1. 主要就是turn\_on\_wheeltec\_robot这个package，

Param\_commom是一些通用的参数设置文件，而我们的小车用的是下面的param\_senior\_mec\_no这个独特的参数文件（因为有摆式悬挂小车，独立悬挂小车，差速，麦轮小车），turn\_on\_wheeltec\_robot.launch 里面可以看出来

1. Robot\_Vel应该是电机编码器解算得到的



1. robot\_ekf\_pos:接收轮式里程计、IMU数据、视觉里程计数据、GPS数据（ROS消息），基本思路就是集成耦合不同的传感器数据. 。每个源给出位态估计和协方差
2. ROS最常用的坐标系为map、odom、base\_link、base\_footprint、base\_laser坐标系。

1）base\_link机器人本体（基座）坐标系

机器人自身的坐标系，传感器对于其的tf就是传感器坐标系和机器人坐标系的关系。Base\_link是base\_footprint的子节点。ROS推荐坐标系（前左天）

2）base\_footprint坐标系

原点为base\_link原点在地面的投影，有些许区别（z值不同）。

3）odom

World-fixed，但是可能随着机器人的运动而漂移，所以不是一个长期有用的全局坐标，必须是连续变化，不能有突变和跳跃，一般是通过里程计来计算的，比如轮式编码器、VO、IMU,是一个短期的局域坐标系。

4）map

一般与机器人所在的世界坐标系一致。Map的z轴向上，机器人相对于map坐标系的坐标不应该随着时间漂移，可以由AMCL、视觉定位等来进行纠正，可以进行突变，所以是一个比较差的局部坐标系（不宜用于避障和局部操作）

坐标系的约定：

开放环境中的全球坐标系：默认x向东，y向北，z向上，z轴的值为WGS84椭球的海拔高度

在结构化的环境中（比如室内），在定义坐标系时和环境保持对应更有用。比如对于有平面图的建筑，坐标系可以和平面图对应。类似的对于室内环境地图可以和建筑物的层相对应。对于有多层结构的建筑物，对每一层单独有一个坐标系也是合理的。

5）earth

多机器人交互设计的，多个机器人会有多个map坐标系，但是之间的关系需要earth表示，才能进行数据共享

如果map是一个全局坐标系，则map到earth的变换是一个静态变换。如果不是，就需要实时计算。

6）base\_laser激光雷达坐标系

与激光雷达的安装点有关，其与base\_link的tf为固定的。

7）坐标系之间的关系

可以用树来表示，每个坐标系只有一个父坐标系，可以多个子坐标系。按道理来说map和odom都是base\_link的父坐标系，但是不符合只有一个父坐标系的规则。

earth -> map -> odom -> base\_link

8）坐标系之间的计算

odom到base\_link的变换由里程计数据源中的一个发布

map到base\_link通过定位组件计算得出。但是定位组件并不发布从map到base\_link的变换。它首先获取odom到base\_link的变换然后利用定位信息计算出map到odom的变换。

earth到map的变换是根据map坐标系选取所发布的一个静态变换。如果没有设置，那么就会使用机器人的初始位置作为坐标原点。

1. launch file ：remap

有一个节点需要订阅“chatter” topic，但是现在只有“hello” topic，但是两者类型一致，则可以使用<remap from=”chatter” to=”hello”/>

1. ros打印错误信息，并且会显示红色

try{

Stm32\_Serial.setPort(usart\_port\_name);//选择哪个口，如果选择的口没有接串口外设初始化会失败

Stm32\_Serial.setBaudrate(serial\_baud\_rate);//设置波特率

serial::Timeout \_time = serial::Timeout::simpleTimeout(2000);//超时等待

Stm32\_Serial.setTimeout(\_time);

Stm32\_Serial.open();//串口开启

}

catch (serial::IOException& e){

ROS\_ERROR\_STREAM("wheeltec\_robot can not open serial port,Please check the serial port cable! ");//如果try失败，打印错误信息

}

1. 代码的理解：

Wheeltec\_robot.cpp里面详细写了如何将底层STM32串口传输的车辆状态数据进行解析，然后将解析得到的IMU、odom数据用topic发出来，由rosbot\_pose\_ekf包去融合这些信息，得到机器人的位姿信息，rosbot\_pose\_ekf.launch只使用了odom和imu数据，没有使用vo和gps。然后就得到了机器人的位置

机器人的控制通过订阅/ cmd\_vel 话题，回调函数为void turn\_on\_robot::Cmd\_Vel\_Callback(const geometry\_msgs::Twist &twist\_aux)，twist\_aux里面为机器人x轴、y轴目标线速度，z轴目标角速度，所以我们控制机器人就只需要发送Vx、Vy、w。

也会订阅AMCL的数据，但是订阅了之后并没有利用这个消息做事情

建图的时候也会同时开启turn\_on\_wheeltec\_robot.launch和algorithm\_gmapping.launch，即同时开启机器人、启用gmapping建图

他的navigation.launch利用就调用turn\_on\_wheeltec\_robot.launch（启动小车），map\_server\_for\_test（提供地图），amcl\_omni.launch（全局定位算法），tel\_local\_planner.launch（局部路径规划）

整个代码应该是amcl定位得到车辆位姿，然后会发送给teb\_local\_planner，然后teb\_local\_planner根据全局地图，根据全局路径规划和局部路径规划，得到车辆需要的速度和角速度，以“cmd/vel”的topic发出去，然后turn\_on\_wheeltec\_robot收到这个消息，然后通过串口发送给小车

车辆坐标原点在车辆中心，右前天？？而雷达的安装根据rplidar说明书是后右地，所以他是不是改了驱动，或者是雷达的参数文件改了一下