# 部署流程

#### 1. 下载自动化解决方案软件包

• 从RTC上下载最新版本的工具包到本地PC



• 确认文件的md5sum值与首页中展示的一致

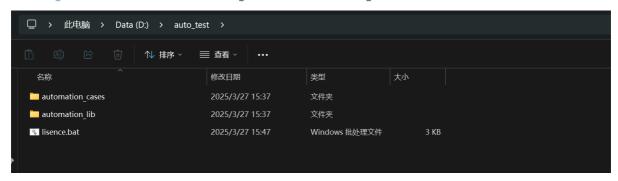
```
yonns@Synne MINGW64 ~/Downloads
$ md5sum vcu_automator_v0.1.zip
c2d99382e2aa8ea8ad4887273ac5b658 *vcu_automator_v0.1.zip
yonns@Synne MINGW64 ~/Downloads
$ |
```

大小 最近一次修改时间

217 MB 2025年3月27日 下午4:14 (昨天)

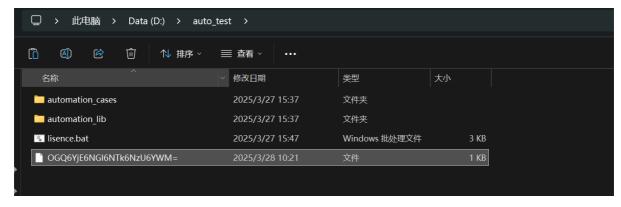
## 2. 解压到指定目录

- 在D盘或E盘根目录下创建一个文件夹: auto\_test (此目录今后将作为自动化测试的根目录)
- 将下载好的软件压缩包文件解压到刚刚创建好的 auto\_test 文件夹内
- 进入 auto\_test, 检查文件内容, 文件夹: automation\_cases, 文件夹: automation\_lib, 文件: lisence.bat



## 3. 配置python/adb环境变量 & 生成注册编号

- 双击lisence.bat文件, 至执行完成
- 确认当前目录下有新的注册编号文件生成



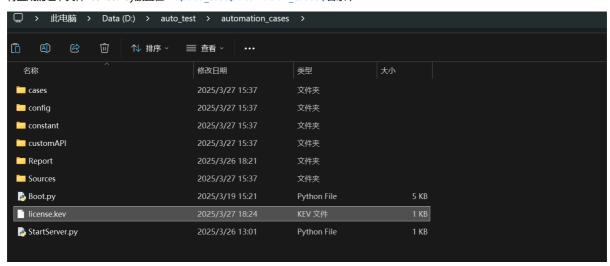
• 可在Windows cmd窗口中使用 where python , where pip , where adb 这3条指令确认python,pip,adb已经正确加入环境变量

#### 4. 申请证书文件

• 将 #3 中生成的注册编号文件发送给VCU软件测试团队License管理员 @Li Zhao, @Cheng Jie 生成证书文件license.key

## 5. 将证书文件放置到正确位置

• 将生成的证书文件license.key放置在 D:\auto\_test\automation\_cases\ 目录下



## 6. 安装Git, 配置环境变量 (若曾经完成, 跳过)

- <u>下载Git并安装</u>
- 假如安装在D:\application\Git\目录下,将 **D:\application\Git\bin** 加入系统环境变量

## 7. PC连接好PCAN硬件,配置驱动

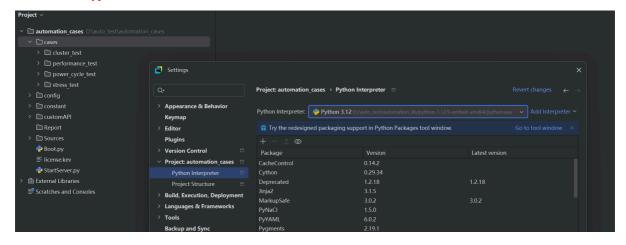
- 将PCAN硬件一端连接PC,另一端连接VCU侧ICAN
- 安装PCAN硬件的驱动软件包,在设备管理器中确认PCAN设备驱动正常



## 8. 在PyCharm中打开automation\_cases工程文件夹,配置python解释器路径

- 在PyCharm中选择 automation\_cases 文件夹并打开
- 在PyCharm中 File > Settings > Python Interpreter 中设置已存在的解释器路径: D:\auto test\automation lib\python-3.12.9-

#### embed-amd64\python.exe



#### 9. PC电脑连接网络

#### 10. 配置Config.json配置文件中关于台架VCU车型参数

• 打开 automation\_cases/config/Config.json 配置文件,正确设置与当前台架匹配的 BENCH\_INFO, PROJECT, MODEL\_YEAR,

VEHICLE PROPULSION TYPE 必要配置的属性值

```
"BENCH_INFO": "CLEA-15",
"PROJECT": "L232",
"MODEL_YEAR": "MY25",
"VEHICLE_PROPULSION_TYPE": "BEV",
```

## 11. 启动车机

- 选择 automation\_cases/StartServer.py 运行,启动can总线相关的服务
- 将 automation\_cases/Boot.py 代码中总线VIN相关的行的代码注释掉,然后运行该脚本,VCU会被唤醒起来,数秒后屏幕会被点亮,且最终PowerMode=Run/Propulsion (如需要使用到总线VIN,需要将此处N配置为实际的VIN值,否则会Theft Lock)

```
def boot(self):
    if self._platform == "G8":
        PowerModeManager.getInstance().set_power_mode("Propulsion")
        **VIN

# CanBusclient.getInstance().send_message(ids=["0x498"], is_mac=("0x498": "False"), datas=("0x498": "4D4131534333138" if not self._vin_number[2] else self._in_n
# CanBusclient.getInstance().send_message(ids=["0x75f"], is_mac=("0x75f": "False"), datas=("0x75f": "5347455228334C37" if not self._vin_number[0] else self._in_n
# CanBusclient.getInstance().send_message(ids=["0x712"], is_mac=("0x712": "False"), datas=("0x712": "4C00000000000000" if not self._vin_number[0] else self._vin_n
else:
        PowerModeManager.getInstance().set_power_mode("Run")

# VIN
# CanBusclient.getInstance().send_message(ids=["0x539"], is_mac=("0x539": "False"), datas=("0x539": "4E41303539323837" if not self._vin_number[2] else self._vin_n
# CanBusclient.getInstance().send_message(ids=["0x53x"], is_mac=("0x53x": "False"), datas=("0x53x": "5347555042414C36" if not self._vin_number[1] else self._vin_n
# CanBusclient.getInstance().send_message(ids=["0x426"], is_mac=("0x426": "False"), datas=("0x426": "4C00000000000000" if not self._vin_number[0] else self._vin_n
# CanBusclient.getInstance().send_message(ids=["0x426"], is_mac=("0x426": "False"), datas=("0x426": "4C00000000000000" if not self._vin_number[0] else self._vin_n
```

• 完成以上所有步骤后,可满足最基本的让台架持续处于正常运行状态的需求

## 12. Power cycle test

• 打开 automation cases/config/Config.json 配置文件,正确设置与当前台架匹配的 BASE TEST DISK, SERIAL PORT 必要配置的值

```
"BASE_TEST_DISK": "D",
"SERIAL PORT": "COM6".
```

• 打开 automation\_cases/cases/power\_cycle\_test/00PowerCycle.py文件,文件头部有许多需要配置的参数: STR\_ENABLE, PULLBACK,

## STRESS TEST, NIGHT WAKEUP STR, SAMPLE TYPE, SERIAL LOG

```
# Configurable
STR ENABLE = True
PULLBACK = False
STRESS_TEST = {"CanStressTest": True, "MonkeyTest": True, "NaviSimulation": False, "
NIGHT_WAKEUP_TIMES = 0
NIGHT_WAKEUP_STR = False
SAMPLE_TYPE = "C"
# Log Obtain
SERIAL_LOG = False # code is in charge of recording serial log, avoid using putty t
...
```

请根据PowerCycle测试文档熟悉各配置项的含义,正确完成配置

• 上述配置完成后,选择 automation\_cases/cases/power\_cycle\_test/00PowerCycle.py 文件并运行,可开启 str/cold 循环测试

#### 13. Stress test

- 开始连续24h/48h/72h压测前,也需要在 automation\_cases/cases/stress\_test/SystemPressureRunner.py 脚本文件内配置参数
- 选择 automation\_cases/cases/stress\_test/SystemPressureRunner.py 运行,开启系统压测