noddle2023readme

@Jun

@July 18, 2023

所有需要的文件在

```
git clone https://github.com/Jun-llj/noddle2023.git
```

第一章 源码使用

1 基础环境配置

1.1 串口规则配置

1. 设备插入电脑
2. 终端执行

```
ll /dev
```

可以查看到ttyACM0

3. 进入目录【/配置Linux环境文件/配置串口/serial_driver/Linux】
4. 找到 ch9012_udev.sh 文件，并运行

```
sh ./ch9102_udev.sh
```

5. ch9012_udev.sh 运行不了可输入指令 sudo chmod +x ch9012_udev.sh 赋权限。
6. 拔插一下设备后输入指令

```
ll /dev
```

可以看到已经配置好，设备名为wheeltec_mic→ttyACM0

1.2 cjson 安装

1. 将 【/配置Linux环境文件/安装cjson/cJSON.rar】文件夹解压到/home 目录下
2. 进入此文件夹输入指令

```
mkdir build
```

3. 创建文件夹并进入此目录，依次输入指令

```
cmake ..
make
sudo make install
```

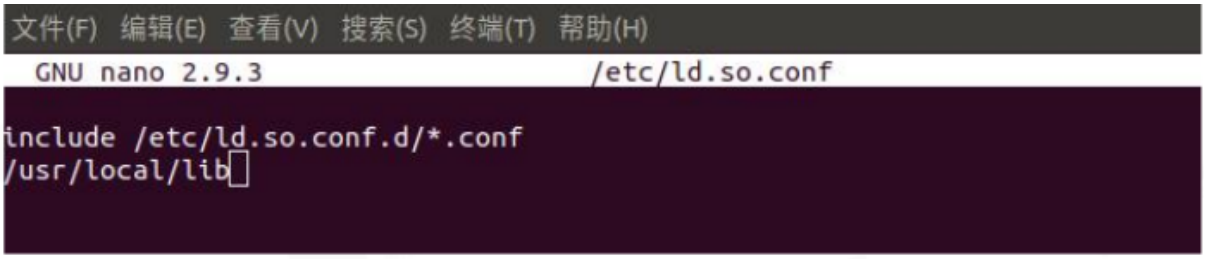
4. 终端执行

```
sudo nano /etc/ld.so.conf
```

在文档末尾一行增加

```
/usr/local/lib
```

即



保存退出后，执行

```
sudo /sbin/ldconfig
```

令其生效

2 ROS 语音功能配置

2.1 so 库配置

1. 将ROS包放入【catkin_ws/src】中——我们只需要不依赖小车的ROS SDK
2. cd ~/catkin_ws/src/xf_mic_asr_offline_line/lib/x64，执行

```
sudo cp lib* /usr/lib
```

3. **（跳过）**gedit ~/catkin_ws/src/xf_mic_asr_offline_line/CMakeLists.txt，将【link_directories】里改为【lib/x64】

2.2 用户参数配置（跳过）

（目前不需要，我已经替换好了，用我的工单就行，失效再换）

1. gedit catkin_ws/src/xf_mic_asr_offline_line/config/appid_params.yaml
2. 修改APPID，目前（2023.7.17）使用梁丽君的工单号（53e84f26），如果这个工单号失效了（出现10121错误码，则自己申请或者问别人是否已经申请了，具体方法在这个板子的官方pdf里）

3. 修改APPID后要替换common.jet
   a. 使用我的common.jet，在u盘里
   b. 使用自己申请的工单里的common.jet，具体方法在这个板子的官方pdf里

   cd ~/catkin_ws/src/xf_mic_asr_offline_line/config/msc/res/asr

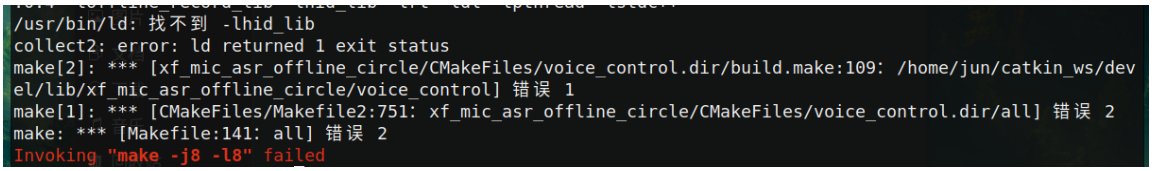
   替换其中的common.jet

2.3 功能包编译

1. cd catkin_ws
2. 终端执行

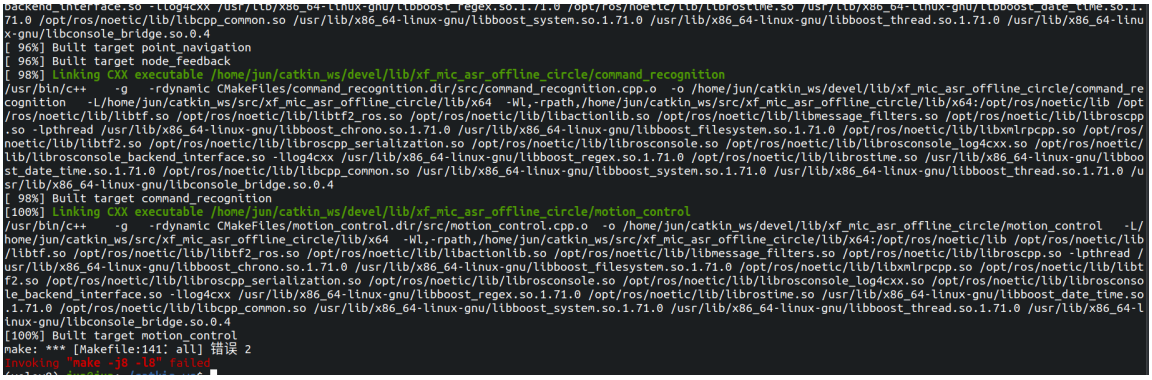
```
catkin_make -DCATKIN_WHITELIST_PACKAGES=xf_mic_asr_offline_line
```

3. 如果出现错误
   a. 错误1：找不到-lhid_lib



解决：进入【catkin_ws/build】和【catkin_ws/devel】中，删掉这两个文件夹里的其中的语音包，再次编译

b. 错误2：makefile:141：all 错误 2



解决：打开【catkin_ws/src/xf_mic_asr_offline_circle/CMakeLists.txt】，删除【target_link_libraries(voice_control ${catkin_LIBRARIES} offline_record_lib **hid_lib** rt dl pthread stdc++)】中的【hid_lib】

3 运行（跳过）

代码已被我修改，语音唤醒功能失效，launch文件已被我删除

唤醒词为“你好小九”

```
roslaunch xf_mic_asr_offline_line base.launch //开启语音控制底层、导航、雷
达扫描节点
roslaunch xf_mic_asr_offline_line mic_init.launch //开启麦克风阵列初始化节
点
```

4 问题

> ... logging to /home/jun/.ros/log/c9f30d88-256f-11ee-97e9-7d0c0959ac9b/roslaunch-jun-25647.log
> Checking log directory for disk usage. This may take a while.
> Press Ctrl-C to interrupt
> WARNING: disk usage in log directory [/home/jun/.ros/log] is over 1GB.
> It's recommended that you use the 'rosclean' command.

同时出现错误码23108

解决：使用ros虚拟环境即可

第二章 espeak安装使用

1. sudo apt install espeak
2. 直接命令行使用：
    a. 中文播报：espeak -vzh "你好”
    b. 英文播报：espeak "hello”
3. 代码调用

```
import os
# cmdline = 'espeak ' +  "hello"
# 程序中说中文（注意格式，espeak与-vzh有空格， 中文：你好之前也有空格）
```

```
cmdline = 'espeak -vzh ' +  "你好"
os.system(cmdline)
```

4. 在命令行使用时发现

> espeak -vzh 你好　#报错：Full dictionary is not installed for 'zh'
> espeak -vzhy 你好 #粤语，但说的不是人话

解决：(espeak-data.zip我的u盘里有)

```
git clone https://github.com/caixxiong/espeak-data.git
解压后直接替换 /usr/lib/x86_64-linux-gnu/espeak-data 这个目录即可
cd /usr/lib/x86_64-linux-gnu
sudo cp -r ~/Downloads/espeak-data/* espeak-data
cd espeak-data
sudo espeak --compile=zh
sudo espeak --compile=zhy # 成功，但是是蹩脚粤语
```

5. 优缺点：
    a. 优点：支持中英文
    b. 缺点：只能命令行使用，机械音较重，不流畅

第三章 修改代码情况与使用

1 修改后文件使用情况

1.1 使用方法

```
终端1：
roslaunch xf_mic_asr_offline_line noddle.launch

终端2：
rosrun jun noddle_example.py
```

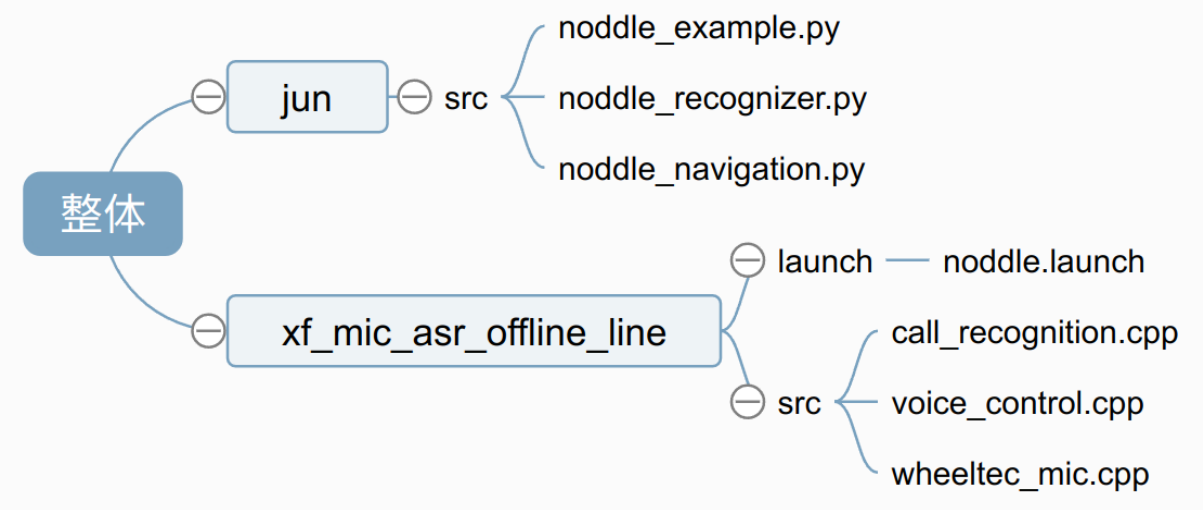
顺序无所谓

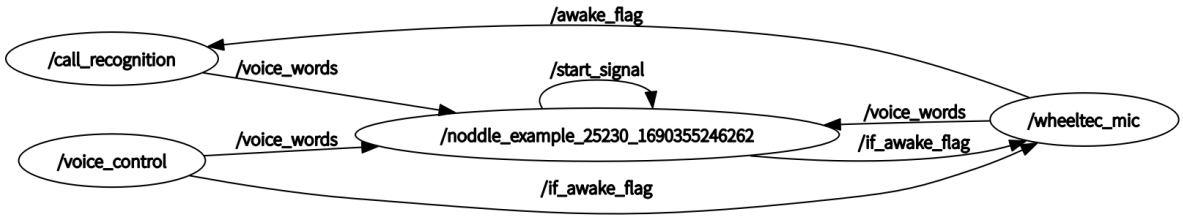
1.2 使用心得

1. 不要轻易修改该功能包名字，我尝试修改，发现要改很多东西，无法运行

2. 不在语法文件【功能包/config/call.bnf】中的句子一律不会被识别成功
3. 原始代码可能要喊两遍才能唤醒，尽量避免这种情况出现，一旦出现直接重启
4. 当语法文件中只有“厨房”，而你说了“去厨房”时，识别置信度会下降，即出现的多余字词越多，置信度越低；当语法文件中只有“厨房”或“是”时，你说了“是的，我要去厨房”，会根据两个关键词的置信度来选择识别结果，只会识别出“厨房”或者“是”（语法文件设定）

2 代码逻辑（通信逻辑）





1. noddle_example.py——开启语音

使用获取命令函数get_cmd()，向**“if_awake_flag”**发布**语音唤醒信号**

noddle_recognizer.py订阅“**voice_words**”，接收**识别结果**

2. wheeltec_mic.cpp——唤醒板子

订阅“**if_awake_flag**”，接收**语音唤醒信号**，唤醒板子

板子一旦被唤醒，向“**awake_flag**”发布**被唤醒信号**，向“**voice_words**”发布**用户提醒消息**

3. voice_control.cpp

建立**服务端**，等待**识别开始指令**

4. call_recognition.cpp——控制录音调用

订阅“**awake_flag**”，接收**被唤醒信号**，并创建**客户端**

如果“**awake_flag**”传入**被唤醒信号**，**客户端发送请求**，发送**识别开始指令**，并得到识别结果的回应

识别结果：

1. 如果语音要求休眠（语法文件里必须有“小车休眠”或该代码中设定的休眠句子），则令awake_flag=0，**停止向服务端发送请求**，并清零识别失败次数
2. 如果识别成功，则清零识别失败次数，
3. 如果识别失败，则统计失败次数
    a. 连续识别失败5次或10次，向“**voice_words**”发布**用户提醒消息**
    b. 连续识别错误次数超过阈值，令awake_flag=0，**停止向服务端发送请求**，向“**voice_words**”发布**用户提醒消息**，并清零识别失败次数

5. voice_control.cpp——录音与识别

服务端得到**识别开始指令**，开始录音与识别，并向“**voice_words**”发布**被识别结果**

若识别成功，则向“**awake_flag**”发布**取消被唤醒信号**

若失败，则继续识别

6. noddle_example.py——处理被识别句子

noddle_recognizer.py

接收到“**voice_words**”发布的**识别结果**，进行逻辑处理，并播报

若结果不对，则使用获取命令函数get_cmd()，向**“if_awake_flag”**发布**语音唤醒信号**，再次开启语音识别