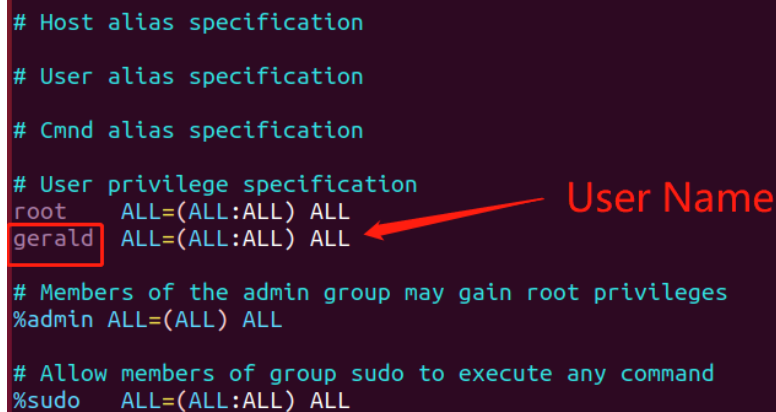


Building STM32 development environment in Linux system

(Using Ubuntu 18.04 + Arm GNU Toolchain 12 + VS Code)

1. Add normal user to root:
 - (1) `$ sudo vim /etc/sudoers`
 - (2) Insert the command:



```
# Host alias specification

# User alias specification

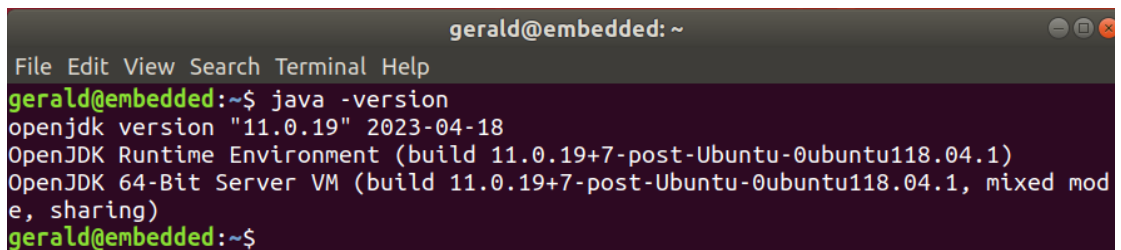
# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
gerald  ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

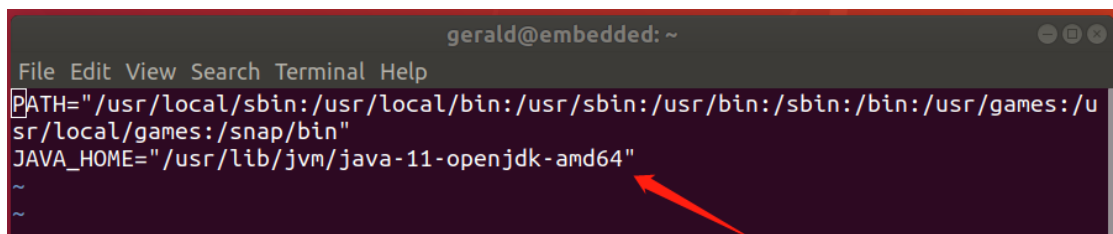
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

2. Installing Java
 - (1) Installation command:
`$ sudo apt install openjdk-11-jdk`
or `$ sudo apt install openjdk-8-jdk` (for other versions)
 - (2) verify installation:
`$ java -version`



```
gerald@embedded: ~
File Edit View Search Terminal Help
gerald@embedded:~$ java -version
openjdk version "11.0.19" 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1, mixed mode, sharing)
gerald@embedded:~$
```

- (3) Add it to environment variable (optional):
`$ sudo vim /etc/environment`
add the java installation path: `/usr/lib/jvm/java-11-openjdk-amd64`



```
gerald@embedded: ~
File Edit View Search Terminal Help
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
~
~
```

- (4) Enable the change:
`$ source /etc/environment`

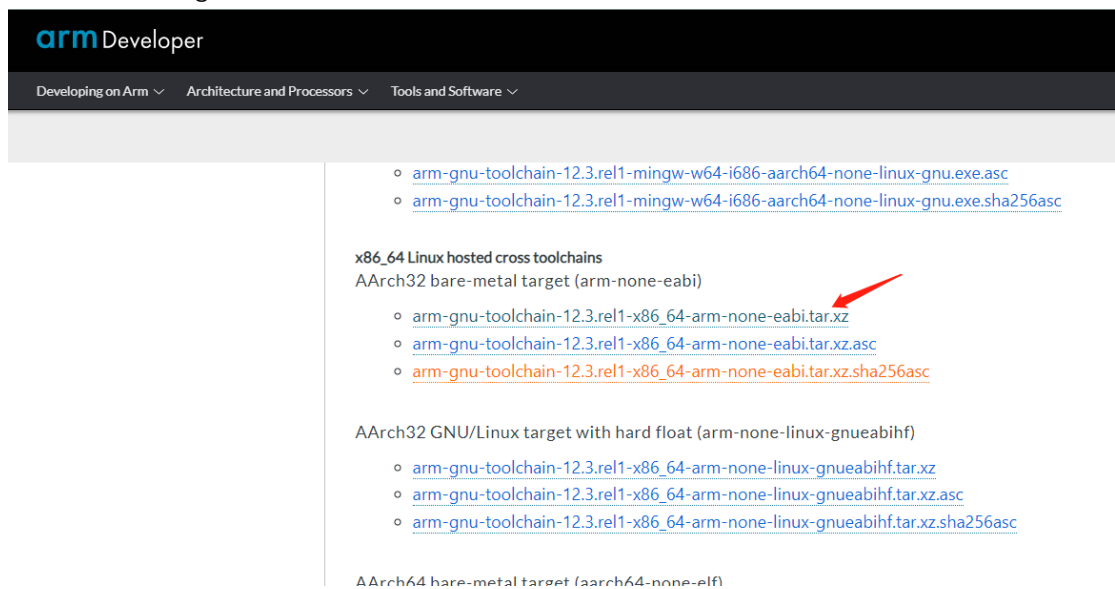
(5) verify the change:

```
$ echo $JAVA_HOME
```

```
gerald@embedded:~$ source /etc/environment
gerald@embedded:~$ echo $JAVA_HOME
/usr/lib/jvm/java-11-openjdk-amd64
gerald@embedded:~$
```

3. Install the GCC-Arm:

(1) Download the gcc-arm toolchain:



The screenshot shows the ARM Developer website. Under the 'Tools and Software' tab, there are links for downloading the GCC-Arm toolchain. A red arrow points to the link for the x86_64 Linux hosted cross toolchains, specifically the arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz file.

arm Developer

Developing on Arm ▾ Architecture and Processors ▾ Tools and Software ▾

- [arm-gnu-toolchain-12.3.rel1-mingw-w64-i686-aarch64-none-linux-gnu.exe.asc](#)
- [arm-gnu-toolchain-12.3.rel1-mingw-w64-i686-aarch64-none-linux-gnu.exe.sha256asc](#)

x86_64 Linux hosted cross toolchains

AArch32 bare-metal target (arm-none-eabi)

- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz](#)
- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz.asc](#)
- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz.sha256asc](#)

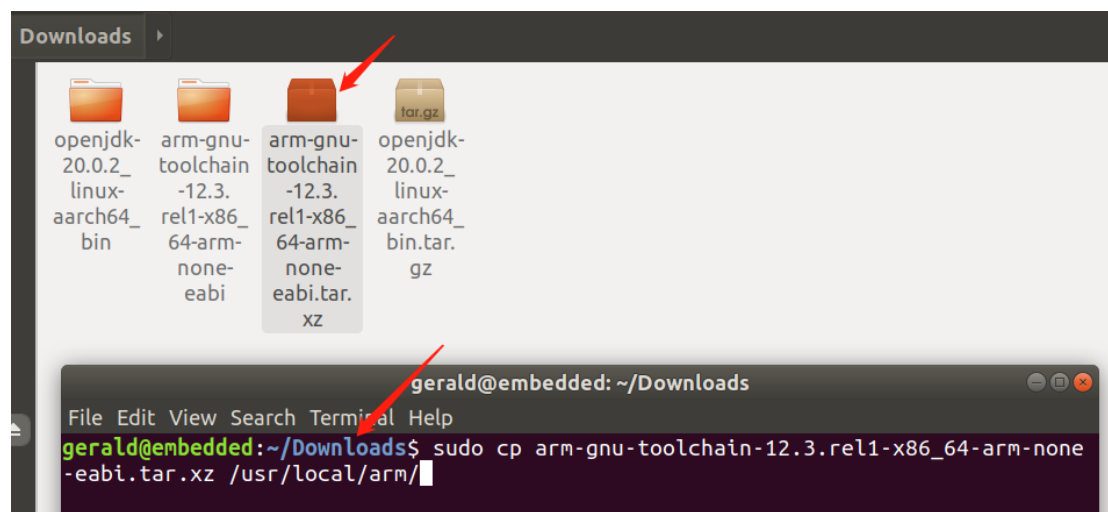
AArch32 GNU/Linux target with hard float (arm-none-linux-gnueabi)

- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-linux-gnueabi.tar.xz](#)
- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-linux-gnueabi.tar.xz.asc](#)
- [arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-linux-gnueabi.tar.xz.sha256asc](#)

AArch64 bare-metal target (aarch64-none-elf)

(2) Copy the package to /usr/local/arm:

```
$ sudo cp arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz /usr/local/arm/
```



The screenshot shows a file manager window with a 'Downloads' folder. Inside, there are four files: 'openjdk-20.0.2-linux-aarch64-bin', 'arm-gnu-toolchain-12.3-rel1-x86_64-arm-none-eabi', 'arm-gnu-toolchain-12.3-rel1-x86_64-arm-none-eabi.tar.xz', and 'openjdk-20.0.2-linux-aarch64-bin.tar.gz'. A red arrow points to the 'arm-gnu-toolchain-12.3-rel1-x86_64-arm-none-eabi.tar.xz' file. Below the file manager, a terminal window shows the command: `gerald@embedded: ~/Downloads$ sudo cp arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz /usr/local/arm/`

(3) Extract the package:

```
$ sudo tar -xvf arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi.tar.xz
```

(4) Add path to the environment:

```
$ sudo vim /etc/profile
```

```
gerald@embedded: /usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi
File Edit View Search Terminal Help
if [ -f /etc/bash.bashrc ]; then
    . /etc/bash.bashrc
fi
else
    if [ "`id -u`" -eq 0 ]; then
        PS1='# '
    else
        PS1='$ '
    fi
fi
fi

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi

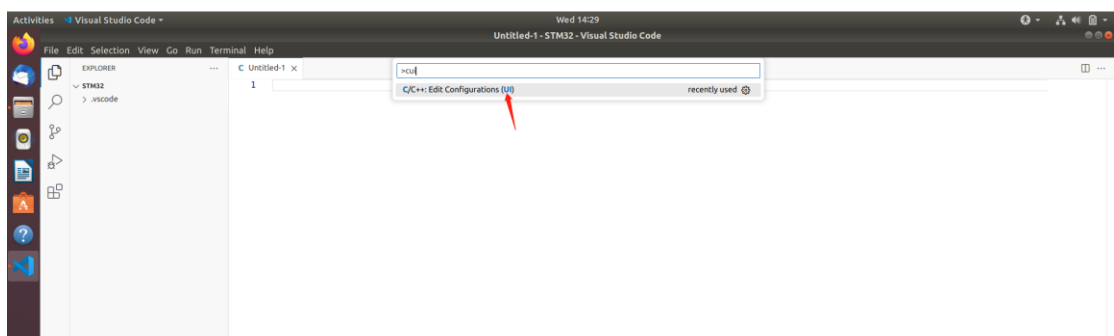
export PATH=$PATH:/usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi/bin

29,1 Bot
```

(5) After restart, verify the change:

```
gerald@embedded: ~
File Edit View Search Terminal Help
gerald@embedded:~$ arm-none-eabi-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=/usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi/bin/./libexec/gcc/arm-none-eabi/12.3.1/lto-wrapper
Target: arm-none-eabi
Configured with: /data/jenkins/workspace/GNU-toolchain/arm-12/src/gcc/configure --target=arm-none-eabi --prefix=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/install --with-gmp=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/host-tools --with-mpfr=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/host-tools --with-mpc=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/host-tools --with-isl=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/host-tools --disable-shared --disable-nls --disable-threads --disable-tls --enable-checking=release --enable-languages=c,c++,fortran --with-newlib --with-gnu-as --with-gnu-ld --with-sysroot=/data/jenkins/workspace/GNU-toolchain/arm-12/build-arm-none-eabi/install/arm-none-eabi --with-multilib-list=aprofile,rmprofile --with-pkgversion='Arm GNU Toolchain 12.3.Rel1 (Build arm-12.35)' --with-bugurl=https://bugs.linaro.org/
Thread model: single
Supported LTO compression algorithms: zlib
gcc version 12.3.1 20230626 (Arm GNU Toolchain 12.3.Rel1 (Build arm-12.35))
```

(6) Set VSCode:



find GCC installation path:

```
gerald@embedded: ~  
File Edit View Search Terminal Help  
gerald@embedded:~$ whereis arm-none-eabi-gcc  
arm-none-eabi-gcc: /usr/local/arm/arm-gnu-toolchain-12.3.rel1-x86_64-arm-none-eabi/bin/arm-none-eabi-gcc  
gerald@embedded:~$
```

IntelliSense Configurations

Use this editor to edit IntelliSense settings defined in the underlying `c_cpp_properties.json` file. Changes made in this editor only apply to the selected configuration. To edit multiple configurations at once go to `c_cpp_properties.json`.

Configuration name
A friendly name that identifies a configuration. `Linux`, `Mac`, and `Win32` are special identifiers for configurations that will be auto-selected on those platforms.
Select a configuration set to edit.
Linux

Compiler path
The full path to the compiler you use to build your project, e.g. `/usr/bin/gcc`, to enable more accurate IntelliSense. The extension will query the compiler to determine the system include paths and default defines to use for IntelliSense.
Specify a compiler path or select a detected compiler path from the drop-down list.

4. Install the OpenOCD:

(1) Use git to download the openocd:

```
$ git clone https://github.com/openocd-org/openocd.git
```

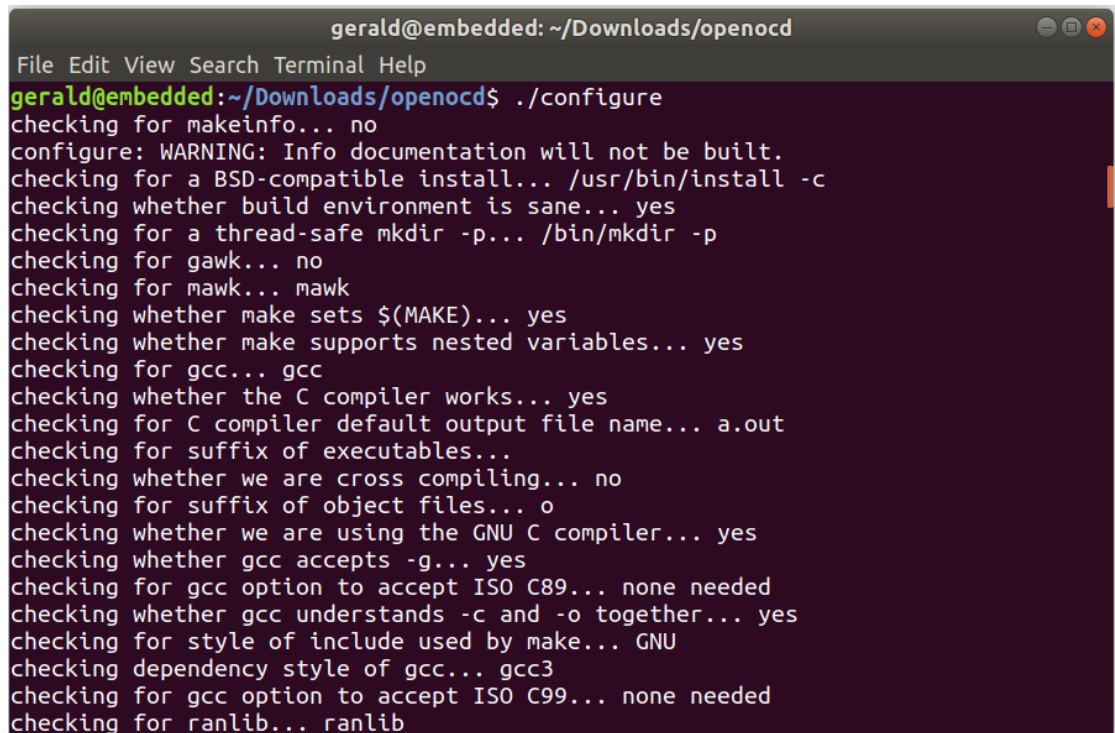
(2) Download the essential library:

```
$ sudo apt install build-essential pkg-config autoconf automake libtool libusb-dev libusb-1.0-0-dev libhidapi-dev libtool libsysfs-dev
```

(3) Open the OpenOCD directory and enter `$ sudo ./bootstrap`

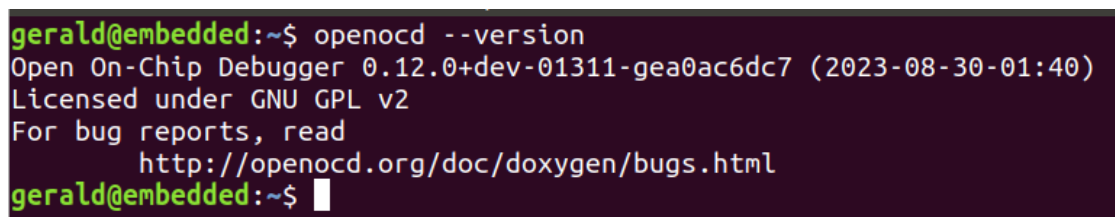
```
gerald@embedded: ~/Downloads/openocd  
File Edit View Search Terminal Help  
gerald@embedded:~/Downloads/openocd$ ./bootstrap  
+ aclocal --warnings=all  
+ libtoolize --automake --copy  
+ autoconf --warnings=all  
+ autoheader --warnings=all  
+ automake --warnings=all --gnu --add-missing --copy  
configure.ac:24: installing 'build-aux/compile'  
configure.ac:42: installing 'build-aux/config.guess'  
configure.ac:42: installing 'build-aux/config.sub'  
configure.ac:19: installing 'build-aux/install-sh'  
configure.ac:19: installing 'build-aux/missing'  
Makefile.am: installing './INSTALL'  
Makefile.am: installing 'build-aux/depcomp'  
Makefile.am:26: installing 'build-aux/mdate-sh'  
Makefile.am:26: installing 'build-aux/texinfo.tex'  
Setting up submodules  
Submodule 'jimtcl' (https://github.com/mstevieb/jimtcl.git) registered for path 'jimtc  
Submodule 'src/jtag/drivers/libjaylink' (https://gitlab.zapb.de/libjaylink/libja  
Submodule 'tools/git2cl' \(https://git.savannah.nongnu.org/git/git2cl.git\) regist  
Cloning into '/home/gerald/Downloads/openocd/jimtcl'
```

- (4) Enter the `$ sudo ./configure`



```
gerald@embedded: ~/Downloads/openocd
File Edit View Search Terminal Help
gerald@embedded:~/Downloads/openocd$ ./configure
checking for makeinfo... no
configure: WARNING: Info documentation will not be built.
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether gcc understands -c and -o together... yes
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking for gcc option to accept ISO C99... none needed
checking for ranlib... ranlib
```

- (5) Enter `$ sudo make` and then enter `$ sudo make install` to finish the installation.
(6) Verify the installation:



```
gerald@embedded:~$ openocd --version
Open On-Chip Debugger 0.12.0+dev-01311-gea0ac6dc7 (2023-08-30-01:40)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
gerald@embedded:~$
```

5. Install STM32CubeMX

- (1) After unzipping the package, set the installation file as an executable file:
`$ chmod 777 SetupSTM32CubeMX-6.9.1`



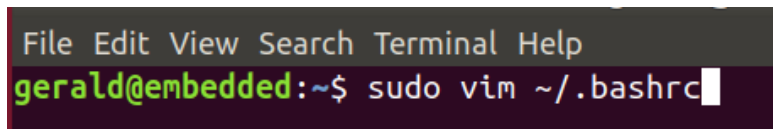
The screenshot shows a file manager window with the path `en.stm32cubemx-lin-v6-9-1`. It contains three items: a folder named `jre`, a file named `Readme.html`, and a file named `SetupSTM32CubeMX-6.9.1`. Below the file manager is a terminal window with the same path. The terminal shows the command `chmod 777 SetupSTM32CubeMX-6.9.1` being executed, and a red arrow points to the file `SetupSTM32CubeMX-6.9.1` in the file manager.

```
Downloads en.stm32cubemx-lin-v6-9-1
jre Readme.html SetupSTM32CubeMX-6.9.1
gerald@embedded: ~/Downloads/en.stm32cubemx-lin-v6-9-1
File Edit View Search Terminal Help
gerald@embedded:~/Downloads/en.stm32cubemx-lin-v6-9-1$ chmod 777 SetupSTM32CubeMX-6.9.1
gerald@embedded:~/Downloads/en.stm32cubemx-lin-v6-9-1$
```

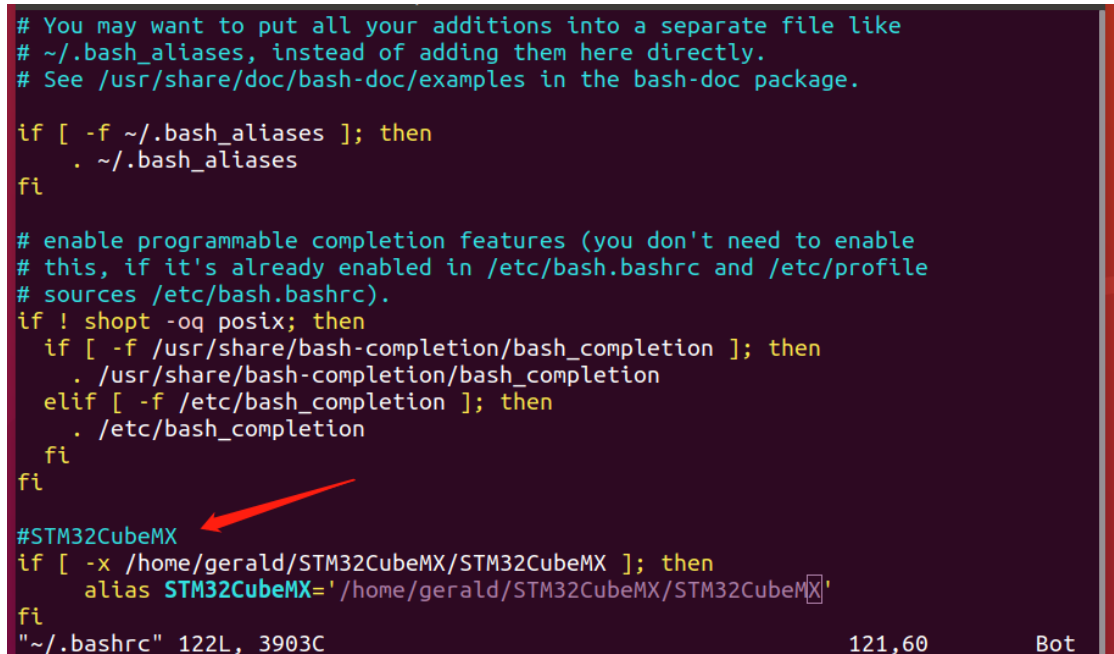
- (2) Run the installation application:
`$./SetupSTM32CubeMX-6.9.1`

- (3) After finishing the installation, add STM32CubeMX path to environment variable:

```
$ sudo vim ~/.bashrc
```



And then adding this command to the end of file:



```
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

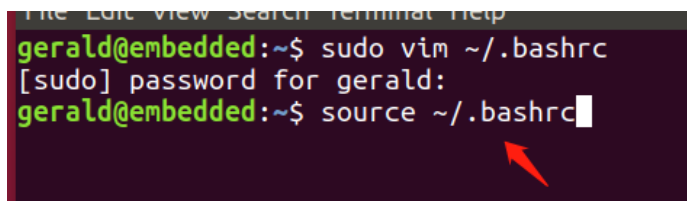
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

#STM32CubeMX
if [ -x /home/gerald/STM32CubeMX/STM32CubeMX ]; then
    alias STM32CubeMX='/home/gerald/STM32CubeMX/STM32CubeMX'
fi

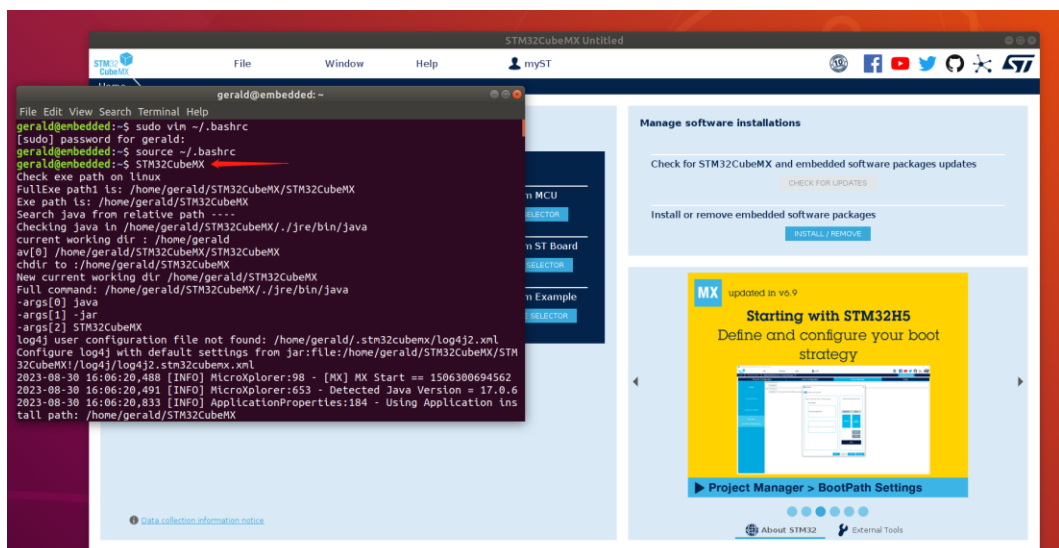
"~/.bashrc" 122L, 3903C 121,60 Bot
```

Save and quit Vim. Next enable the change:

```
$ source ~/.bashrc
```

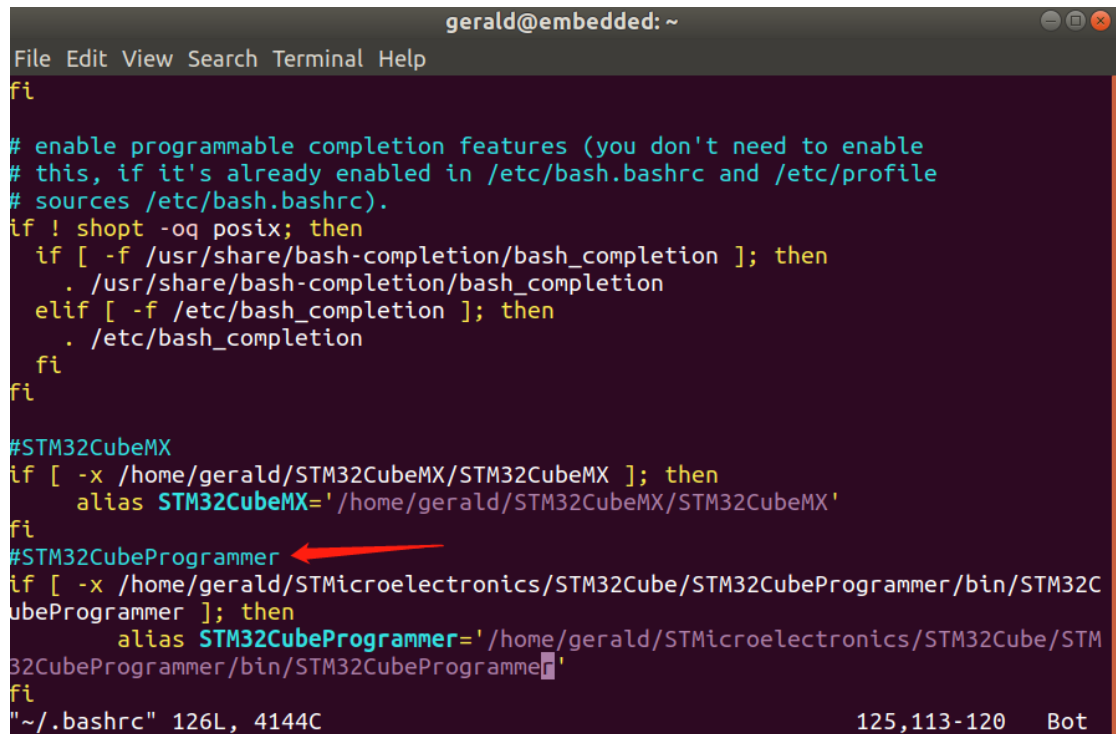


- (4) Verify the Change:



6. Add STM32CubeProgrammer to the environment path:

\$ `sudo vim ~/.bashrc` and then \$ `source ~/.bashrc`



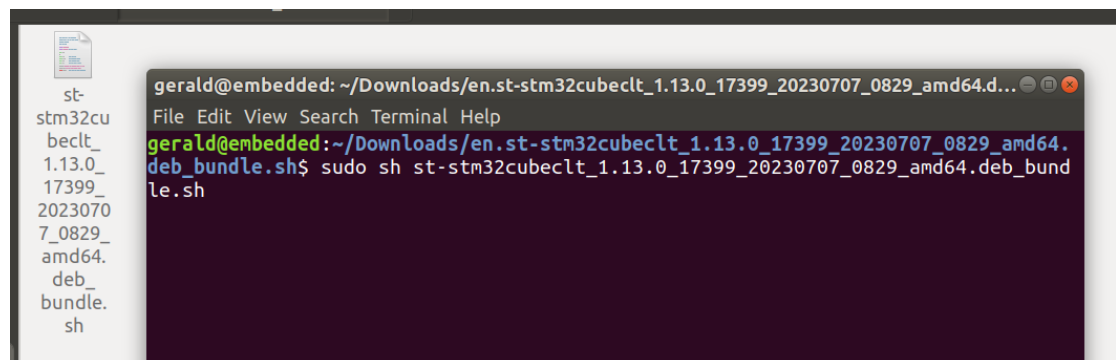
```
gerald@embedded: ~
File Edit View Search Terminal Help
fi
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#STM32CubeMX
if [ -x /home/gerald/STM32CubeMX/STM32CubeMX ]; then
  alias STM32CubeMX='/home/gerald/STM32CubeMX/STM32CubeMX'
fi

#STM32CubeProgrammer
if [ -x /home/gerald/STMicroelectronics/STM32Cube/STM32CubeProgrammer/bin/STM32C
  alias STM32CubeProgrammer='/home/gerald/STMicroelectronics/STM32Cube/STM
  32CubeProgrammer/bin/STM32CubeProgramme'
fi

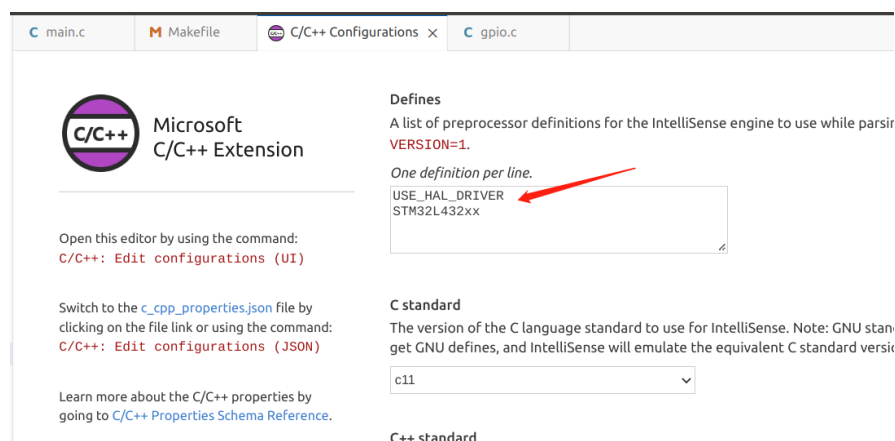
"~/.bashrc" 126L, 4144C 125,113-120 Bot
```

7. Install the STM32CubeCLT:



8. VSCode and Makefile setting:

(1) VSCode

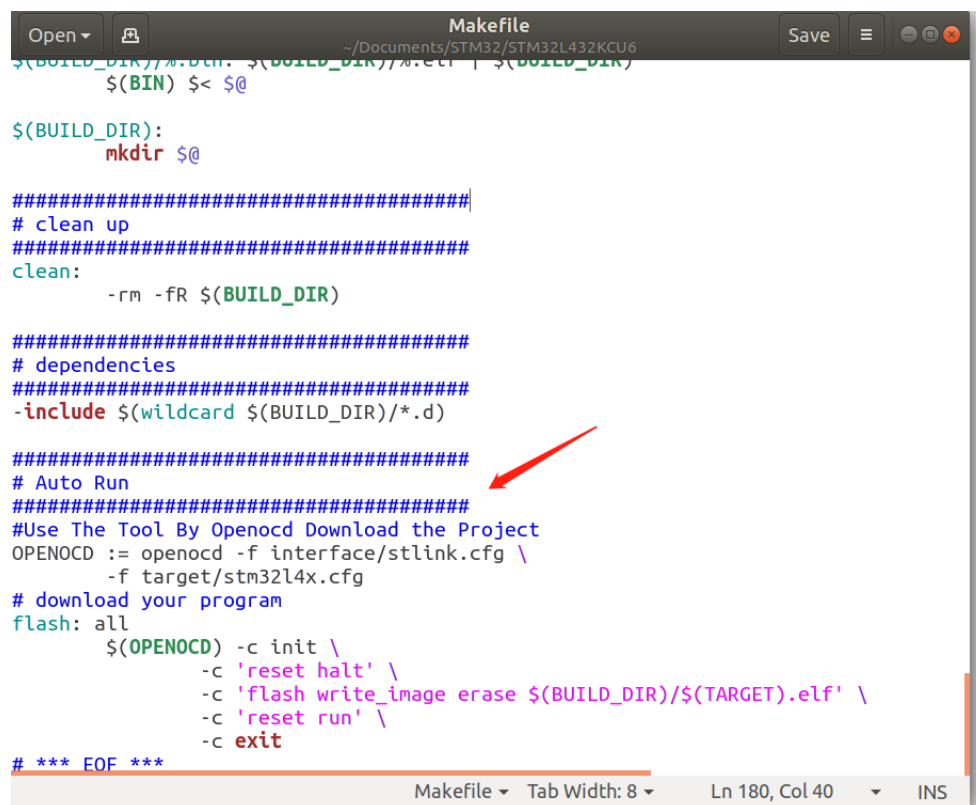


get Defines from Makefiles:



```
90
91 # fpu
92 FPU = -mfpu=fpv4-sp-d16
93
94 # float-abi
95 FLOAT-ABI = -mfloat-abi=hard
96
97 # mcu
98 MCU = $(CPU) -mthumb $(FPU) $(FLOAT-ABI)
99
100 # macros for gcc
101 # AS defines
102 AS_DEFS =
103
104 # C defines
105 C_DEFS = \
106 -DUSE_HAL_DRIVER \
107 -DSTM32L432xx
108
109 # AS includes
110 AS_INCLUDES =
```

(2) Add **make flash** command to Makefile file:



```
Open Makefile Save
~/Documents/STM32/STM32L432KCU6

$(BUILD_DIR)/%.bin: $(BUILD_DIR)/%.elf | $(BUILD_DIR)
    $(BIN) $< $@

$(BUILD_DIR):
    mkdir $@

#####
# clean up
#####
clean:
    -rm -fR $(BUILD_DIR)

#####
# dependencies
#####
-include $(wildcard $(BUILD_DIR)/*.d)

#####
# Auto Run
#####
#Use The Tool By Openocd Download the Project
OPENOCD := openocd -f interface/stlink.cfg \
    -f target/stm32l4x.cfg
# download your program
flash: all
    $(OPENOCD) -c init \
        -c 'reset halt' \
        -c 'flash write_image erase $(BUILD_DIR)/$(TARGET).elf' \
        -c 'reset run' \
        -c exit

# *** EOF ***

Makefile Tab Width: 8 Ln 180, Col 40 INS
```


9. Set full screen display in Virtualbox:

