ESP32

开发环境搭建

淘宝店铺：

PC 端：

http://n-xytrt8gqu585po94mwj5atokcyd4.taobao.com/index.htm

手机端：

https://shop.m.taobao.com/shop/shop_index.htm?sellerId=755668508&shopId=104493595&inShopPageId=423890608&pathInfo=shop/index2

资料下载地址：

链接：https://pan.baidu.com/s/1kCjD8yktZECSGmHomx_veg?pwd=q8er
提取码：q8er

源码下载地址：

https://gitee.com/vi-iot/esp32-board.git

一、前言

开发 ESP32，最头疼的事情莫过于这个开发环境的搭建了，网上各种教程简直是多不胜数，但完全照着做一遍又不行，而官方的资料也一头雾水。其实主要原因还是因为有多种开发环境、多种工具、不同平台都可以编译我们的 ESP32 工程。在本教程中，我们坚持使用 esp-idf 库来进行开发，因为 esp-idf 库是官方目前主推的库，最新的特性更新以及 bug 修复都在 esp-idf 上进行发布，而且大部分源码是开源的，大家可以随意获取和查看。基于 esp-idf 开发还不够，我们还需要开发工具，esp-idf 可以在 windows、linux、mac 上进行开发，但官方主推的还是 linux 平台上开发，我之前试过 windows 平台下开发，编译慢不说，安装工具还很不方便。因此本教程所有代码均是在 Linux 环境下编译的，好了，我们先看下，如何把这个开发环境搭建好。

二、软件安装

因为我们需要在 Linux 下开发，但同时也想用 windows 下的工具或其他软件，因此最好的方法就是装虚拟机，我们的原则是，使用 windows 下的工具来进行代码浏览和编辑，然后在 linux 下编译，需要装的工具有如下

vmware 虚拟机软件，可以自行去官网下载，安装也很简单，这里不做过多讲解

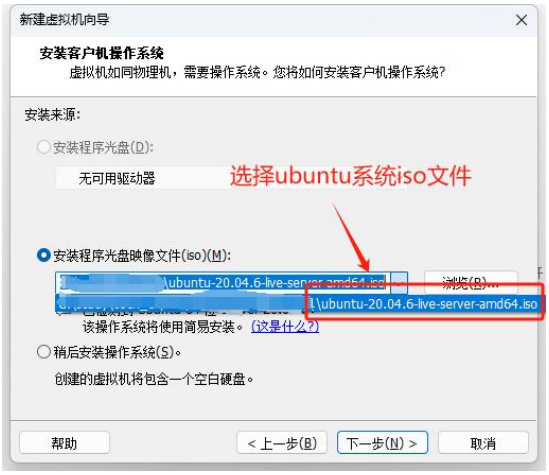
ubuntu 我们选用 ubuntu 作为 Linux 操作系统，这也是官方推荐的，但我这里推荐大家用 ubuntu 20.04 版本，用其他版本可能都会有问题，还有我推荐大家用服务器版本，这样虚拟机开起来就会减少内存占用。

vscode VSCode 作为我们远程连接 ubuntu 工具和代码浏览工具，十分好用，大家自行下载安装即可，这里不做过多讲解。

MobaXterm Mobaxterm 作为一款免费的终端工具，我推荐大家用这个工具，SSH 登录到虚拟机上，进行项目的配置和编译，功能强大

另外对电脑要求如下：
16G 内存（推荐 32G 或以上）
200G 以上硬盘空间

现在打开我们 VMWare 软件，在左边的窗口右键，新建虚拟机



后续就是下一步到底了，然后创建完成后，虚拟机就会启动，开始安装 ubuntu 系统，安装步骤大家看提示操作即可。请注意，如果有设置用户名的地方，请设置成和刚才创建虚拟机设置用户名步骤那里一致的用户名和密码；有叫你升级 update 的地方，一律 cancel；有提示你装 openssl，务必装上。安装虚拟机应该问题比较少，有问题的话私信我。

接下来才是重要环节

虚拟机开机后，是如下界面

```
Ubuntu 20.04.6 LTS vi tty1

vi login: [   27.636395] cloud-init[1256]: Cloud-init v. 23.1.2-0ubuntu0~20.04.2 running 'modules:co
nfig' at Wed, 08 May 2024 13:51:38 +0000. Up 27.36 seconds.
[   28.359336] cloud-init[1262]: Cloud-init v. 23.1.2-0ubuntu0~20.04.2 running 'modules:final' at We
d, 08 May 2024 13:51:39 +0000. Up 28.12 seconds.
[   28.359766] cloud-init[1262]: Cloud-init v. 23.1.2-0ubuntu0~20.04.2 finished at Wed, 08 May 2024
13:51:39 +0000. Datasource DataSourceNone.  Up 28.34 seconds
[   28.360119] cloud-init[1262]: 2024-05-08 13:51:39,342 - cc_final_message.py[WARNING]: Used fallba
ck datasource

vi login:
```

看起来就是 shell，啥都没有，首先我们要登录，用刚才创建虚拟机和安装系统时的用户名和密码登录。我们要安装一些工具，输入如下命令

sudo apt-get install git wget flex bison gperf python3-pip python3-venv cmake ninja-build ccache libffi-dev libssl-dev dfu-util libusb-1.0-0 net-tools

（注意 ccache 和 libffi-dev 之间有空格！）

Ubuntu 20.04 自带了 Python3.8 版本（esp-idf 的编译需要 python 工具），我们不用去更新，也不要用其他版本，就用这个版本即可。

因为 github 的访问问题，我们很难直接从 github 上拉取完整的 esp-idf 源码，但乐鑫官方提供了一个下载工具，我们通过这个工具可以轻松的获取到 esp-idf。

首先我们新建一个 esp32 目录，存放我们所有 esp32 相关的东西

mkdir esp32

cd esp32

我们拉取 esp-gitee-tools 工具

git clone https://gitee.com/EspressifSystems/esp-gitee-tools.git

完成后到这个工具目录里面

cd esp-gitee-tools

./jihu-mirror.sh set

上面这句命令，会将 github 的地址自动替换成 jihulab 上的镜像地址，这样就不用担心 github 的访问问题了

然后我们可以理直气壮的拉取 github 源码了，先回到 esp32 目录，

cd ..

然后拉取 esp-idf

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

速度很快，就不存在访问问题了。

上面步骤执行完后，应该有这两个目录（如下红框）

然后我们切换到 esp-idf 目录

cd esp-idf

由于本教程所有例程都是基于 esp-idf v5.2 版本进行开发的，因此我们需要切换一下版本

git checkout v5.2

这样就切换完成了，我们输入 git branch 看一下

```
vi@vi:~/esp32/esp-idf$ git branch
* (HEAD detached at v5.2)
  master
vi@vi:~/esp32/esp-idf$
```

发现是 v5.2 版本
然后需要执行
git submodule update --init --recursive 让子模块也切换但对应版本

在实验中发现 pip 安装一些包的时候速度很慢，输入如下两条命令切换源解决
pip config set global.index-url http://mirrors.aliyun.com/pypi/simple
pip config set global.trusted-host mirrors.aliyun.com

然后需要安装编译工具，需要 esp-gitee-tools 工具目录下的 install.sh 脚本
我们在 esp-idf 目录下直接执行
../esp-gitee-tools/install.sh

等待完成后，我们的 esp-idf 就部署完成了。

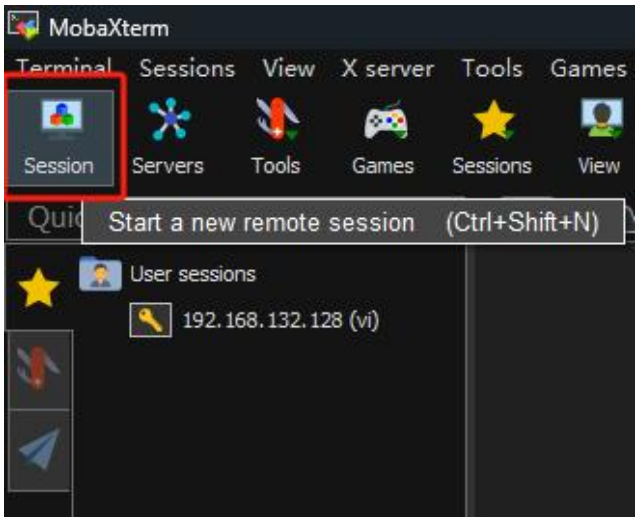
三、工程目录和编译

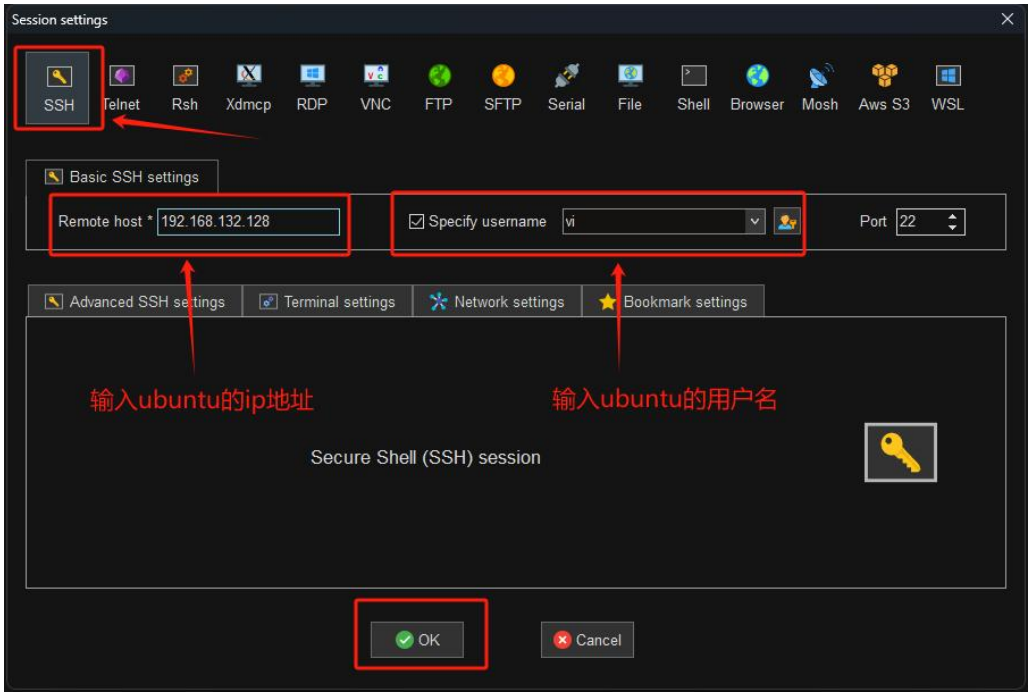
在上一章，我们把 esp-idf 完成的拉取完成了，我们先来看下 esp-idf 里面有什么
最主要的目录如下：
components:组件代码，esp-idf 的核心，包含通用的功能部件，比如 freeRTOS、wifi、GPIO 驱动、PWM 驱动等等
examples:官方例程，大部分的应用都可以在这里找到相关例子
docs:一些介绍文档
tools：一些工具，如编译、烧录

接下来，大家把我的例程下载到 esp32 目录里面
cd ~/esp32
git clone --recursive https://gitee.com/vi-iot/esp32-board.git
这个仓库是我给大家提供的例程，包含了很多个工程，每个工程都是可以单独编译的。

3.1 代码编译

目前我们只是安装了一些工具，以及下载好了 esp-idf 和工程源码，但还没编译。所谓编译就是要使用 esp-idf 里面的工具，对我们的工程源码进行编译、汇编、链接等操作，生成 bin 文件的过程。为了方便起见，我们使用 mobaxterm 工具 SSH 登录到我们的 ubuntu，mobaxterm 显示内容更丰富，功能也更强大。先打开 mobaxterm 工具，新建一个 session。





点击 OK 后，输入密码，就可以登录 ubuntu 了，

操作和在 VMWare 上面差不多，也同样是用 shell 命令操作。现在我们要设置 esp-idf 的环境变量。

cd esp32/esp-idf

source export.sh

执行完 export.sh 脚本后，会出现如下

```
Added the following directories to PATH:
  /home/vi/esp32/esp-idf/components/espcoredump
  /home/vi/esp32/esp-idf/components/partition_table
  /home/vi/esp32/esp-idf/components/app_update
  /home/vi/.espressif/tools/xtensa-esp-elf-gdb/12.1_20231023/xtensa-esp-elf-gdb/bin
  /home/vi/.espressif/tools/riscv32-esp-elf-gdb/12.1_20231023/riscv32-esp-elf-gdb/bin
  /home/vi/.espressif/tools/xtensa-esp-elf/esp-13.2.0_20230928/xtensa-esp-elf/bin
  /home/vi/.espressif/tools/riscv32-esp-elf/esp-13.2.0_20230928/riscv32-esp-elf/bin
  /home/vi/.espressif/tools/esp32ulp-elf/2.35_20220830/esp32ulp-elf/bin
  /home/vi/.espressif/tools/openocd-esp32/v0.12.0-esp32-20230921/openocd-esp32/bin
  /home/vi/.espressif/tools/xtensa-esp-elf-gdb/12.1_20231023/xtensa-esp-elf-gdb/bin
  /home/vi/.espressif/tools/riscv32-esp-elf-gdb/12.1_20231023/riscv32-esp-elf-gdb/bin
  /home/vi/.espressif/tools/xtensa-esp-elf/esp-13.2.0_20230928/xtensa-esp-elf/bin
  /home/vi/.espressif/tools/riscv32-esp-elf/esp-13.2.0_20230928/riscv32-esp-elf/bin
  /home/vi/.espressif/tools/esp32ulp-elf/2.35_20220830/esp32ulp-elf/bin
  /home/vi/.espressif/tools/openocd-esp32/v0.12.0-esp32-20230921/openocd-esp32/bin
  /home/vi/.espressif/python_env/idf5.2_py3.8_env/bin
  /home/vi/esp32/esp-idf/tools
Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build

vi@vi:~/esp32/esp-idf$
```

说明设置环境变量成功。因为这次设置只是针对当前登录的用户，我们退出了重新登录这些环境变量就没有了，我们又要重新执行一下以上命令。解决方法是在~/.profile 中追加如下命令

source esp32/esp-idf/export.sh

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
source esp32/esp-idf/export.sh
```

这样每次登录就会自动执行这条语句。

然后我们回到 esp32-board 库

cd ~/esp32/esp32-board　　（这里大家自行切换到 esp32-board 目录中）

我们选 helloworld 工程试下

```
vi@vi:~/esp32/esp32-board$ ls
dht11  display  helloworld  ledc  ntc  sdcard  smartconfig  spiffs  sr04  wifi
```

cd helloworld

重点来了！

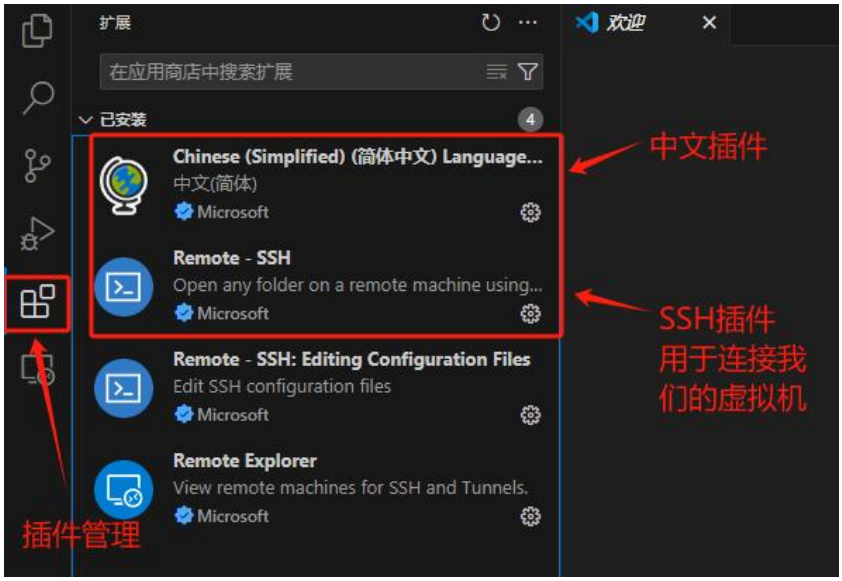
在这个目录下，输入 idf.py build 即可自动完成编译，生成 bin 文件

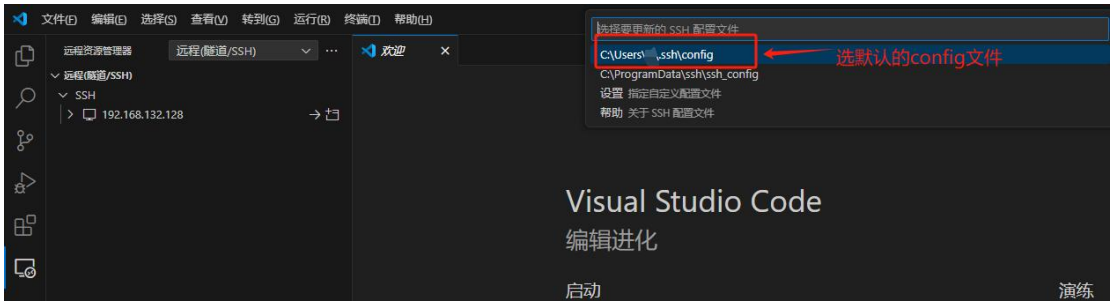
```
Merged 2 ELF sections
Successfully created esp32 image.
Generated /home/vi/esp32/esp32-board/helloworld/build/main.bin
[40/40] cd /home/vi/esp32/esp32-board/helloworld/build/esp-idf/esptool_py && /home/vi/.espressif/pyth...oworl
main.bin binary size 0x2b260 bytes. Smallest app partition is 0x100000 bytes. 0xd4da0 bytes (83%) free.

Project build complete. To flash, run:
 idf.py flash
or
 idf.py -p PORT flash
or
 python -m esptool --chip esp32 -b 460800 --before default_reset --after hard_reset write_flash --flash_mode
on_table/partition-table.bin 0x10000 build/main.bin
or from the "/home/vi/esp32/esp32-board/helloworld/build" directory
 python -m esptool --chip esp32 -b 460800 --before default_reset --after hard_reset write_flash "@flash_args"
vi@vi:~/esp32/esp32-board/helloworld$
```

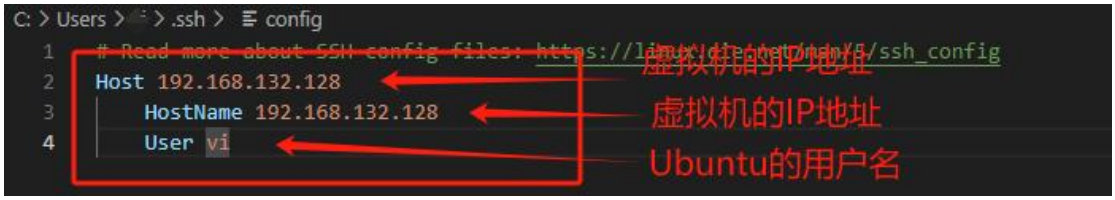
大家看到如上信息就说明编译成功了！！

3.2 代码浏览和编辑

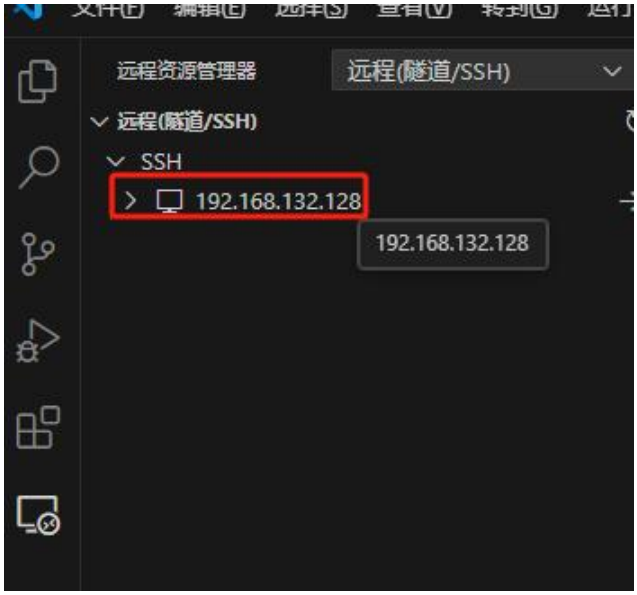
那我们怎么看代码方便呢？又如何修改代码方便呢？不会要用 vim 这工具吧？当然不是。接下来我们要用到 VSCode 工具，我们在 windows 上打开 VSCode 工具，先安装几个插件



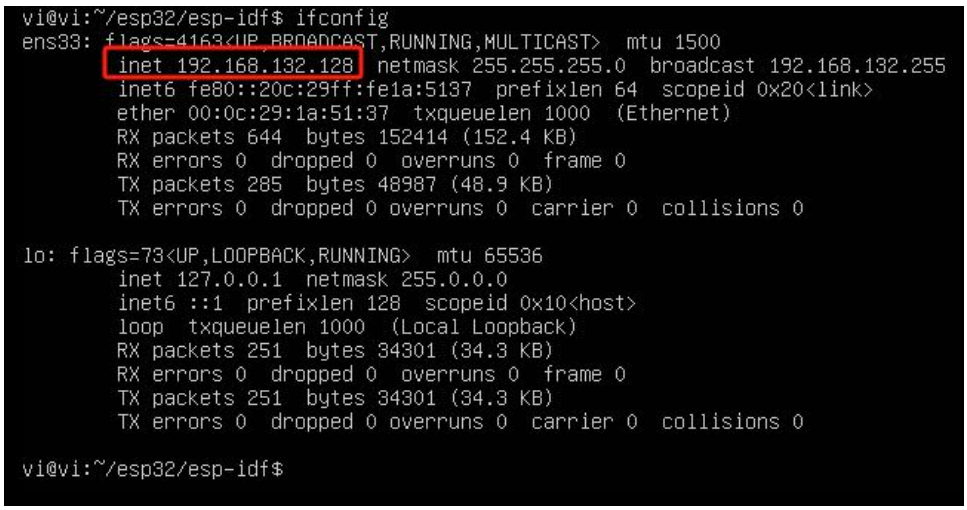


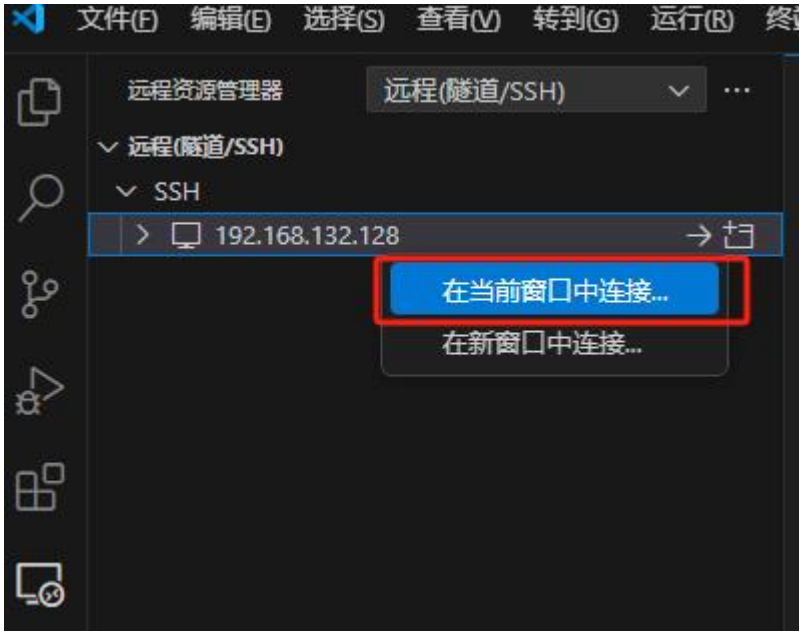


保存，重启一下 VSCode，就能看见可以连接的远程主机了

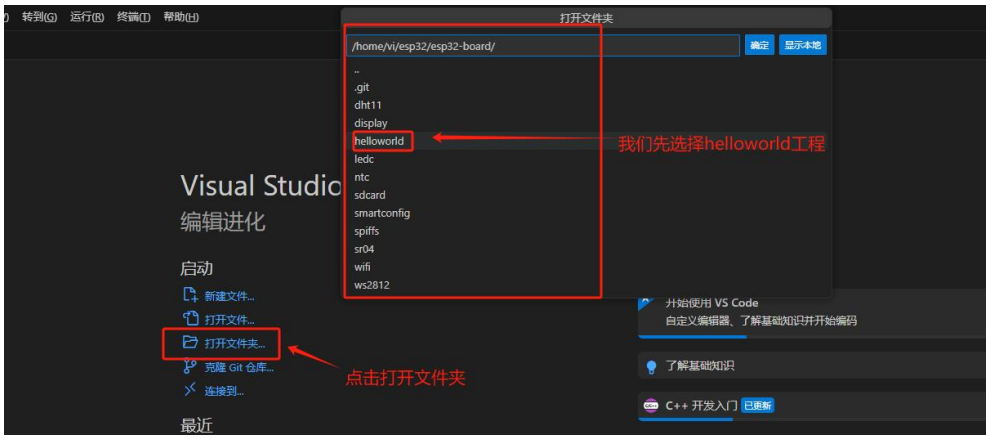


注意 IP 地址要换成你虚拟机的 IP 地址。在 ubuntu 里面输入 ifconfig 命令可以查看 ip 地址

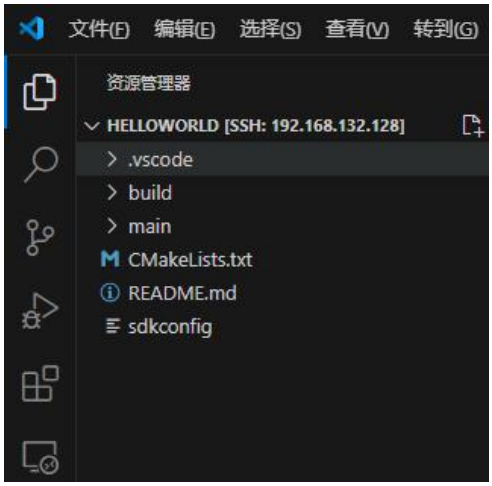




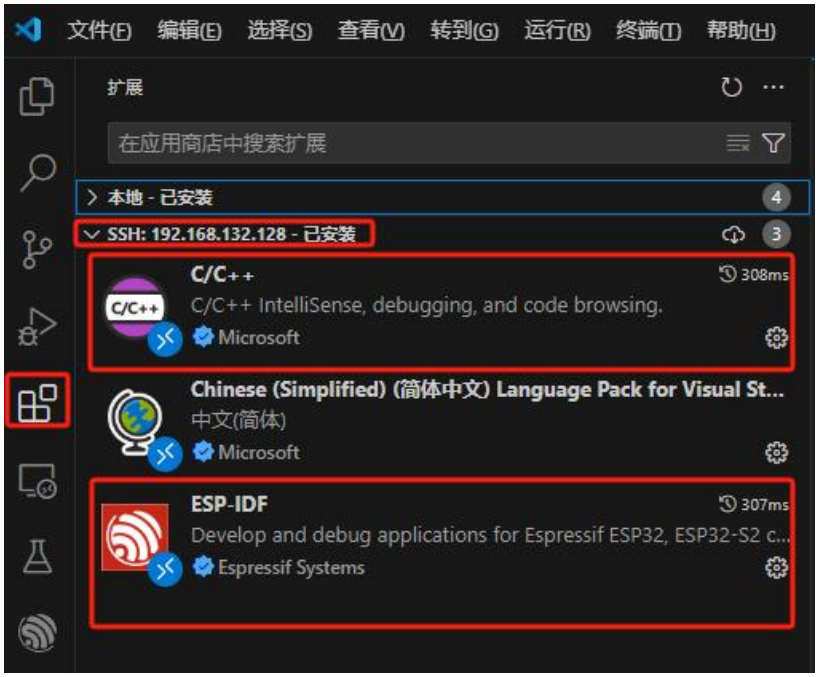
会让你输入密码，输入对应的用户名密码，就登录成功了，但目前我们还看不到文件，我们需要打开文件夹，按下图，选择好对应的文件夹后，点确定，我们选 helloworld 工程



这里会让你再输入一次密码，输入正确后，在左边资源管理器就会出现文件夹里面的内容了



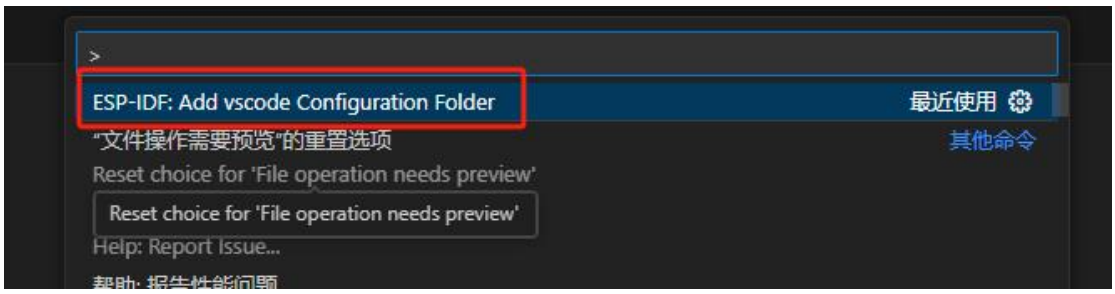
VSCode 还没部署完，我们 SSH 远程登录虚拟机后，需要安装两个插件



需要安装 C/C++、ESP-IDF，如果不装这两个插件，我们看代码没法跳转

装好后，最好重启一下 VSCode，重复上述步骤登录 ubuntu 和打开 helloworld 文件夹，然后我们回到我们的 helloworld 工程

按下 ctrl+shift+p，弹出搜索框，搜索 ESP-IDF:Add vscode Configuration Folder，在下拉列表出现后，点击一下就可以了，这步的目的是为了把 esp-idf 里面的源码路径也加到我们的工程中，现在我们可以通过按着 ctrl 键点击函数或变量进行跳转看代码了



大家可以方便在 VSCode 上对代码文件进行编辑了！！

综上所述，esp-idf 的开发环境是需要一些专业技术的，但开发环境的搭建属于一劳永逸的事情，环境搭建完，我们后续的开发工作就会非常顺利，我们宁愿在前期花多点功夫，也不要落得后期各种奇葩问题。

当然后续还有很多要调整的细节，随着教程我会慢慢指导大家的。