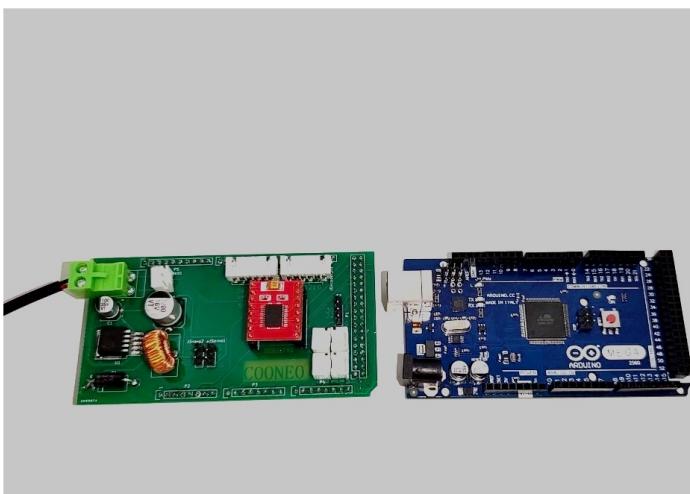


Arduino_Raspberry_ROS_Car Tutorials



chapter 1: Construction A ROS Car



Download codes from Git:

```
git clone https://github.com/COONEO/Arduino_Jetson_nano_ROS_Car.git
```

Step One: Download programs for Arduino Mega 2560

打开你的 Arduino IDE，然后进入 Arduino_code 文件夹找到 ROS 小车电机驱动程序，可能需要安装附带的库；然后对照我们微信公众号 ‘COONEO’ 中的文章《开源！手把手教你驱动 Arduino + ROS 小车的电机》中的描述改动对应位置的程序，最后编译烧录进 Arduino mega 2560 中。

```

RobotPIDDriver_tb6612 Make4e2ndChassis.cpp Make4e2ndChassis.h commands.h encoder_driver.h encoder_driver motor_driver

double input;
double output;
}

PIDInfo;
PIDInfo leftInfo, rightInfo;

//车轮配置
***** 第一步修改 电机外输出轴 转动一圈 所输出的总脉冲数 *****
*
* 由于是采用的中断方式捕获电机的霍尔脉冲，并且使用的是边沿触发方式，所以电机的编码值计算方法如下：
*   encoder = (边沿触发) 2 × 霍尔编码器相数量 (如：2) × 霍尔编码器线束 (如 13 ) × 电机减速比 (如：30) /
*

*****
double wheeldiameter = 0.064;           //车轮直径 单位 米 (m)
double encoderresolution = 2496.0; //编码器输出脉冲数/圈 2*2*13*48 = 2496      TT-motor encoder
//double encoderresolution = 1560.0; //编码器输出脉冲数/圈 2*2*13*30 = 1560      37-motor encoder

***** 第四步修改 PID 参数，优化电机的调速性能 *****
//PID参数配置
double Kp_L = 7.0, Ki_L = 10.0, Kd_L = 0.003; //2.0 5.0 0.003
double Kp_R = 5.0, Ki_R = 10.0, Kd_R = 0.003; //2.0 5.0 0.003
double Sum_count_L = 0;
double Sum_count_R = 0;

PID leftPID(&leftInfo.input, &leftInfo.output, &leftInfo.target, Kp_L, Ki_L, Kd_L, DIRECT);
PID rightPID(&rightInfo.input, &rightInfo.output, &rightInfo.target, Kp_R, Ki_R, Kd_R, DIRECT);
double pid_rate = 100.0; // default is 100 Hz
double pidinterval = 1000.0 / pid_rate; // PID每次运算结果的执行时间
long nextmotion;
int moving;

// A pair of variables to help parse serial commands (thanks Fergs)
int arg = 0;
int index = 0.

```

For more details, please see the Document in our Wechat ID COONEO :

2022/1/1 17:09

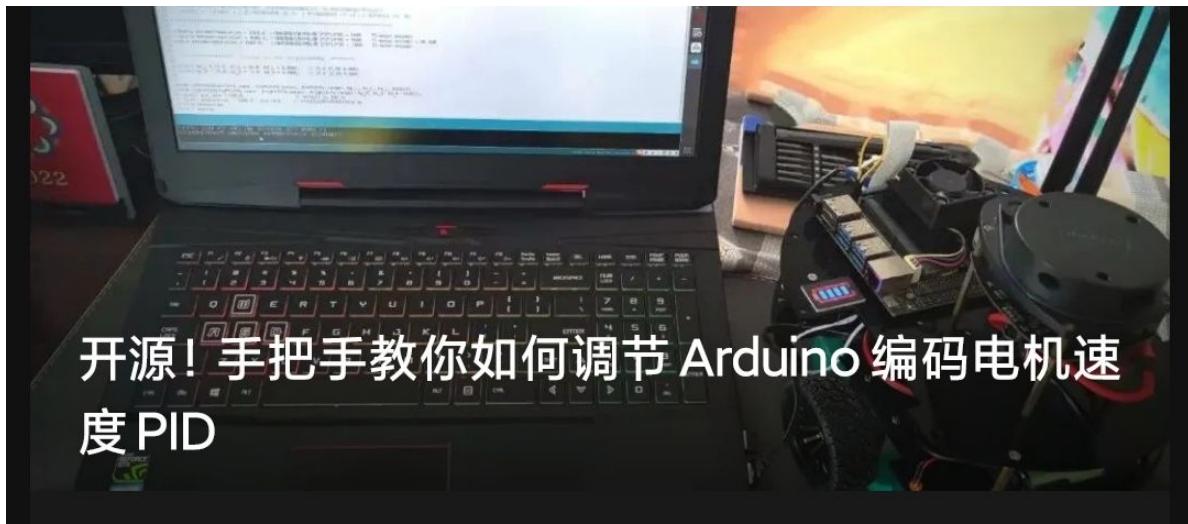
...

开源！手把手教你驱动Arduino+ROS小车的电机 原创



264 3 1

调节霍尔编码电机的速度PID参数方法，则参照微信公众号推文《开源！手把手教你如何调节Arduino编码电机速度PID》：



开源！手把手教你如何调节 Arduino 编码电机速度 PID

Step Two: FLASH OS and LAUNCH ROS NODE

1. 由于我们二次制作的Jetson-nano镜像（在英伟达官方的基础上安装了ROS环境以及一些常用软件）较大，您要是没有购买内存卡的又想节约时间的话，建议购买我们的内存卡服务哟，将为您安装好镜像；
2. 将 Jetson_nano_ROS_code 文件夹中的工作空间移动至你新烧录的镜像中，需要自行检查的配置有两点：
 - a. 配置 ros_arduino_bridge/ros_arduino_python 包中的配置文件（my_arduino_params.yaml），需要根据你使用的电机参数，轮胎左右安装的间距更新对应的参数，配置文件中需要检查的地方均写了注释；
 - b. 我们在镜像中根据 lsusb 的信息绑定了 3 个串口（A1M8雷达、IMU、Arduino），绑定之后，他们的访问

```
cooneo_nvi@cooneo-nvi:~$ lsusb
Bus 002 Device 002: ID 0bda:0411 Realtek Semiconductor Corp.
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 8087:0a2b Intel Corp.
Bus 001 Device 010: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 001 Device 009: ID 2341:0042 Arduino SA Mega 2560 R3 (CDC ACM)
Bus 001 Device 012: ID 2717:ff80
Bus 001 Device 002: ID 0bda:5411 Realtek Semiconductor Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
cooneo_nvi@cooneo-nvi:~$
```

名字分别变为了（LIDAR_PORT、IMU_PORT、Arduino_PORT）;该 rule 文件放置在 /etc/udev/rules.d 路径下，名为: cooneo_nvi_robot.rules。如果你没有使用我们的硬件，但想使用我们的软件，请结合自己的设备修改 rule 文件中的两个 ID；

```
KERNEL=="ttyACM*", ATTRS{idVendor}=="2341", ATTRS{idProduct}=="0042", MODE=="0777", SYMLINK+="Arduino_PORT"
KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE=="0777", SYMLINK+="LIDAR_PORT"
KERNEL=="ttyUSB*", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", MODE=="0777", SYMLINK+="IMU_PORT"

```

· 启动小车：

```
cd catkin_ws
catkin_make -j
source devel/setup.bash
roslaunch ros_arduino_python arduino.launch
```

· 然后使用 rqt_robot_steering 尝试控制小车

```

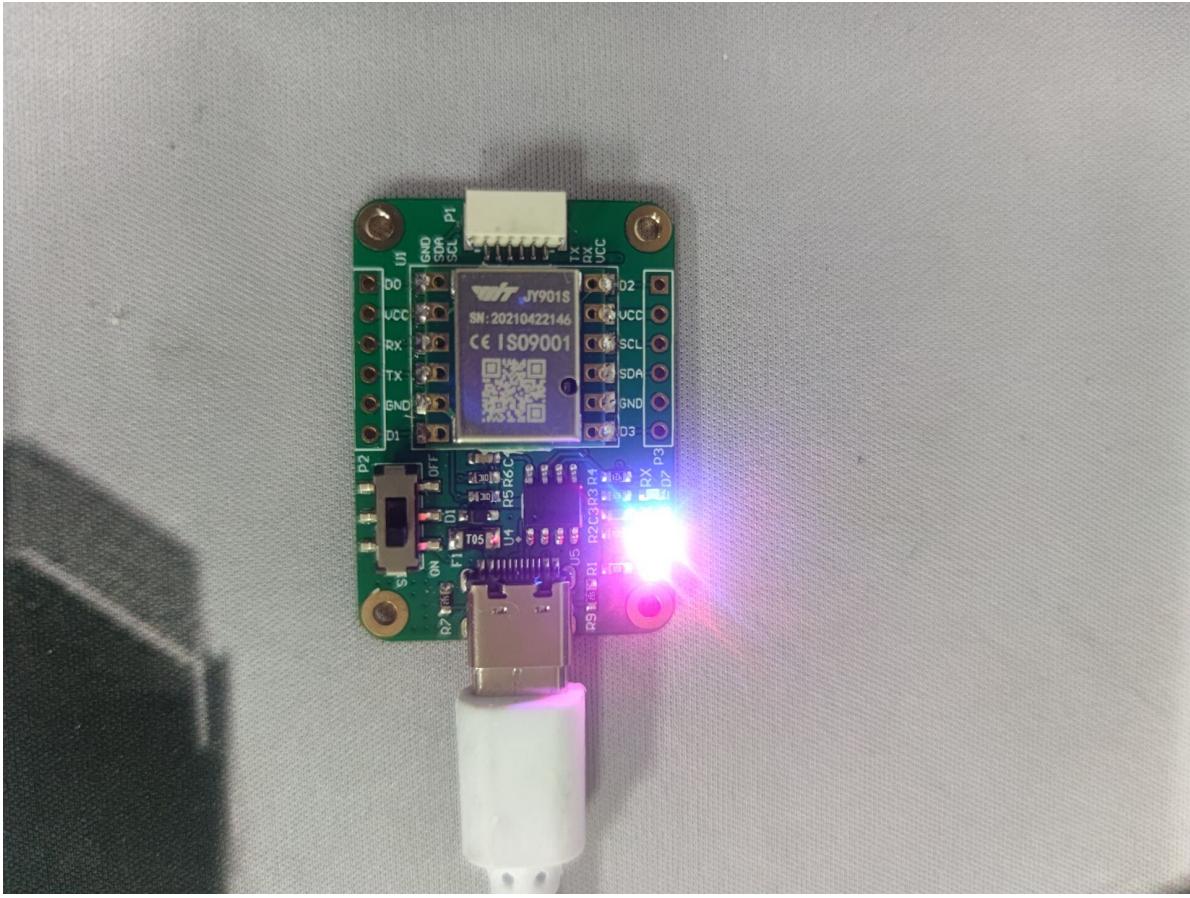
cooneo_nvi@cooneo-nvi:~/catkin_ws$ roslaunch ros_arduino_python arduino.launch
... logging to /home/cooneo_nvi/.ros/log/b63aa388-810f-11ec-aaf9-0028f8145eb5/rosvi-32124.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://10.42.1.1:33581/
SUMMARY
=====
PARAMETERS
* /arduino/Kd: 12
* /arduino/Ki: 0
* /arduino/Ko: 50
* /arduino/Kp: 10
* /arduino/acel_limit: 1.0
* /arduino/base_controller_rate: 10
* /arduino/base_frame: base_link
* /arduino/baud: 115200
* /arduino/encoder_resolution: 52
* /arduino/gear_reduction: 30
* /arduino/motors_reversed: False
* /arduino/port: /dev/Arduino_PORT
* /arduino/rate: 30
* /arduino/sensors/arduino_led/direction: output

```

chapter 2: Configure JY901 IMU

我们使用了一个带 温度和磁力计的 9轴 imu 模块儿，用来融合 轮式里程计 数据，然后实现 gmapping 建图。使用之前需要对该模块儿设置一番，对应的 ROS 包为 imu_901。其中的 resource_folder 文件夹中 有win10 下的配置软件，具体的设置方法，参考 imu_901/ReadMe.pdf 文件。



设置完毕后，将模块接入到 Jetson nano 的USB端口上即可 (ps: 记得结合自己的安装位置修改 imu_901.launch 中的 静态 tf 后面的参数)

chapter 3: gmapping_ekf

如果您没有购买我们的整体小车，建议您自己按照我们的推文 配置 ROS 主从机，然后再参照我们的步骤运行 gmapping 建图演示：

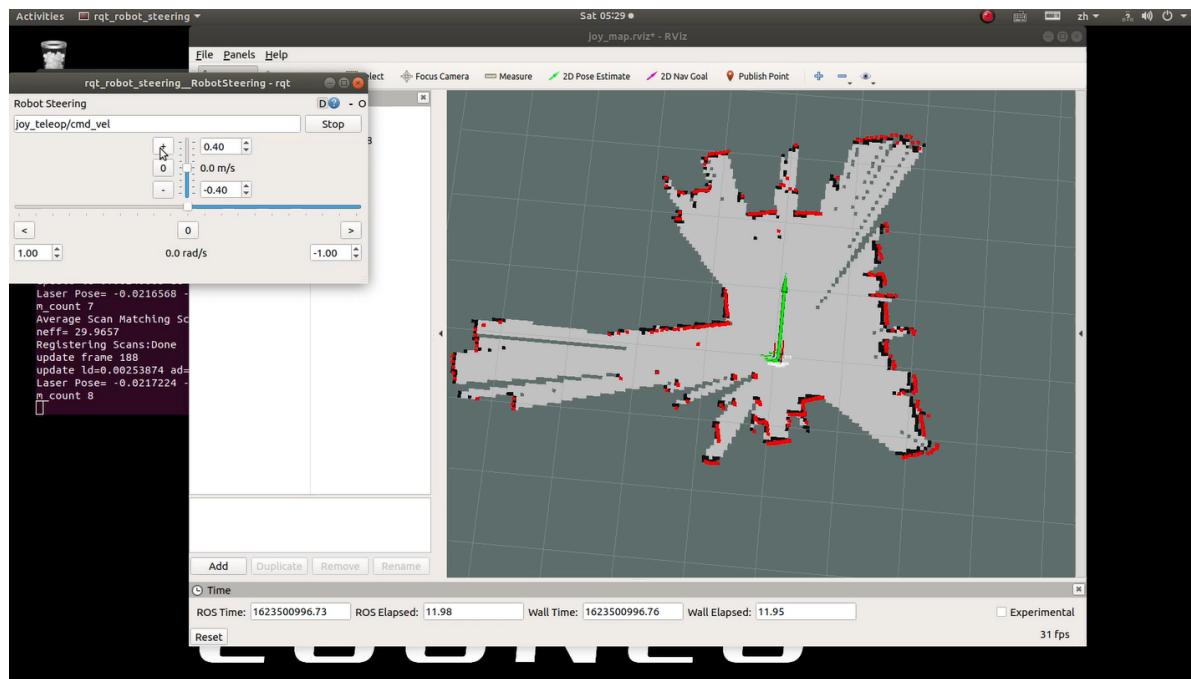
```
# Jetson_nano 默认开启 热点模式  
Wifi SSID name : cooneo_nvi  
Wifi SSID password: cooneo_nvi  
  
# 若使用热点模式 ssh 进 Jetson_nano  
ssh cooneo_nvi@10.42.1.1  
密码: cooneo_nvi
```

- 运行 建图 节点：

```
# 打开Jetson nano 中的终端 或者 ssh 进Jetson  
cd catkin_ws  
catkin_make -j  
source devel/setup.bash  
roslaunch launch_file gmapping_ekf.launch
```

- 然后打开我们配套虚拟机中的程序：

```
# 打开虚拟机中的一个终端  
cd catkin_ws  
source devel/setup.bash  
roslaunch remote_gmapping joy_gmapping.launch
```



- 保存地图

```
# 打开Jetson nano 中的终端 或者 ssh 进Jetson  
cd catkin_ws/src/launch_file/map/  
rosrun map_server map_saver -f map  
  
# 随后在该路径下会产生 两个文件，分别是 地图文件（map.pgm） 和 地图描述文件（map.yaml）
```

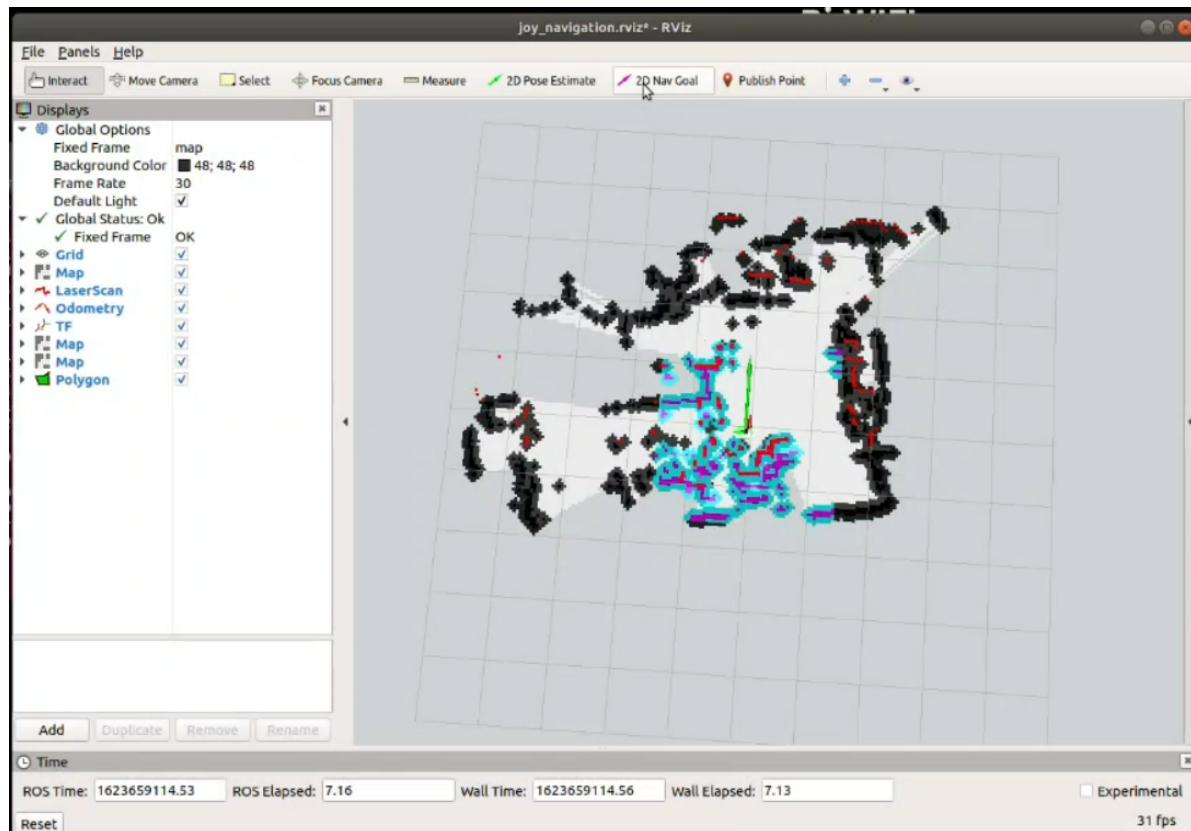
chapter 4: navigation_ekf

- 运行导航节点

```
# 打开Jetson nano 中的终端 或者 ssh 进Jetson
cd catkin_ws
source devel/setup.bash
roslaunch launch_file navigation_ekf.launch
```

- 打开虚拟机中配套的程序

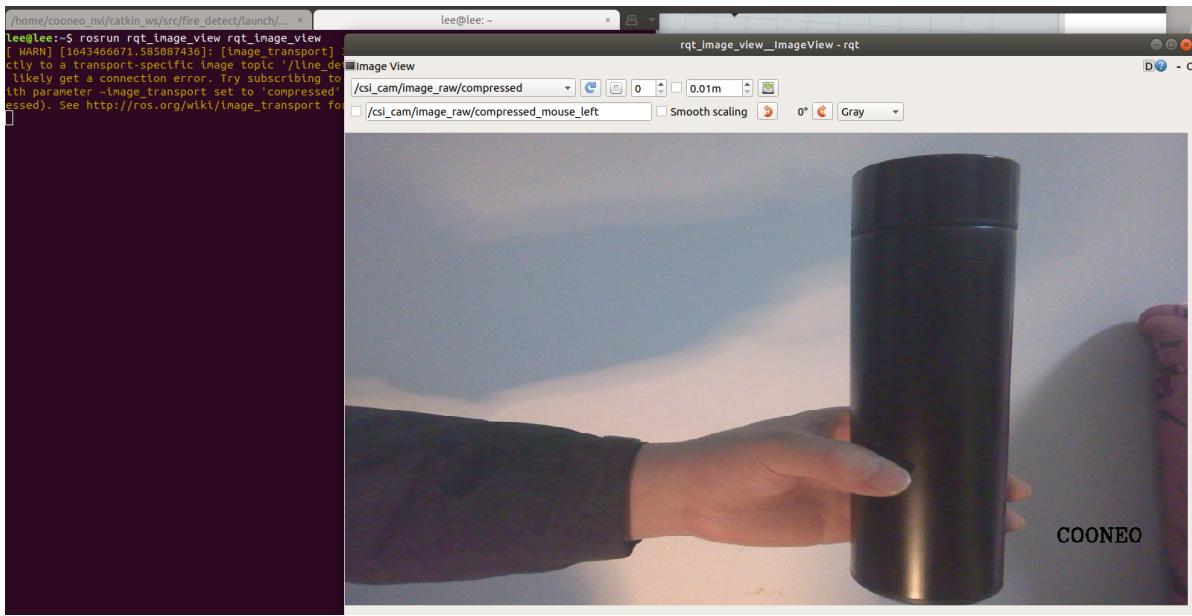
```
# 打开虚拟机中的一个终端
cd catkin_ws
source devel/setup.bash
roslaunch remote_gmapping joy_navigation.launch
```



chapter 5 : camera applications

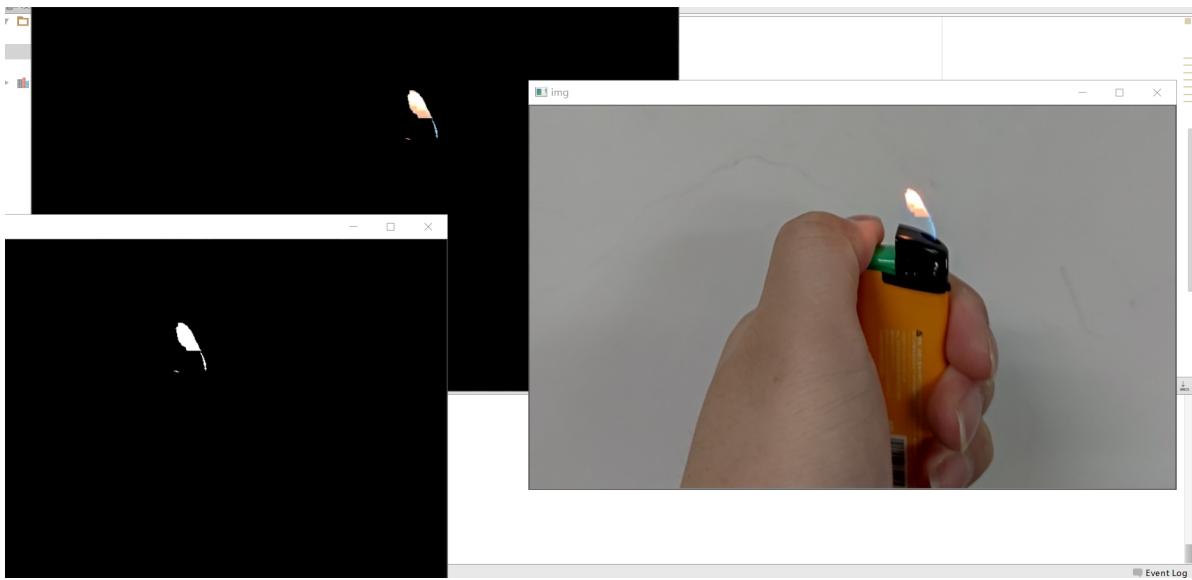
- 打开Jetson-nano 的CSI 摄像头

```
cd catkin_ws
source devel/setup.bash
roslaunch fire_detect Jetson_nano_csi_camera_node.launch
```



· 打开火焰检测节点

```
cd catkin_ws
source devel/setup.bash
roslaunch fire_detect fire_detect.launch
```



这里的火焰识别主要是基于颜色实现的 demo 级别功能，你可能需要根据你的 火苗大小动态修改代码中的值：

```
lower = [0, 43, 46]
upper = [10, 255, 255]
lower = np.array(lower, dtype="uint8")
upper = np.array(upper, dtype="uint8")
mask = cv2.inRange(hsv, lower, upper)

output = cv2.bitwise_and(frame, hsv, mask=mask)
no_red = cv2.countNonZero(mask)
#修改这个判别值
if int(no_red) > 40000:
    text = "Fire Detect"
```

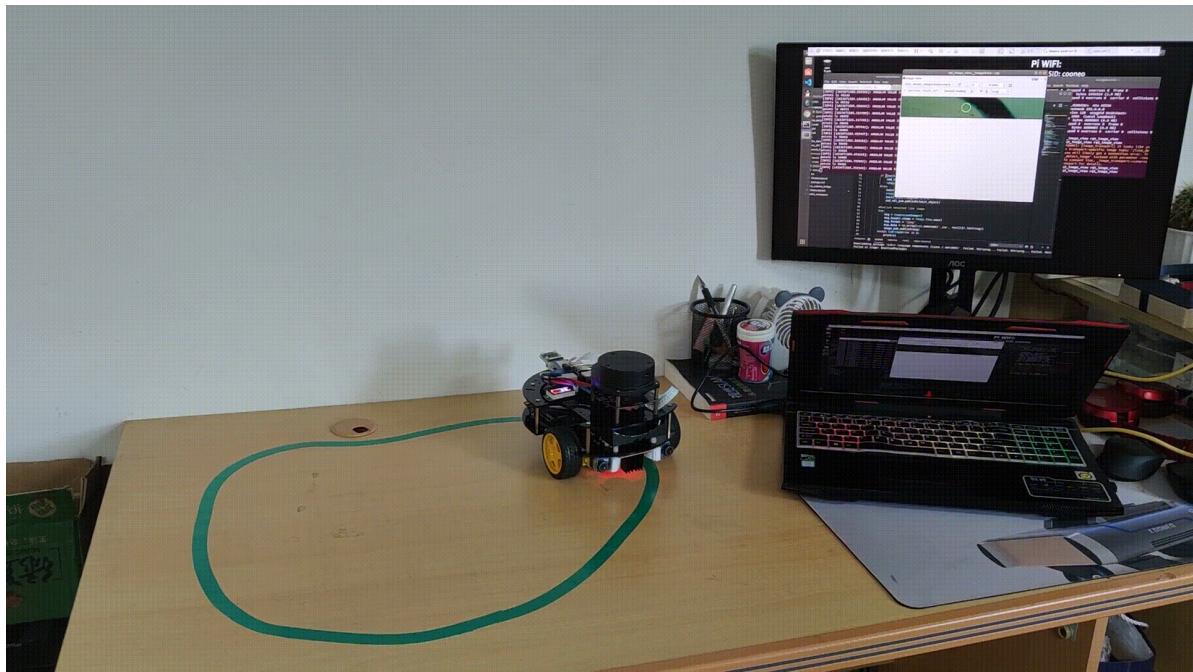
```
cv2.putText(frame, text, (100, 100), cv2.FONT_HERSHEY_COMPLEX, 2.0, (0, 0, 255), 5)
# cv2.imshow("output", frame)

else:
    cv2.putText(frame, "No Fire", (200, 100), cv2.FONT_HERSHEY_COMPLEX, 2.0, (0, 255, 0), 5)
    # cv2.imshow("output", frame)
print(int(no_red))
```

· 打开摄像头寻迹节点



```
cd catkin_ws
source devel/setup.bash
roslaunch linetrack linetrack_red.launch
```



这也是根据颜色来寻迹的，所以你需要根据你的巡线颜色修改 对应的HSV 颜色空间的范围。

```
...
# If your line color changes, it's not a red color,you need to change the HSV[min,max] value in
np.array[ , , ]
...
# 修改 对应的巡线颜色 HSV 范围
lower_red = np.array([35,43,46])
upper_red = np.array([77,255,255])
#Threshold the HSV image to get only yellow colors
mask = cv2.inRange(hsv,lower_red,upper_red)
#cv2.imshow("MASK",mask)
no_red = cv2.countNonZero(mask)
```

温馨提示：我们这个过程视频也放在 COONEO Bilibili 账号中了，欢迎观看。

详细的过程解读，请参见我们的微信公众号: "COONEO"。

2022.01.27

author:ZhaoXiang Lee

COONEO Co.,Ltd

Web:<http://cooneo.cc>

E: cooneo@outlook.com

For more details,you can search "COONEO" in your WeChat.



or search "COONEO" in Bilibili.