自己动手做一台 SLAM 导航机器人

第三章：感知与大脑

作者：

知乎@小虎哥哥爱学习

目录

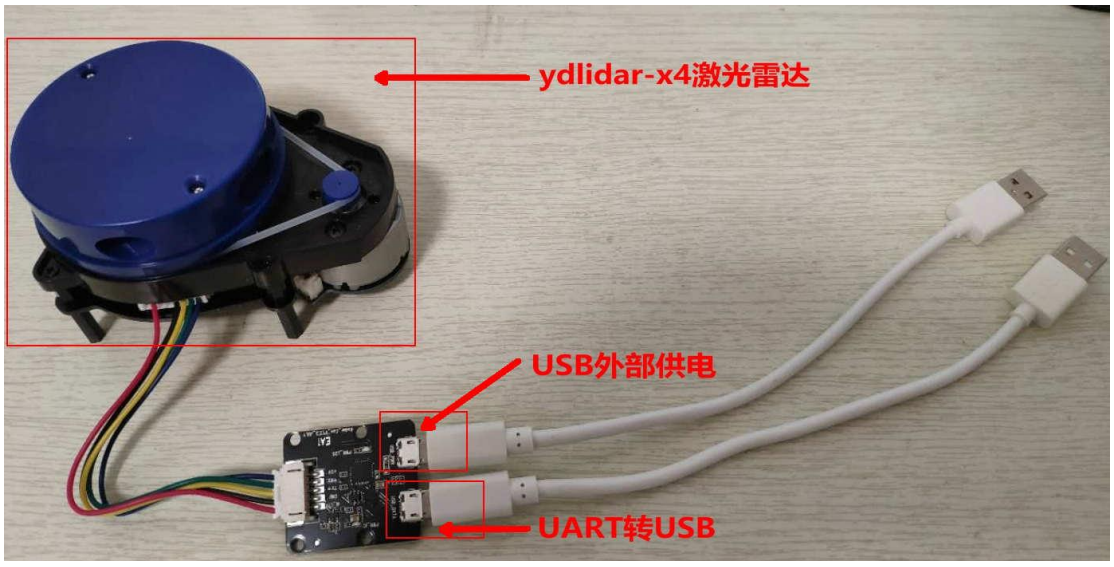
在我的想象中机器人首先应该能自由的走来走去，然后应该能流利的与主人对话。朝着这个理想，我准备设计一个能自由行走，并且可以与人语音对话的机器人。实现的关键是让机器人能通过传感器感知周围环境，并通过机器人大脑处理并输出反馈和执行动作。本章节涉及到的传感器有激光雷达、IMU、轮式里程计、麦克风、音响、摄像头，和用于处理信息的嵌入式主板。关于传感器的 ROS 驱动程序开发和在机器人上的使用在后面的章节会展开，本章重点对机器人传感器和嵌入式主板进行讲解，主要内容：

1.ydlidar-x4 激光雷达
2.带自校准九轴数据融合 IMU 惯性传感器
3.轮式里程计与运动控制
4.音响麦克风与摄像头
5.机器人大脑嵌入式主板性能对比
6.做一个能走路和对话的机器人

1. ydlidar-x4 激光雷达

在移动机器人中，获取机器人周围障碍物和环境的轮廓形状是非常重要的。使用激光雷达正是为了实现这个目的。利用扫描得到的障碍物信息，机器人就可以利用 SLAM 建立地图、并进行避障和自主导航。考虑到成本，推荐大家选用低成本的 2D 激光雷达，这里推荐的 ydlidar-x4 激光雷达正是一款极低成本的 2D 激光雷达，作为学习性能足够用了。

1.1.硬件概述



（图 1）ydlidar-x4 激光雷达实物

雷达主体由激光测距模组和电机构成，雷达主体需要连接转接板，用于实现外部供电和 UART 转 USB，如图 1。

项目	最小值	典型值	最大值	单位	备注
测距频率	-	5000	-	Hz	每秒测距 5000 次
扫描频率	6	-	12	Hz	PWM 或电压调速
测距范围	0.12	-	>10	m	室内环境
扫描角度	-	0~360	-	Deg	-
测距分辨率	-	<0.5	-	mm	测距范围<2m
		<实际距离的 1%			测距范围>2m
角度分辨率	0.48	0.50	0.52	Deg	扫描频率 为 7Hz 时
使用寿命	-	1500	-	h	持续工作寿命

（图 2）ydlidar-x4 激光雷达性能参数

测距频率是指每秒钟测距的次数（即激光测距模组的采样率），测距频率默认为 5KHz；扫描频率是指电机每秒钟转过的圈数，扫描频率默认为 8Hz；测距范围是指落在此距离范围的障碍物才能被测量，测距范围为 0.12~10m；扫描角度为 0~360 度，也就是能够 360 度全方位扫描；测距分辨率就是测距精度；角度分辨率就是两个相邻扫描点之间的夹角。ydlidar-x4 激光雷达性能参数，如图 2。

关于激光雷达的性能参数对 SLAM 建图与避障导航的影响，这里做一个简短的分析。

扫描频率：

扫描频率越高，电机转动一圈的时间约短，扫描获取 1 帧雷达数据的时间越短，这样可以使 SLAM 建图和避障导航实时性更好。简单点说就是机器人运动速度很快时，扫描到的点云数据连续性也比较好，这有利于 SLAM 算法建立稳定的地图，有利于避障导航算法及时发现并避开障碍物。

测距范围：

小于测距范围最小值的区域就是雷达的测量盲区，处于盲区中的障碍物无法被探测，所以，盲区当然是越小越好以保证机器人不发生碰撞；大于测距范围最大值的区域就是雷达超量程的区域，在超量程的区域的障碍物无法被探测或者可以探测但误差很大，所以，在比较开阔的环境下应该采用远距离量程的雷达。

扫描角度：

大部分雷达都是 0~360 度全方位扫描的，所以就没什么太大的区别了，其实就是机器人不用转动身体的情况下就能一次性扫描出四周的障碍物信息。

测距分辨率：

测距分辨率也就是测距精度，测距精度越高当然有利于 SLAM 建图和避障导航，但是测距精度越越高的雷达成本当然也越高，现在国产低成本的雷达普遍为厘米级（cm）的精度，差一点的雷达 5cm 左右的精度，稍微好一点的雷达 2cm 左右的精度，如果要达到毫米级（mm）的精度成本就非常高了。

角度分辨率：

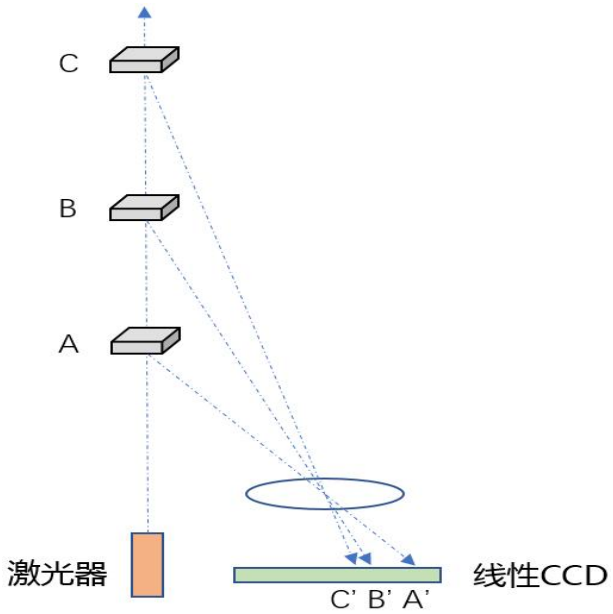
角度分辨率决定了两个相邻点云之间的夹角，由于雷达是通过旋转进行扫描的，随着距离增加点云会越来越稀疏。如果角度分辨率比较低，在扫描远距离物理时只能得到非常稀疏的几个点云，这样的点云基本上没有什么用处了。

角度分辨率 = 360 /（测距频率 / 扫描频率）

从上面的角度分辨率计算公式来看，一般测距频率为常数值（由激光模组特性决定），那么通过降低扫描频率可以提升角度分辨率，但同时扫描频率降低会影响雷达的实时性，所以这是一个权衡的过程，根据实际情况做选择。

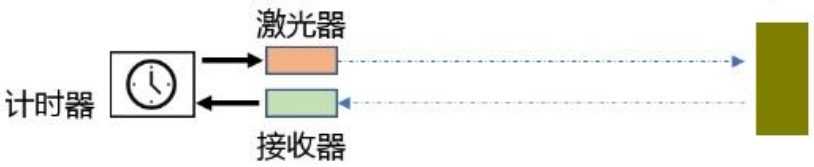
1.2.工作原理

激光雷达之所以流行，得益于它能够精确的测距。主流的激光雷达基于两种原理：一种是三角测距法，另一种是飞行时间（TOF）测距法。其实很好理解，就是利用了最基本的数学与物理知识。



（图 3）三角测距原理

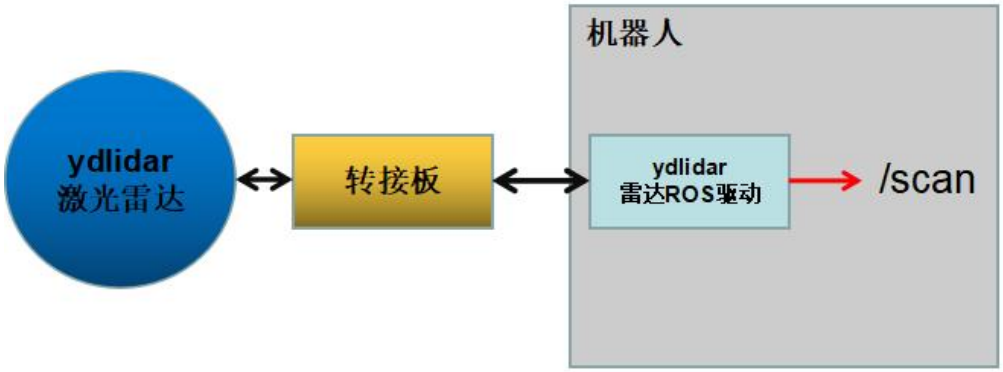
三角测距原理如图 3，激光器发射一束激光，被物体 A 反射后，照射到图像传感器的 A'，这样就形成了一个三角形，通过解算可以求出物体 A 到激光器的距离。激光束被不同距离的物体反射后，形成不同的三角形。我们不难发现随物体距离不断变远，反射激光在图像传感器上的位置变化会越来越小，也就是越来越难以分辨。这正是三角测距的一大缺点，物体距离越远，测距误差越大。



（图 4）TOF 测距原理

飞行时间（TOF）测距原理如图 4，激光器发出激光时，计时器开始计时，接收器接收到反射回来的激光时，计时器停止计时，得到激光传播的时间后，通过光速一定这个条件，很容易计算出激光器到障碍物的距离。由于光速传播太快了，要获取精确的传播时间太难了。所以这种激光雷达自然而然成本也会高很多，但是测距精度很高。

1.3.在机器人中使用 ydlidar-x4 激光雷达

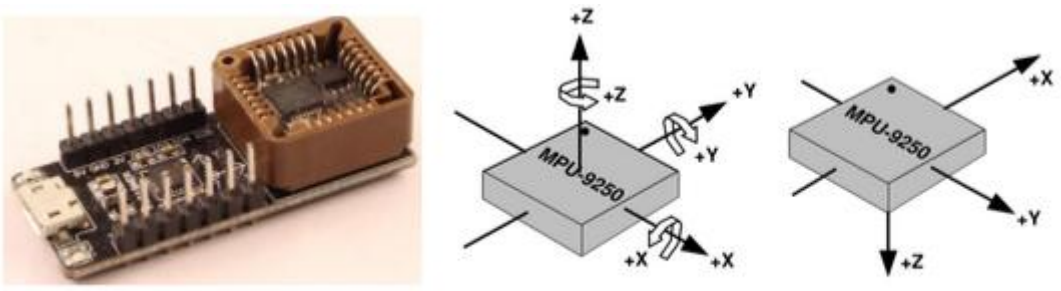


（图 5）在机器人中使用 ydlidar-x4 激光雷达

ydlidar-x4 激光雷达通过串口与机器人相连接，机器人中通过运行雷达 ROS 驱动，来实现读取串口的雷达数据和将雷达数据发布到/scan 这个主题，这样机器人上的其它节点就可以通过订阅/scan 主题来获取激光雷达数据了。关于雷达 ROS 驱动在机器人上的具体使用，将在后面的章节中具体讲解。

2. 带自校准九轴数据融合 IMU 惯性传感器

IMU 是惯性测量单元的简称，用于测量物体的三轴姿态角（roll、pitch、yaw）、三轴加速度（acc_x、acc_y、acc_z）、三轴角速度（w_x、w_y、w_z）等。IMU 惯性测量单元在制造过程中，由于物理因素，导致 IMU 惯性测量单元实际的坐标轴与理想的坐标轴之间会有一定的偏差，同时三轴加速度、三轴角速度、三轴磁力计的原始值会与真实值有一个固定的偏差等。这里提到的自校准就是要通过给的补偿值来减小或消除坐标轴的偏差及原始值的固定偏差，也就是所谓的 IMU 内部标定。如果将 IMU 安装到机器人或摄像头上后，需要知道 IMU 与机器人或摄像头的相对位置，这个时候进行的标定就是所谓的 IMU 外部标定。特此说明，这里提到的自校准指 IMU 内部标定。这里提到的九轴数据融合，是指通过三轴加速度、三轴角速度数据融合得到更加精准的三轴加速度、三轴角速度，同时通过三轴加速度、三轴角速度、三轴磁力计数据融合得到测量物体的三轴姿态角。选用一款带自校准九轴数据融合的 IMU，能很好的提升机器人的 SLAM 建图与导航性能，同时降低机器人上软件的开发难度。出于这一点，我们选用了一款基于 MPU9250 的.带自校准九轴数据融合 IMU，如图 6。



（图 6）带自校准九轴数据融合 IMU 惯性传感器

2.1.自校准

IMU 误差模型：

误差主要来自三部分：噪声（bias and noise）、尺度误差（scale error）、轴偏差（axis misalignment）。通过对这些误差的补偿来实现 IMU 测量数据的校准，校准的数学表达如图 7。

acc_calib = Ta*Sa*(acc+Ba+Na)

w_calib = Tw*Sw*(w+Bw+Nw)

mag_calib = Tm*Sm*(mag+Bm+Nm)

mag_calib2 = Tm2a*mag_calib

其中，加速度原始测量值向量 acc 加上零偏补偿 Ba 和白噪声 Na 补偿后，再用尺度矫正矩阵 Sa 和轴偏差矫正矩阵 Ta 变换后，就可以得到校准后的三轴加速度向量 acc_calib；通过类似的方法可以得到校准后的三轴角速度 w_calib 和三轴磁力 mag_calib；最后还要将磁力计坐标系下的磁力数据 mag_calib 变换到加速度坐标系，变换得到的 mag_calib2 将用于加速度/磁力计的数据融合。

（图 7）校准的数学表达

噪声部分考虑零偏 Bias 和高斯白噪声 noise。零偏 Bias 也叫随机游走，一般是由传感器内部构造、温度变换多方面综合影响的结果；高斯白噪声 noise，一般是由于 AD 转换引起的一种外部噪声。

尺度误差部分，来自于 AD 转换中量化过程，比如采样电压值 1V 对应 acc(x)轴的 1.4g，同样采样电压值 1V 对应 acc(y)轴的 1.6g，也就是说不通的轴上 AD 转换量化是不同的。

轴偏差部分，三轴加速度、三轴角速度、三轴磁力计的坐标轴严格上都不是正交坐标系。但是最后我们期望的使用值默认是在正交坐标系下测量的，所以就需要将在非正交坐标系测量的原始值变换到正交坐标系中。

加速度校准：

校准过程中需要判断传感器是否处于静止状态，其实很简单，在时间 t 内（t 一般取 50s），分别计算 acc(x)、acc(y)、acc(z)三轴数据的方差 var[acc(x)]、var[acc(y)]、var[acc(z)]，如果 var[acc(x)]+var[acc(y)]+var[acc(z)]小于阈值 H（H 为经验值，需要通过实验法确定），就判断传

感器静止。

在静止状态下，加速度计测量值的二范数等于当地重力加速度 g。在这一约束条件下，利用最小二乘法进行优化问题求解，由于加速度采样值取得是一个小窗口采样区间的平均值，所以可以忽略白噪声 Na 的影响，便可以求解出待标定参数 Ta、Sa、Ba。在静止状态取一串加速度采样值共 M 个，构建代价函数如图 8，对代价函数进行最优化求解即可，可以选用 ceres 或 g2o 等优化工具来完成具体的优化计算过程。

$$\min \sum_{i=1}^{M} (\|g\|^2 - \|Ta*Sa*(acc+Ba)\|^2)^2$$

（图 8）代价函数

角速度校准：

角速度校准分为两部分：用 Allan 方差校准零偏 Bias、用最优化方法求解尺度误差 Sw 和轴偏差 Tw。加速度校准中需要用到加速度校准信息，所以加速度校准好坏影响整个 IMU 校准。

同加速度校准一样，也需要在静止状态下采集角速度，简便的做法是在采集加速度的同时也采集角速度。利用角速度采样值计算 Allan 方差，Allan 方差计算比较繁琐就不展开，请直接参考 wiki 百科：https://en.wikipedia.org/wiki/Allan_variance。

Allan 方差中共有 5 个噪声参数：量化噪声 Q、角度随机游走 N、零偏 B、角速度随机游走 K、角速度斜坡 R。通过图 9 的步骤便可以求出角速度的零偏 Bw，由于角速度采样时是一个小窗口采样区间的平均值，所以可以忽略白噪声 Nw 的影响。

$$\sigma_{total}^2(\tau) = \sigma_Q^2(\tau) + \sigma_N^2(\tau) + \sigma_B^2(\tau) + \sigma_K^2(\tau) + \sigma_R^2(\tau)$$
$$= \frac{3Q^2}{\tau^2} + \frac{N^2}{\tau} + \frac{2B^2}{\pi} ln2 + \frac{K^2\tau}{3} + \frac{R^2\tau^2}{2}$$

其实，Allan 方差就是各部分误差的平方和，Allan 方差进一步整理

$$\sigma_{total}^2(\tau) = \sum_{i=-2}^{2} C_i \tau^i$$

在 Allan 方差的计算中，我们已经有不同T取值下 $\sigma_{total}^2(\tau)$ 值，这样就可以通过小二乘法进行优化问题求解，求出 C_i 的值。利用 C_i 可以反求出各噪声参数

$$\begin{cases} Q = \frac{\sqrt{C_{-2}}}{3600\sqrt{3}}(^\circ) \\ N = \frac{\sqrt{C_{-1}}}{60}(^\circ)\sqrt{h} \\ B = \frac{\sqrt{C_0}}{0.664}(^\circ)h \\ K = 60\sqrt{3C_1}[(^\circ)\cdot h^{-1}]/\sqrt{h} \\ R = 3600\sqrt{2C_2}[(^\circ)\cdot h^{-1}]/h \end{cases}$$

这样，我们就求出了零偏 B

（图 9）用 Allan 方差校准零偏 Bias

角速度的零偏 Bw 校准完后，将零偏值 Bw 带入，接着校准尺度误差 Sw 和轴偏差 Tw。挑选加速度校准过程中两静止状态夹杂的动态片段，这样经过校准后的静态值的平均值可作为 acc_calib 向量的起始值 acc_calib(k-1)和旋转完成后的结束值 acc_calib(k)。

通过图 10 的步骤便可以求出角速度的尺度误差 Sw 和轴偏差 Tw。

将加速度起始值 acc_calib(k-1)用 $U_{a,k-1}$ 表示

将加速度结束值 acc_calib(k)用 $U_{a,k}$ 表示

n 个离散时间角速度向量测量值 Wi，i=1,2,...,n

使用四阶龙格库塔法（RK4n）对离散时间角速度积分可以得到旋转后的姿态角，记操作 P

由于角速度最终校准值 w_calib=Tw*Sw*(w+Bw)，由角速度积分得到的加速度旋转结束值

$U_{g,k} = P[\ Tw*Sw*(Wi+Bw)\ , U_{a,k-1}\ ]$

通过角速度积分得到的加速度 $U_{g,k}$ 应该尽量逼近加速度计的测量值 $U_{a,k}$，构造代价函数

$$\min \sum_{k=2}^{M} \left\| u_{a,k} - u_{g,k} \right\|^2$$

采集 M 组两静止状态夹杂的动态片段，利用最小二乘法进行优化问题求解，便可以求出待标定参数 Tw 和 Sw

（图 10）求角速度的尺度误差 Sw 和轴偏差 Tw

磁力计校准：

与加速度校准比较类似，磁力计测量值的二范数等于当地磁场强度 m，不同的是磁力的测量不需要在静止状态。在这一约束条件下，利用最小二乘法进行优化问题求解，由于磁力采样值取得是一个小窗口采样区间的平均值，所以可以忽略白噪声 Nm 的影响，便可以求解出待标定参数 Tm、Sm、Bm。构建代价函数如图 11，对代价函数进行最优化求解即可。

$$\min \sum_{i=1}^{M} \left(\|m\|^2 - \|Tm*Sm*(mag+Bm)\|^2 \right)^2$$

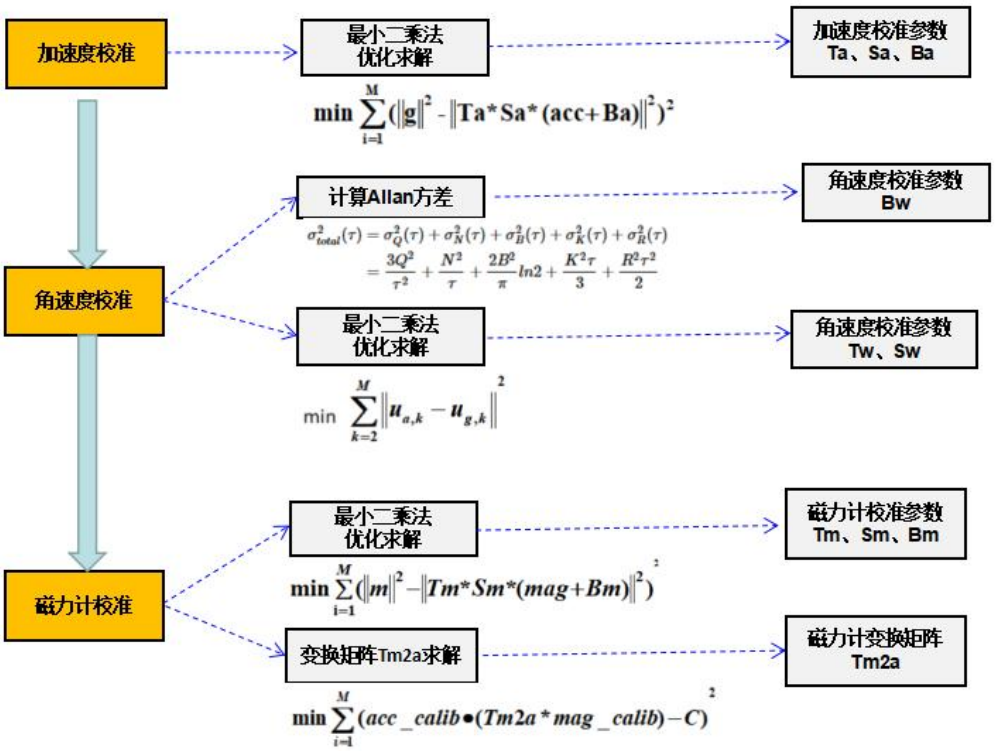
（图 11）代价函数

由于需要用加速度/磁力计进行融合，需要将磁力计测量值变换到加速度坐标系。变换矩阵 Tm2a 求解比较简单，通过多组关联的加速度 acc_calib 和磁力计 mag_calib，通过最小二次法优化求解，可以求出变换矩阵 Tm2a。这里基于一个假设，磁力向量与加速度向量成固定夹角，也就是两向量的点乘为一个常数 C，利用这个约束可以构建代价函数如图 12，对代价函数进行最优化求解即可。

$$\min \sum_{i=1}^{M} (acc\_calib \bullet (Tm2a * mag\_calib) - C)^2$$

（图 12）代价函数

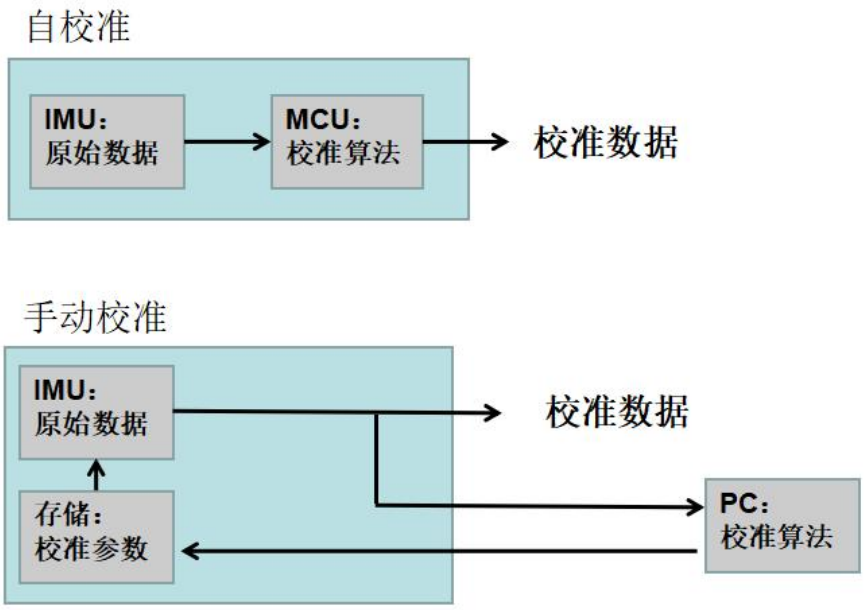
IMU 校准软件流程图：



（图 13）代价函数

IMU 手动校准与自校准对比：

通常的校准过程是，将 IMU 接入 PC 端，手动将 IMU 置不同的状态，PC 端通过采集这些数据，完成校准。另一种校准过程是，将校准算法内置在 IMU 模块上的 MCU 中，在 IMU 工作的过程中实时采集数据，并自动完成校准，不需要人为的干预。本文介绍的 MPU9250 模块就属于自校准这一方式。手动校准与自校准对比如图 14。



（图 14）手动校准与自校准对比

2.2.九轴数据融合

IMU 的校准完成后，就可以利用校准后的九轴数据进行数据融合，得到 IMU 的空间姿态了。

MPU9250 模块包括三轴加速度计、三轴陀螺仪和三轴磁力计。因为加速度/磁力计具有高频噪声，也就是说它们的瞬时值不够精确，解算出来的姿态会震荡，但长期来看姿态方向是对的。而陀螺仪具有低频噪声，即每个时刻的得到的角速度是比较精确的，使用积分就能得到旋转角度（姿态），但是积分会累积误差，因此积分到后面姿态就不对了，也就是漂移现象。 加速度/磁力计和陀螺仪在频域上的特性互补，可以融合这三种传感器的数据，提高精度和系统的动态特性。

由于欧拉角在求解姿态时存在万向节锁，不能用于全姿态解算；故采用四元数 Quaternion 进行姿态解算。常用的九轴数据融合算法包括：高低通互补滤波、扩展卡尔曼滤波 EKF、Mahony 滤波。在实际情况中根据不同的用途进行选择，这里对这三种滤波方法进行原理性的介绍。

高低通互补滤波：

$$Quaternion(t) = (1-\alpha)\hat{Quaternion}(t) + \alpha \cdot accMagQuaternion(t)$$

$$\hat{Quaternion}(t) = Quaternion(t-1)gyroSensorQ(t)$$

其中，Quaternion(t)是经融合后的姿态四元数

$\hat{Quaternion}(t)$ 是通过角速度积分推算出来的估计姿态四元数

accMagQuaternion(t)是加速度/磁力计融合得到的姿态四元数

gyroSensorQ(t)是陀螺仪测量到的角速度积分得到的旋转量的四元数

α是融合陀螺仪估计姿态和加速度/磁力计融合姿态的加权系数，为达到高低通互补滤波效果，一般取比较小的值，如α=0.1

（图 15）高低通互补滤波的过程

高低通互补滤波的过程如图 15,通过加权系数融合陀螺仪估计姿态和加速度/磁力计融合姿态，达到对陀螺仪低通滤波、对加速度/磁力计高通滤波的效果。

扩展卡尔曼滤波 EKF：

和高低通互补滤波过程类似，也是分为估计、校正。

Predict [edit]

Predicted state estimate $\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k)$

Predicted covariance estimate $\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_k \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}_k^\top + \boldsymbol{Q}_k$

Update [edit]

Innovation or measurement residual $\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})$

Innovation (or residual) covariance $\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top + \boldsymbol{R}_k$

Near-optimal Kalman gain $\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1} \boldsymbol{H}_k^\top \boldsymbol{S}_k^{-1}$

Updated state estimate $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \tilde{\boldsymbol{y}}_k$

Updated covariance estimate $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_{k|k-1}$

where the state transition and observation matrices are defined to be the following Jacobians

$$\boldsymbol{F}_k = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_k}$$

$$\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}}$$

（图 16）EKF 滤波的过程

EKF 滤波的过程如图 16，想要了解 EKF 滤波的详细推导请直接参考 wiki 百科：https://en.wikipedia.org/wiki/Extended_Kalman_filter。估计过程中，利用陀螺仪积分推算出估计姿态；在校正过程中，计算估计姿态下的加速度/磁力计，再用在实际姿态下测量的加速度/磁力计按照 EKF 校正方程进行校正。

Mahony 滤波：

同样，Mahony 滤波也分为估计、校正。主要使用了 PI 控制思想。

$$\dot{\hat{q}} = \frac{1}{2}\hat{q} \otimes \mathbf{P}(\overline{\Omega} + \delta) \quad ---------- (a)$$

$$\delta = k_P \cdot e + k_I \cdot \int e \quad ---------- (b)$$

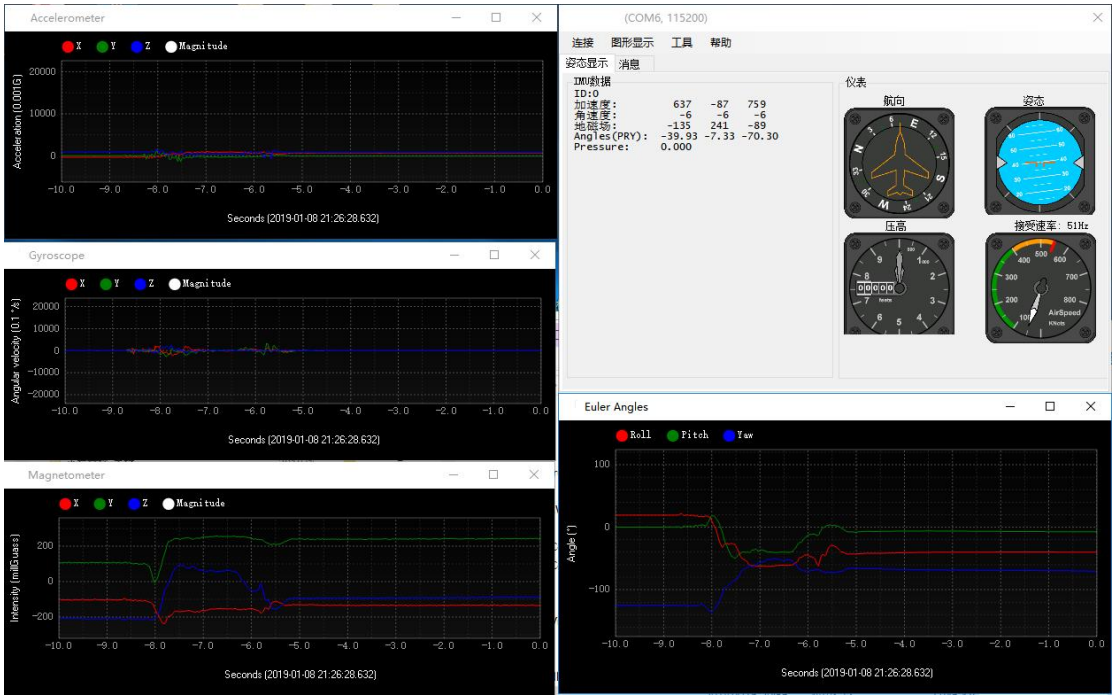
$$e = \bar{v} \times \hat{v} \quad ---------- (c)$$

式中，$\hat{q}$ 表示系统姿态估计的四元数表示；δ 是经过 PI 调节器产生的新息；e 表示实测的惯性向量 $\bar{v}$ 和预测的向量 $\hat{v}$ 之间的相对旋转（误差）。

$\mathbf{P}(\cdot)$ —— pure quaternion operator（四元数实部为0），表示只有旋转。

（图 17）Mahony 滤波的过程

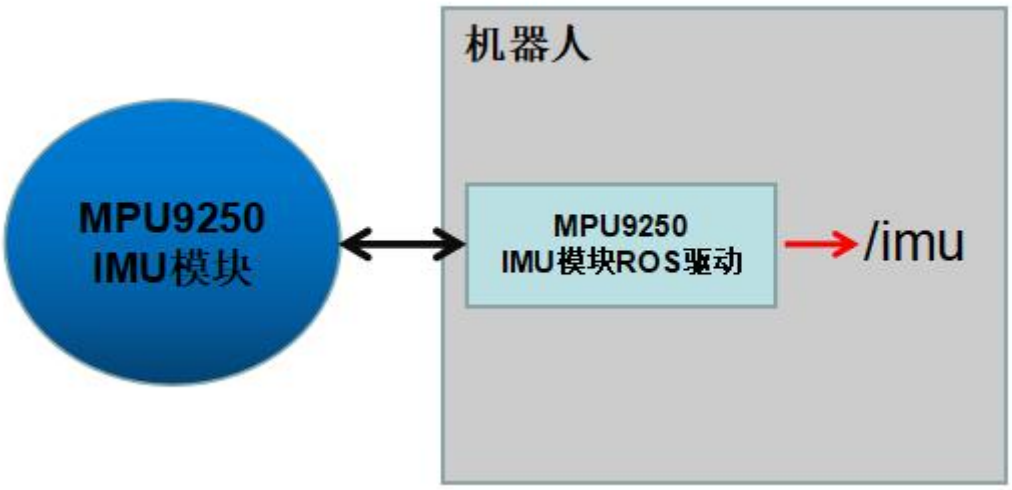
Mahony 滤波的过程如图 17。在估计过程中，同样利用陀螺仪积分推算出估计姿态；在校正过程中，计算估计姿态下的加速度/磁力计，测量在实际姿态下的加速度/磁力计，这两个向量叉乘便可以得到姿态误差 e，姿态误差 e 通过 PI 控制过程可以实现对陀螺仪测量角速度的补偿，利用补偿后的角速度重新估计出来的姿态，就是滤波后的姿态了。



（图 18）MPU9250 模块九轴数据融合后的效果

MPU9250 模块九轴数据融合后的效果如图 18，图中左侧显示的分别是加速度、角速度、磁力计的实时数据，图中右侧显示的是融合后的姿态实时数据，可以看出融合后的姿态还是非常平稳的。

2.3.在机器人中使用 IMU



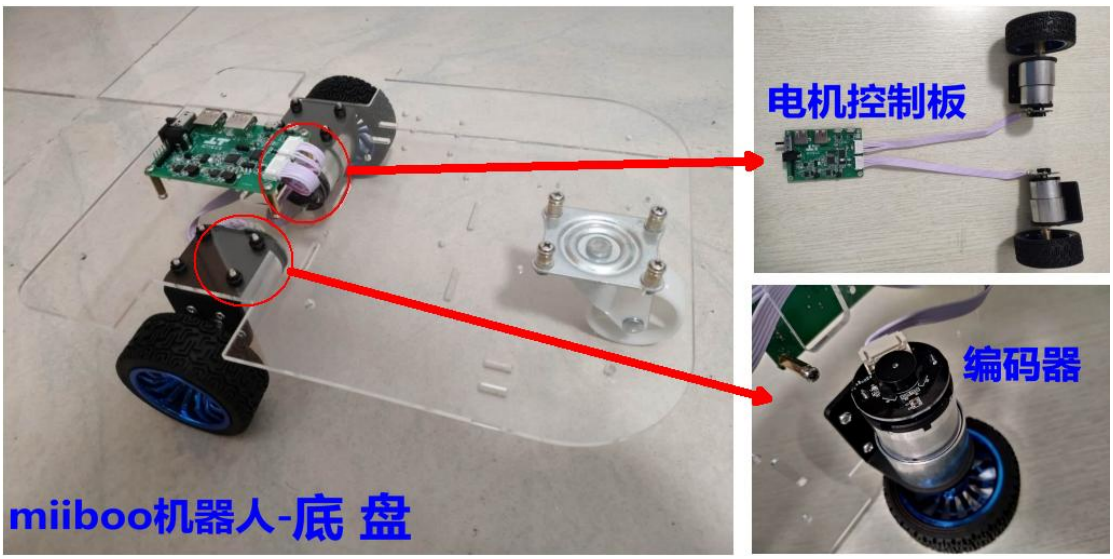
（图 19）在机器人中使用 MPU9250-IMU 模块

MPU9250-IMU 模块通过串口与机器人相连接，机器人中通过运行 IMU 的 ROS 驱动，来实现读取串口的 IMU 数据和将 IMU 数据发布到/imu 这个主题，这样机器人上的其它节点就可以通过订阅/imu 主题来获取 IMU 数据了。关于 IMU 的 ROS 驱动在机器人上的具体使用，将在后面的章节中具体讲解。

3. 轮式里程计与运动控制

底盘提供轮式里程计与运动控制，是机器人 SLAM 建图与避障导航的基础。所以对底盘进行了解，和熟悉轮式里程计与运动控制的底层原理是很有益处的。这里以我们的 miiboo 机器人的底盘为例，对底盘上的轮式里程计和运动控制的原理进行分析。

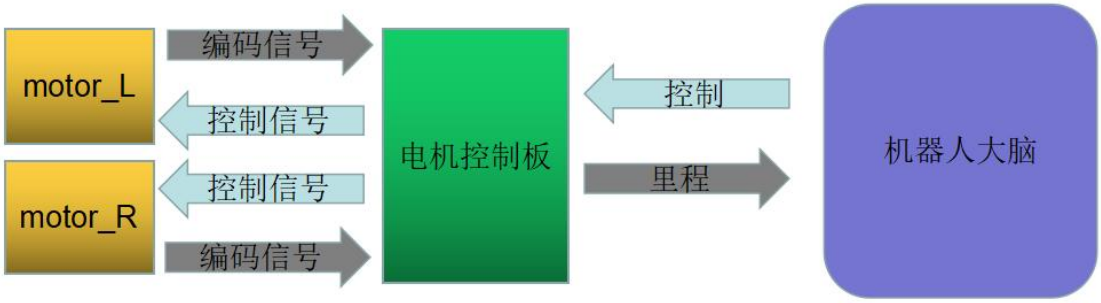
3.1.硬件概述



（图 20）miiboo 机器人底盘

底盘主要由电机控制板和带编码器的减速电机构成，如图 20。电机控制板通过串口与机器人的大脑（如树莓派 3）相连接，通过接收大脑下发的控制指令，利用 PID 算法对电机进行控制；同时，采集电机上的编码器数据发送给大脑，利用航迹推演算法得到底盘的里程计信息。

3.2.轮式里程计与运动控制



（图 21）轮式里程计与运动控制

如图 21，为轮式里程计与运动控制的系统框图。首先是机器人大脑发送控制命令，其实就是期望左、右电机达到的目标转速，我们都知道在一个控制系统中，被控对象很难完全按照期望目标来运行，这就需要引入反馈对被控对象进行实时的闭环控制，让被控对象尽量

逼近期望目标，电机控制板主要就是用来实现这个过程。同时，电机控制板还负责对电机编码信号进行采样，将单位采样时间（一般为 10ms）内的编码脉冲累计值作为里程数据发送给机器人大脑，机器人大脑利用航迹推演算法求解出里程计信息。

通信协议：

电机控制板与机器人大脑之间采用串口通信。电机左、右轮期望转速被封装到串口的字符串中，作为控制命令发送给电机控制板；单位时间（一般 10ms）内采样到的电机编码脉冲累计值（等效为实际电机速度）作为里程数据，以同样的方式被封装到串口的字符串中发送给机器人大脑。可以看出，控制命令与里程数据遵循一样的封装协议，协议具体形式如图 22。

帧头		左轮速度				右轮速度				校验和
top		enc1_sig	enc1_val			enc2_sig	enc2_val			checksum
ff	ff	xx	xx	xx	xx	xx	xx	xx	xx	xx

数据帧由 11 个字节组成，如上表所示，从左往右依次定义为：

top[0],top[1]:帧头，固定取值 ff ff

enc1_sig：左轮速度符号位，速度为负时取 0，否则取非 0 值

enc1_val：左轮速度，依次为高、中、低位，拼起来得到一个 24 位的正整数

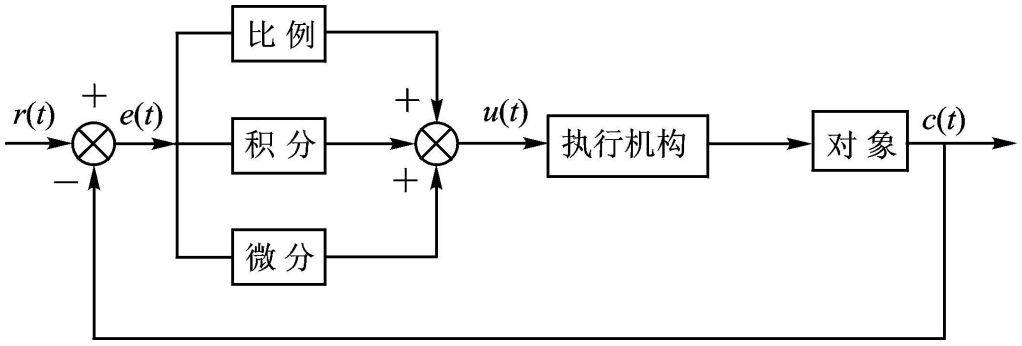
enc2_sig：右轮速度符号位，速度为负时取 0，否则取非 0 值

enc2_val：右轮速度，依次为高、中、低位，拼起来得到一个 24 位的正整数

checksum：前面所有字节求累加和取低 8 位

（图 22）通信协议

电机控制：



（图 23）PID 算法流程

电机控制最常用的就是 PID 控制算法，如图 23 为 PID 算法流程。以电机转速控制为例，r(t)就是给定的目标转速，c(t)就是电机实际运行时的转速，通过闭环反馈可以求得 r(t)与 c(t)的偏差值 e(t)，PID 控制算法中的比例（P）、积分（I）、微分（D）调节器利用 e(t)生成新的控制量 u(t)，u(t)通过执行机构（电机驱动器）作用于被控对象（电机），电机的实际运行速度 c(t)通过闭环反馈，进入下一次 PID 调节。就这样，不断的通过闭环反馈调节，使电机实际运行速度 c(t)最终逼近给定的目标速度 r(t)。

在连续和离散时间域上 PID 会有不同的表现形式，在连续时间域上积分、微分调节通过积分计算、微分计算实现，而在离散时间域上积分、微分调节通过累加和、差分计算实现。由于电机控制需要在程序上进行实现，所以需要采用离散域的 PID。按照 PID 算法生成的调

节量的形式，又可以分为位置式 PID 和增量式 PID；位置式 PID 生成的 u(t)为直接的控制量，增量式 PID 生成的 Δu(t)是控制量的修正量，需要叠加上一次的 u(t-1)才能作为控制量。离散形式的位置式与增量式 PID 数学表达如图 24。

位置型：

$$u(k) = K_P[e(k) + \frac{1}{T_I}\sum_{i=0}^{k} Te(i) + T_D\frac{e(k)-e(k-1)}{T}]$$

$$u(k) = K_P[e(k) + \frac{T}{T_I}\sum_{i=0}^{k} e(i) + T_D\frac{e(k)-e(k-1)}{T}]$$

$$u(k) = K_P e(k) + K_I\sum_{i=0}^{k} e(i) + K_D[e(k)-e(k-1)]$$

其中：

e(k) = input_target – feedback_current，输入目标值与当前反馈值之差；

K_P 为比例系数；

$K_I = K_P\frac{T}{T_I}$，为积分系数；

$K_D = K_P\frac{T_D}{T}$，为微分系数；

T 为采样周期；

T_I 为积分时间；

T_D 为微分时间。

增量型：

$$u(k) = K_P e(k) + K_I\sum_{i=0}^{k} e(i) + K_D[e(k)-e(k-1)]$$

$$u(k-1) = K_P e(k-1) + K_I\sum_{i=0}^{k-1} e(i) + K_D[e(k-1)-e(k-2)]$$

$$\Delta u(k) = u(k) - u(k-1)$$

$$\Delta u(k) = K_P[e(k)-e(k-1)] + K_I e(k) + K_D[e(k)-2e(k-1)+e(k-2)]$$

（图 24）离散形式的位置式与增量式 PID 数学表达

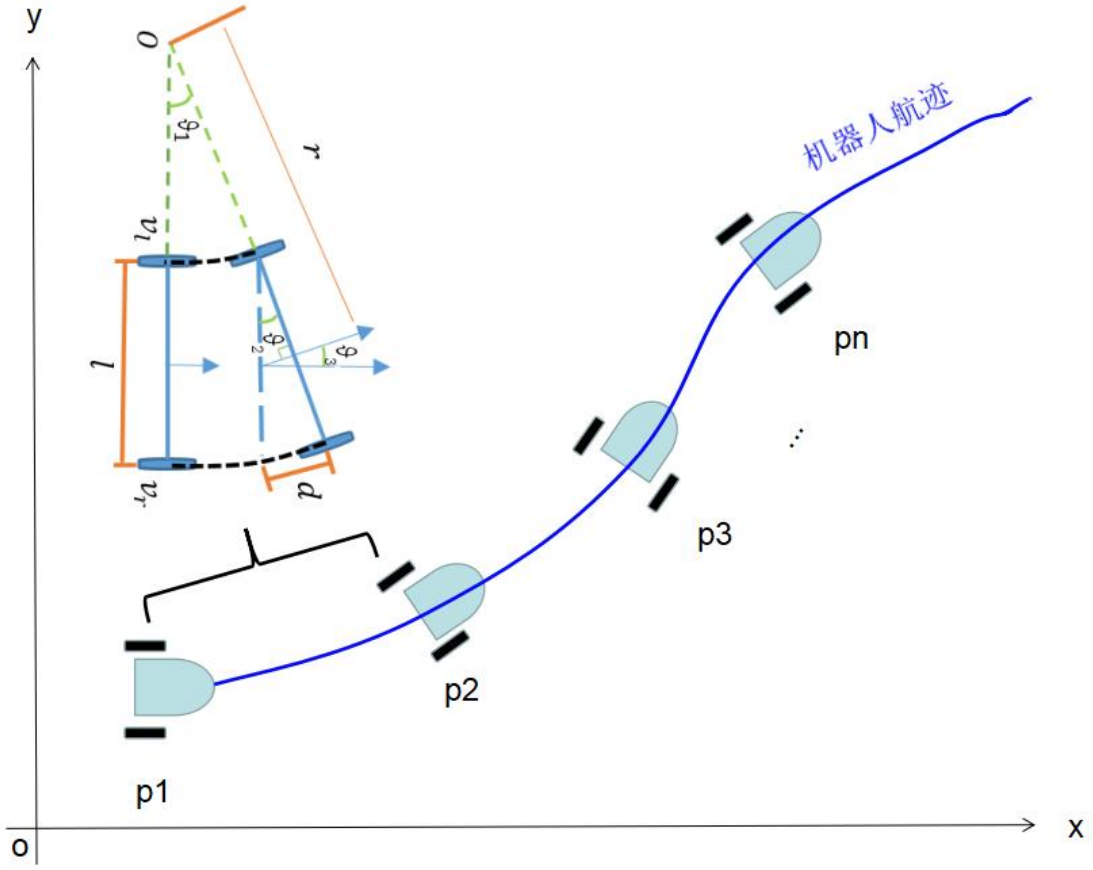
可以看出，利用位置 PID 的数学表达式经过简单的变形就能得到增量 PID 的数学表达式。增量算法不需要做累加，控制量增量的确定仅与最近几次误差采样值有关，计算误差或计算精度问题，对控制量的计算影响较小。而位置算法要用到过去的误差的累加值，容易产生大的累加误差。增量式算法得出的是控制量的增量，例如阀门控制中、只输出阀门开度的变化部分，误动作影响小，必要时通过逻辑判断限制或禁止本次输出，不会严重影响系统的工作。而位置算法的输出是控制量的全量输出，误动作影响大。增量型 PID 的程序实现，如如图 25。关于 PID 参数的整定，将放在后面的文章进行详细讲解。

```
//PWM_limit
s16 PWM_limit = 30000;//s16:[-32768 ~ +32767]

//line fit,pwm-enc model,pwm=Ka*enc
float Ka = 2558.2559;

//PID params
float Kp = 0.1;
float Ki = 0.03;
float Kd = 0.002;

int Incremental_PID_motor1(int target_enc,int current_enc)
{
  static int err,last_err,last_last_err;
  static int u_output;

  //PID caculate
  err = target_enc - current_enc;
  u_output = u_output + Ka*(Kp*(err-last_err) + Ki*err + Kd*(err-2*last_err + last_last_err));
  //debug
  //printf("pid1:%d %d %d\r\n",u_output,target_enc,current_enc);

  //PWM_max is 35999,set PWM_limit is 30000
  if(u_output>PWM_limit) u_output = PWM_limit;
  if(u_output<(-1*PWM_limit)) u_output = -1*PWM_limit;

  //iteration update
  last_last_err = last_err;
  last_err = err;

  //Brake stop
  if(target_enc==0)
  {
    u_output = 0;
    last_last_err = 0.0;
    last_err = 0.0;
  }
  return u_output;
}
```

（图 25）增量型 PID 的程序实现

差动两轮底盘轮式里程计：

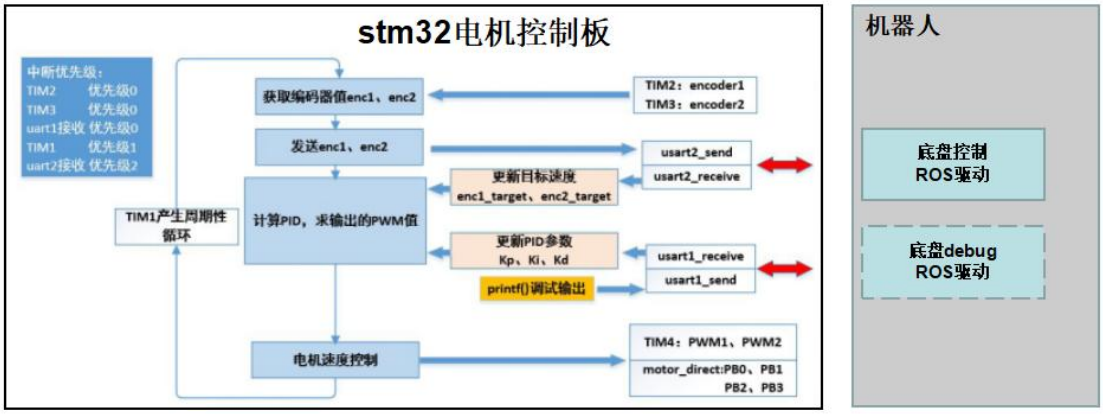
轮式里程计是机器人底盘的重要组成部分，采用航迹推演算法对机器人的位姿进行估计，并对机器人当前的速度、旋转速度、左右轮速度进行转换。无论是机器人的定位导航还是普通的运动控制，都需要轮式里程计。



（图 26）通过航迹推演计算里程计

如图 26，为通过航迹推演计算里程计的过程。随时间推移机器人底盘的实时位姿 p1、p2、p3、…、pn 连接起来就形成了机器人的航迹，考虑很短的时间内两相邻机器人位姿 p1 和 p2，在已知机器人位姿 p1 和机器人当前左右轮速度 v_l、v_r 的条件下，利用微积分的思想可以推算出机器人在下一个时刻的位姿 p2，通过这样不断的推演，就可以计算出机器人当前的位姿以及速度、角速度等信息，这就是所谓的航迹推演。关于航迹推演的具体数学推导和程序实现，将在后面的文章中进行展开讲解。

底盘电机控制板软件框架：



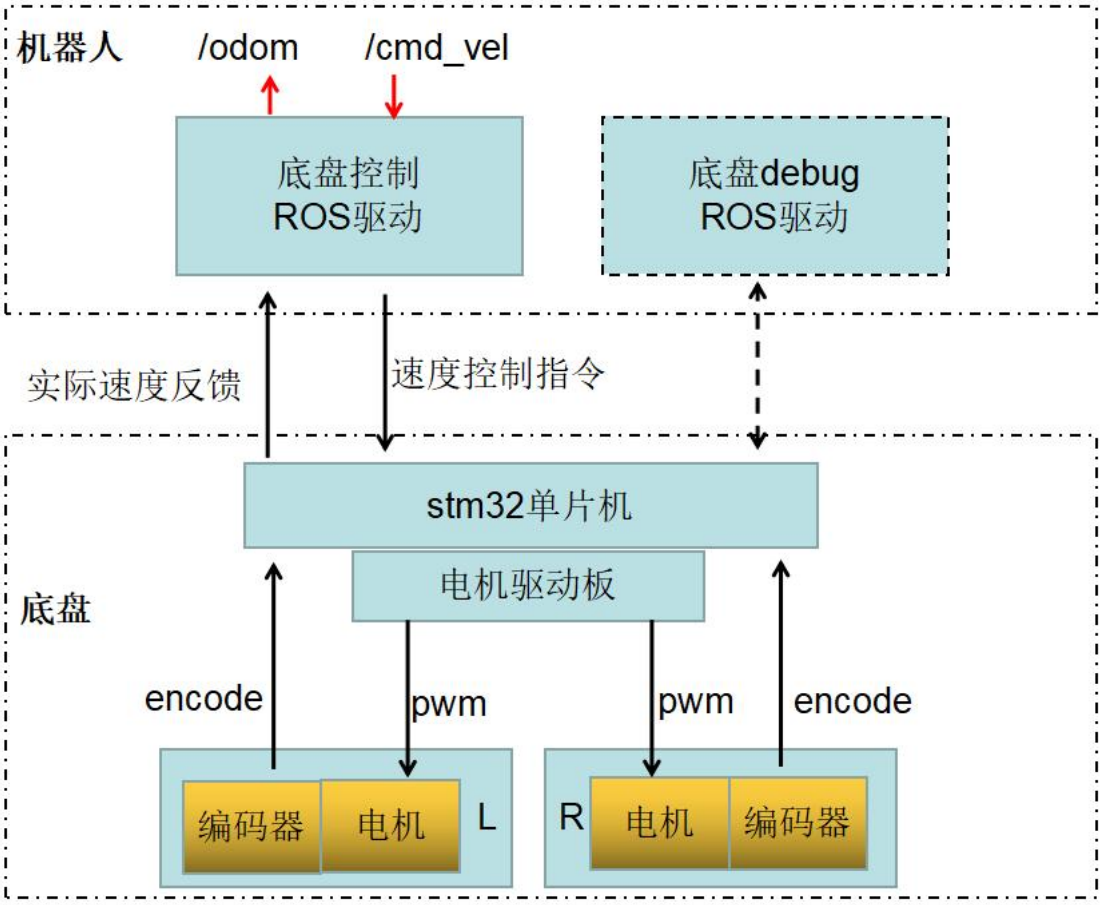
（图 27）stm32 电机控制板软件架构

如图 27，为 stm32 电机控制板软件架构。底盘中的电机控制与里程数据采集的程序在 stm32 单片机上实现，TIM1 定时器产生周期性的循环，循环中进行电机编码器数据采集、PID 计算、电机速度 pwm 控制，剩下的就是 usart1 串口与 usart2 串口跟机器人大脑之间的

通信了，底盘 debug 接口是用于 stm32 程序开发阶段使用的，所以在机器人正常运行的过程中只需要使用底盘控制接口。关于 stm32 部分的代码和对应机器人大脑中 ROS 驱动代码将在后面介绍。

3.3.在机器人中使用底盘

通过前面的讲解，我们已经对机器人底盘的用途及工作原理有了一定的了解，并且知道了电机控制和里程计的工作过程。这时候肯定很想知道如何在机器人中把底盘使用起来呢？其实很简单，和激光雷达、IMU 这些传感器一样，底盘也可以当做一个传感器来使用，只不过不同之处是这个传感器与机器人大脑是双向交互的，机器人大脑向底盘发送控制命令，底盘反馈里程数据给机器人大脑。但是，不论交互的细节如何，只需要装上底盘的 ROS 驱动包，上层算法只需要发布和订阅相应的主题就能达到使用底盘的目的。



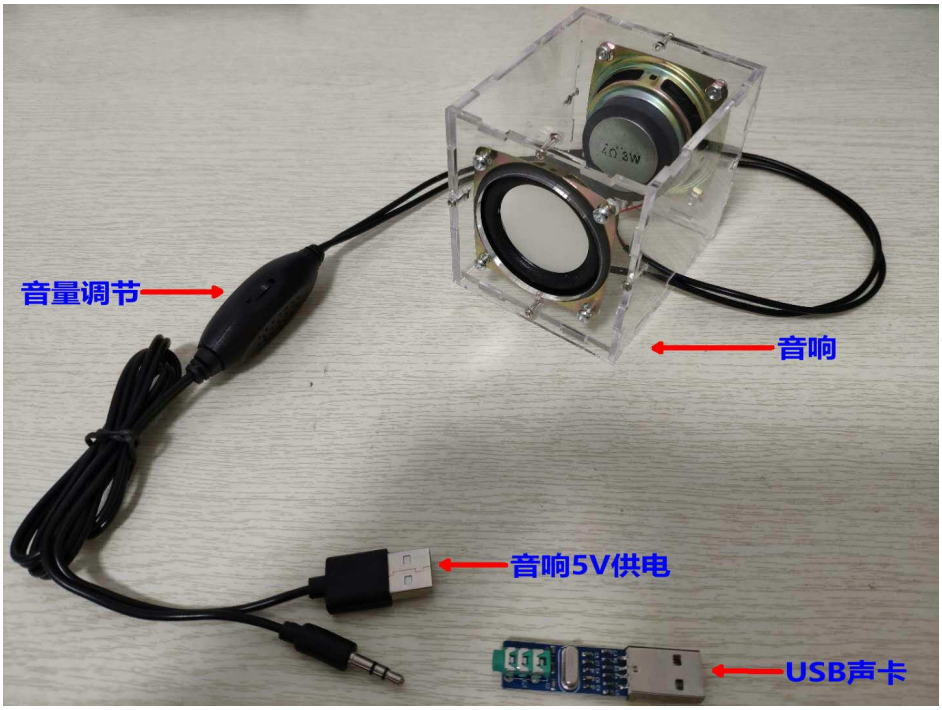
（图 28）在机器人中使用底盘

底盘通过串口与机器人相连接，机器人中通过运行底盘控制 ROS 驱动，来实现读取串口的速度反馈，利用航迹推演算法计算得到里程计并发布到/odom 这个主题；底盘控制 ROS 驱动订阅/cmd_vel 主题的运动控制数据，并转换为速度控制指令通过串口发送给底盘。这样机器人上的其它节点就可以通过发布/cmd_vel 主题来对底盘进行控制，通过订阅/odom 主题获取底盘的里程计。关于底盘控制 ROS 驱动、底盘里程计标定、底盘的 debug，将在后面的章节中具体讲解。

4. 音响麦克风与摄像头

要让机器人能跟人进行对话，需要麦克风和音响。其实麦克风就相当于机器人的耳朵，音响就相当于机器人的嘴巴。摄像头作为机器人的眼睛，摄像头可以用来帮助机器人定位或认识环境。

4.1.音响



（图 29）音响

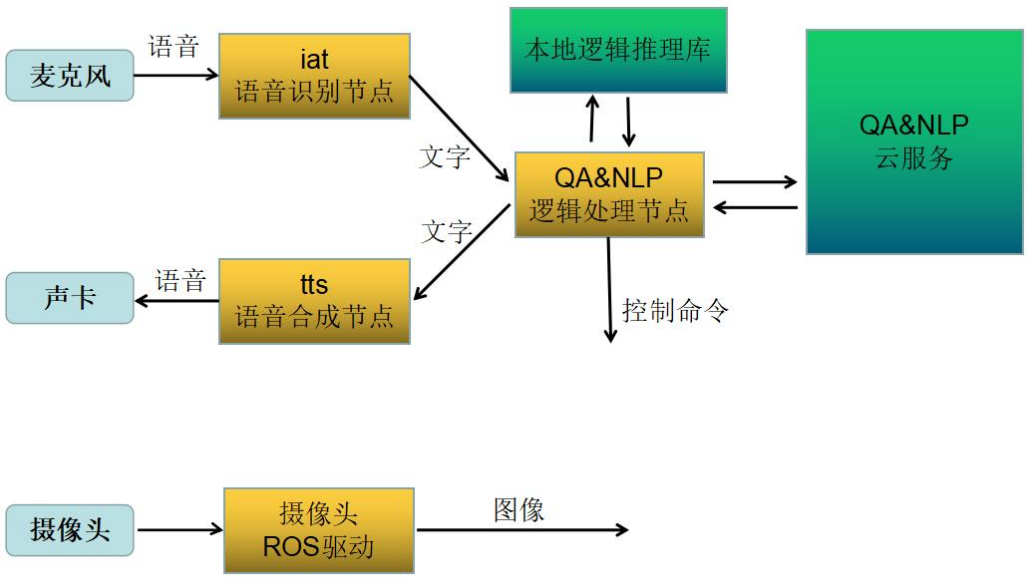
如图 29，为音响的基本配件。这里需要说明一下，由于后面使用树莓派 3 作为机器人的大脑，不过树莓派 3 的自带声卡不好用，驱动容易崩溃，所以这里推荐使用免驱的 USB 声卡。其实，声卡就是 DA 转换器，就是将数字音频信号转换为模拟电压信号；音响就是将声卡输出的模拟电压进行放大并通过喇叭播放出来。

4.2.麦克风与摄像头

（图 30）麦克风与摄像头

如图 30，摄像头上直接集成了麦克风。这里选用的是四麦阵列指向性麦克风，可以对特定方向上的声音拾取，并过滤其他方向上的杂音。这里的摄像头是 640x480 像素 60fps。

4.3.在机器人中使用音响麦克风与摄像头



（图 31）麦克风与摄像头

我们只需要在机器人上安装对应的 ROS 驱动，就可以通过发布和订阅相应的主题来实现对传感器的访问了。如图 31，iat 语音识别节点用于驱动麦克风，并将麦克风采集的语音转换为文字；问答（QA）和自然语言处理（NLP）节点处理 iat 语音识别节点发布的文字，

并将处理结果发布给 tts 语音合成节点；tts 语音合成节点订阅问答（QA）和自然语言处理（NLP）节点发布的文字，并驱动声卡将文字转换为语音；摄像头 ROS 驱动将摄像头数据直接发布到相应的话题。这样机器人上的其他节点都可以通过订阅与发布相应的节点来访问麦克风、声卡和摄像头。关于图像、语音、文字等处理的具体应用将在后面详细展开。

5. 机器人大脑嵌入式主板性能对比

从事 SLAM 与机器人导航也有几年时间了，期间用过不少的嵌入式主板做开发。机器人是软硬件结合的一个实体，这里就对机器人的大脑（嵌入式主板）进行一些讨论。结合我用过的一些嵌入式开发板，展开对比分析，具体型号如图 32。

（图 32）我用过的嵌入式板型号

5.1.树莓派 3

树莓派一直很火，现在已经推出第三代了。这里放一张树莓派 0、树莓派 2、树莓派 3 的全家福吧，如图 33。

（图 33）树莓派全家福

接下来看看具体的性能参数，如图 34。如果想了解更多关于树莓派的资料，可以阅读树莓派的 wiki 教程 https://en.wikipedia.org/wiki/Raspberry_Pi。

型号	A型	A+型	B型	B+型	2代B型	3代B型
SOC[4]	BroadcomBCM2835（CPU，GPU，DSP和SDRAM，USB）				Broadcom BCM2836	Broadcom BCM2837
CPU	ARM1176JZF-S核心（ARM11系列）700MHz				ARM Cortex-A7 (ARMv7系列) 900MHz (四核心)	ARM Cortex-A53 1.2GHz 64-bit quad-core ARMv8 CPU
GPU	Broadcom VideoCrore IV，OpenGL ES 2.0,1080p 30 h.264/MPEG-4 AVC高清解码器					
内存	256 MB（与GPU共享，可以理解为集成显卡的显存与内存共享）		512MB		1GB (LPDDR2)	
USB 2.0接口个数	1（支持USB hub扩展）		2	4		
视频输入	15-针头 MIPI 相机 (CSI) 界面，可被树莓派相机或树莓派相机(无红外线版)使用					
影像输出	RCA视频接口输出（仅1代B型有此接口），支持PAL和NTSC制式，支持HDMI (1.3和1.4)，分辨率为640 x 350 至 1920 x 1200 支持PAL 和NTSC制式。					
音源输出	3.5mm插孔，HDMI电子输出或I²S					
板载存储	SD/MMC/SDIO卡插槽	MicroSD卡插槽	SD / MMC / SDIO卡插槽	MicroSD卡插槽		
网络接口	无		10/100以太网接口			• 10/100以太网接口 • 802.11n Wireless LAN • Bluetooth 4.1 • Bluetooth Low Energy (BLE)
外设	8个GPIO, 外加下列外设(亦可当作GPIO使用): UART、I²C、带两个选择的SPI总线，+3.3 V、+5 V、地线	17个GPIO及HAT规格铺设	除A型所拥有之外设之外，亦有4个GPIO可供用户使用(需自行焊接电路)	17个GPIO及HAT规格铺设		
额定功率	300mAH（1.5W）	200mAH(1W)	700mAH（3.5W）	600mAH（3.0W）	800mAH(4.0W)	
电源输入	5V / 通过MicroUSB或GPIO头					
	45g	23g		45g		
总体尺寸	85.60 x 53.98 mm	65 × 56.5 × 10 mm		85 x 56 x 17mm		
操作系统	Debian GNU/Linux 、Fedora、Arch Linux 、RISC OS，详见: [5] 2代B型以上型号还将支持Windows10 IoT[6]					

（图 34）树莓派性能参数

树莓派 3，售价 200RMB 左右，CPU 是 1.2Ghz ARM-Cortex-A53，内存 1GB，板载 wifi 模块，还有一个多媒体显示 GPU（不过感觉没什么用）。CPU 和内存配置算的的上是同等价位

嵌入式主板的战斗机了，板载 wifi 这个也很实用。虽然树莓派支持安装很多种 linux 系统，由于我这里要跑 ROS 机器人系统，所以我选择了安装 ubuntu-mate-16.04, ubuntu-mate-16.04 LTS 实际上是 ubuntu-16.04 LTS 的一部分，为桌面、Raspberry Pi 2 和 3 单片机准备的，这也是 Ubuntu MATE 的首个 LTS 长期支持版，亮点在于包含 MATE 1.12.1 桌面环境，针对平板支持多点触控和“自然滚动”，对多屏幕设置提供更好的支持，更好的会话管理，扩展的 systemd 支持，改进过的 Power 小程序——可显示产品型号和提供商信息等内容。

（图 35）ubuntu-mate-16.04 系统界面

安装好 ubuntu-mate-16.04 操作系统后，上电可以看到图 35 所示的系统界面，就可以安装 kinetic 版本的 ROS 了，然后就可以验证 SLAM 算法了。Gmapping 激光 SLAM 建图和 ros-navigation 自动导航跑的都很顺畅；google-cartographer 进行建图和重定位也没什么问题；跑 ORB-SLAM2 的 Mono 模式就不太行了，帧率 5 帧以内。

5.2.Firefly-RK3399

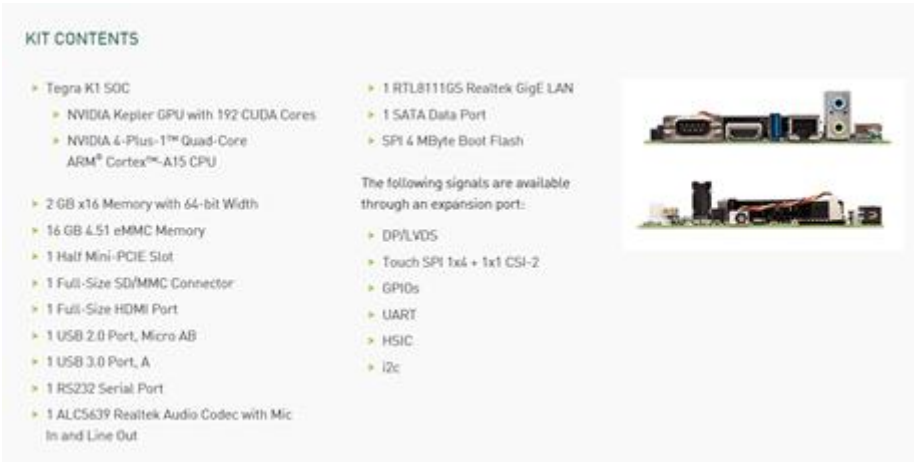
萤火虫开发板还是很强大的板子分 RK3288 和 RK3399 两个版本，先看一下官方的宣传广告。

类型	规格参数
CPU	RK3399，双 Cortex-A72 大核+四 Cortex-A53 小核，主频 2GHz
GPU	四核 ARM Mali-T860，支持 OpenGL ES 1.1/2.0 /3.0，OpenVG1.1, OpenCL, Directx11
DDR	双通道 DDR-1866/DDR3L-1866/LPDDR3-1866/LPDR4
	2GB/4GB(可选)
内置存储器	支持 eMMC5.1，SDIO3.0
	8GB/16GB/32GB/64G/128G（可选）
解码分辨率/多媒体	支持 4K VP9 and 4K 10bits H265/H264 视频解码，高达 60fps
	1080P 多格式视频解码（VC-1，MPEG-1/2/4，VP8）
	1080P 视频编码，支持 H.264，VP8 格式
显示	双 VOP 显示：分辨率分别支持 4096X2160 及 2560X1600
	支持双通道 MIPI-DSI（每通道 4 线）
	HDMI2.0 支持 4K 60Hz 显示，支持 HDCP 1.4/2.2
	支持 DisplayPort 1.2（4 线，最高支持 4K 60Hz）
	支持 eDP 1.3（4 线，10.8Gbps）
	支持 Rec.2020 和 Rec.709 色域转换
接口	双 ISP 像素处理能力高达 13MPix/s，支持双路摄像头数据同时采集
	支持双 USB3.0 Type-C 接口
	支持 PCIe 2.1（4 full-duplex lanes）
	内置低功耗 MCU
	支持 8 路数字麦克风阵列输入
网络支持	RJ45 接口千兆以太网
	板载 WIFI/BT 模块，支持 WiFi 2.4GHz/5GHz 双频，支持 802.11a/b/g/n/ac 协议，支持 Bluetooth 4.0（支持 BLE）
USB	2 x USB2.0 HOST，1 x USB2.0 OTG，2xUSB3.0
红外接收	1 路红外接收头，支持红外遥控功能
输入电源	12/2A
尺寸	124MMx93.5MM

（图 36）Firefly-RK3399

如图 36 所示，RK3399 双核 CortexA72+四核 CortexA53 的 CPU 和 2GB/4GB 可选配的内存，一看这些配置就知道很强大，的确跑各种视觉算法很不错，不过 1000RMB 的售价感觉有点小贵。

5.3.Nvidia-jetson-TK1



（图 37）Nvidia-jetson-TK1

Tegra K1 是 Nvidia 推出的一款 AI 级别的嵌入式主板，ARM-Cortex-A15 的 CPU，192 个 CUDA 核心的 kepler 架构的 GPU，2GB 内存，如图 37。有人用 Tegra K1 做了一个计算集群，感觉还挺有意思的，如图 38。

（图 38）Nvidia-jetson-TK1 计算集群

5.4.Nvidia-jetson-TX2

Jetson TX2 的是可以作为核武器的处理器的（@~@），性能是十分强大的。简单的智能小车或者机器人不推荐使用 TX2，性价比比较低。利用 TX2 做处理器，控制移动平台（高精度的小车底盘）做 SLAM 我觉得是一个相当有意思的项目，TX2 的处理能力非常适合实现机器视觉。

（图 39）Nvidia-jetson-TX2

图 39 中左边是官方的开发板和扩展板，不过由于官方扩展板体积太大了用在很多地方不方便，于是网上推出了一款小巧的扩展板 Connect Tech Inc 很不错。

	Jetson TX2	Jetson TX1
GPU	NVIDIA Pascal™, 256 CUDA cores	NVIDIA Maxwell ™, 256 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2	Quad ARM® A57/2 MB L2
Video	4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)	4K x 2K 30 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (10-Bit Support)
Memory	8 GB 128 bit LPDDR4 59.7 GB/s	4 GB 64 bit LPDDR4 25.6 GB/s
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4	2x DSI, 1x eDP 1.4 / DP 1.2 / HDMI
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.1 (1.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2	Gen 2 | 1x4 + 1x1
Data Storage	32 GB eMMC, SDIO, SATA	16 GB eMMC, SDIO, SATA
Other	CAN, UART, SPI, I2C, I2S, GPIOs	UART, SPI, I2C, I2S, GPIOs
USB	USB 3.0 + USB 2.0	
Connectivity	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth	
Mechanical	50 mm x 87 mm (400-Pin Compatible Board-to-Board Connector)	

（图 40）Nvidia-jetson-TX1/2 性能参数

这个是性能表，看上去也是叼叼的，不过售价不便宜接近 5000RMB。。。

5.5.Intel-NUC

前面介绍的都是 ARM 架构的主板，现在介绍一款 X86 架构的主板 NUC。

intel 功能和配置

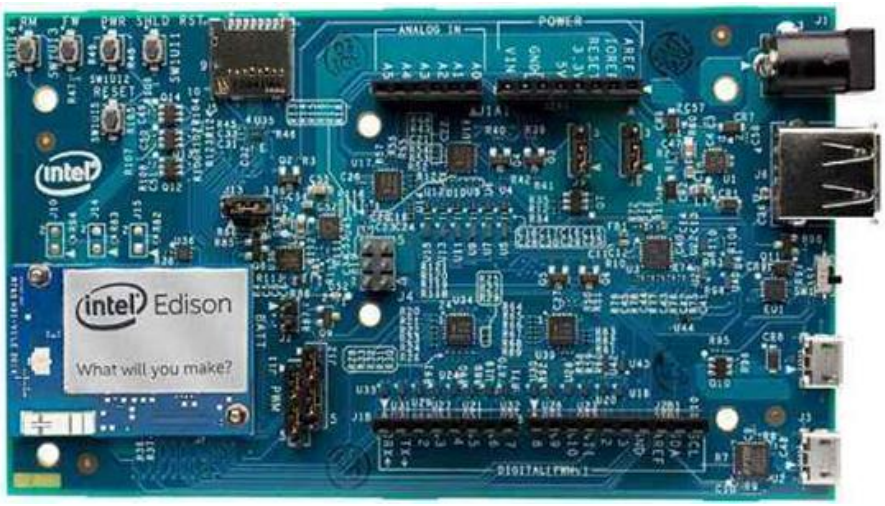
尺寸	115mm x 111mm x 51mm
处理器	第七代智能英特尔® 酷睿™ i7-7567U （3.5GHz 高达4.0GHz Turbo，双核，4MB缓存，28W热设计功耗（TDP））
内存	双通道DDR4-2133 SODIMMs,1.2V 可达32GB
图形	英特尔®锐炬™ Plus 显卡650
音频	7.1环绕立体声（通过微型HDMI和微型DisplayPort） 耳机和话筒接口（前面板） 组合扬声器/ TOSLINK *光纤音频插孔 双阵列前麦克风(前面板)
外设连接	1 x Thunderbolt™3 接口(40Gbps USB3.1 Gen 2 10Gbps 和DisplayPort*1.2)via USB-C 1 x HDMI*2.0接口 2 x USB 3.0端口（前面板，其中一个可用于充电） 2 x USB 3.0(后面板) 2 x 内部USB 2.0（通过接头） 消费者红外传感器（前面板）
存储	M.2 22x42/80（key M） SATA3 /PCIe*X4 Gen 3 NVMe*/AHCI SSD Micro SDXC插槽支持UHS-I 内部SATA3接口支持2.5英寸机械硬盘/固态硬盘（厚度可达9.5mm）
网络	英特尔® I219-V 10/100/1000 Mbs网络连接 英特尔®Wireless-AC 8265的无线天线（IEEE 802.11ac 2x2 + 蓝牙v4.2, internal antennas）
电源适配器	19V ,65 瓦，壁挂式AC-DC电源适配器

（图 41）Intel-NUC

之前用过 Intel-NUC7-i7,19V 供电 65W 功耗感觉不适合嵌入式级别的应用场合，而且 3000RMB 的售价也没法和 Nvidia-jetson-TX2 比较性价比，所以不推荐在机器人上使用。

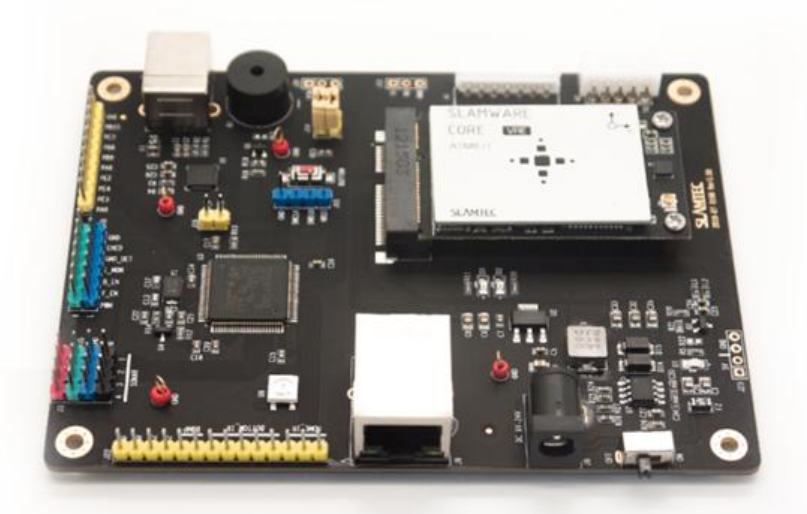
5.6.Intel-Edison

其实这是 intel 一个失败的尝试，主打物联网应用，CPU 采用 intel 的 Atom 处理器，最大的亮点是可以在主板上直接扩展 Ardunio 单片机开发板。如图 42。



（图 42）Intel-Edison

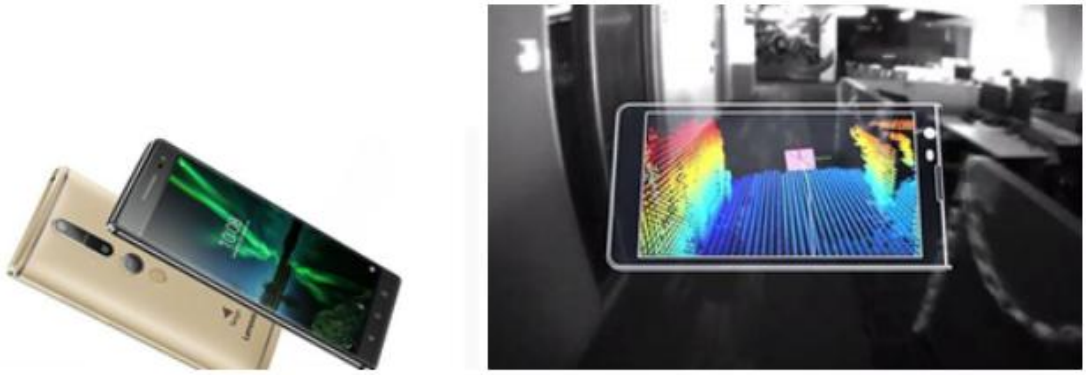
其实个人不推荐用 Edison 来开发 SLAM 算法，不过也有公司做这方面的尝试，比如上海思岚科技的 SLAMWARE-CORE 就是 Edison 的模仿者。



（图 43）SLAMWARE-CORE

5.7.Google-Tango-phone

其实 Tango-phone 是一个完整的 AR 方案，手机内集成了深度相机和 VO 视觉里程计。



（图 44）Tango-phone

看网上的演示视频也是十分的炫酷，不过最终这个项目还是没有在 google 中火起来，可能还是存在不少问题的吧。

5.8.总结

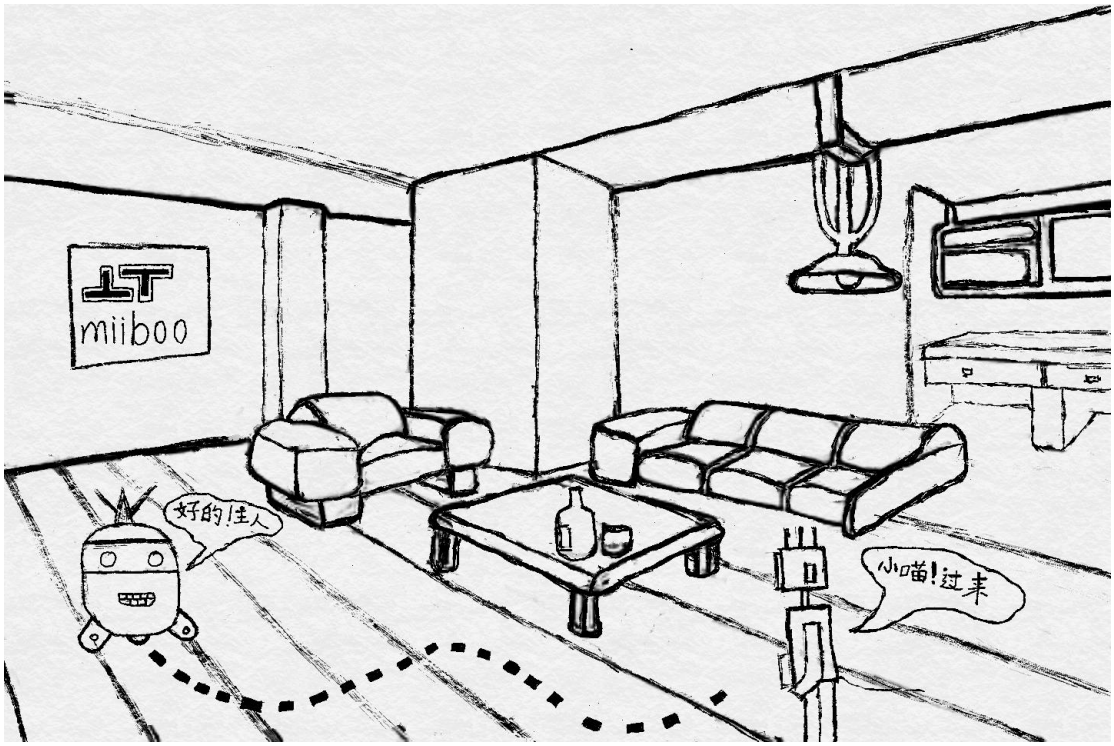
型号	CPU	RAM	GPU	用途	亮点	价格（RMB）
树莓派 3	ARM-Cortex-A53	1GB	多媒体显示 GPU	教育，创客，家庭娱乐	性价比高	200
Firefly-RK3399	ARM 2*Cortex-A72+4*Cortex-A53	2GB/4GB 可选	4 核 ARM-mali-T860	计算机视觉，3D 视觉	CPU 很强大	1000
Nvidia-jetson-TK1	ARM-Cortex-A15	2GB	Kepler 架构 192 个 CUDA 核心	计算机视觉，深度学习	带 GPU 的超级嵌入式板	1600
Nvidia-jetson-TX2	4*Cortex-A57+2 个自研的 Denver（丹佛）核心	8GB	Pascal 架构 256 个 CUDA 核心	计算机视觉，深度学习	最强大的嵌入式 AI 板	4000
Intel-NUC	Intel-i7	最高支持 32GB		迷你电脑		3000
Intel-Edison	Intel-atom			物联网	支持板载 Ardunio	700
Google-Tango-phone					AR 全套方案集成	3500

（图 45）性能对比

最后，总结一些各个开发板的性能对比，见图 45。玩机器人和 SLAM 的朋友们，如果是中低端需求推荐树莓派 3，高端需求推荐 jetson-TX2。

6. 做一个能走路和对话的机器人

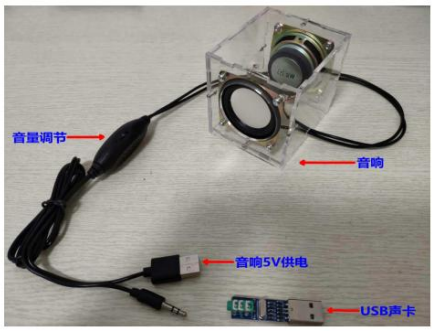
在我的想象中机器人首先应该能自由的走来走去，然后应该能流利的与主人对话。朝着这个理想，我准备设计一个能自由行走，并且可以与人语音对话的机器人。为了更形象的表达我的想法，我的小学五年级的绘画水平也是豁出去了，如果有大神路过，恳请多多包涵，画好后的样子大概就是这样啦，如图 46。

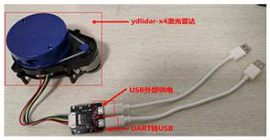


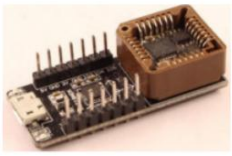
（图 46）想象中的机器人

有了想法，接下来就要开始亲自动手 DIY 了。体验动手乐趣的同时，以玩耍的形式融入当下前沿的 SLAM、自动导航、图像识别、语音识别、自然语言处理等技术，提高自己的同时找到属于儿时的那一份快了。

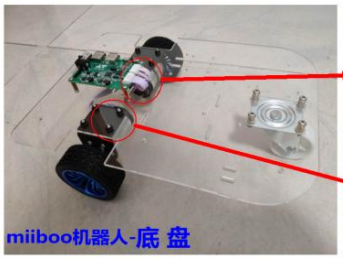
说干就干，首先需要准备好激光雷达、IMU、底盘、音响、麦克风、摄像头这些传感器，然后还要准备一块树莓派 3 开发板作为机器人的大脑。其实这些知识已经通过前面的文章打好了基础，所以就好办多了，准备好这些东西就好了，如图 47。





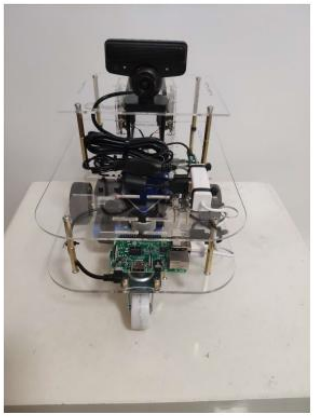


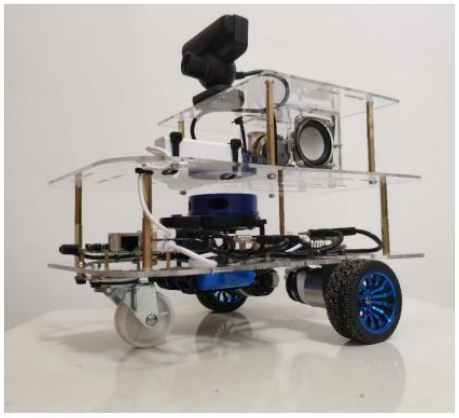
MPU9250 模块

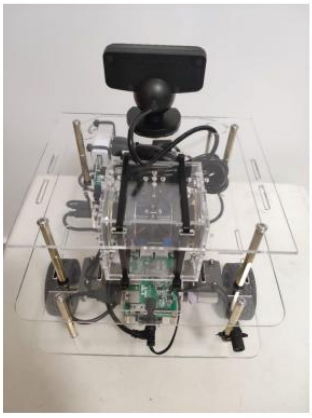


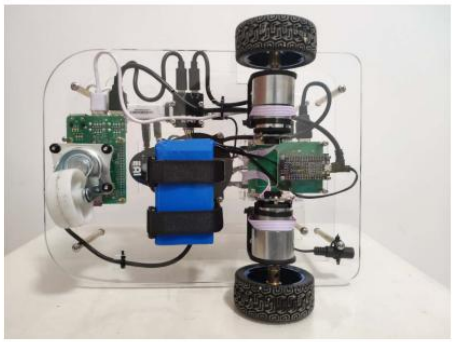
（图 47）DIY 部件准备

准备好传感器和树莓派 3 后，就可以进行组装了。经过紧张有趣的组装后，一台有趣的机器人就被 DIY 出来了，这时候终于可以看到传说中的机器人的真容了，如图 48，虽然机器人的真实样子没有想象中的那么有艺术感，但科技感十足有木有@^@





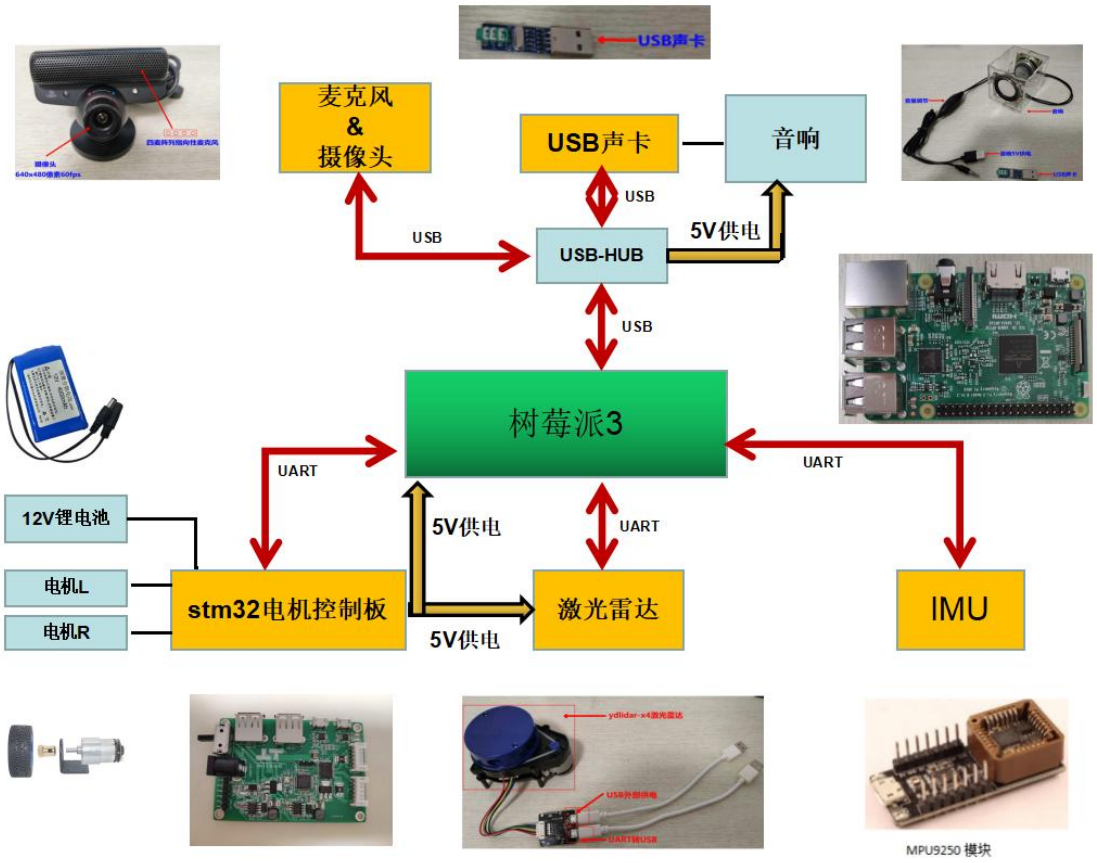




（图 48）DIY 出来的机器人真容

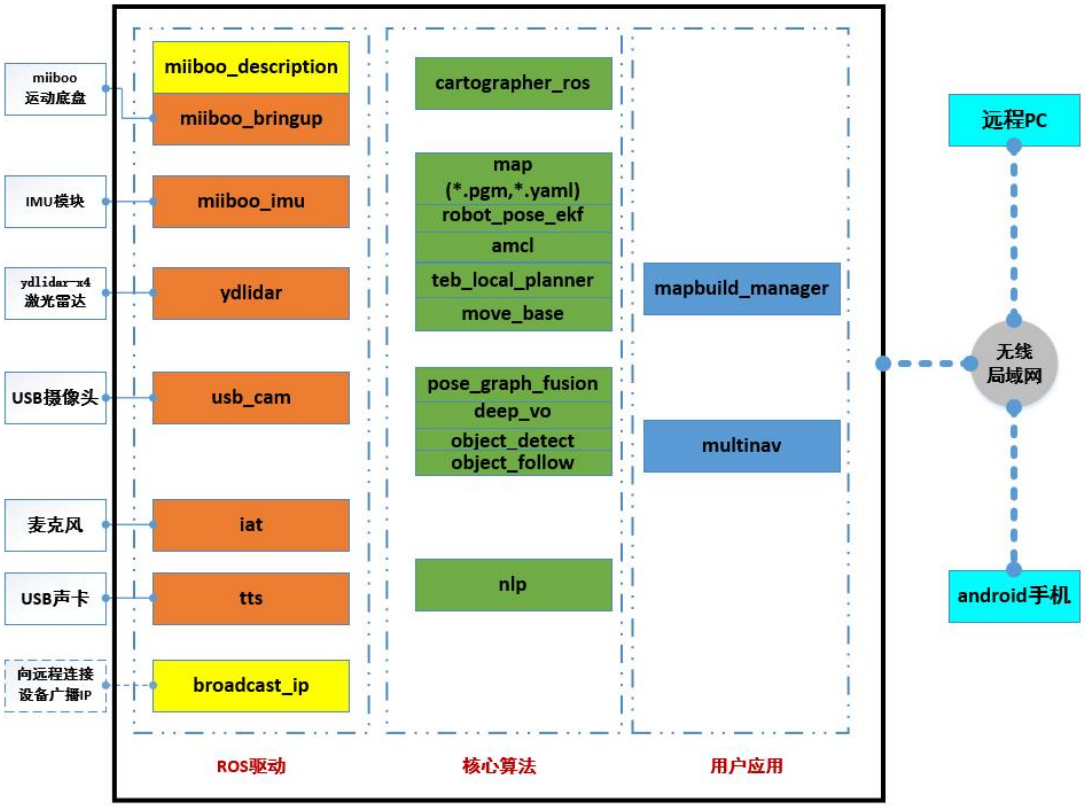
麻雀虽小五脏俱全，现在就来对这个会说话会走路的机器人做一做剖析吧。机器人的骨架是由亚克力板和铜柱组合而成；两个带编码器的减速电机和一个万向轮作为运动执行机构；

可充电锂电池给整个机器人供电；stm32 电机控制板用于控制电机运动并提供里程数据，是底盘的核心部件；激光雷达提供环境障碍信息，用于 SLAM 建图和避障导航；IMU 用于在里程计数据融合、SLAM 建图、导航中提供惯导数据；免驱 USB 声卡、音响、四麦阵列指向性麦克风作为语音交互过程中的输入输出部件；摄像头用于物体识别、物体跟踪、环境监控、视觉辅助定位；树莓派 3 是整个机器人的中央处理单元，各个传感器的 ROS 驱动、SLAM 算法、导航算法、语音交互、自然语言处理算法、图像处理算法都将运行在上面。为了更好的理解机器人的工作原理，这里给出机器人的硬件框架，如图 49。



（图 49）机器人硬件框架

机器人的硬件搭建完毕后，就要赋予机器人灵魂了。这里说的灵魂就是我们的软件及算法，包括各个传感器的 ROS 驱动、轮式里程计与 IMU 融合、激光 SLAM 建图、自主导航避障、语音识别、语音合成、自然语言处理、物体识别、物体跟踪、远程视频监控、视觉辅助定位、机器人与工作 PC、Android 手机之间的通信等。为了更好的理解机器人的工作原理，这里给出机器人的软件框架，如图 50。



（图 50）机器人软件框架

做了如此长的铺垫，大家肯定已经等不及要亲自动手去开发机器人上的软件代码了。别急，接下来的章节将跟你娓娓道来，让你体验动手写程序真正的快乐$^$

后记

为了防止后续大家找不到本篇文章，我同步制作了一份文章的 pdf 和本专栏涉及的例程代码放在 github 和 gitee 方便大家下载，如果下面给出的 github 下载链接打不开，可以尝试 gitee 下载链接：

- github 下载链接：
  https://github.com/xiihoo/DIY_A_SLAM_Navigation_Robot

- gitee 下载链接：
  https://gitee.com/xiihoo-robot/DIY_A_SLAM_Navigation_Robot

参考文献

张虎,机器人 SLAM 导航核心技术与实战[M].机械工业出版社,2022.