eYFi-Mega

Software Manual v0.1

1 December 2019

Installation of VS Code in Linux

There are several IDEs available to program AVR Microcontrollers and ESP32 but for eYFi-Mega, we have developed a special extension for VS Code which makes it convenient to program and use features of the development board. In this section, we will discuss how to install the VS Code and in the next section, we will see how to install the extension.

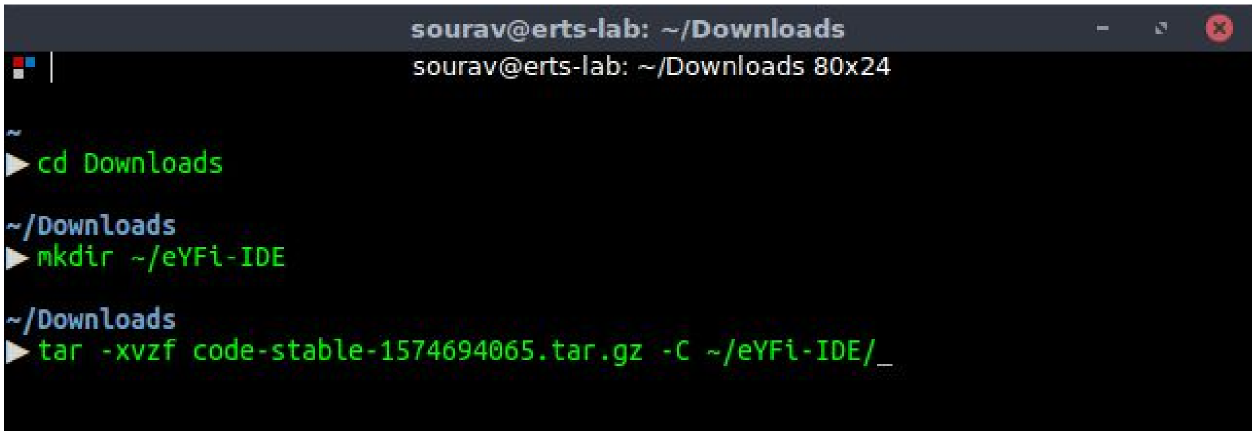
NOTE: For all the software required to work properly make sure that Python 2 is set as your default Python Interpreter. We recommend you to go through Appendix C before you proceed with the installation.

Download and Install VS Code

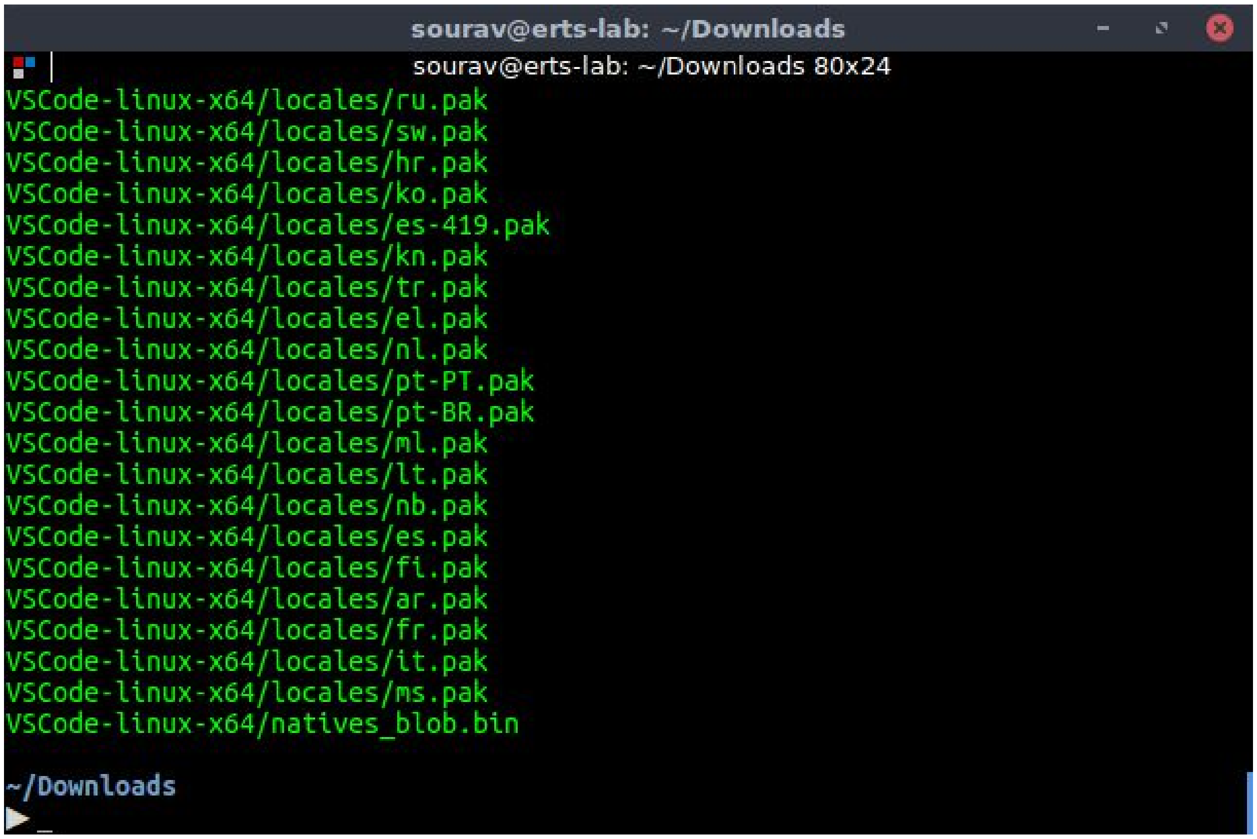
1. Download the portable version of the IDE which can be found at this URL
    a. 64 Bit Linux: https://code.visualstudio.com/docs/?dv=linux64
    b. 32 Bit Linux: https://code.visualstudio.com/docs/?dv=linux32

2. Extract the **.tar.gz** file at any desired location to install the VS Code in your machine. We recommend to extract it in a folder in your home directory. To do so execute the following commands,

```
$ cd ~/Downloads
$ mkdir ~/eYFi-IDE
$ tar -xvzf code-stable-1574694065.tar.gz -C ~/eYFi-IDE/
```



```
sourav@erts-lab: ~/Downloads
sourav@erts-lab: ~/Downloads 80x24

~
▶ cd Downloads

~/Downloads
▶ mkdir ~/eYFi-IDE

~/Downloads
▶ tar -xvzf code-stable-1574694065.tar.gz -C ~/eYFi-IDE/_
```



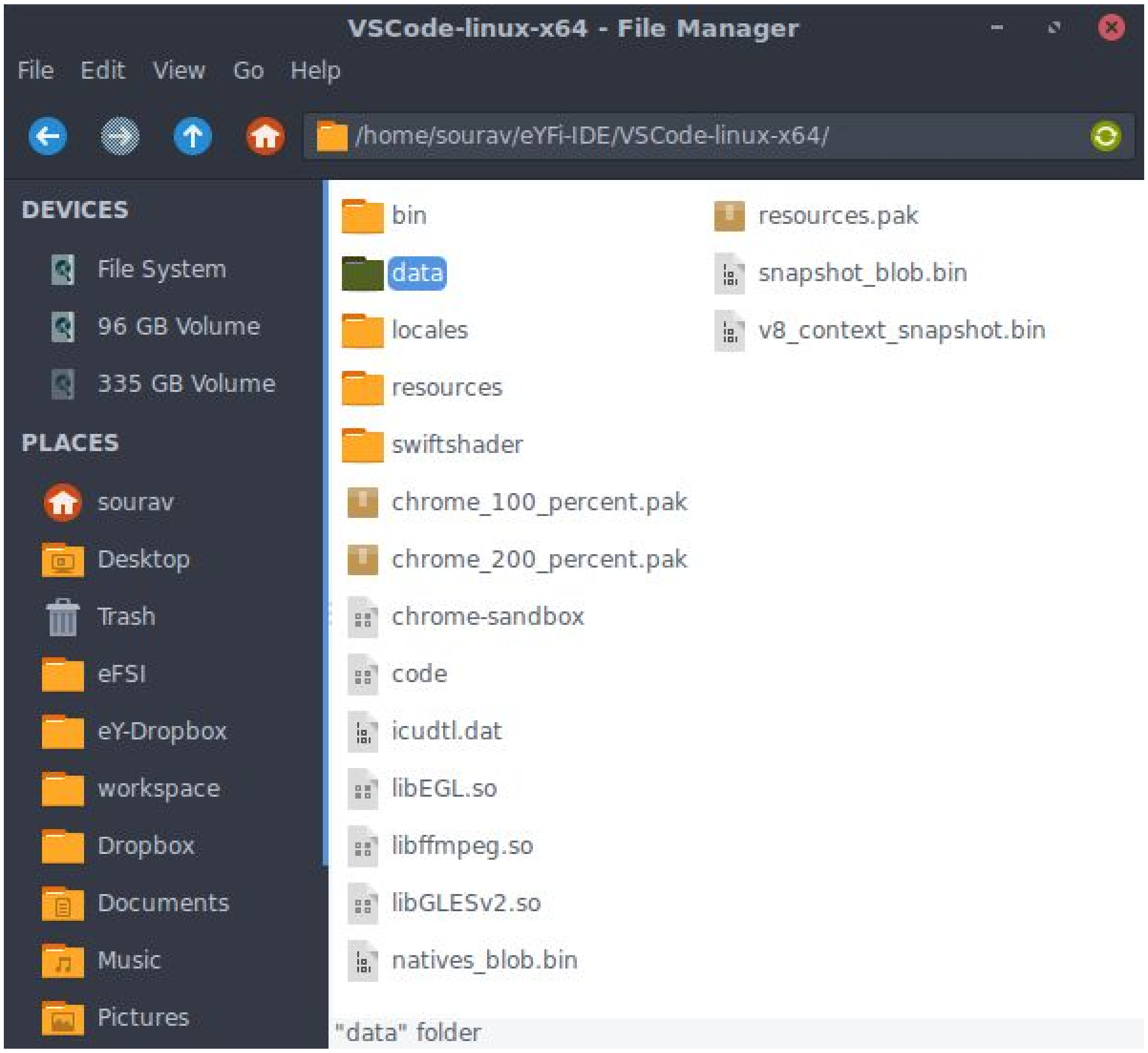
```
sourav@erts-lab: ~/Downloads
sourav@erts-lab: ~/Downloads 80x24
VSCode-linux-x64/locales/ru.pak
VSCode-linux-x64/locales/sw.pak
VSCode-linux-x64/locales/hr.pak
VSCode-linux-x64/locales/ko.pak
VSCode-linux-x64/locales/es-419.pak
VSCode-linux-x64/locales/kn.pak
VSCode-linux-x64/locales/tr.pak
VSCode-linux-x64/locales/el.pak
VSCode-linux-x64/locales/nl.pak
VSCode-linux-x64/locales/pt-PT.pak
VSCode-linux-x64/locales/pt-BR.pak
VSCode-linux-x64/locales/ml.pak
VSCode-linux-x64/locales/lt.pak
VSCode-linux-x64/locales/nb.pak
VSCode-linux-x64/locales/es.pak
VSCode-linux-x64/locales/fi.pak
VSCode-linux-x64/locales/ar.pak
VSCode-linux-x64/locales/fr.pak
VSCode-linux-x64/locales/it.pak
VSCode-linux-x64/locales/ms.pak
VSCode-linux-x64/natives_blob.bin

~/Downloads
▶ _
```

Enable Portable Mode

Visual Studio Code needs to be run in Portable Mode. This mode enables all data created and maintained by the VS Code to live near itself, so it can be moved around across environments.

After unzipping the VS Code download, simply create a “**data**” folder within Code's folder:

```
|- ~/eYFi-IDE/VSCode-linux-x64
|   |- code (the executable file)
|   |- data
|   |- ...
```



From then on, that folder will be used to contain all Code data, including session state, preferences, extensions, etc.

The **data** folder can be moved to other VS Code installations. This is useful for updating your portable Code version: simply move the **data** folder to a newer extracted version of the VS Code.

Start VS Code Portable

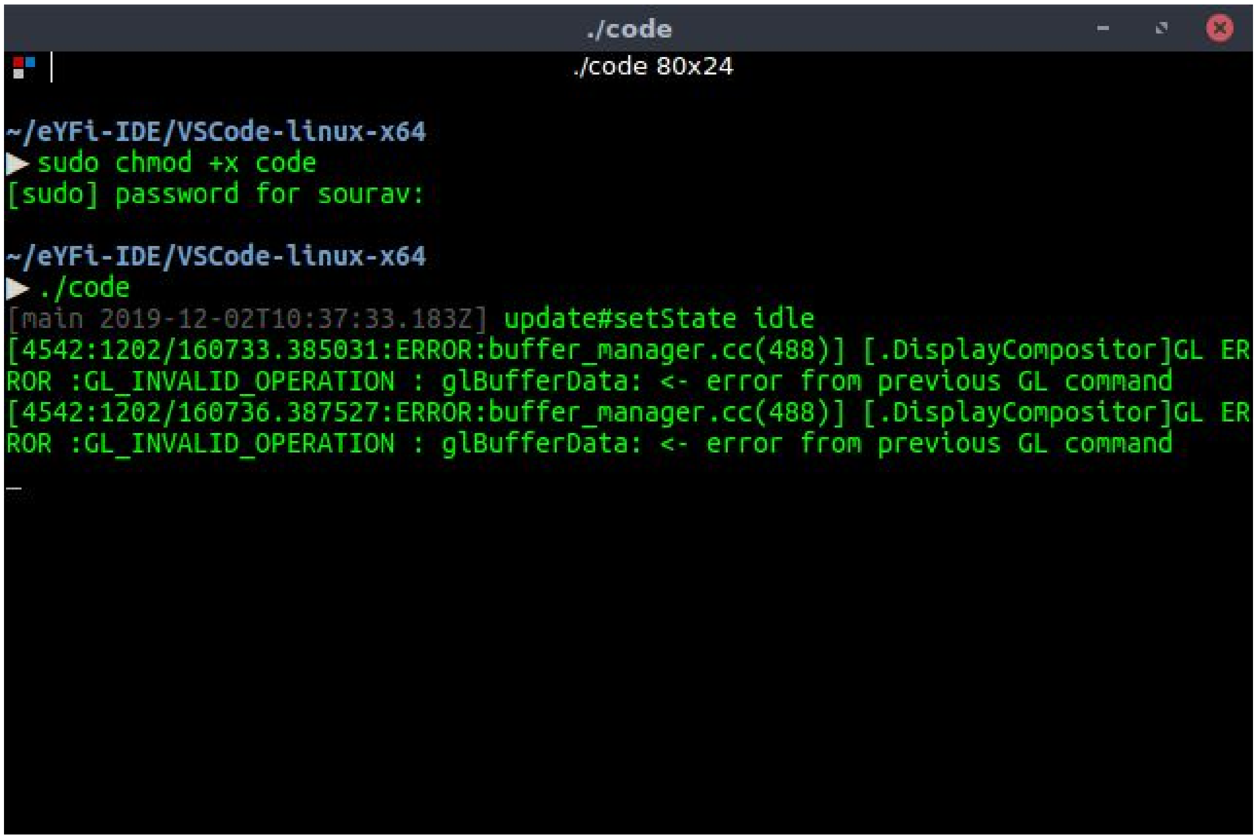
Open terminal in the root folder of the VS Code portable.

1. Make the “**code**” file executable. To do this use this command:

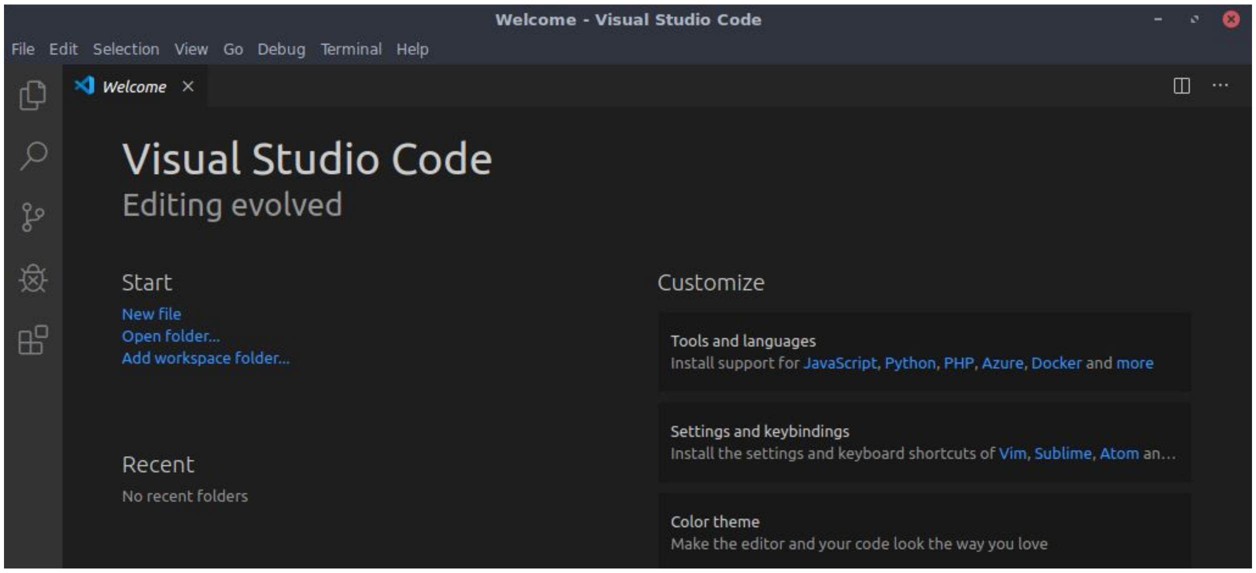
```
$ sudo chmod +x code
```

2. Now to start VS code use this command:

```
$ ./code
```



```
./code 80x24

~/eYFi-IDE/VSCode-linux-x64
▶ sudo chmod +x code
[sudo] password for sourav:

~/eYFi-IDE/VSCode-linux-x64
▶ ./code
[main 2019-12-02T10:37:33.183Z] update#setState idle
[4542:1202/160733.385031:ERROR:buffer_manager.cc(488)] [.DisplayCompositor]GL ER
ROR :GL_INVALID_OPERATION : glBufferData: <- error from previous GL command
[4542:1202/160736.387527:ERROR:buffer_manager.cc(488)] [.DisplayCompositor]GL ER
ROR :GL_INVALID_OPERATION : glBufferData: <- error from previous GL command
```


Welcome - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Welcome

Visual Studio Code
Editing evolved

Start
New file
Open folder...
Add workspace folder...

Recent
No recent folders

Customize

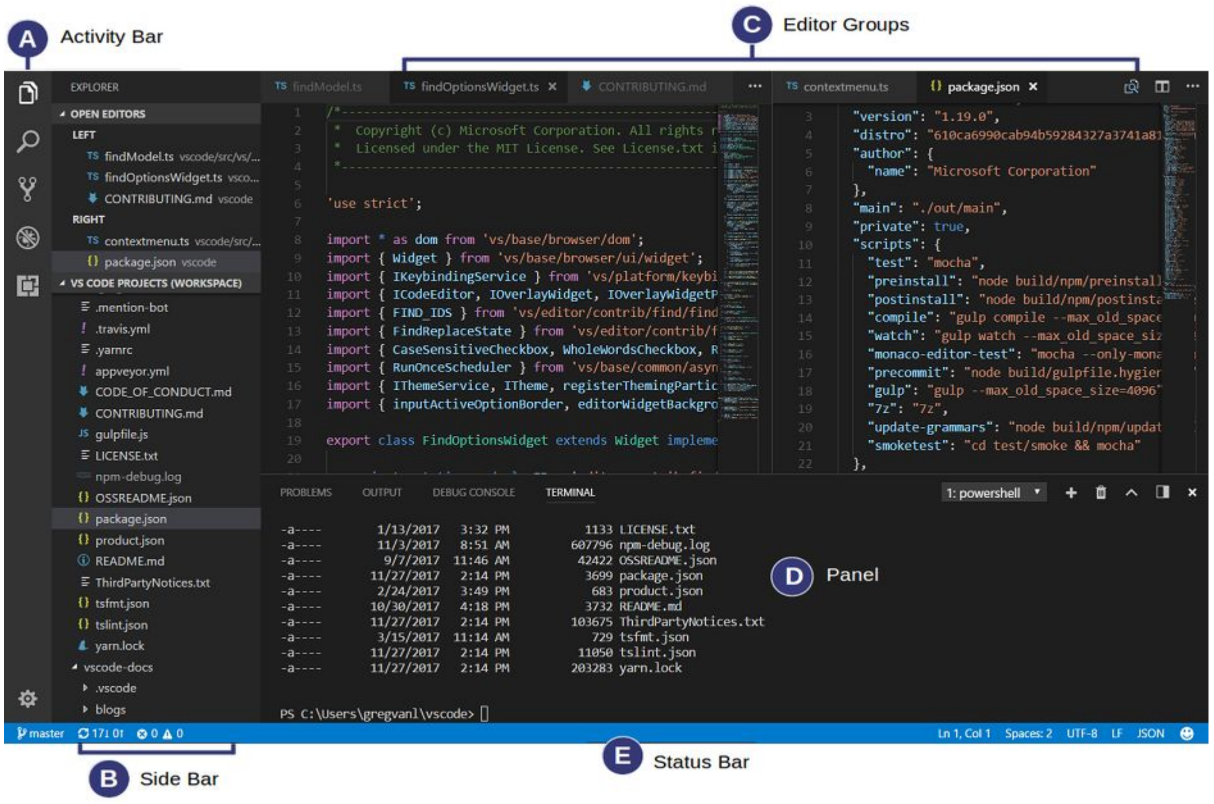
Tools and languages
Install support for JavaScript, Python, PHP, Azure, Docker and more

Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom an...

Color theme
Make the editor and your code look the way you love

VS Code User Interface

At its heart, Visual Studio Code is a code editor. Like many other code editors, VS Code adopts a common user interface and layout of an explorer on the left, showing all of the files and folders you have access to, and an editor on the right, showing the content of the files you have opened.



```
/*---------------------------------------------------------------
 *  Copyright (c) Microsoft Corporation. All rights r
 *  Licensed under the MIT License. See License.txt i
 *---------------------------------------------------------------

'use strict';

import * as dom from 'vs/base/browser/dom';
import { Widget } from 'vs/base/browser/ui/widget';
import { IKeybindingService } from 'vs/platform/keybi
import { ICodeEditor, IOverlayWidget, IOverlayWidgetP
import { FIND_IDS } from 'vs/editor/contrib/find/find
import { FindReplaceState } from 'vs/editor/contrib/f
import { CaseSensitiveCheckbox, WholeWordsCheckbox, R
import { RunOnceScheduler } from 'vs/base/common/asyn
import { IThemeService, ITheme, registerThemingPartic
import { inputActiveOptionBorder, editorWidgetBackgro

export class FindOptionsWidget extends Widget impleme
```

```
"version": "1.19.0",
"distro": "610ca6990cab94b59284327a3741a81
"author": {
  "name": "Microsoft Corporation"
},
"main": "./out/main",
"private": true,
"scripts": {
  "test": "mocha",
  "preinstall": "node build/npm/preinstall
  "postinstall": "node build/npm/postinsta
  "compile": "gulp compile --max_old_space
  "watch": "gulp watch --max_old_space_siz
  "monaco-editor-test": "mocha --only-mona
  "precommit": "node build/gulpfile.hygier
  "gulp": "gulp --max_old_space_size=4096"
  "7z": "7z",
  "update-grammars": "node build/npm/updat
  "smoketest": "cd test/smoke && mocha"
},
```

```
-a----        1/13/2017   3:32 PM           1133 LICENSE.txt
-a----        11/3/2017   8:51 AM         607796 npm-debug.log
-a----         9/7/2017  11:46 AM          42422 OSSREADME.json
-a----       11/27/2017   2:14 PM           3699 package.json
-a----        2/24/2017   3:49 PM            683 product.json
-a----       10/30/2017   4:18 PM           3732 README.md
-a----       11/27/2017   2:14 PM         103675 ThirdPartyNotices.txt
-a----        3/15/2017  11:14 AM            729 tsfmt.json
-a----       11/27/2017   2:14 PM          11050 tslint.json
-a----       11/27/2017   2:14 PM         203283 yarn.lock


PS C:\Users\gregvanl\vscode> 
```

Basic Layout

VS Code comes with a simple and intuitive layout that maximizes the space provided for the editor while leaving ample room to browse and access the full context of your folder or project. The UI is divided into five areas:

Editor - The main area to edit your files. You can open as many editors as you like side by side vertically and horizontally.

SideBar - Contains different views like the Explorer to assist you while working on your project.

Status Bar - Information about the opened project and the files you edit.

Activity Bar - Located on the far left-hand side, this lets you switch between views and gives you additional context-specific indicators, like the number of outgoing changes when Git is enabled.

Panels - You can display different panels below the editor region for output or debug information, errors, and warnings, or an integrated terminal. Panel can also be moved to the right for more vertical space.

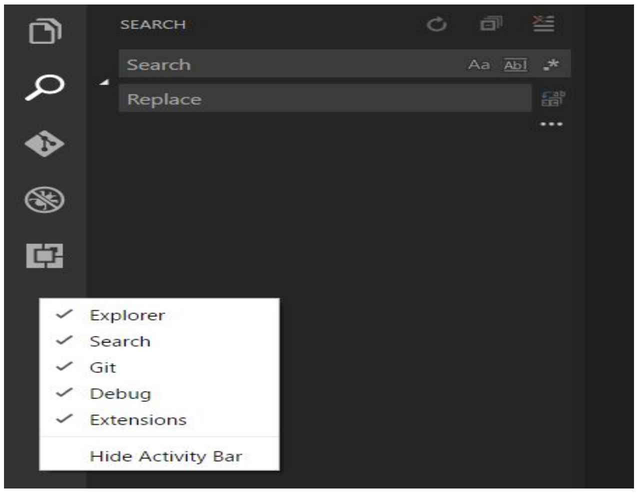
Each time you start VS Code, it opens up in the same state it was in when you last closed it. The folder, layout, and opened files are preserved.

Open files in each editor are displayed with tabbed headers (Tabs) at the top of the editor region. To learn more about tabbed headers, see the Tabs section below.

Tip: You can move the Side Bar to the right-hand side (View > Move Side Bar Right) or toggle its visibility (Ctrl+B).

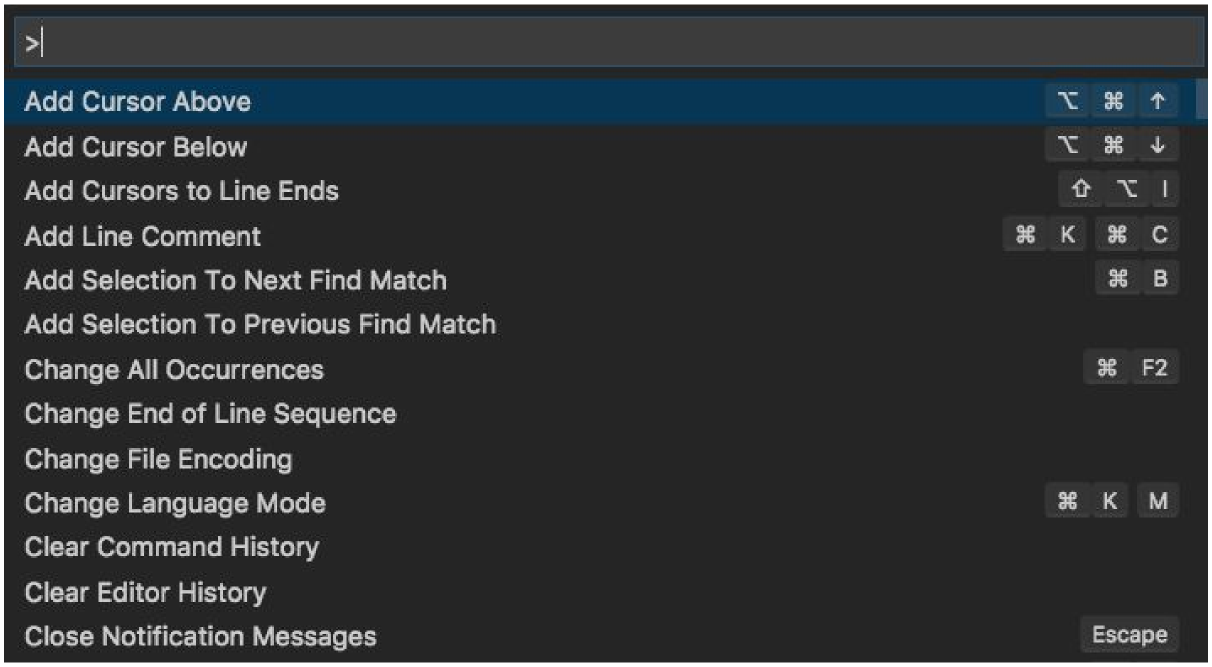
Activity Bar

The Activity Bar on the left lets you quickly switch between Views. You can also reorder Views by dragging and dropping them on the Activity Bar or remove a View entirely (right-click Hide from Activity Bar).



Command Palette

VS Code is equally accessible from the keyboard. The most important key combination to know is Ctrl+Shift+P, which brings up the Command Palette. From here, you have access to all of the functionality of the VS Code, including keyboard shortcuts for the most common operations.

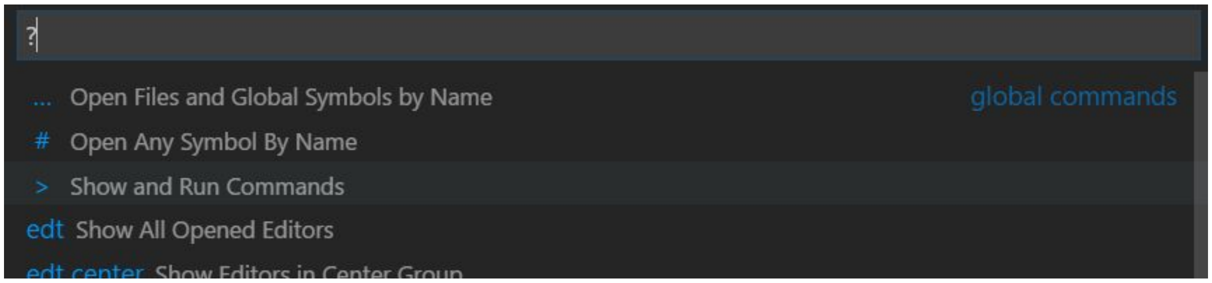


```
>
Add Cursor Above                          ⌥ ⌘ ↑
Add Cursor Below                          ⌥ ⌘ ↓
Add Cursors to Line Ends                  ⇧ ⌥ I
Add Line Comment                          ⌘ K ⌘ C
Add Selection To Next Find Match          ⌘ B
Add Selection To Previous Find Match
Change All Occurrences                    ⌘ F2
Change End of Line Sequence
Change File Encoding
Change Language Mode                      ⌘ K M
Clear Command History
Clear Editor History
Close Notification Messages               Escape
```

The Command Palette provides access to many commands. You can execute editor commands, open files, search for symbols, and see a quick outline of a file, all using the same interactive window. Here are a few tips:

- Ctrl+P will let you navigate to any file or symbol by typing its name
- Ctrl+Shift+Tab will cycle you through the last set of files opened
- Ctrl+Shift+P will bring you directly to the editor commands
- Ctrl+Shift+O will let you navigate to a specific symbol in a file
- Ctrl+G will let you navigate to a specific line in a file

Type ? into the input field to get a list of available commands you can execute from here:

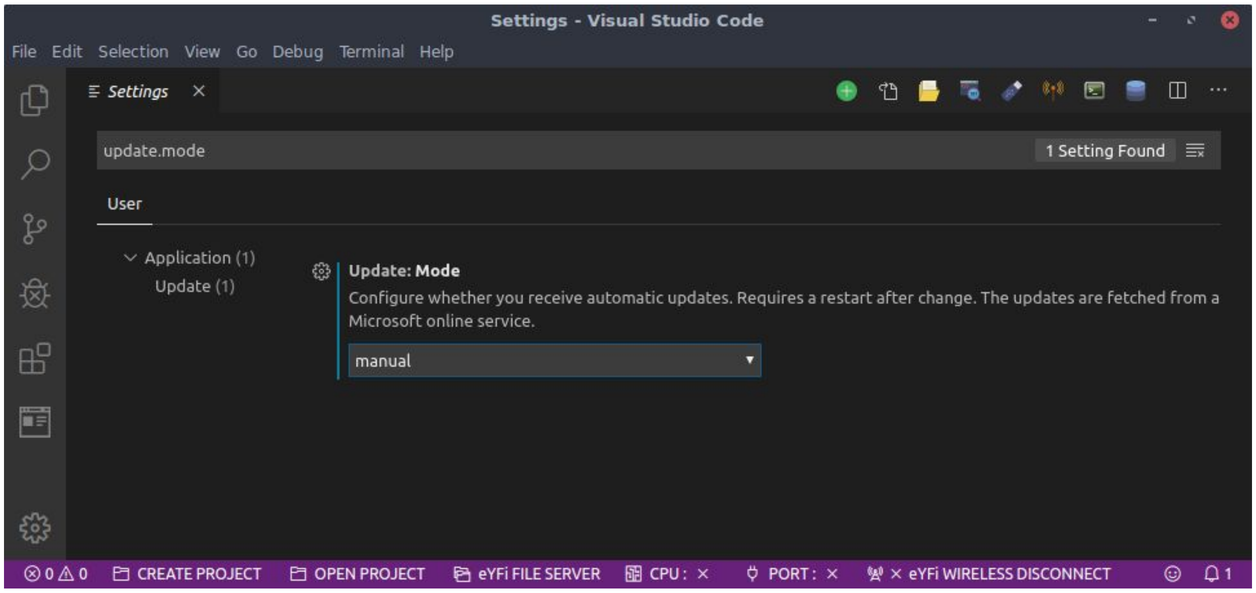


```
?
...  Open Files and Global Symbols by Name        global commands
#    Open Any Symbol By Name
>    Show and Run Commands
edt  Show All Opened Editors
edt center  Show Editors in Center Group
```

Disable Update VS Code Pop Up

When you start VS Code you may get pop up to update VS Code. As the eYFi-Mega Extension is tested on **VS Code 1.36.0**, we suggest you not to update your VS Code. To disable Update Pop up you can do the following,

1. Press **Ctrl+comma** to open settings.
2. Search for “Update: Mode” by typing **update.mode** in the search bar.
3. Set “Update: Mode” to **manual**.



update.mode

Update: Mode
Configure whether you receive automatic updates. Requires a restart after change. The updates are fetched from a Microsoft online service.
manual

4. Now restart VS Code for this change to take effect.

Installation of eYFi-Mega Extension and Toolchains in Linux

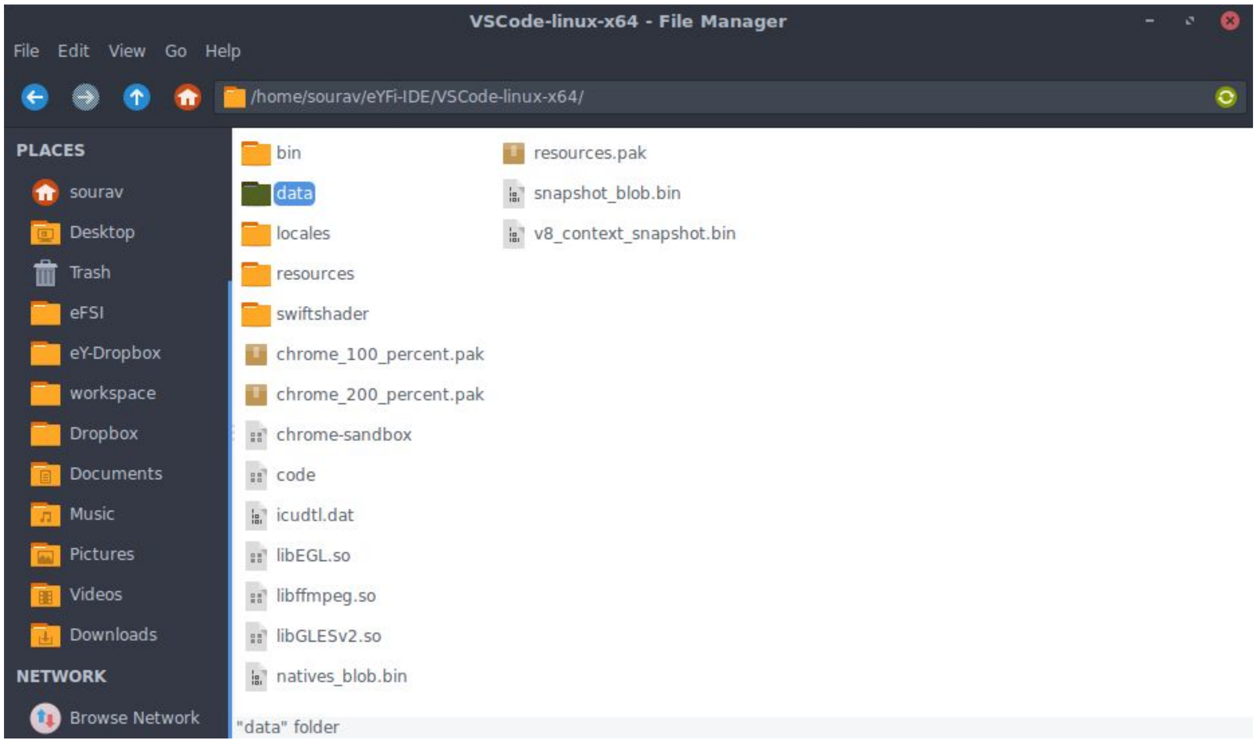
Installation of eYFi-Mega Extension

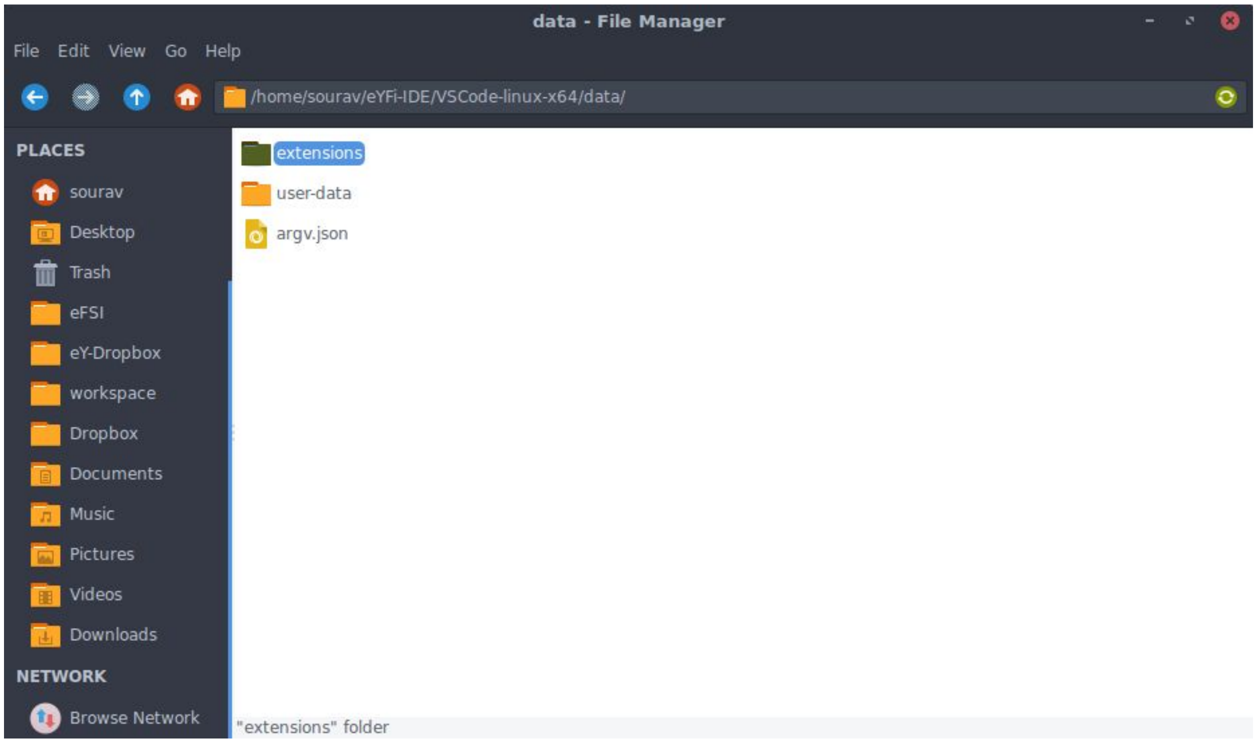
To install eYFi-Mega Extension two zip files need to be downloaded.,

- extensions.zip
- eY-Toolchain.zip

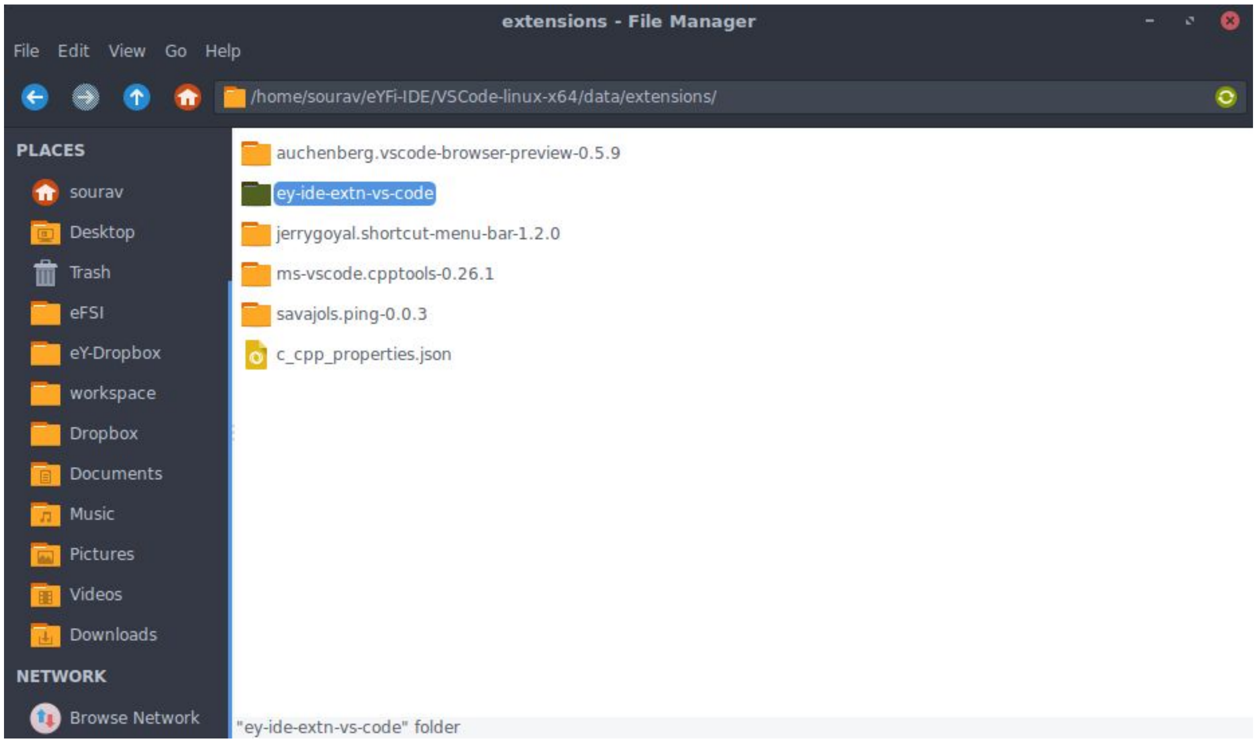
These two zip files can be downloaded from eYFi-Mega’s website which can be found at e-Yantra’s products page at https://e-yantra.org/products.

1. Extract the extension.zip file in the “data” folder of VS Code.



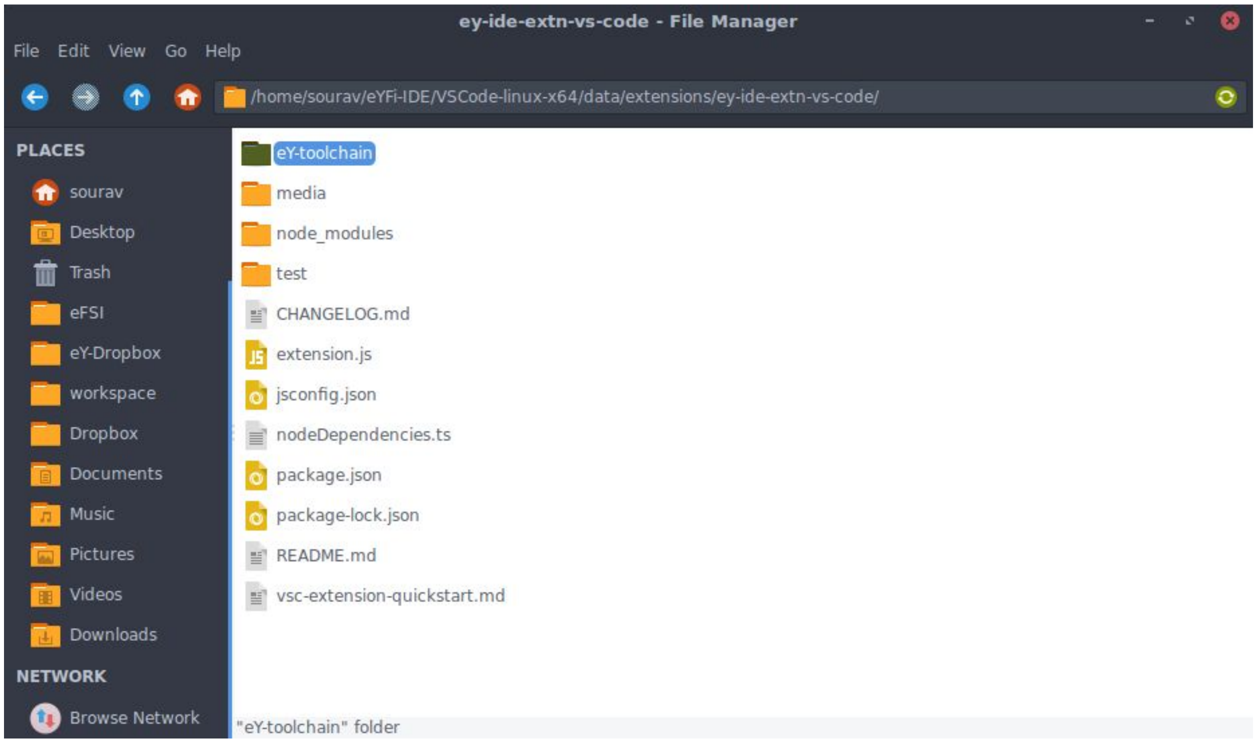


2. Extract the eY-Toolchain.zip in the “ey-ide-extn-vs-code” folder inside the “extensions” folder.



auchenberg.vscode-browser-preview-0.5.9
ey-ide-extn-vs-code
jerrygoyal.shortcut-menu-bar-1.2.0
ms-vscode.cpptools-0.26.1
savajols.ping-0.0.3
c_cpp_properties.json

"ey-ide-extn-vs-code" folder



eY-toolchain
media
node_modules
test
CHANGELOG.md
extension.js
jsconfig.json
nodeDependencies.ts
package.json
package-lock.json
README.md
vsc-extension-quickstart.md

"eY-toolchain" folder

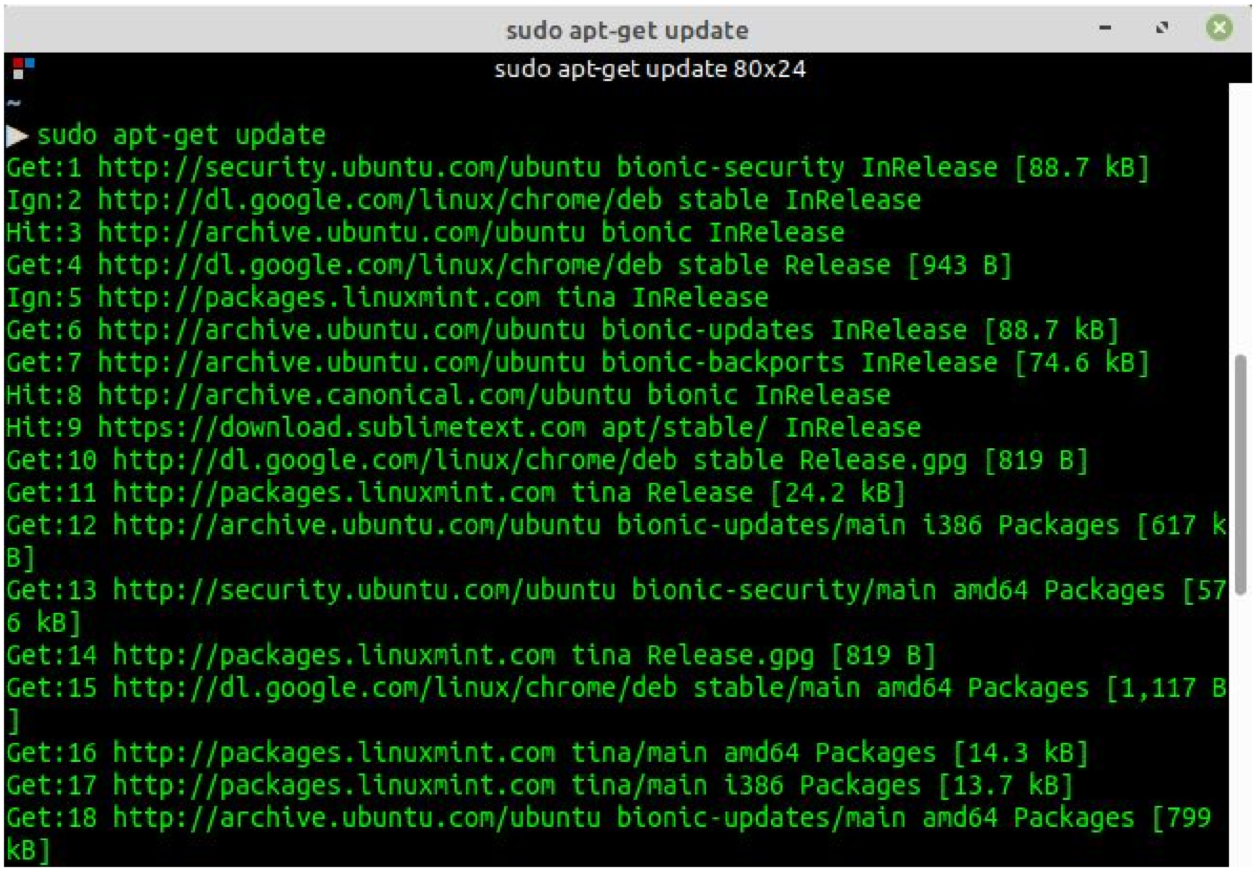
3. After these two steps, eYFi-Mega Extension will get installed for your VS Code.

To compile ATMega 2560 and ESP32 programs, their respective toolchain has to be installed in your system. In the next subsection, we will discuss how to install the two toolchains.

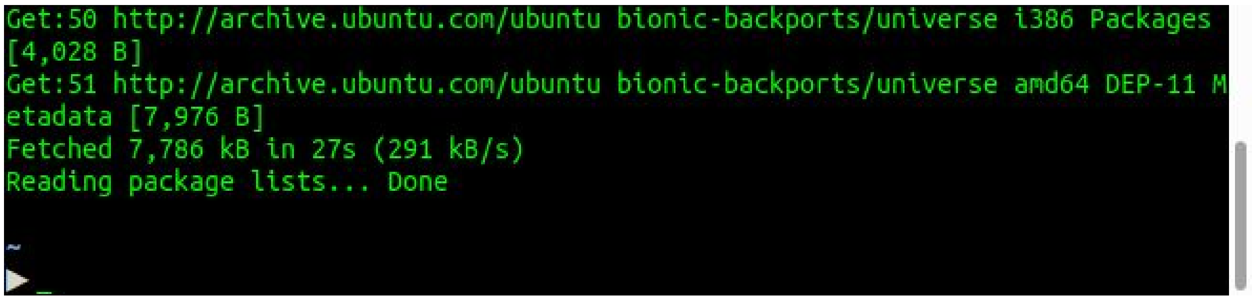
Installation of AVR Toolchain for ATMega 2560

Execute the following commands to install AVR Toolchain in your system,

```
$ sudo apt-get update
```

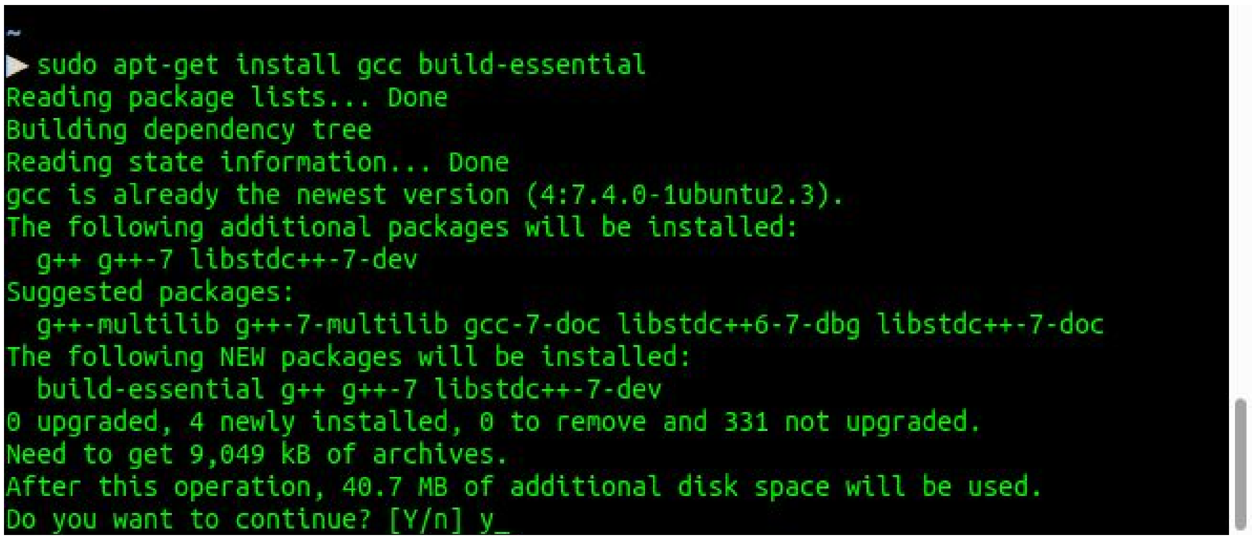


```
sudo apt-get update
sudo apt-get update 80x24
~
▶ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Ign:2 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic InRelease
Get:4 http://dl.google.com/linux/chrome/deb stable Release [943 B]
Ign:5 http://packages.linuxmint.com tina InRelease
Get:6 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:8 http://archive.canonical.com/ubuntu bionic InRelease
Hit:9 https://download.sublimetext.com apt/stable/ InRelease
Get:10 http://dl.google.com/linux/chrome/deb stable Release.gpg [819 B]
Get:11 http://packages.linuxmint.com tina Release [24.2 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic-updates/main i386 Packages [617 k
B]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [57
6 kB]
Get:14 http://packages.linuxmint.com tina Release.gpg [819 B]
Get:15 http://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,117 B
]
Get:16 http://packages.linuxmint.com tina/main amd64 Packages [14.3 kB]
Get:17 http://packages.linuxmint.com tina/main i386 Packages [13.7 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [799
kB]
```



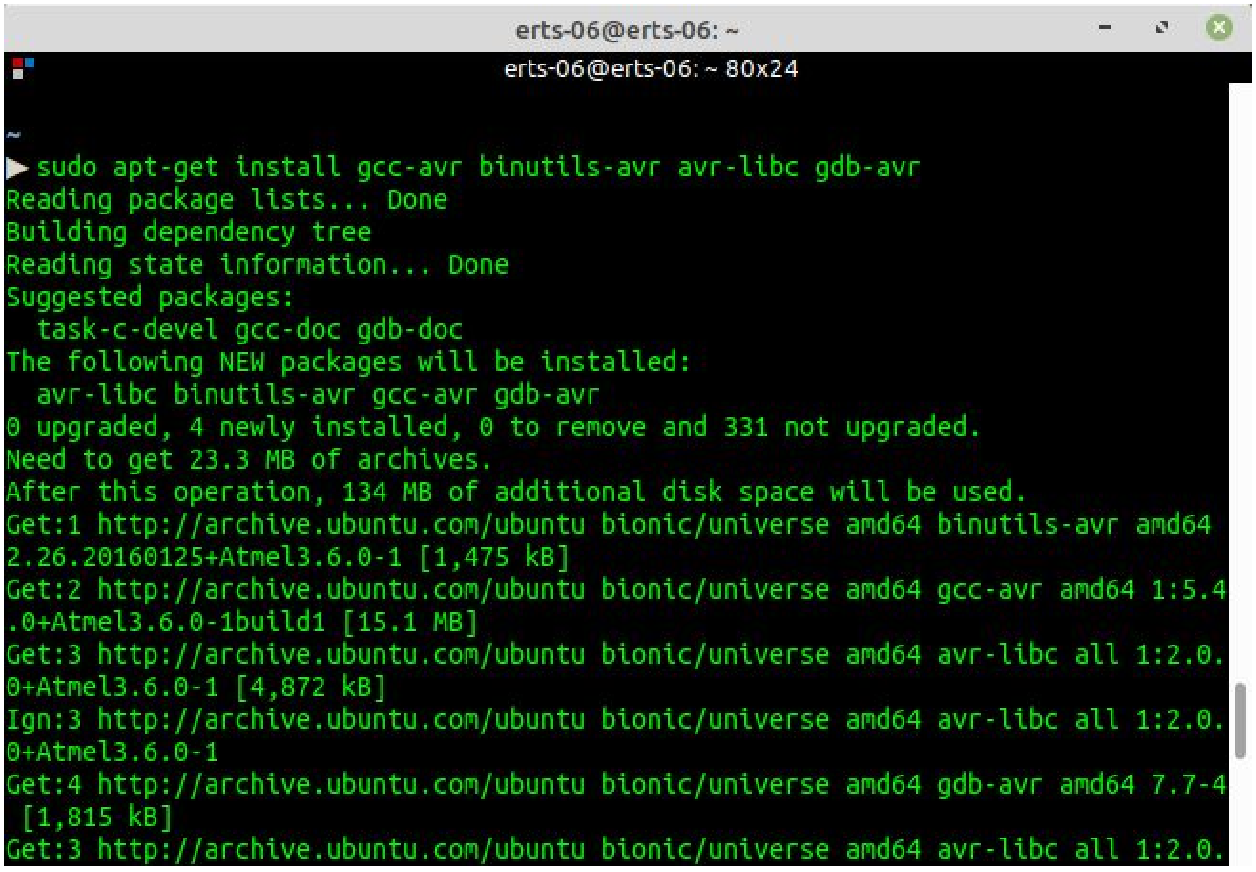
```
Get:50 http://archive.ubuntu.com/ubuntu bionic-backports/universe i386 Packages
[4,028 B]
Get:51 http://archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 M
etadata [7,976 B]
Fetched 7,786 kB in 27s (291 kB/s)
Reading package lists... Done

~
▶ _
```

```
$ sudo apt-get install gcc build-essential
```



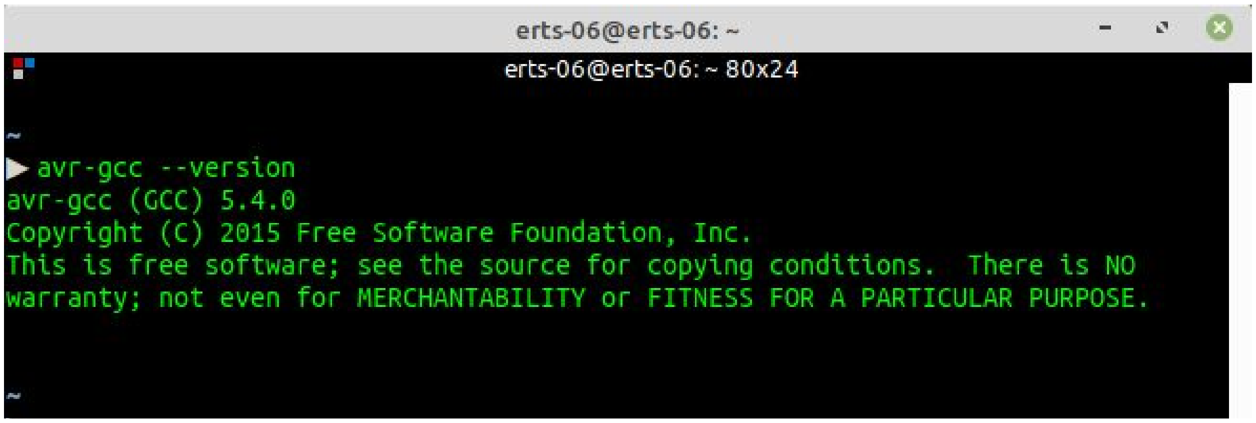
```
~
▶ sudo apt-get install gcc build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc is already the newest version (4:7.4.0-1ubuntu2.3).
The following additional packages will be installed:
  g++ g++-7 libstdc++-7-dev
Suggested packages:
  g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg libstdc++-7-doc
The following NEW packages will be installed:
  build-essential g++ g++-7 libstdc++-7-dev
0 upgraded, 4 newly installed, 0 to remove and 331 not upgraded.
Need to get 9,049 kB of archives.
After this operation, 40.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y_
```

```
$ sudo apt-get install gcc-avr binutils-avr avr-libc gdb-avr
```



```
~
▶ sudo apt-get install gcc-avr binutils-avr avr-libc gdb-avr
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  task-c-devel gcc-doc gdb-doc
The following NEW packages will be installed:
  avr-libc binutils-avr gcc-avr gdb-avr
0 upgraded, 4 newly installed, 0 to remove and 331 not upgraded.
Need to get 23.3 MB of archives.
After this operation, 134 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 binutils-avr amd64
2.26.20160125+Atmel3.6.0-1 [1,475 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/universe amd64 gcc-avr amd64 1:5.4
.0+Atmel3.6.0-1build1 [15.1 MB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/universe amd64 avr-libc all 1:2.0.
0+Atmel3.6.0-1 [4,872 kB]
Ign:3 http://archive.ubuntu.com/ubuntu bionic/universe amd64 avr-libc all 1:2.0.
0+Atmel3.6.0-1
Get:4 http://archive.ubuntu.com/ubuntu bionic/universe amd64 gdb-avr amd64 7.7-4
 [1,815 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/universe amd64 avr-libc all 1:2.0.
```

```
$ avr-gcc --version
```



erts-06@erts-06: ~

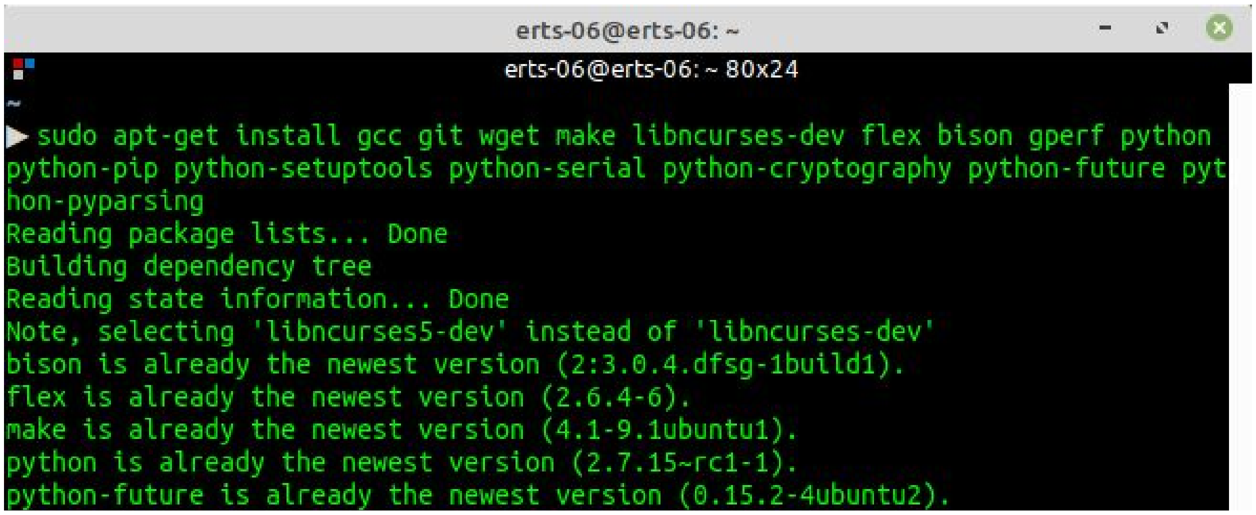
erts-06@erts-06: ~ 80x24

```
~
▶ avr-gcc --version
avr-gcc (GCC) 5.4.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


~
```

Installation of ESP-IDF and Xtensa Compiler for ESP32

To install ESP-IDF and Xtensa Compiler for ESP32 execute the following commands in the given sequence,

```
$ sudo apt-get install gcc git wget make libncurses-dev flex bison gperf python
python-pip python-setuptools python-serial python-cryptography python-future
python-pyparsing
```



erts-06@erts-06: ~

erts-06@erts-06: ~ 80x24

```
~
▶ sudo apt-get install gcc git wget make libncurses-dev flex bison gperf python
python-pip python-setuptools python-serial python-cryptography python-future pyt
hon-pyparsing
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses5-dev' instead of 'libncurses-dev'
bison is already the newest version (2:3.0.4.dfsg-1build1).
flex is already the newest version (2.6.4-6).
make is already the newest version (4.1-9.1ubuntu1).
python is already the newest version (2.7.15~rc1-1).
python-future is already the newest version (0.15.2-4ubuntu2).
```

```
$ mkdir -p ~/esp
```

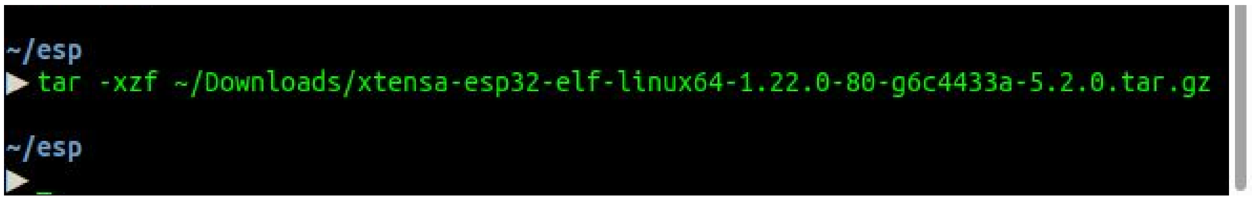
```
$ cd ~/esp
```

Download Xtensa Compiler

for 64-bit Linux: https://dl.espressif.com/dl/xtensa-esp32-elf-linux64-1.22.0-80-g6c4433a-5.2.0.tar.gz

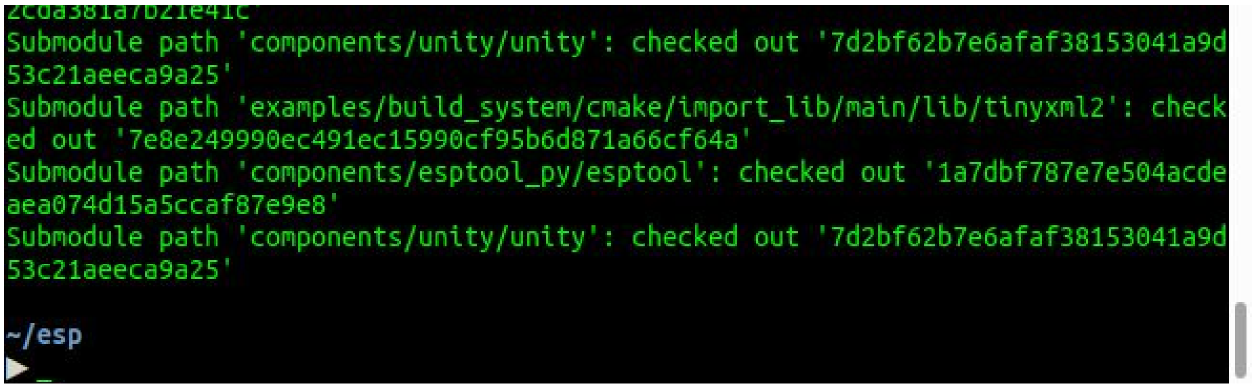
for 32-bit Linux: https://dl.espressif.com/dl/xtensa-esp32-elf-linux32-1.22.0-80-g6c4433a-5.2.0.tar.gz

```
$ tar -xzf ~/Downloads/xtensa-esp32-elf-linux64-1.22.0-80-g6c4433a-5.2.0.tar.gz
```



```
~/esp
▶ tar -xzf ~/Downloads/xtensa-esp32-elf-linux64-1.22.0-80-g6c4433a-5.2.0.tar.gz

~/esp
▶ _
```

```
$ git clone -b v3.3 --recursive https://github.com/espressif/esp-idf.git
```



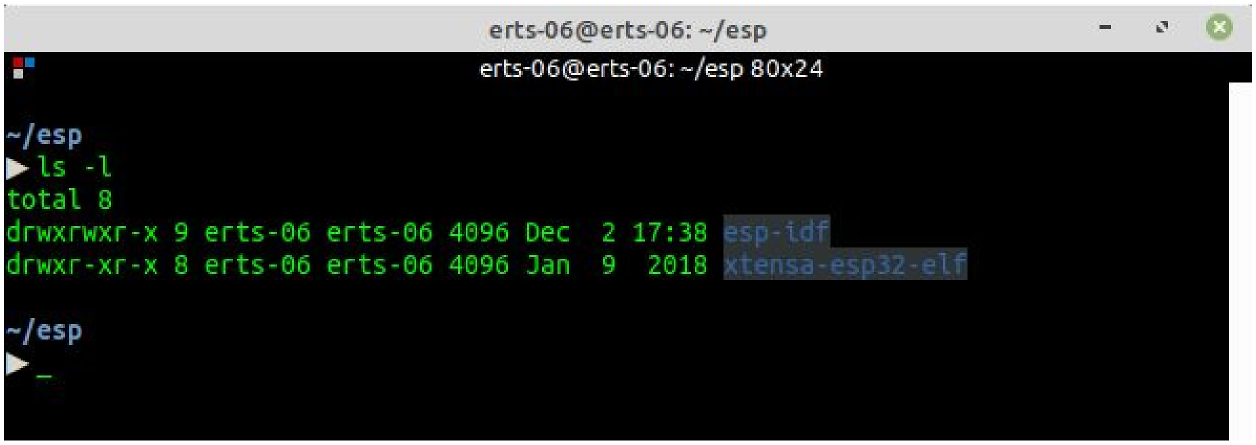
```
2cda381a7b21e41c'
Submodule path 'components/unity/unity': checked out '7d2bf62b7e6afaf38153041a9d
53c21aeeca9a25'
Submodule path 'examples/build_system/cmake/import_lib/main/lib/tinyxml2': check
ed out '7e8e249990ec491ec15990cf95b6d871a66cf64a'
Submodule path 'components/esptool_py/esptool': checked out '1a7dbf787e7e504acde
aea074d15a5ccaf87e9e8'
Submodule path 'components/unity/unity': checked out '7d2bf62b7e6afaf38153041a9d
53c21aeeca9a25'

~/esp
▶ _
```

Now the esp folder should have following two folders

1. esp-idf
2. xtensa-esp32-elf

Use the following command to check folders in the esp folder

```
$ ls -l
```

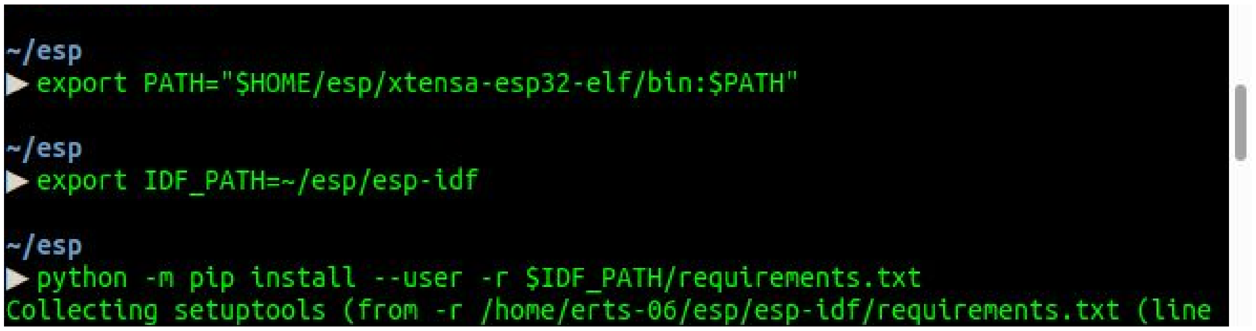


```
erts-06@erts-06: ~/esp
erts-06@erts-06: ~/esp 80x24

~/esp
▶ ls -l
total 8
drwxrwxr-x 9 erts-06 erts-06 4096 Dec  2 17:38 esp-idf
drwxr-xr-x 8 erts-06 erts-06 4096 Jan  9  2018 xtensa-esp32-elf

~/esp
▶ _
```

```
$ export PATH="$HOME/esp/xtensa-esp32-elf/bin:$PATH"
```

```
$ export IDF_PATH=~/esp/esp-idf
```

```
$ sudo python -m pip install --user -r $IDF_PATH/requirements.txt
```



```
~/esp
▶ export PATH="$HOME/esp/xtensa-esp32-elf/bin:$PATH"

~/esp
▶ export IDF_PATH=~/esp/esp-idf

~/esp
▶ python -m pip install --user -r $IDF_PATH/requirements.txt
Collecting setuptools (from -r /home/erts-06/esp/esp-idf/requirements.txt (line
```

Do the following to set the environment variables for ESP32

If you are using BASH shell, execute the following command

```
$ sudo gedit ~/.bashrc
```

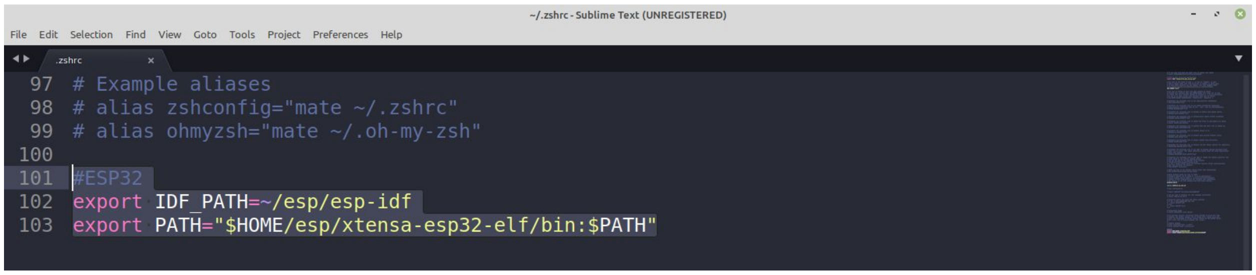
If you are using ZSH shell, execute the following command

```
$ sudo gedit ~/.zshrc
```

NOTE: Most of the major Linux Distributions like Ubuntu uses BASH Shell.

Once the text editor opens up add the following lines at the end of the file,

```
#ESP32
export IDF_PATH=~/esp/esp-idf
export PATH="$HOME/esp/xtensa-esp32-elf/bin:$PATH"
```


~/.zshrc - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
.zshrc

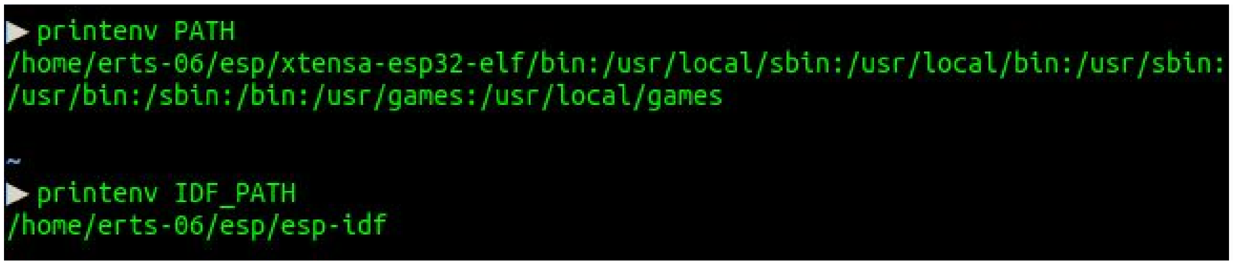
```
# Example aliases
# alias zshconfig="mate ~/.zshrc"
# alias ohmyzsh="mate ~/.oh-my-zsh"

#ESP32
export IDF_PATH=~/esp/esp-idf
export PATH="$HOME/esp/xtensa-esp32-elf/bin:$PATH"
```

Save and close the text editor.

To check the Environment Variables execute the following commands,

```
$ printenv IDF_PATH
```

```
$ printenv PATH
```



```
▶ printenv PATH
/home/erts-06/esp/xtensa-esp32-elf/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:
/usr/bin:/sbin:/bin:/usr/games:/usr/local/games

~
▶ printenv IDF_PATH
/home/erts-06/esp/esp-idf
```

Dial Out

The currently logged user should have read and write access to the serial port over USB. On most Linux distributions, this is done by adding the user to dialout group with the following command:

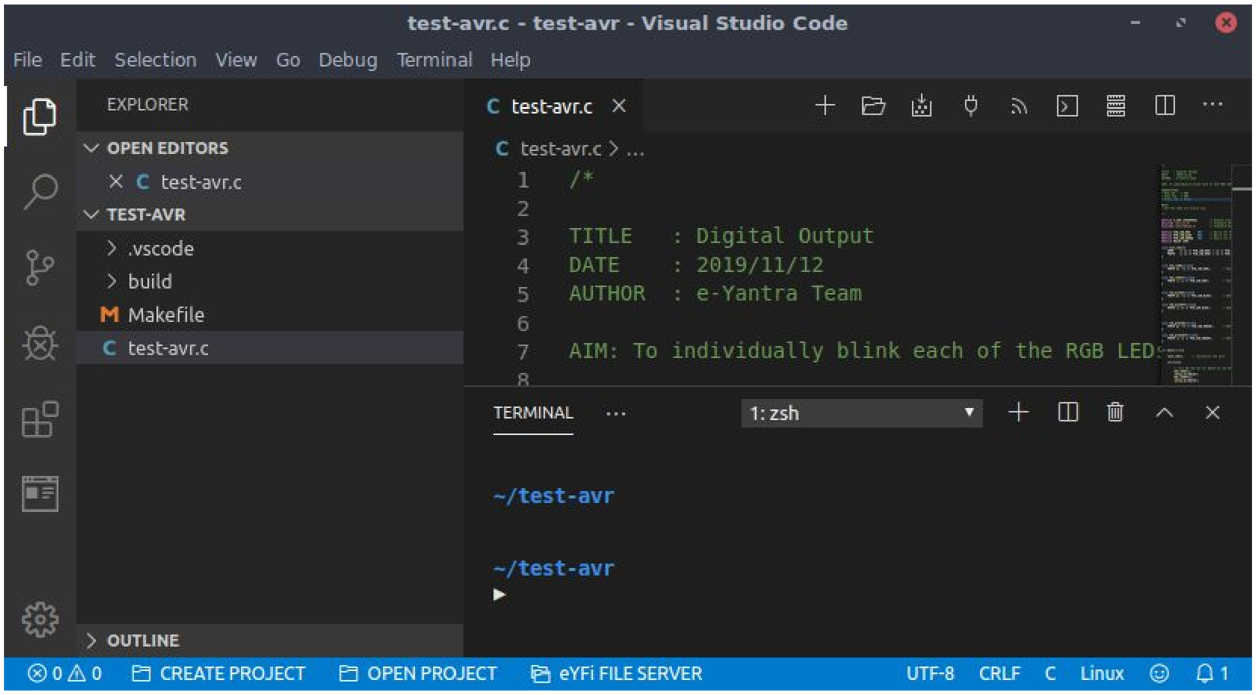
```
$ sudo usermod -a -G dialout $USER
```

If the above command does not work, use the following command

```
$ sudo chmod 777 /dev/ttyUSB0
```

Using eYFi-Mega Extension

Buttons of eYFi-Mega Extension



```
/*

TITLE    : Digital Output
DATE     : 2019/11/12
AUTHOR   : e-Yantra Team

AIM: To individually blink each of the RGB LEDs
```

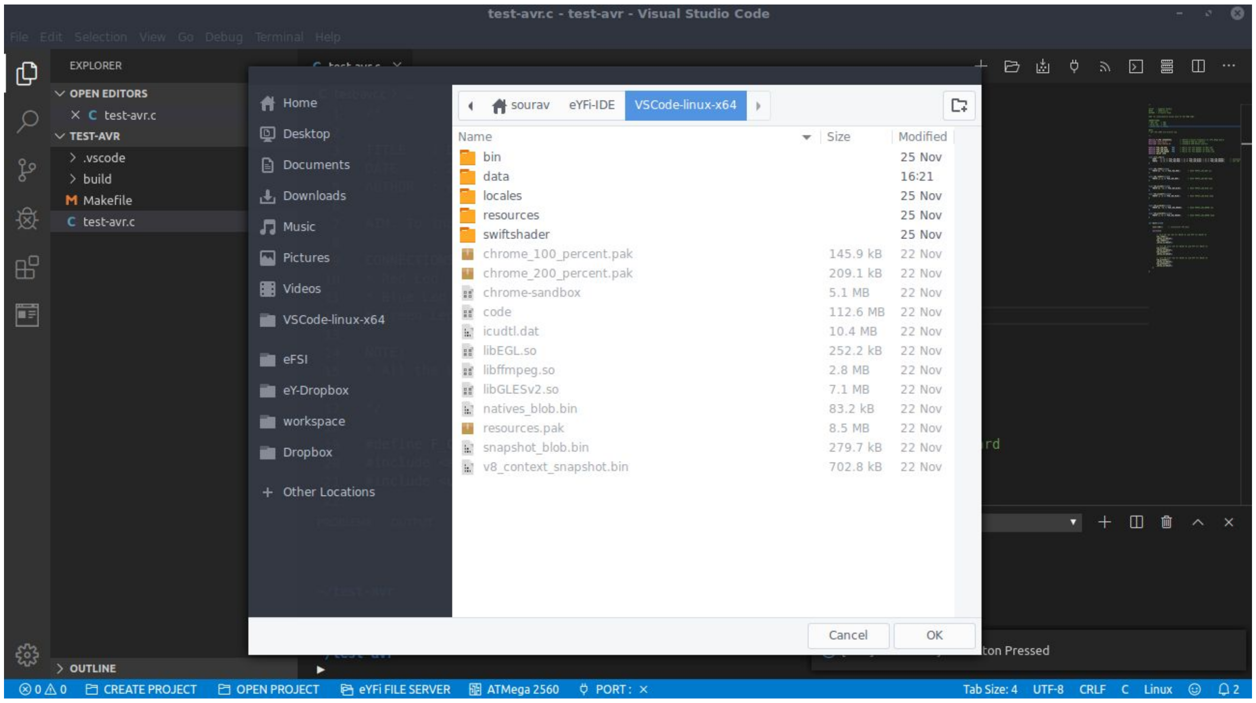
```
~/test-avr

~/test-avr
▶
```

On the top right corner, you can find several buttons that are useful for development on the eYFi-Mega Board.

Button - Create Project

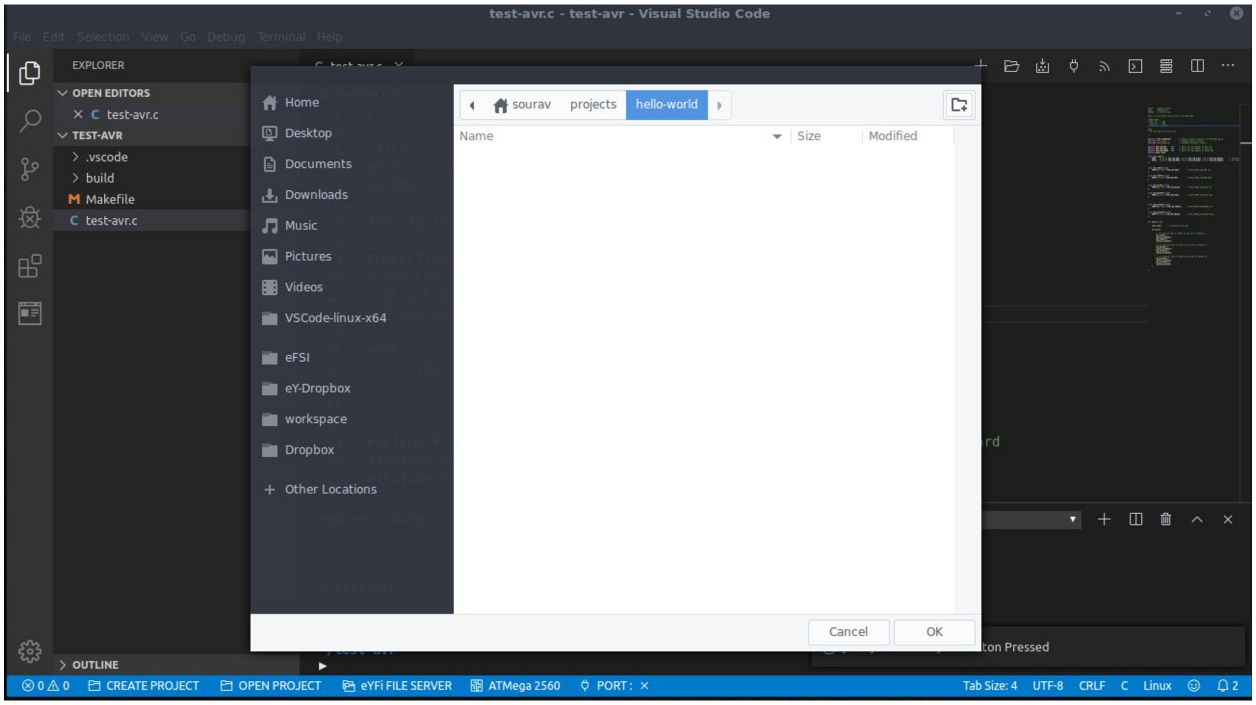
Using this button you can create projects for ESP32 and ATMega2560.

When you press this button a window to select target microcontroller will pop-up. Using this window you can select one of the microcontroller for which you want to create project.



Name	Size	Modified
bin		25 Nov
data		16:21
locales		25 Nov
resources		25 Nov
swiftshader		25 Nov
chrome_100_percent.pak	145.9 kB	22 Nov
chrome_200_percent.pak	209.1 kB	22 Nov
chrome-sandbox	5.1 MB	22 Nov
code	112.6 MB	22 Nov
icudtl.dat	10.4 MB	22 Nov
libEGL.so	252.2 kB	22 Nov
libffmpeg.so	2.8 MB	22 Nov
libGLESv2.so	7.1 MB	22 Nov
natives_blob.bin	83.2 kB	22 Nov
resources.pak	8.5 MB	22 Nov
snapshot_blob.bin	279.7 kB	22 Nov
v8_context_snapshot.bin	702.8 kB	22 Nov

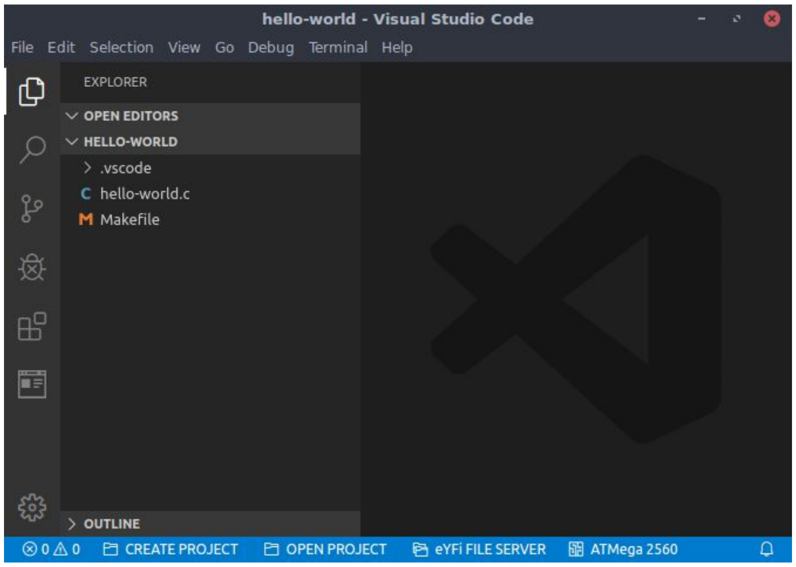
After selecting the target microcontroller, a file explorer will pop up. Using this file explorer first navigate to the folder where you want to save your project and then create a folder with your project name. Then open your project folder and click OK.



NOTE: Make sure your project name does not contain any whitespace.

After that, the extension will populate your project folder with the necessary files.

NOTE: For ESP32 projects, you can find the source file in the **main** folder



Button - Open Project

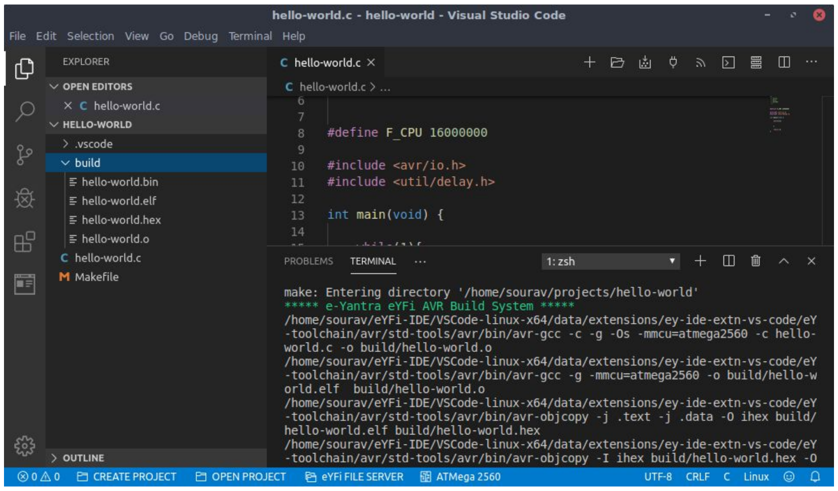
You can use this button to open your existing eYFi-Mega projects. When you click this button a file explorer will pop-up. Using this you can navigate to your project folder and then open it.

NOTE: We recommend not to use this button to open your non eYFi-Mega projects. Use VS Code’s native menubar for that.

Button - Compile

Use this button to compile and generate the binary and hex file of your application.

If the build is successful for ATMega 2560, all the object files, the hex file, and the binary file of your application will get generated inside the **build folder** in the project directory.

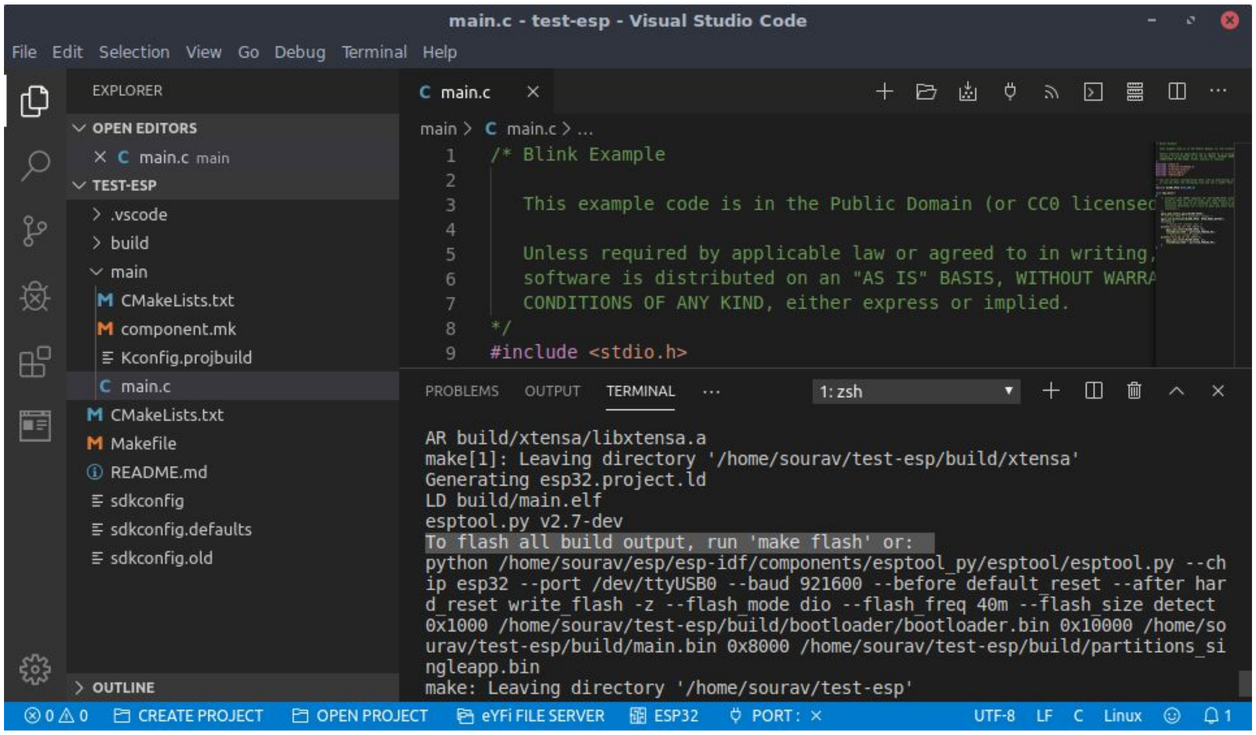


```c
#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main(void) {
```

```
make: Entering directory '/home/sourav/projects/hello-world'
***** e-Yantra eYFi AVR Build System *****
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-gcc -c -g -Os -mmcu=atmega2560 -c hello-
world.c -o build/hello-world.o
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-gcc -g -mmcu=atmega2560 -o build/hello-w
orld.elf  build/hello-world.o
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-objcopy -j .text -j .data -O ihex build/
hello-world.elf build/hello-world.hex
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-objcopy -I ihex build/hello-world.hex -O
```

When the build is successful for ESP32, bin file will get generated in the build folder and you will get the following message in the terminal,

“To flash all build output, run 'make flash' or:”



```
/* Blink Example

   This example code is in the Public Domain (or CC0 licensed

   Unless required by applicable law or agreed to in writing,
   software is distributed on an "AS IS" BASIS, WITHOUT WARRA
   CONDITIONS OF ANY KIND, either express or implied.
*/
#include <stdio.h>
```

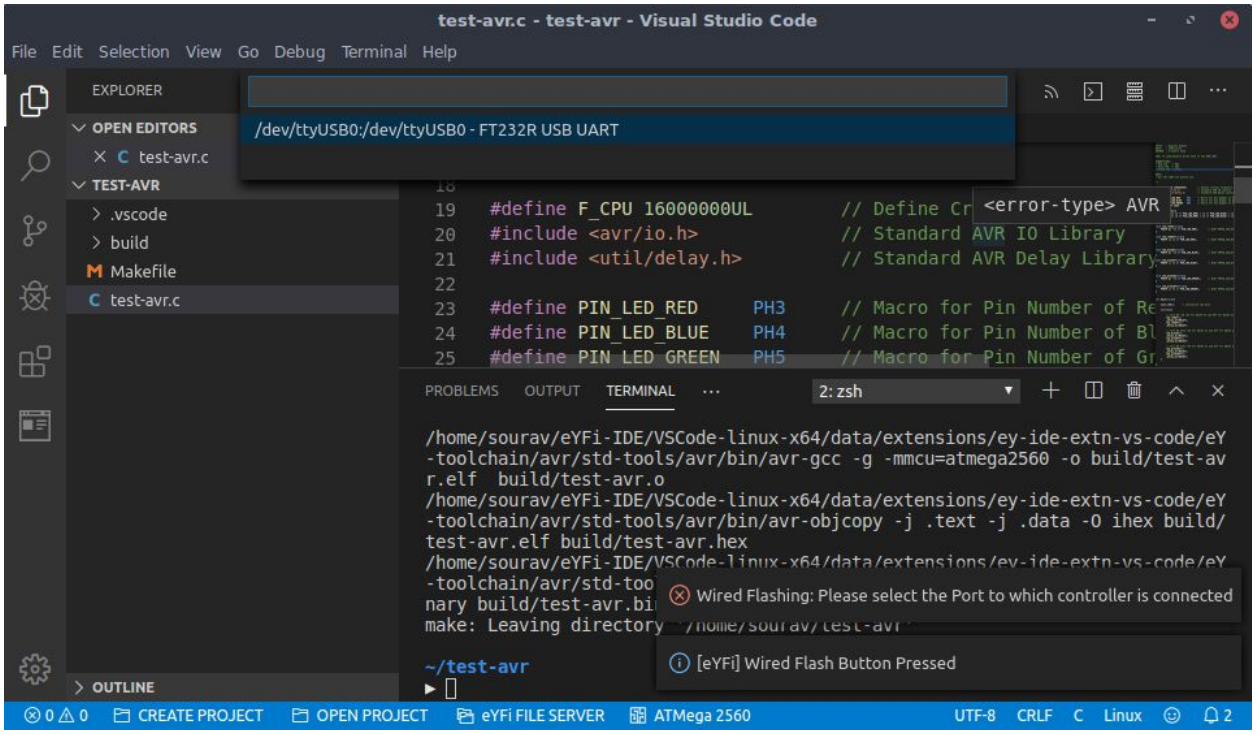
```
AR build/xtensa/libxtensa.a
make[1]: Leaving directory '/home/sourav/test-esp/build/xtensa'
Generating esp32.project.ld
LD build/main.elf
esptool.py v2.7-dev
To flash all build output, run 'make flash' or:
python /home/sourav/esp/esp-idf/components/esptool_py/esptool/esptool.py --ch
ip esp32 --port /dev/ttyUSB0 --baud 921600 --before default_reset --after har
d_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size detect
0x1000 /home/sourav/test-esp/build/bootloader/bootloader.bin 0x10000 /home/so
urav/test-esp/build/main.bin 0x8000 /home/sourav/test-esp/build/partitions_si
ngleapp.bin
make: Leaving directory '/home/sourav/test-esp'
```

All the errors and warnings, if there is any, will get displayed in the same terminal.

Button - Wired Flash

Use this button to flash the binary file in the microcontroller using the serial port.

When you press this button and if the serial port to which your eYFi-Mega is connected is not set, a window to select serial port will pop up. Once you select the serial port, the flashing process will start.



```
#define F_CPU 16000000UL          // Define Cr
#include <avr/io.h>               // Standard AVR IO Library
#include <util/delay.h>           // Standard AVR Delay Library

#define PIN_LED_RED      PH3      // Macro for Pin Number of Re
#define PIN_LED_BLUE     PH4      // Macro for Pin Number of Bl
#define PIN_LED_GREEN    PH5      // Macro for Pin Number of Gr
```

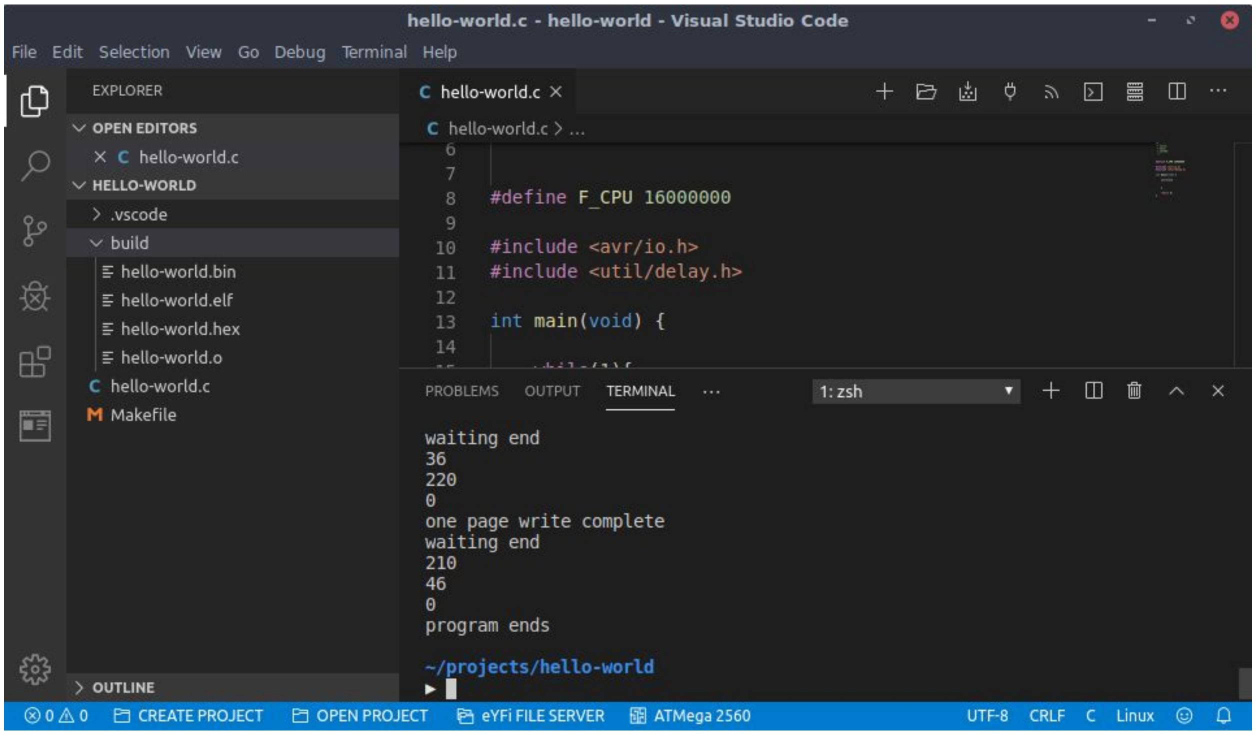
```
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-gcc -g -mmcu=atmega2560 -o build/test-av
r.elf  build/test-avr.o
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-tools/avr/bin/avr-objcopy -j .text -j .data -O ihex build/
test-avr.elf build/test-avr.hex
/home/sourav/eYFi-IDE/VSCode-linux-x64/data/extensions/ey-ide-extn-vs-code/eY
-toolchain/avr/std-too
nary build/test-avr.bi
make: Leaving directory '/home/sourav/test-avr'

~/test-avr
▶
```

Wired Flashing: Please select the Port to which controller is connected

[eYFi] Wired Flash Button Pressed

When flashing is complete for ATMega 2560, the following message will get printed in the terminal,

“program ends”

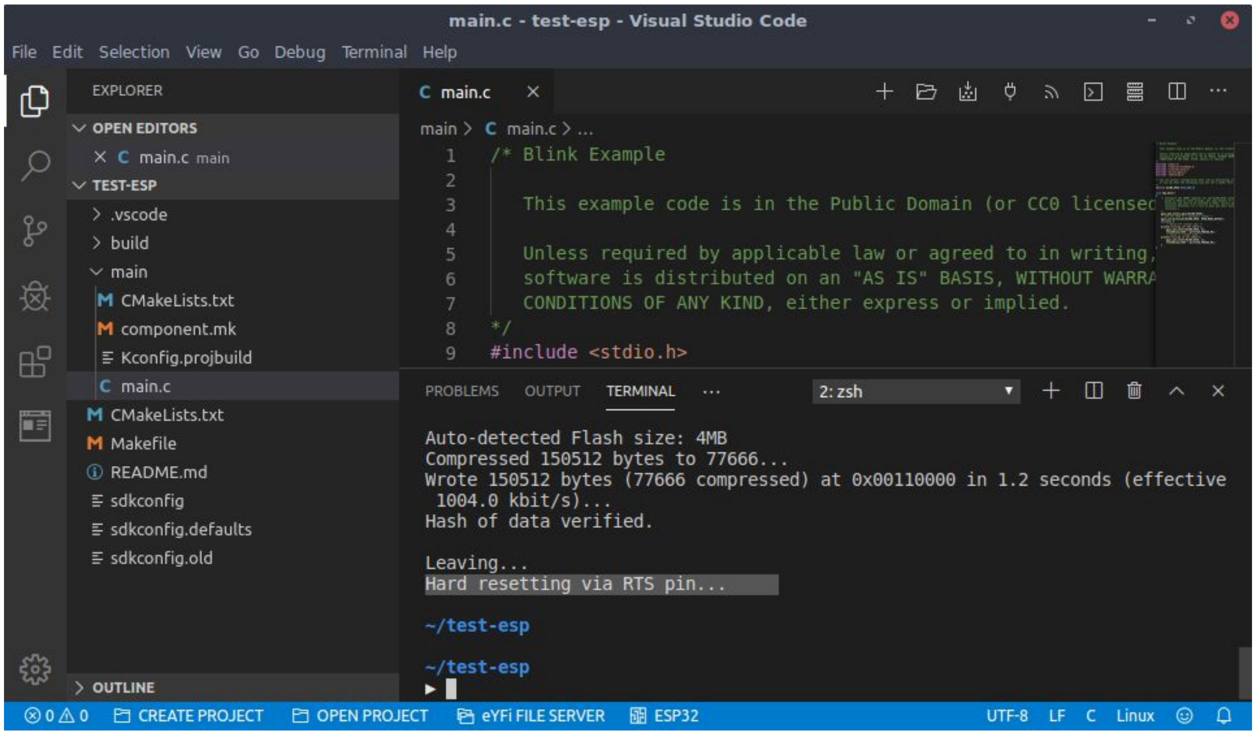


```
#define F_CPU 16000000

#include <avr/io.h>
#include <util/delay.h>

int main(void) {
```

```
waiting end
36
220
0
one page write complete
waiting end
210
46
0
program ends

~/projects/hello-world
▶
```

When flashing is complete for ESP32, the following message will get printed in the terminal,

“Hard resetting via RTS pin…”



```c
/* Blink Example

   This example code is in the Public Domain (or CC0 licensec
   
   Unless required by applicable law or agreed to in writing,
   software is distributed on an "AS IS" BASIS, WITHOUT WARRA
   CONDITIONS OF ANY KIND, either express or implied.
*/
#include <stdio.h>
```

```
Auto-detected Flash size: 4MB
Compressed 150512 bytes to 77666...
Wrote 150512 bytes (77666 compressed) at 0x00110000 in 1.2 seconds (effective
 1004.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

~/test-esp

~/test-esp
▶
```

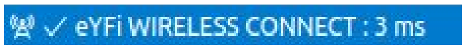
NOTE: When flashing make sure that switch S1 is in the correct position.

Button – Wireless Flash

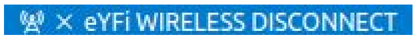
Use this button to flash the binary file in the microcontroller over Wi-Fi.

Before you start Wireless Flashing make sure that you are connected to the Wireless Access Point of eYFi-Mega.

Once you are connected to the Wireless Access Point, the following message will get displayed in the status bar below.



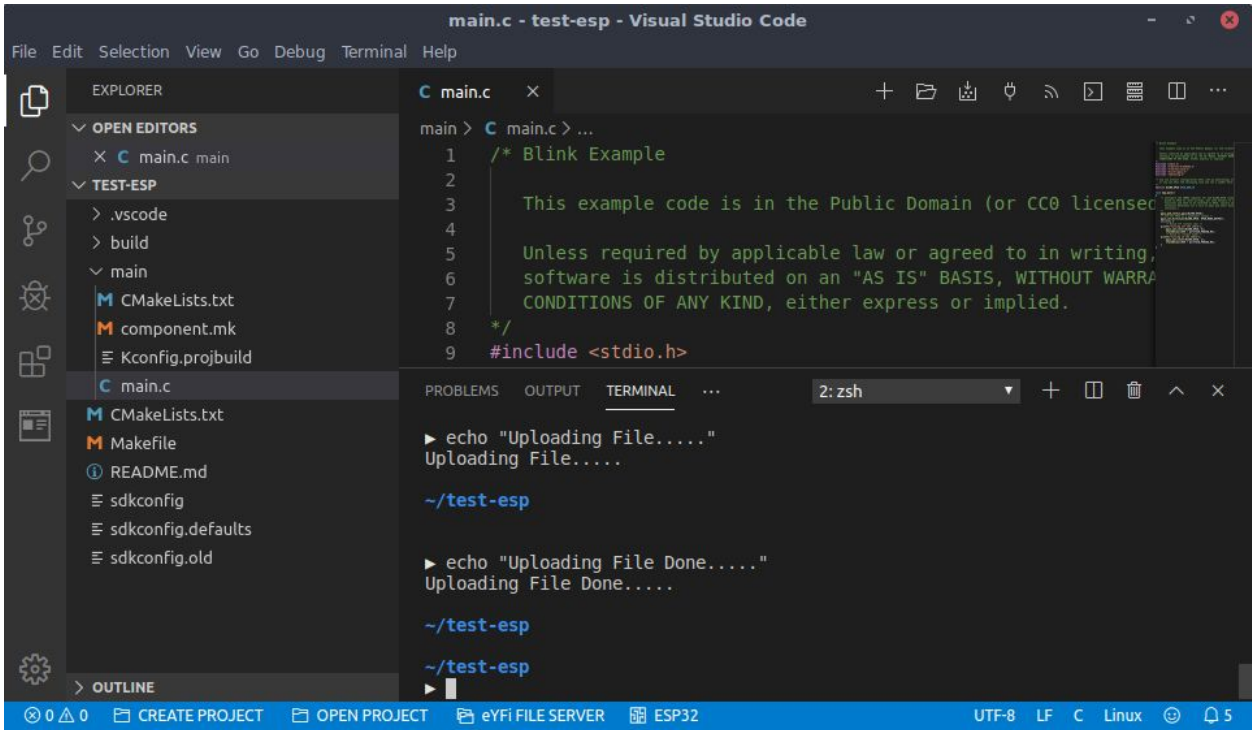
If you are not connected to the Wireless Access Point, the following message will get displayed instead,



Once you are connected to eYFi-Mega's Wireless Access Point you can do Wireless Flashing.

After you press the button first compilation will start and after compilation uploading of the generated bin file will start. When this starts, "**Uploading File**" will get printed in the terminal.

During this time keep an eye on the Wi-Fi Status LED on the board. When the file is getting uploaded to the file server Wi-Fi Status LED will turn OFF. Once the upload is done, the flashing process will start and the Wi-Fi Status LED will start blinking fast. After flashing is done Wi-Fi Status LED will start blinking slowly and "**Uploading File Done**" will get printed in the terminal.



```c
/* Blink Example

   This example code is in the Public Domain (or CC0 licensed

   Unless required by applicable law or agreed to in writing,
   software is distributed on an "AS IS" BASIS, WITHOUT WARRA
   CONDITIONS OF ANY KIND, either express or implied.
*/
#include <stdio.h>
```

```
▶ echo "Uploading File....."
Uploading File.....

~/test-esp


▶ echo "Uploading File Done....."
Uploading File Done.....

~/test-esp

~/test-esp
▶
```

NOTE: For ESP32 you need to switch to user application manually using the S3 switch present on the board once Wireless Flashing is done.

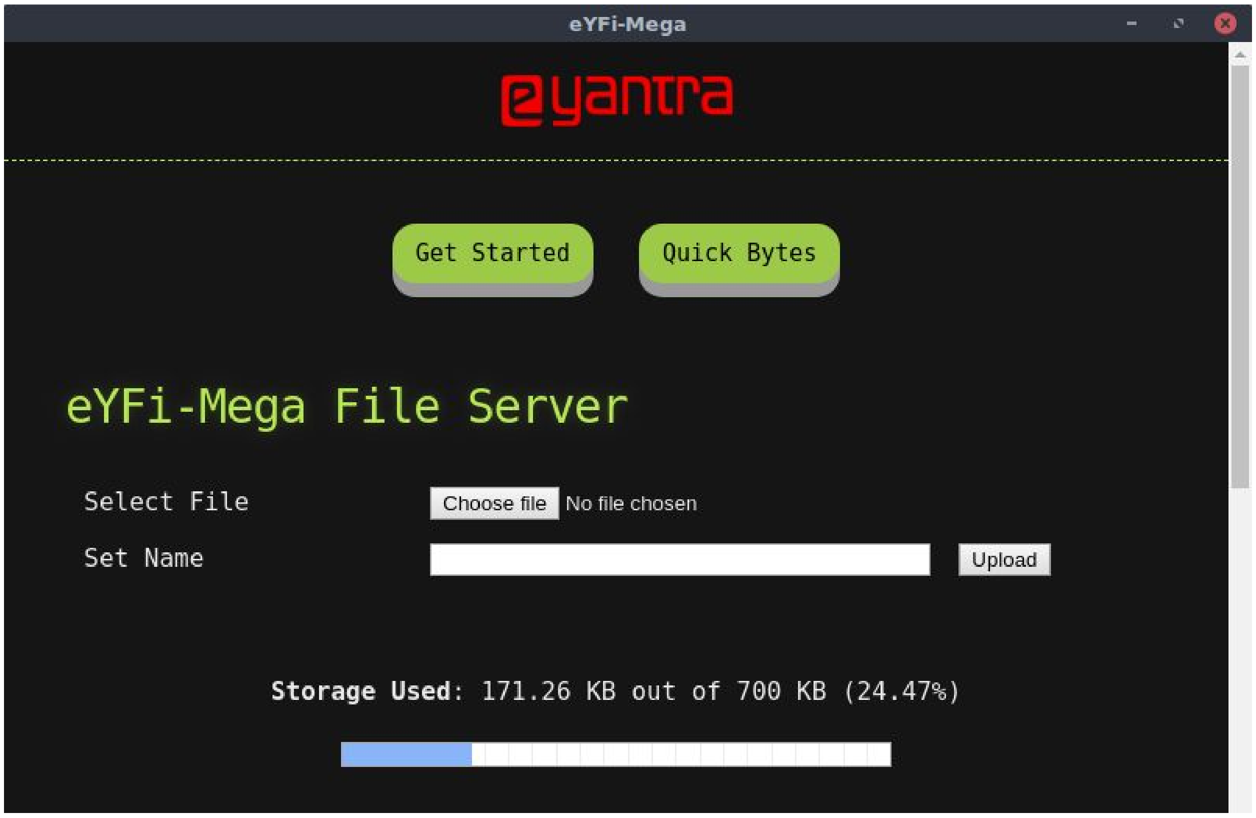
Button – eYFi File Server

eYFi-Mega File Server is like a file explorer for the 700 KB on-board storage. You can use this storage to store any files like you can use this storage to configure the WiFi-SSID and Password of eYFi-Mega's Wireless Access Point (explained in the next section), log data from sensors, you can even store .bin or .hex files of your firmware and flash it in any of the two controllers through this file server.

NOTE: When you are setting the name of your file which is to be uploaded, keep the name of the file including the name of the extension less than 31 characters in length.

Before you press this button make sure that you are connected to eYFi-Mega's Wireless Access Point.

When you press this button, application to access file server hosted by eYFi-Mega will start.



Storage Used: 171.26 KB out of 700 KB (24.47%)

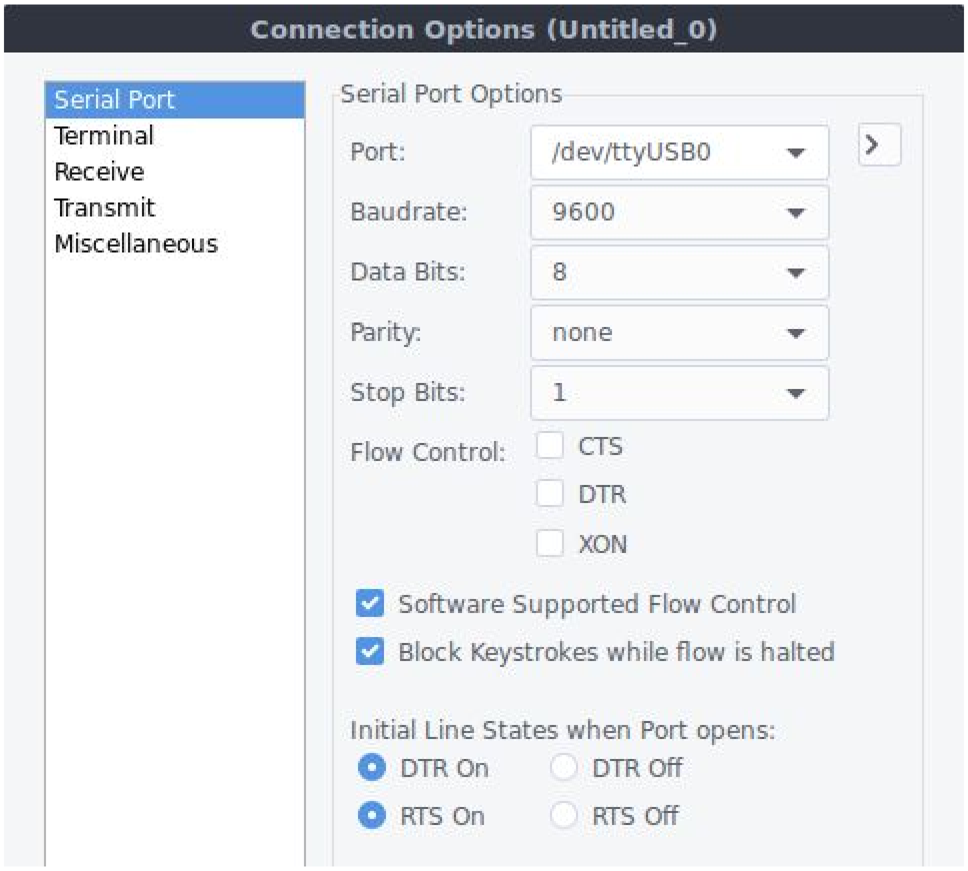
Button - Wired Serial Monitor

This button will open up an instance of CoolTerm.

To start Serial Terminal first press the **Options** button of CoolTerm and select the appropriate port and baud rate.

For ATMega 2560, turn **ON** DTR and RTS.

For ESP32, turn **OFF** DTR and RTS.



Connection Options (Untitled_0)

Serial Port
Terminal
Receive
Transmit
Miscellaneous

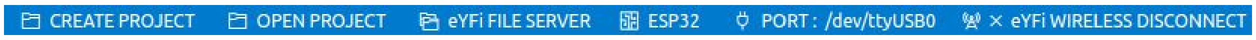
Serial Port Options

Port: /dev/ttyUSB0
Baudrate: 9600
Data Bits: 8
Parity: none
Stop Bits: 1
Flow Control: [ ] CTS [ ] DTR [ ] XON

[✓] Software Supported Flow Control
[✓] Block Keystrokes while flow is halted

Initial Line States when Port opens:
[✓ DTR On] DTR Off
[✓ RTS On] RTS Off

Status Bar



The status bar contains some useful buttons and indicators for the eYFi-Mega board.

Indicators

1. Selected Microcontroller - This will indicate the microcontroller used in the project.
2. eYFi Wireless Connection Status - This will indicate whether your computer is connected to eYFi-Mega's Wireless Access Point or not.

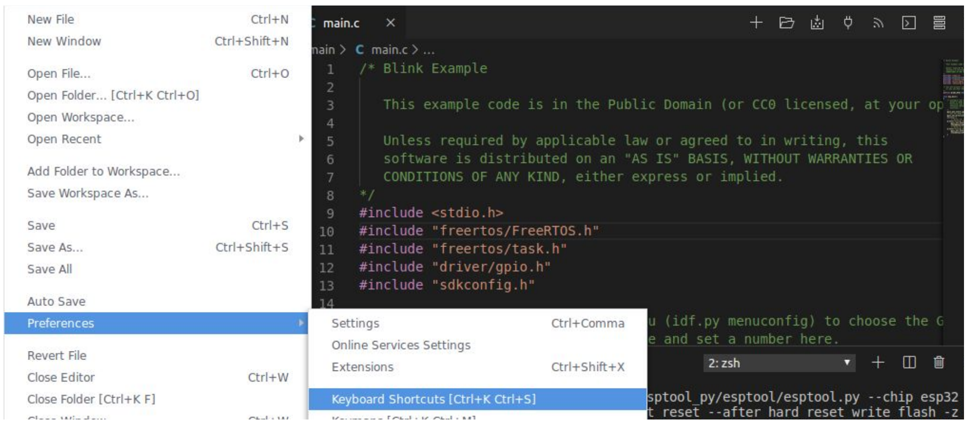
Buttons

1. Create Project - This button can also be used to create project for eYFi-Mega.
2. Open Project - This button can also be used to Open any existing eYFi-Mega Project.
3. eYFi File Server - This button can also be used to access eYFi File Server.

Keyboard Shortcuts

VS Code allows you to set your own keyboard shortcuts for various buttons. To set your own shortcuts for the buttons of eYFi-Mega extension follow the following steps,

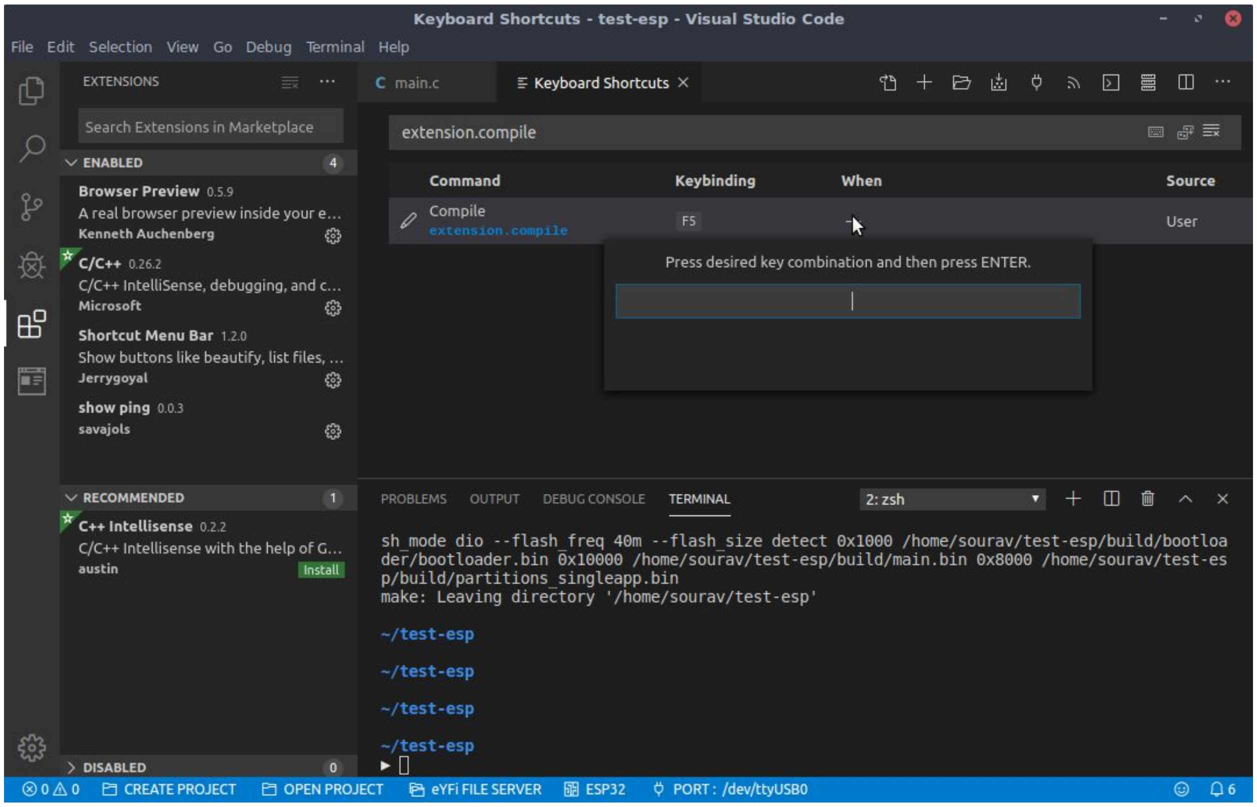
1. First, open keyboard shortcuts settings by going to File -> Preferences -> Keyboard Shortcuts



```
/* Blink Example

   This example code is in the Public Domain (or CC0 licensed, at your op

   Unless required by applicable law or agreed to in writing, this
   software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
   CONDITIONS OF ANY KIND, either express or implied.
*/
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "sdkconfig.h"
```

2. Then in the search bar, search the following things to select eYFi-Mega Extensions buttons
    a. Create Project Button - **extension.create**
    b. Open Project Button - **extension.open**
    c. Build Button - **extension.compile**
    d. Wired Flash Button - **extension.wiredFlash**
    e. Wireless Flash Button - **extension.wirelessFlash**
    f. Wired Serial Monitor Button - **extension.wiredSerial**
    g. eYFi File Server Button - **extension.fileServer**

3. After selecting one button double click in the **When cell** and press the key combination which you want to set as the shortcut for the button.


Keyboard Shortcuts - test-esp - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
EXTENSIONS
Search Extensions in Marketplace
ENABLED 4
Browser Preview 0.5.9
A real browser preview inside your e...
Kenneth Auchenberg
C/C++ 0.26.2
C/C++ IntelliSense, debugging, and c...
Microsoft
Shortcut Menu Bar 1.2.0
Show buttons like beautify, list files, ...
Jerrygoyal
show ping 0.0.3
savajols
RECOMMENDED 1
C++ Intellisense 0.2.2
C/C++ Intellisense with the help of G...
austin
Install
DISABLED 0
main.c
Keyboard Shortcuts
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: zsh
0 0 CREATE PROJECT OPEN PROJECT eYFi FILE SERVER ESP32 PORT : /dev/ttyUSB0 6

extension.compile

Command	Keybinding	When	Source
Compile extension.compile	F5	-	User

Press desired key combination and then press ENTER.

```
sh_mode dio --flash_freq 40m --flash_size detect 0x1000 /home/sourav/test-esp/build/bootloa
der/bootloader.bin 0x10000 /home/sourav/test-esp/build/main.bin 0x8000 /home/sourav/test-es
p/build/partitions_singleapp.bin
make: Leaving directory '/home/sourav/test-esp'

~/test-esp

~/test-esp

~/test-esp

~/test-esp
▶ 
```

Build System

ATMega 2560 Build System

We have developed our own Build System for ATMega 2560 projects. At the core of this build system we have a GNU Make Makefile which ties all the necessary AVR tools together.

This Makefile also allows you to add any third-party library to your project by simply copying the library files in the project folder.

There are few commands for this Makefile which you can use,

1. **make all**: This command will compile your source code and generate binary and hex files.
2. **make clean**: This command will clean your project directory by removing all the object files, binary file and hex file.

ESP32 Build System

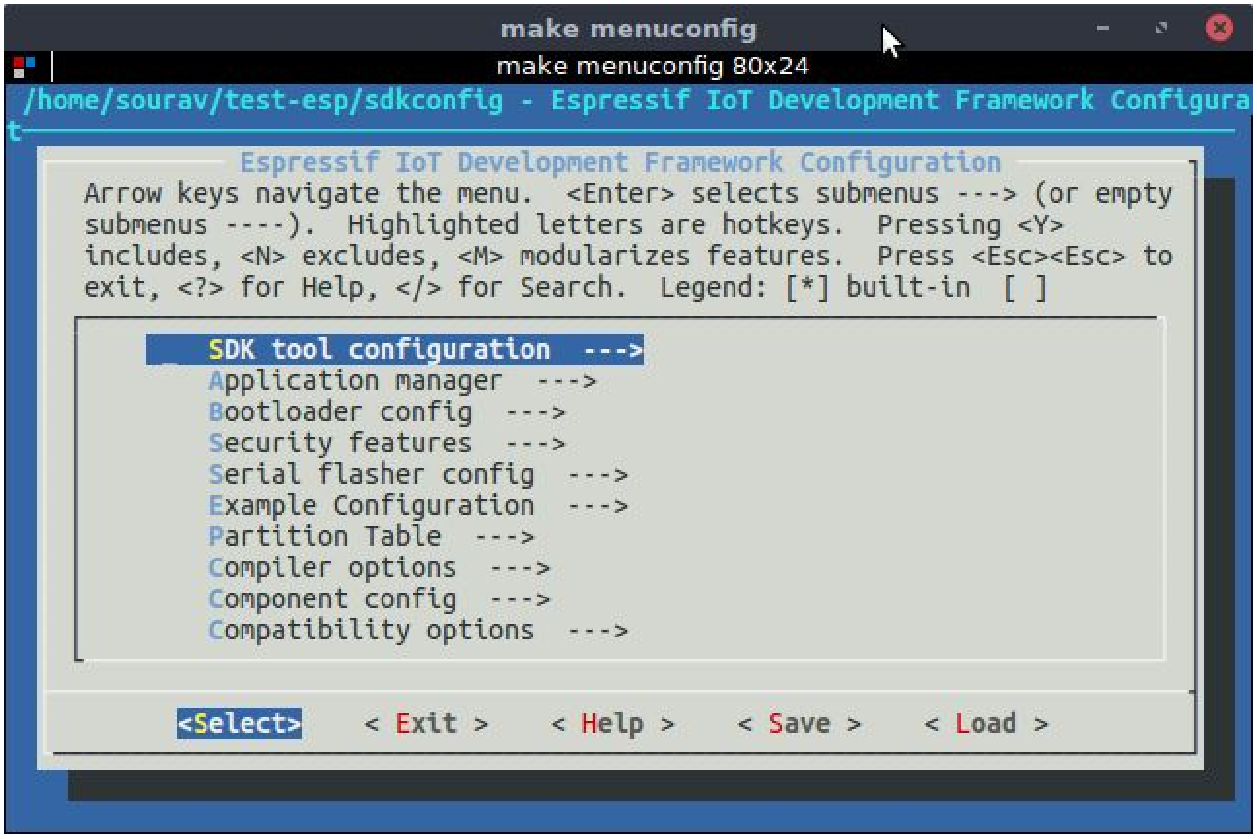
For ESP32 we are using the default Build System developed by Espressif which comes with its own Makefile.

There are several handy commands for this Makefile which you can use,

1. **make all**: This command will compile your source code and generate binary and executable files.
2. **make -j5 all**: The “j5” flag allows you to use all the 5 CPU cores for the build process. This way build process can complete quickly if you use more than one core for the build process. Note: If you have 3 CPU cores use “j3” flag instead.
3. **make clean**: This command will clean your project directory by removing all the object files.
4. **make erase_flash**: This will erase the entire 4 MB flash of ESP32. **NOTE**: Using this command will erase OTA Application, Partition Table and File Storage of eYFi-Mega.
5. **make flash**: This command will flash your application in the factory partition of ESP32. **WARNING**: eYFi-Mega OTA Application resides in

factory partition. If you execute this command OTA Application will get replaced with your application.

6. **make monitor**: This will open a serial monitor for ESP32 in the terminal.
7. **make menuconfig**: This command will open a menu-driven user interface, which will allow you to choose the features of ESP32 that you want to have in your application.

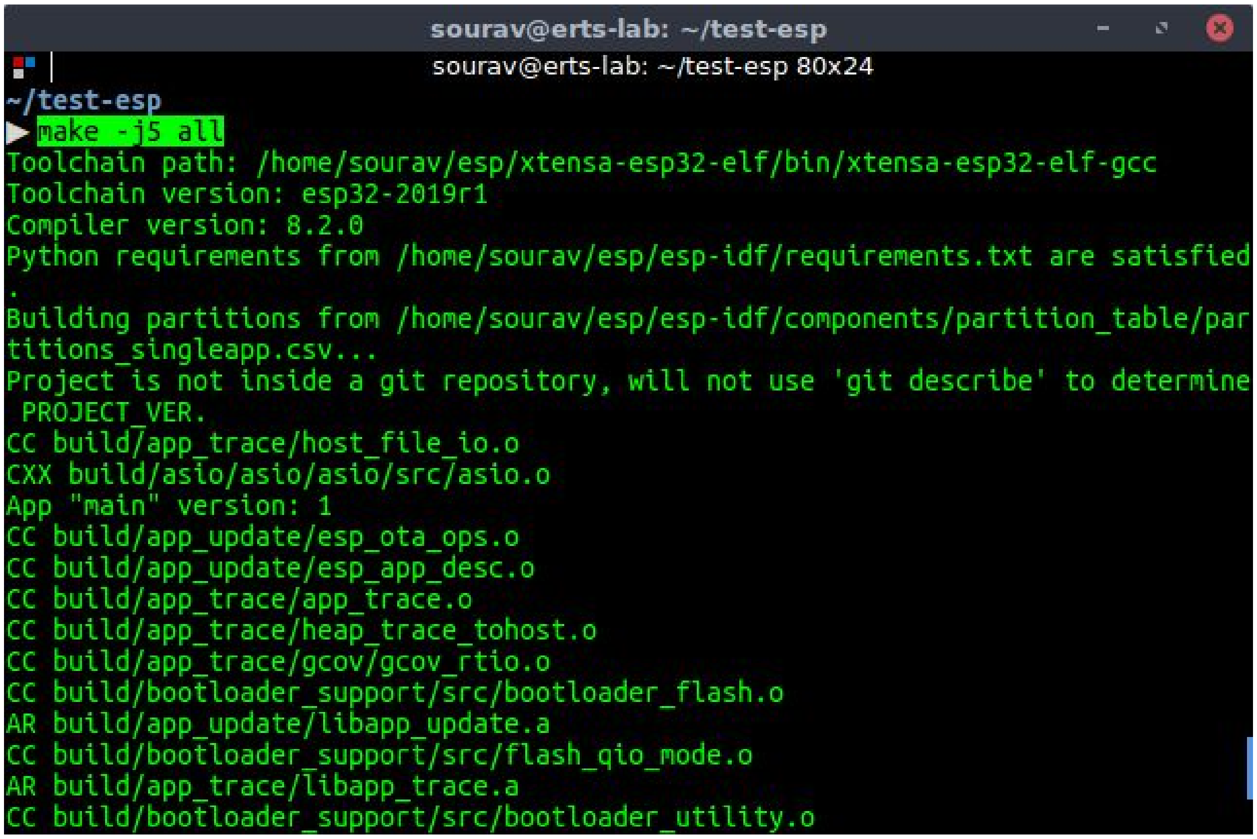


```
make menuconfig
make menuconfig 80x24
 /home/sourav/test-esp/sdkconfig - Espressif IoT Development Framework Configura
t
            Espressif IoT Development Framework Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

            SDK tool configuration  --->
            Application manager  --->
            Bootloader config  --->
            Security features  --->
            Serial flasher config  --->
            Example Configuration  --->
            Partition Table  --->
            Compiler options  --->
            Component config  --->
            Compatibility options  --->

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

Using Makefile Commands

To use Makefile commands navigate to the eYFi-Mega project directory and open the terminal there. Make sure that the Makefile is present in the project directory.

Now in the terminal type the commands which you want to execute.



sourav@erts-lab: ~/test-esp
sourav@erts-lab: ~/test-esp 80x24

```
~/test-esp
▶ make -j5 all
Toolchain path: /home/sourav/esp/xtensa-esp32-elf/bin/xtensa-esp32-elf-gcc
Toolchain version: esp32-2019r1
Compiler version: 8.2.0
Python requirements from /home/sourav/esp/esp-idf/requirements.txt are satisfied
.
Building partitions from /home/sourav/esp/esp-idf/components/partition_table/par
titions_singleapp.csv...
Project is not inside a git repository, will not use 'git describe' to determine
 PROJECT_VER.
CC build/app_trace/host_file_io.o
CXX build/asio/asio/asio/src/asio.o
App "main" version: 1
CC build/app_update/esp_ota_ops.o
CC build/app_update/esp_app_desc.o
CC build/app_trace/app_trace.o
CC build/app_trace/heap_trace_tohost.o
CC build/app_trace/gcov/gcov_rtio.o
CC build/bootloader_support/src/bootloader_flash.o
AR build/app_update/libapp_update.a
CC build/bootloader_support/src/flash_qio_mode.o
AR build/app_trace/libapp_trace.a
CC build/bootloader_support/src/bootloader_utility.o
```

Adding Third-Party Libraries

To add any third-party library to your ATMega 2560 project, simply copy and paste the library source files in the project directory where your main C file exists. For ESP32, copy and paste the library source files in the main folder where your main C file exists. The Makefiles will take care of compiling and linking your library with your main source file.

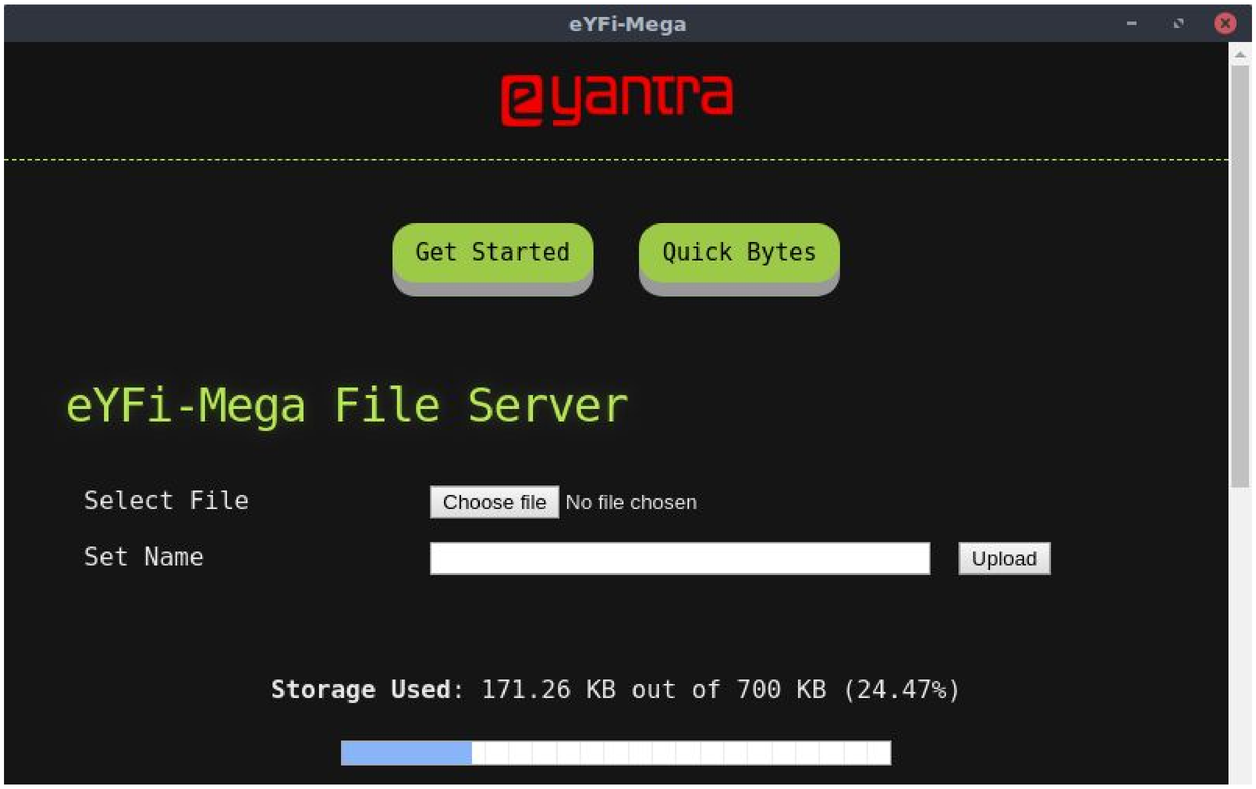
eYFi-Mega OTA Application

The eYFi-Mega OTA Application which resides in the factory partition of ESP32, allows user to flash the firmware of ATMega 2560 or ESP32 wirelessly and gives access to the file storage of eYFi-Mega.

eYFi-Mega File Server

eYFi-Mega File Server is like a file explorer for the 700 KB onboard storage. To access this file server you need to first connect to eYFi-Mega's Wireless Access Point. After connecting to the Access Point you can use VS Code with eYFi-Mega Extension to open the file server window. You can use this storage to store any files like you can use this storage to configure your WiFi-SSID and password (explained in the next section), log data from sensors, you can even store .bin or .hex files of your firmware and flash it in any of the two controllers through this file server.

NOTE: When you are setting the name of your file which is to be uploaded, keep the name of the file including the name of the extension less than 31 characters in length.



eYFi-Mega File Server

Select File No file chosen

Set Name

Storage Used: 171.26 KB out of 700 KB (24.47%)

Set Wi-Fi SSID and Password

To set your own Wi-Fi SSID and Password of the board, upload a config.txt file to your eYFi-Mega File Server with your SSID and Password in the following format in the file. After the upload is done, reset the ESP32 by pressing the ESP_RESET button, and you are all set!

Filename: **config.txt**

```
WiFi-SSID [your-wifi-ssid]

WiFi-Pass [your-wifi-pass]
```

For example,

```
WiFi-SSID [eYFi-Mega]

WiFi-Pass [eyantra123]
```

will set the SSID to "eYFi-Mega" and Password to "eyantra123".

Default SSID and Password of eYFi-Mega Wireless Access Point

Wi-Fi SSID:	**eYFi-Mega**
Wi-Fi Password:	**eyantra123**

In case the config.txt file is missing from your file server or the format of the config.txt file is not proper then, this default SSID and Password will be set for your Wireless Access Point.

Wi-Fi Status LED Indication

When you are connected to your eYFi-Mega over a Wi-Fi link, it's important to know the different states in which your board is. These states are indicated by the Blue Wi-Fi Status LED, which is right next to Red ESP_ON LED. Below is the description of different LED Patterns associated with different states of the board.

State	LED Pattern
Wi-Fi Client is connected to ESP32	ON
Wi-Fi Client is disconnected from ESP32	OFF
File Upload Start	OFF
File Upload End	Blink Fast for 100 ms
Firmware Flash Start	Blink Fast
Firmware Flash End for AVR	Blink Slow for 5 s
Firmware Flash End for ESP32	Blink Slow till application Switch #1 (S1) is toggled

eYFi-Mega Wireless Serial Terminal Application

We have developed an ESP32 application which allows ATMega 2560 to transmit its data on UART #0 over Wi-Fi. So with the help of this application, you can receive serial data over Wi-Fi at your side.

To use this first, you would have to flash this application which you can download from eYFi-Mega’s website in the user-app partition of ESP32.

After that make sure the S2 switch is towards the Wi-Fi symbol. This will connect UART#0 of ATMega 2560 with UART#1 of ESP32.

To start this application switch S1 away from the Wi-Fi symbol and then press ESP_RESET button, to start the application.

In your ATMega 2560 application make sure that you are sending data to UART #0 at a baud rate of 115200.

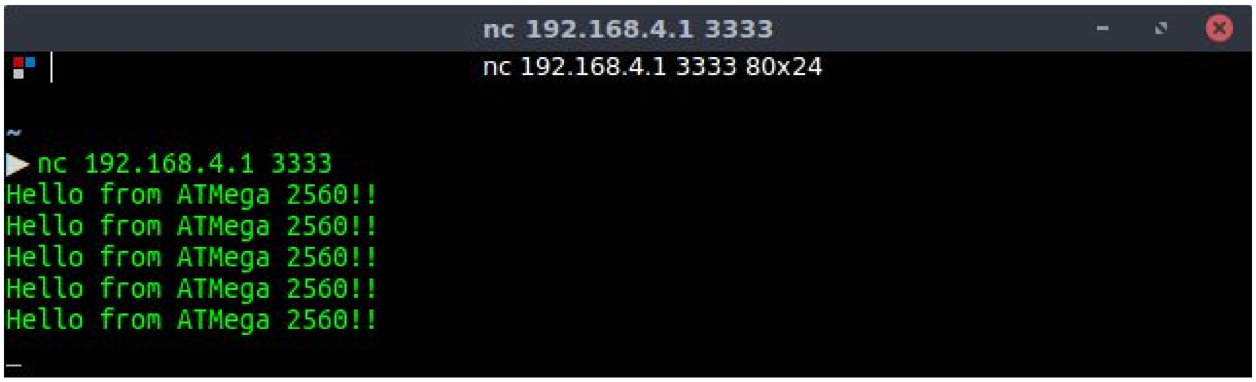
To receive data on your Linux system you can use **netcat** utility. Run the following command in the terminal to receive data from the UART #0 of ATMega 2560 over Wi-Fi,

```
nc 192.168.4.1 3333
```

You can also run this command on your Android Device. To run terminal commands on your Android Device use **Termux App** which you can download from Google Play Store. Termux App Download URL

Before you run this command, make sure that you are connected to the Wireless Access Point of eYFi-Mega. For this application the Wi-Fi SSID is **eYFi-Wireless-Serial** and there is no password.

In the Quick Bytes page of eYFi-Mega which you can find at eYFi-Mega’s website, you can find the example code for using this Wireless Serial Terminal.



```
~
▶ nc 192.168.4.1 3333
Hello from ATMega 2560!!
Hello from ATMega 2560!!
Hello from ATMega 2560!!
Hello from ATMega 2560!!
Hello from ATMega 2560!!
_
```

eYFi-Mega ESP32 Partition Table

We have partitioned the 4 MB flash of onboard ESP32 to accommodate 1 MB OTA App, 2 MB for User App and 700 KB for File Storage. This partition table can be modified by you as per your requirement. For example, if you want to increase or decrease the size of the 700 KB File Storage you can update the partition table. If you decide to modify the partition table for your project, you might not be able to use the OTA App because our OTA App is compatible with the partition table described below. In case you accidentally erase the partition you can always download our Partition Table and our OTA App from our website. The current partition table is described below,

Partition Name	Start Address	Size
reserved area (nvs, otadata, phy_init)	-	300 KB
factory (ota-app)	0x10000	1 MB
app1 (user-app)	0x110000	2 MB
storage (spiffs)	0x310000	700 KB

Flashing ESP32 Bootloader, Partition Table, OTA Application, Wireless Serial Terminal Application and ESP32 User Application in eYFi-Mega

In case you accidentally erase any of the important firmware which is required for eYFi-Mega to function properly, you can always download them from our website and flash them again. In this section, we will discuss how you can do that.

To flash firmware in ESP32 you would need **esptool**.

The **esptool** and the firmware described in the following sections can be downloaded from eYFi-Mega’s website which can be found at e-Yantra’s products page at https://e-yantra.org/products.

Flashing eYFi-Mega ESP32 Bootloader

Copy the bootloader file in the esptool folder and execute the following command to flash the bootloader

```
$ python2 esptool.py -b 921600 --port /dev/ttyUSB0 write_flash 0x1000
eyfi-mega_esp32_bootloader.bin
```

Flashing eYFi-Mega ESP32 Partition Table

Copy the partition table file in the esptool folder and execute the following command to flash the partition table.

```
$ python2 esptool.py -b 921600 --port /dev/ttyUSB0 write_flash 0x8000
eyfi-mega_esp32_partitions.bin
```

Flashing eYFi-Mega ESP32 OTA Application

Copy the OTA Application file in the esptool folder and execute the following command to flash the OTA Application

```
$ python2 esptool.py -b 921600 --port /dev/ttyUSB0 write_flash 0x10000
eyfi-mega_esp32_ota_app.bin
```

Flashing eYFi-Mega ESP32 User Application and Wireless Serial Terminal Application

Copy the Wireless Serial Terminal Application or User Application file in the esptool folder and execute the following command to flash the file

```
$ python2 esptool.py -b 921600 --port /dev/ttyUSB0 write_flash 0x110000
eyfi-mega_esp32_user_app.bin
```

APPENDIX

APPENDIX A: Kill Any Command

While any command is being executed either in your terminal or in VS Code's terminal and you want to kill it, just press CTRL+C in the terminal.

APPENDIX B: Unix and Blank Spaces

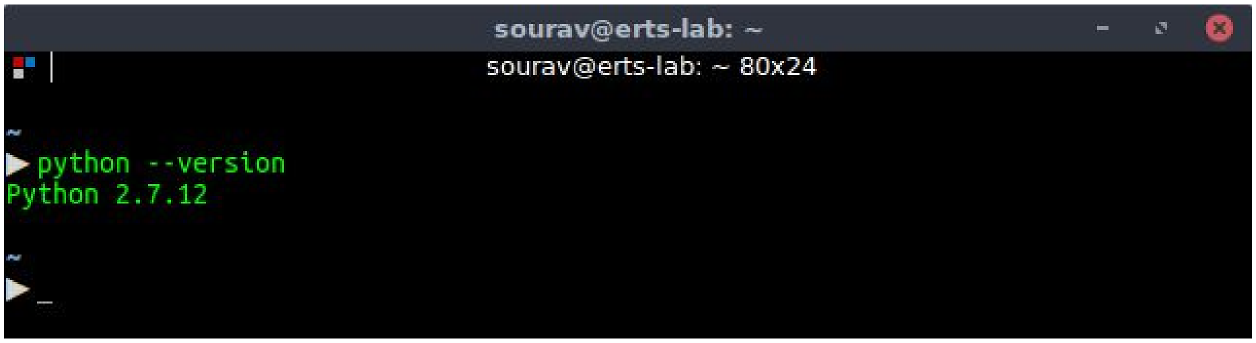
Unix systems are not good in handling blank spaces so, we recommend not to have any blank spaces in your folder name or filename. Use camelCase or snake_case instead of having a blank space between two words.

APPENDIX C: Set Python 2 as your default Python Interpreter

It might be possible that Python 3 is your default Python interpreter in your environment. If that is the case some of the python commands used by the ESP32 build system and eYFi-Mega VS Code Extension might not work because they use Python 2.

To check your default python interpreter version enter the following command in your terminal,

```
$ python --version
```



```
~
▶ python --version
Python 2.7.12

~
▶ _
```

If your Python version is 2.x then you are good to go. But, if it is 3.x then you would have to set Python 2 as your default interpreter for your environment. To do so, follow the following steps,

If you are using BASH shell, execute the following command

```
$ sudo gedit ~/.bashrc
```

If you are using ZSH shell, execute the following command

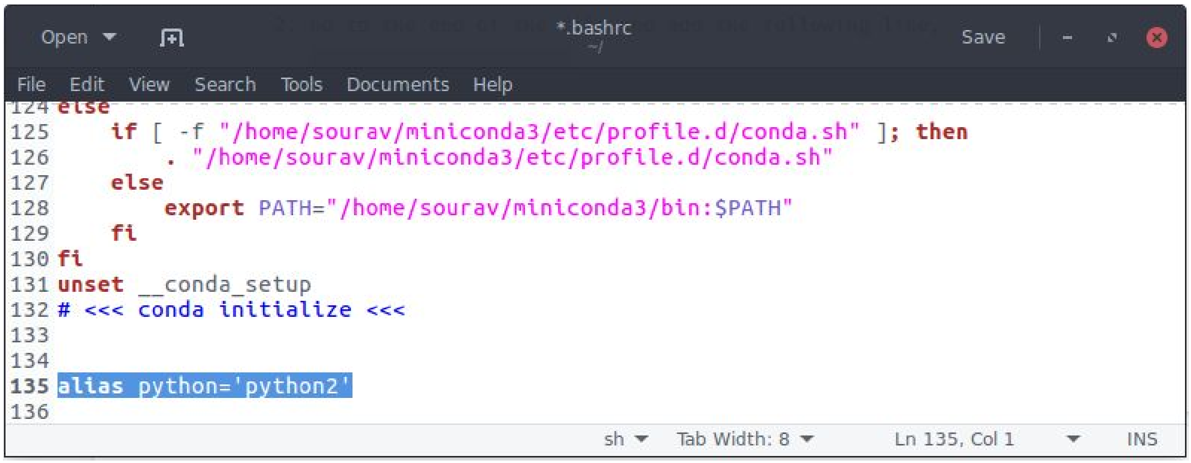
```
$ sudo gedit ~/.zshrc
```

This command will open up the shell settings file in gedit text editor.

NOTE: Most of the major Linux Distributions like Ubuntu uses BASH Shell.

Go to the end of the file and add the following line,

```
alias python='python2'
```

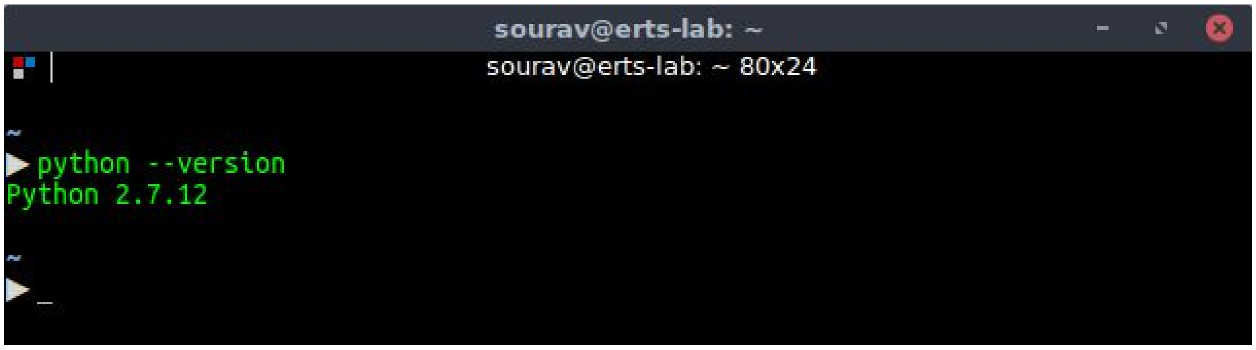


```
else
    if [ -f "/home/sourav/miniconda3/etc/profile.d/conda.sh" ]; then
        . "/home/sourav/miniconda3/etc/profile.d/conda.sh"
    else
        export PATH="/home/sourav/miniconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<


alias python='python2'

```

Make sure to save and then exit the text editor.

Close the terminal and open it again and enter the following command,

```
$ python --version
```



```
~
▶ python --version
Python 2.7.12

~
▶ _
```

Now Python 2 will be set as you default python interpreter.

****