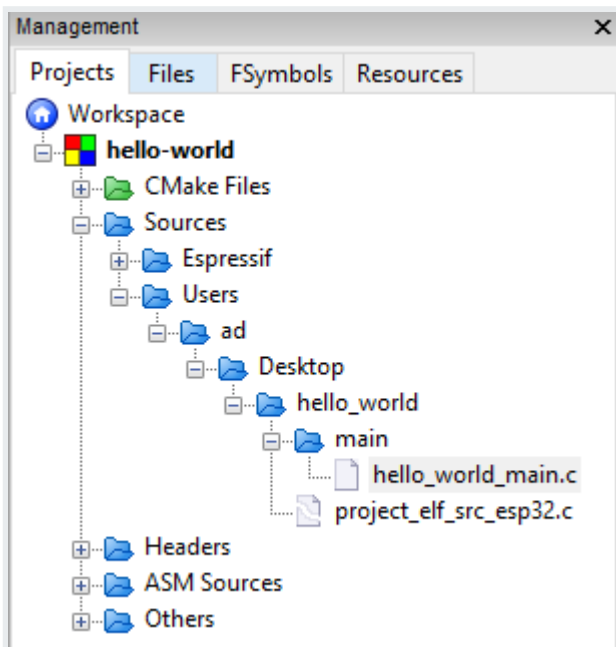
C:\Espressif\esp-idf>cd C:\Users\ad\Desktop\hello_world
C:\Users\ad\Desktop\hello_world>cmake -G "CodeBlocks - MinGW Makefiles"
```

```
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/ad/Desktop/hello_world
C:\Users\ad\Desktop\hello_world>
```

- Run CodeBlocks. Click on "File → Open...", find .cbp type file and click on “Open”.



- After this, there should be a project tree available and code is ready to be modified.



- To build or flash project, use ESP-IDF CMD.

Thonny IDE

- Download and install Python interpreter.

- Download the package with example projects for IoT Gateway.
- Select one of the demos and copy it to a new directory.
- Run ESP-IDF CMD, erase flash using:

```
esptool.py --port <COMX> erase_flash
```

```
C:\Users\ad\Desktop\hello_world>esptool.py --port COM15 erase_flash
esptool.py v3.3.2-dev
Serial port COM15
Connecting.....
Detecting chip type... Unsupported detection protocol, switching and trying again...
Connecting.....
Detecting chip type... ESP32
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: b4:e6:2d:fb:b8:1d
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 4.1s
Hard resetting via RTS pin...
```

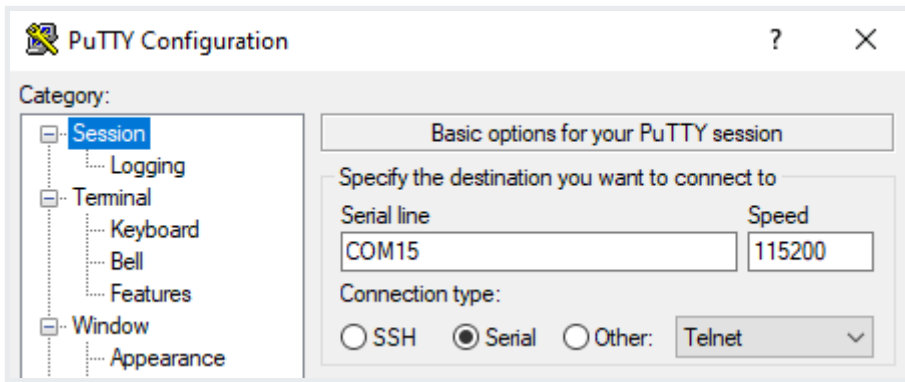
- Download [firmware](#) which allows to run microPython.
- Flash .bin file using:

```
esptool.py --port <COMX> write_flash -z 0x1000 <name_of_the_bin_file>
```

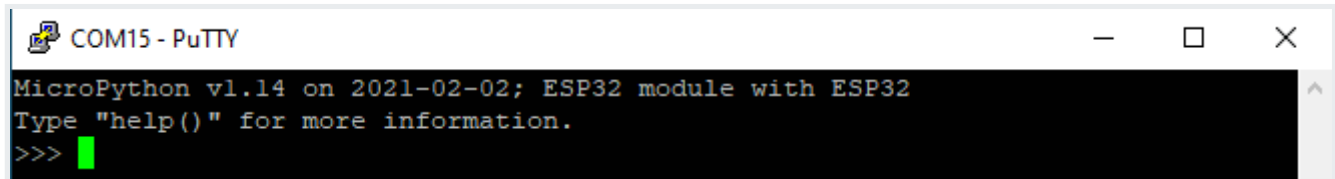
```
C:\Espressif\esp-idf>cd c:\users\ad\desktop
c:\Users\ad\Desktop>cd micropython
c:\Users\ad\Desktop\micropython>esptool.py --port COM15 write_flash -z 0x1000 esp32-idf4-20210202-v1.14.bin
esptool.py v3.3.2-dev
Serial port COM15
Connecting.....
Detecting chip type... Unsupported detection protocol, switching and trying again...
Connecting.....
Detecting chip type... ESP32
Chip is ESP32-D0WD (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: b4:e6:2d:fb:b8:1d
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00001000 to 0x0016bfff...
Compressed 1484624 bytes to 951640...
Wrote 1484624 bytes (951640 compressed) at 0x00001000 in 84.6 seconds (effective 140.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
c:\Users\ad\Desktop\micropython>
```

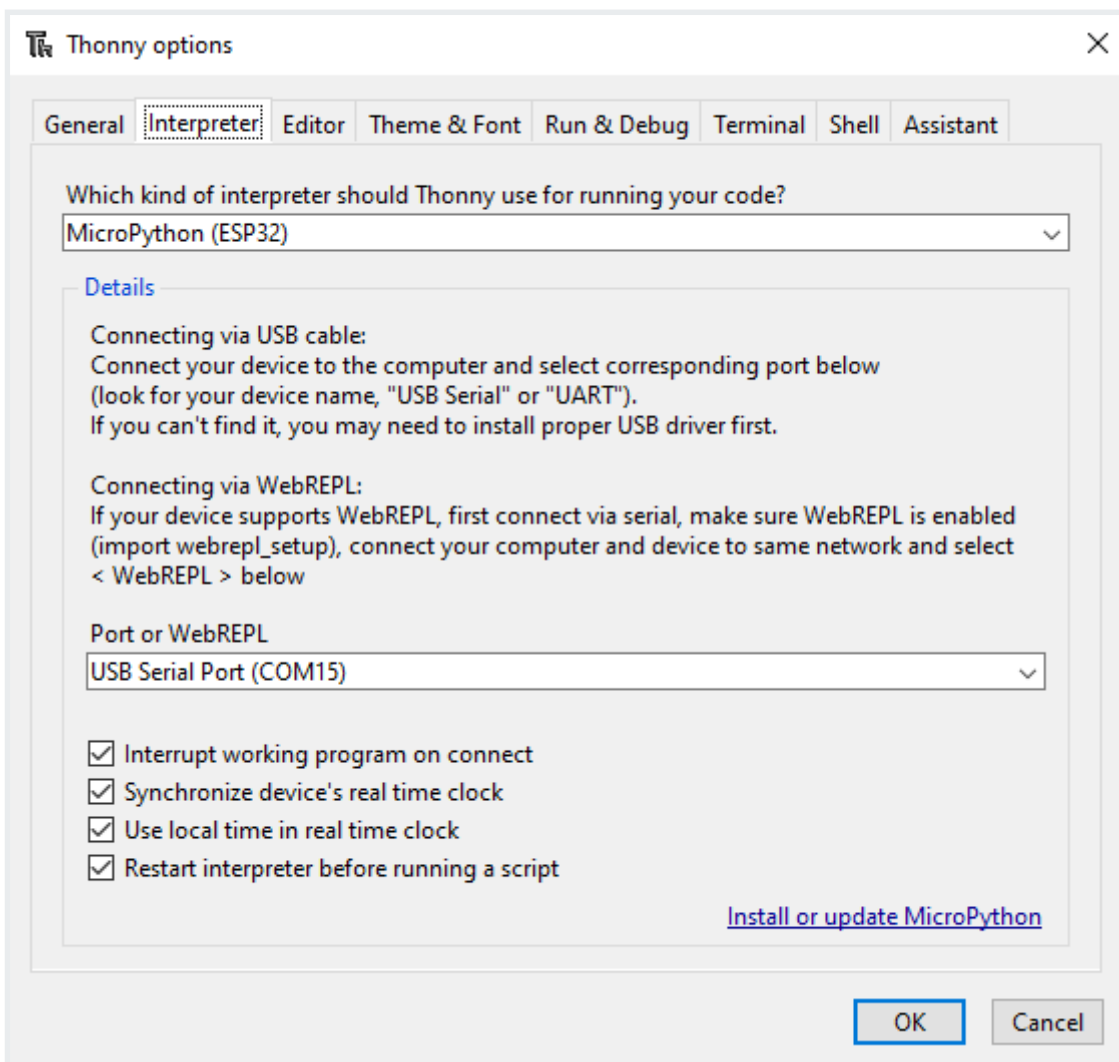
- Open serial port monitor like “Putty”, set COM port, baudrate, connection type to serial and click on “Open”.



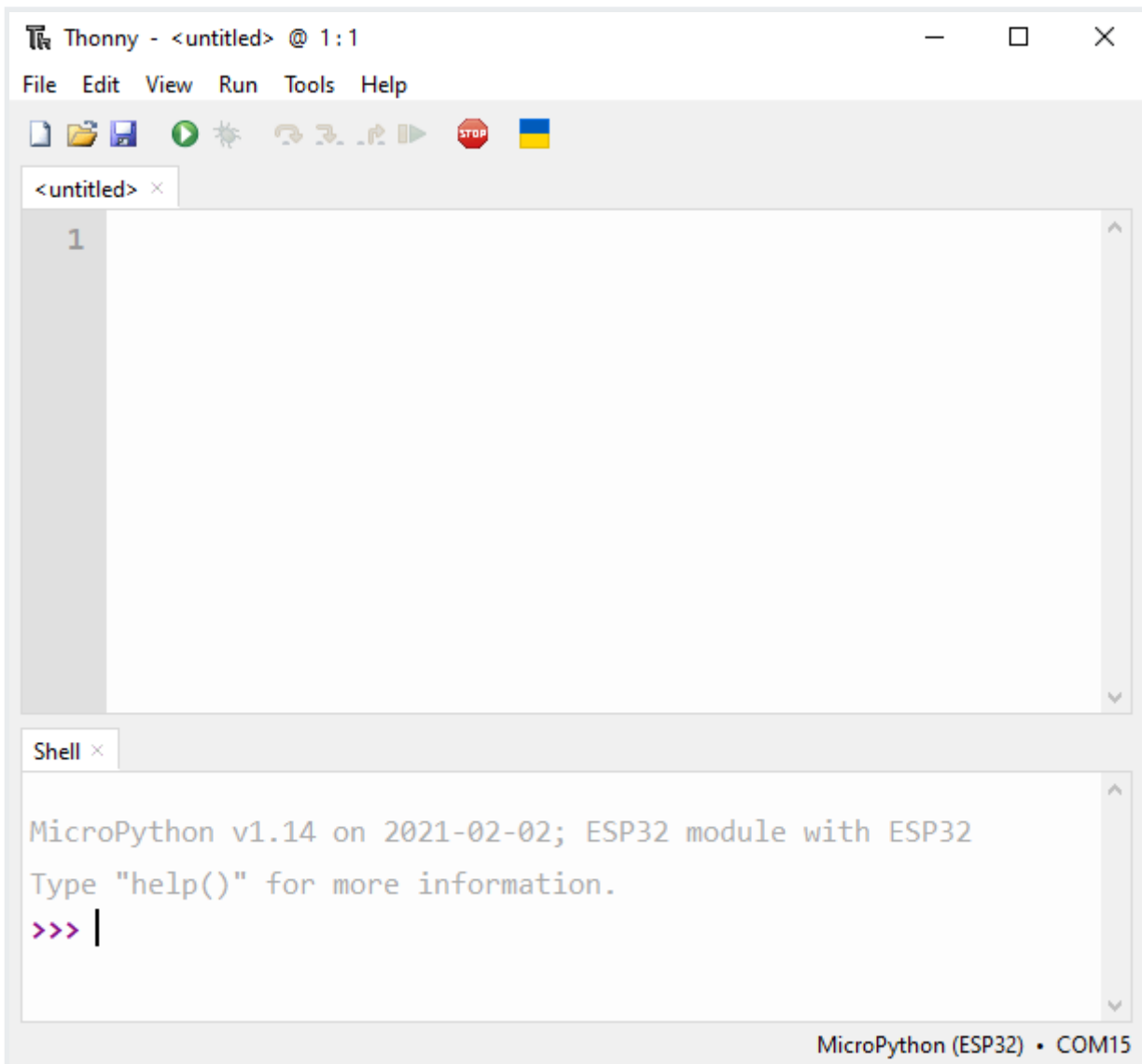
- Now you can run Python console on ESP32.



- Download and install [Thonny IDE](#).
- Click on "Tools → Options → Interpreter". In this tab you can choose an interpreter and COM port.



- After all the process Thonny IDE is ready to work.



- To flash the demo, open file in Thonny IDE. Next click on green button to run the current script.

Uploading firmware

To upload firmware, you need to use external programmer with Tag-connect.



Revision #13

Created 3 April 2024 13:20:21

Updated 20 June 2024 13:28:31