

**TyRan移动机器人开发平台**

**用户手册 V1.0**

# 目录

安全注意事项 2

操作注意事项 3

机器人系统概述 4

系统架构概述 4

机器人结构说明 7

充电步骤 8

开关机与急停 8

emake用户手册 9

1. 概述 9

2. 安装 9

3. 使用 9

4. 编写程序 10

开始编程 11

1. 开始编程 11

2. 阅读编程手册 12

3. 查看范例程序 12

4. 编写第一个程序 12

4. 编译 16

5. 运行 16

手把手教学————编写一个简单的MORO机器人程序 17

机器人开发与调试 27

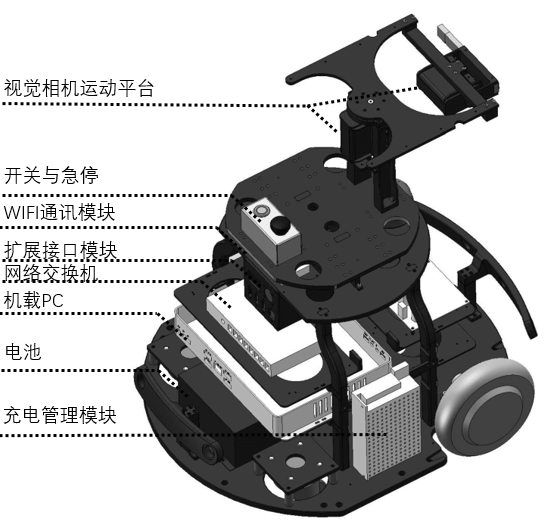
# 安全注意事项

* 任何操作者在使用TyRan之前必须熟知安全须知；
* 严格按照操作流程操作。任何不正确的操作可能会引起对人或机器人的伤害；
* 遇到紧急情况请立即关闭总开关电源；在确定找到问题所在时，再打开总开关；
* 当TyRan机器人处于上电运行状态时，禁止对机器人的各个部件插拔拆装；
* 对TyRan机器人的操作只能是有机器人基础的操作人员来进行；
* 在高速运行时，建议不要让机器人撞击障碍物，可能会缩短机器人说明或损坏外界物体；
* TyRan使用电池工作，请严格使用本机所配置的充电器，安照充电规范对电池充电；
* TyRan内部单元期间都有严格的电压限制，如果要改动，请严格按照电气标准进行变换；
* 请使用TyRan系统自带的电源工作，禁止使用任何未经许可的电源；外接电源和电池之间可以相互谢欢。当机器人接上外接电源时，机器人的电源由外接电源提供，电池的供电线路被自动切断；当拔外接电源时，机器人的供电电源自动切换到有机器人内部的电池供电；
* TyRan不具备防火功能，请远离火源避免电池爆炸；
* TyRan不具备防水特征，禁止淋雨，避免用水冲洗机器人；

# 操作注意事项

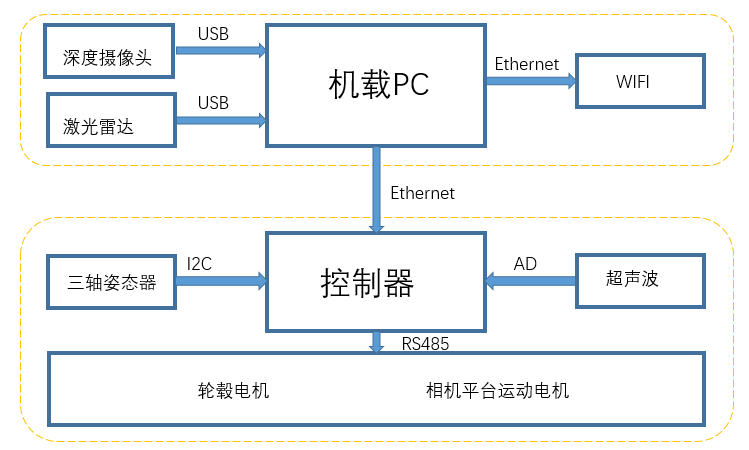
* 当打开系统电源开关时，双电机处于位置伺服状态，会自动锁死当前位置，为了较好地保护电机，建议不要用外力使机器人运动：
* 电池长时间不用时，为了保持电池的性能，每隔一个月需要对电池重新充电；
* 原装充电器具有智能充电功能，充电结束后会自动转入电池维护状态。通常电池充电时间在啊4-8小时；
* 用户加装自己的扩展应用时应该保证安全固定，否则，机器人长时间运行时，可能出现松动的螺丝掉入机器人控制系统中，可能导致系统的崩溃，甚至引起火灾；（尽管TyRan机器人已经加入多重电气保护，但为了安全考虑，建议用户扩展时要小心！）
* TyRan 机器人负载能力70公斤左右。加装超过此重量的负载，电机可能会损坏；
* TyRan机器人的各种连线时有方向的，而且接口是对应的。如果接错或者插反都无法正常工作，甚至可能会导致火灾等灾难性事故。因此用户（尤其是还不熟悉本机器人的用户）在连接这些线缆时请仔细阅读本用户手册，并小心操作。当连接线不能轻松插进去时，请检查是否插错，千万不要强行插入。
* 电池电压过低时，电源管理电路会自动切断系统电源，此时用户应该及时给机器人充电或使用交流电源供电。

# 机器人系统概述

TyRan是一款开源的移动地面运动系统开发平台。使用者通过机器人算法可实现对移动平台的自主移动、智能避障、环境识别、智能跟踪、SLAM导航、人机交互等功能。

# TyRan Plus正面功能图系统架构概述

TyRan移动机器人平台采用系统式控制方式控制，由机载PC完成机器人的数据采集和算法处理，交由控制实现运动控制，标准配置的TyRan控制架构如下：

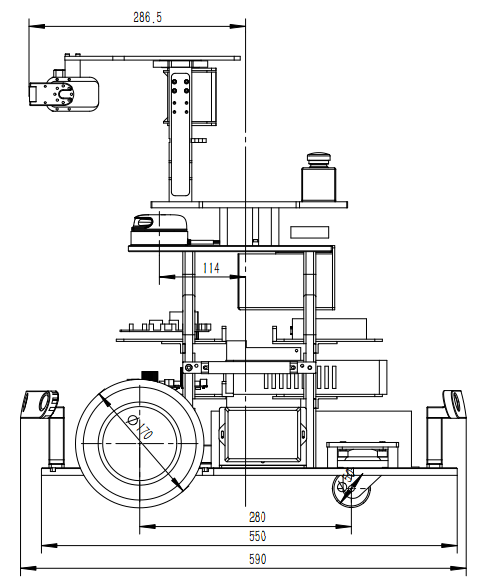
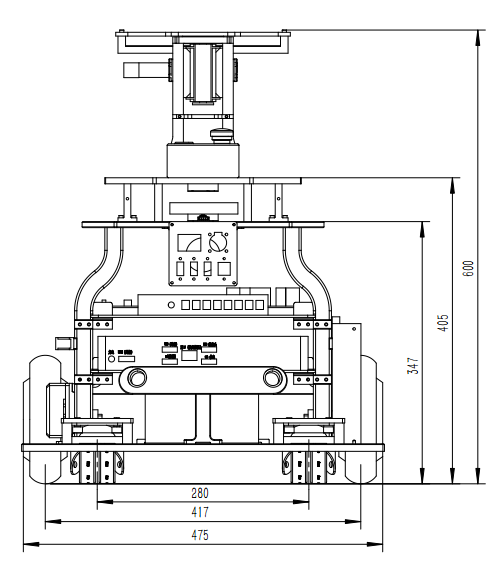
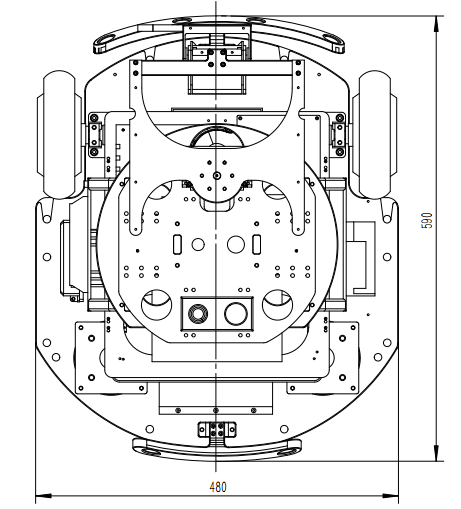


**机器人参数**

|  |  |  |
| --- | --- | --- |
| 内容 | | 规格/指标 |
| 基本 | 底盘尺寸 | 590（D）×480（W）mm |
| 标准高度 | 610mm |
| 自身重量 | 21kg（仅Tyran） |
| 机身材料 | 铝合金（阳极氧化黑色） |
| 电源模组 | 锂电池组：DC 25.2V，20Ah |
| 充电方式 | 充电器外置 |
| 通讯接口 | Ethernet、WIFI |
| 运动 | 移动方式 | 两轮差速驱动，2个从动轮 |
| 驱动轮 | 6.5寸轮毂电机，内置增量编码器 |
| 越障能力 | 10mm |
| 爬坡角度 | 10° |
| 避障方式 | 深度摄像头 + 超声波传感器 + 激光雷达（仅Tyran Plus） |
| 移动速度 | 标准： 3m/s |
| 导航方式 | 视觉+激光 SLAM导航（仅Tyran Plus） |
| 姿态感知 | 标配六轴IMU |
| 视觉 | 传感器 | RGB-D摄像机 |
| 视域 | 最远10米，水平180°及垂直90°视角可控调整。 |
| 算法 | 人脸识别、物体识别、maker识别 |
| 控制器 | 机器人控制器 | X86架构，intel i7 CPU, NVIDA GM107 GPU |
| 运动控制器 | STM32嵌入式系统 |
| 操作系统 | EwayOS | 国内自主知识产权智能机器人专用操作系统 |

# 机器人结构说明

TyRan采用模块化的结构，以便用户根据要求选择和定制配置，并在需要的时候方便更换和添加其他模块。标准配置的机器人主要尺寸如下：



# 充电步骤

当机器人人充电指示灯带变红时，需要连接原装电源适配器为机器人充电，充电时，请打开位于机器人接口的总开关，充电完成后，可从机器人手机测试终端中观测机器人充电电量

# 开关机与急停

* 机器人开机，首先需要打开机器人总开关，同时悬起急停按钮，按下机器人启动按键，即可完成机器人的启动。
* 机器人关机，再次按下机器人启动按键，关闭机器人总开关即可。
* 出现紧急情况，请直接按下急停按键，使机器人停止。

# emake用户手册

### 1. 概述

emake是EwayOS对外开发接口的一部分，负责生成、配置开发项目。

emake基于Ubuntu 14.04 - Desktop系统，开发IDE使用QtCreator。

### 2. 安装

EwayOS\_SDK.run为SDK安装包，下载后运行命令 chmod 755 EwayOS\_SDK.run添加执行权限，sudo ./EwayOS\_SDK.run运行安装。

### 3. 使用

安装成功后在任意目录打开终端，键入emake即可运行emake。

**参数列表**

**(1) -t 指定生成工程类型，目前支持fapp、flib和slib。**

**(a) fapp**： 生成Function应用程序，此项目可运行在操作系统为ubuntu14.04的电脑上，通过wifi连接MoRo HighLevelAI及LowLevelAI，可获取Sensor数据、调用FuncInterface中相关接口发送命令控制机器人运动以及调用他人FunctionLib库。

此类程序不可上传至MoRo本地运行。

**(b) flib：**生成FunctionLib程序，此项目生成为一个MoRo Function动态库，Function库可被其它Function程序加载，加载后可获取Sensor数据、调用FuncInterface中相关接口发送命令控制机器人运动以及调用他人FunctionLib库。

Function库可通过WebUI上传至MoRo本地运行。

**(c) slib：**生成ElementLib程序，此项目生成为一个MoRo Element动态库，Element库可被Sensor程序加载，加载后可根据连接信息创建Feature。

**(2) -n 指定生成工程的名称，名称必须以字母开头，中间不可出现除字母及数字之外字符。**

**(3) -i 设置MoRo IP地址。**

**(4) -v 显示emake版本号**

**(5) -h 显示emake帮助信息**

**(6) -u 升级emake**

生成成功后用QtCreator打开文件夹下的.pro工程文件即可。

### 4. 编写程序

emake生成的项目一般包含一个.h文件，一个.cpp文件，一个.pro文件，应用程序还有一个main.cpp文件。其中.pro文件为QTCreator工程文件,.h文件及.cpp文件为主要编程区域。**注意：请不要删除任何 emake 自动生成的代码。**

***Example：***

创建一个名为TestApp的Function，此Function可以直接运行。

emake -t fapp -n TestApp -i 127.0.0.1

创建一个名为TestLib的Function，此Function需要通过App类的Function加载。

emake -t flib -n TestApp -i 127.0.0.1

创建名为TestSensor的Element

emake -t slib -n TestSensor

# 开始编程

### 1. 开始编程

emake是EwaySDK的主要入口，可使用emake创建、配置机器人工程；打开帮助文档；打开调试工具。

emake使用请参考[emake使用文档](file:///D:\SD、\moro\doc\emake%20user%20manual.html)。

#### (1) 创建一个机器人应用程序

打开终端，键入 emake -t fapp -n XXX（程序名） -i xxx.xxx.xxx.xxx（MoRo的IP地址）；其中程序名可自行设定，规则与C++语法对参数命名的规则一致；MoRo的IP地址需使用实际MoRo机器人的IP地址，出厂MoRo机器人的IP为192.168.8.100。

回车，emake即会根据输入参数在当前目录下创建一个与程序名一致的文件夹，文件夹中为机器人应用程序框架。主要包含 .pro工程文件、main.cpp 程序入口、程序名.cpp、程序名.h的机器人程序入口类。

打开QtCreator， 打开 .pro文件即可加载工程。

应用程序编程请[FuncSDK用户编程手册](file:///D:\SD、\moro\doc\FuncSDK\FuncSDK%20user%20manual.html)。

#### (2) 创建一个机器人应用模块

打开终端，键入 emake -t flib -n XXX（程序名）； 其中程序名可自行设定，规则与C++语法对参数命名的规则一致，回车，emake即会根据输入参数在当前目录下创建一个与程序名一致的文件夹，文件夹中为机器人应用模块框架。主要包含 .pro工程文件、程序名.cpp、程序名.h的机器人程序入口类。

打开QtCreator， 打开 .pro文件即可加载工程。

应用程序编程请[FuncSDK用户编程手册](file:///D:\SD、\moro\doc\FuncSDK\FuncSDK%20user%20manual.html)。

#### (3) 创建一个机器人感知模块

打开终端，键入 emake -t slib -n XXX（程序名）； 其中程序名可自行设定，规则与C++语法对参数命名的规则一致，回车，emake即会根据输入参数在当前目录下创建一个与程序名一致的文件夹，文件夹中为机器人感知模块框架。主要包含 .pro工程文件、程序名.cpp、程序名.h的机器人程序入口类。

打开QtCreator， 打开 .pro文件即可加载工程。

应用程序编程请[SensorSDK用户编程手册](file:///D:\SD、\moro\doc\SensorSDK\SensorSDK%20user%20manual.html)。

### 2. 阅读编程手册

打开终端，键入 emake -d即可打开帮助文档。

### 3. 查看范例程序

EwaySDK当前提供[FaceExample](file:///D:\SD、\moro\doc\Example\CFaceExample.html)，[MarkerExample](file:///D:\SD、\moro\doc\Example\CMarkerExample.html)，[CmdExample](file:///D:\SD、\moro\doc\Example\CmdExample.html)，[Mother-KidExample](file:///D:\SD、\moro\doc\Example\CMother-KidExample.html)，[MsgExample](file:///D:\SD、\moro\doc\Example\CMsgExample.html)，[SensorExample](file:///D:\SD、\moro\doc\Example\SensorExample.html) 6个范例程序，[FeedbackMove](file:///D:\SD、\moro\doc\Example\FeedbackMove.html) 1个综合范例程序。

Example安装在/usr/share/ewaybot/example目录下。使用时将其拷贝至用户目录下，使用QtCreator打开工程文件即可。

### 4. 编写第一个程序

编写HeadMove程序，程序实现一个控制机器人头部向左转动90度，并且实时获取头部位置。

1. 打开终端，输入emake -t fapp -n HeadMove -i 192.168.8.100。
2. 打开QtCreator，加载HeadMove.pro文件
3. 打开HeadMove.cpp。

**编辑Initialize函数，申请资源**

eint HeadMove::Initialize(std::vector<eint> &vFeatureList, ebool& bLimbAuthority, edouble &dTimeOut)

{

vFeatureList.push\_back(SysCom\_ID\_LMsg\_HeadPos); //获取机器人头部位置

bLimbAuthority = true; //申请机器人控制权

dTimeOut = 10;//设置超时时间，若超时申请权限无法满足，框架调JobFailed接口进行通知

return ERR\_NONE;

}

此处为初始化函数，他会在所有函数调用之前调用。

vFeatureList.push\_back() 函数的作用是提供机器人的各种消息，可用的参数有



bLimbAuthority 是用来申请肢体控制权限的 如果bLimbAuthority = true 申请肢体控制；bLimbAuthority = false 不申请肢体控制。

dTimeOut 设置整个过程所需时间，如果到了设置时间系统没有申请成功并且TimeOut没有赋值，则Function会处于一直等待的状态；如果申请没有成功但TimeOut有赋值，则会在等待时间后进入JobFailed。

最后返回值，返回一个ERR\_NONE代表成功，就代表我们的初始化已经成功了，程序就会自动执行下一步。

**编辑CheckMsgCode函数，设置消息处理方式**

eint HeadMove::CheckMsgCode(eint nMsgCode)

{

return MsgProcCode\_Notify; //收到消息后调用对应回调函数

}

**CheckMsgCode（）函数的主要功能是设置消息处理方式。**

**若CheckMsgCode()返回MsgProcCode\_Ignore，该消息会被直接丢弃; 若返回MsgProcCode\_Notify，框架回调用对应的Proc开头的函数进行回调通知(此程序就是采用的这一条); 若返回MsgProcCode\_Record，框架会将消息数据存储，Function模块可调用GetMsgData函数获取最新一次消息数据。**

**编辑JobStart函数，开始头部运动**

void HeadMove::JobStart()

{

eint nErrorCode = 0; //定义返回值

CHeadMoveCmd iCmd; //头部命令实体

iCmd.m\_fYaw = 1.57; //Yaw（绕机器人坐标系Z轴）运动绝对位置，单位弧度

iCmd.m\_fYawSpeed = 0.5; //Yaw方向运动平均速度，单位 弧度/秒

iCmd.m\_fPitch = 0; //Pitch（绕机器人坐标系Y轴）运动绝对位置，单位弧度

iCmd.m\_fPitchSpeed = 0; //Yaw方向运动平均速度，单位 弧度/秒

nErrorCode = SendHeadMove(&iCmd); //发送运动指令，ERR\_NONE为无错误

OSG\_WriteLog(LogPriNormal,"call head move, return code %d",nErrorCode); //将返回值写入日志

return ;

}

**此模块会在系统初始化完成后自动调用。上述示例程序中使用了cHeadMoveCmd这个类，我们在其中设置好了他的运动参数。参数列表如下：**

|  |  |  |  |
| --- | --- | --- | --- |
| **成员变量** | **数据类型** | **含义** | **值域** |
| **m\_fRZ** | **efloat** | **头部水平偏航角度（rad）** | **机器人关节定义** |
| **m\_fRZSpeed** | **efloat** | **头部水平偏航角度运动速度（rad/s）** | **[0.1, 1.0]** |
| **m\_fRY** | **efloat** | **头部倾斜角度（rad）** | **机器人关节定义** |
| **m\_fSpeed** | **efloat** | **头部倾斜角度运动速度（rad/s）** | **[0.1, 1.0]** |
|  | **CBaseCommand** | **基础命令数据** |  |

**编辑ProcHeadPos函数，编写收到头部位置变化消息的响应**

eint HeadMove::ProcHeadPos(edouble dTimeStamp,CHeadPosMessage\* piHeadPos)

{

for(euint m = 0; m < piHeadPos->m\_viHeadJointList.size (); m++)

  {

       OSG\_WriteLog(LogInfo, "头部关节电机，ID = %d, 当前位置 = %.4f, 当前速度 = %.4f",

                    piHeadPos->m\_viHeadJointList[m].m\_unJointID,

                    piHeadPos->m\_viHeadJointList[m].m\_dPosition,

                    piHeadPos->m\_viHeadJointList[m].m\_dSpeed);

  } //头部运动中，将当前位置写入日志

​

return ERR\_NONE;

}

**此部分可以将相关的机器人参数返回传递到控制台。**



**编辑ProcMotCmdDone函数，编写运动结束后的处理**

eint HeadMove::ProcMotCmdDone(edouble dTimeStamp,CCmdDoneMessage\* piCmdDone)

{

OSG\_WriteLog(LogPriNormal,"head move stop, code %d",piCmdDone->m\_nCmdStatus);//运动结束，将执行返回值写入日志

​

return ERR\_NONE;

​

}

此回调表示肢体运动结束，由于运动精度问题，最终位置与预期位置有一定偏差。此处用的是命令执行状态



### 4. 编译

QtCreator菜单项中选择Build -> Build All，编译工程。

### 5. 运行

QtCreator菜单项中选择Debug -> Start Debugging ->Start Debugging，开始调试工程。

机器人头部会运动到左侧90度的位置，同时界面会显示多条head moving日志；机器人停止运动后，界面会显示head move stop.

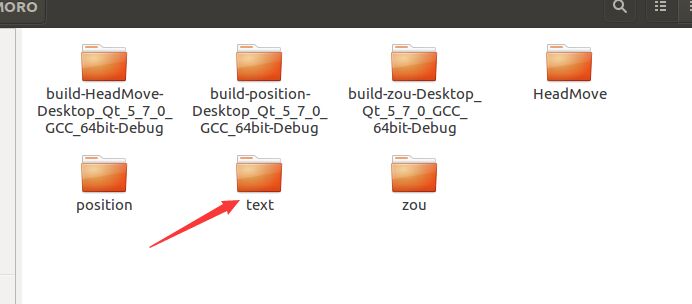
# 手把手教学————编写一个简单的MORO机器人程序

1.找到你要创建工程的地方，打开终端，再输入emake -t fapp -n text -i 192.168.8.100

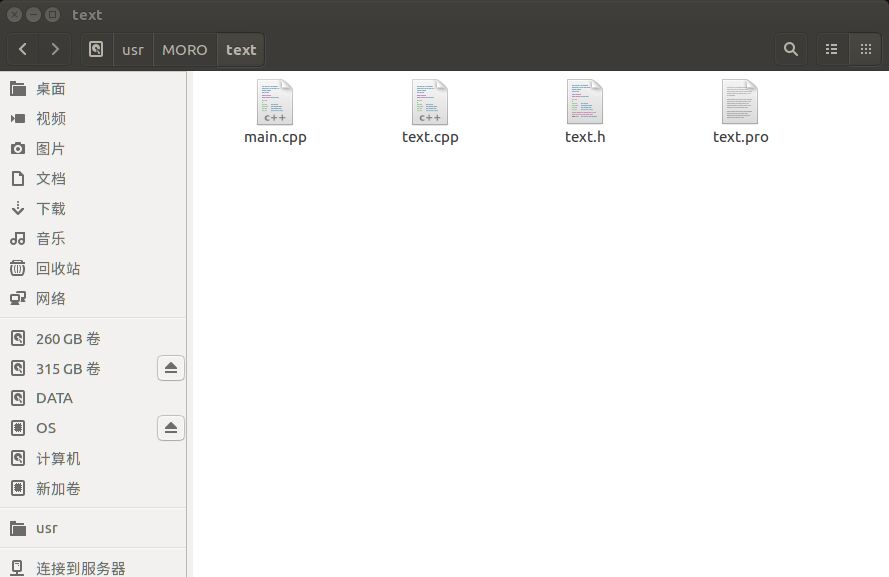
这样就创建了一个名为 text的工程。



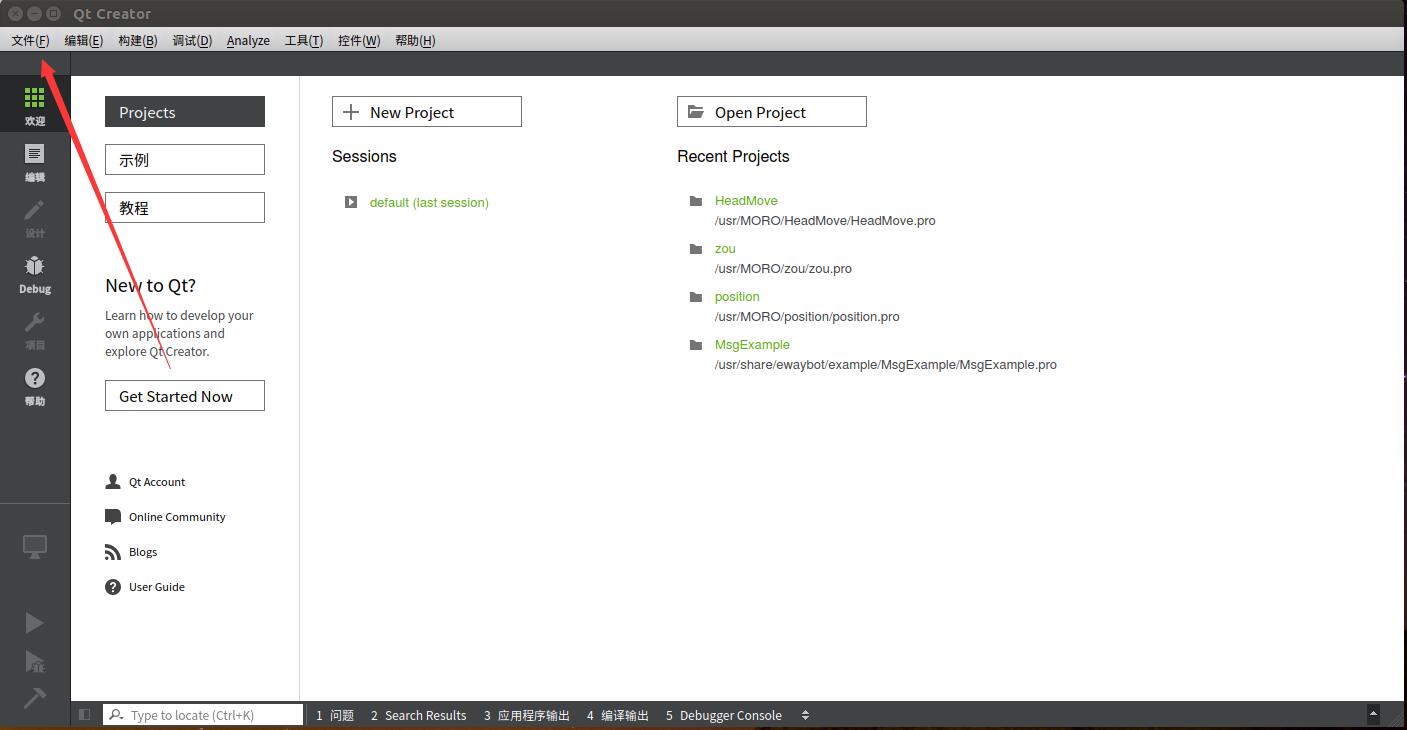
敲下回车过后文件夹就创建好了。



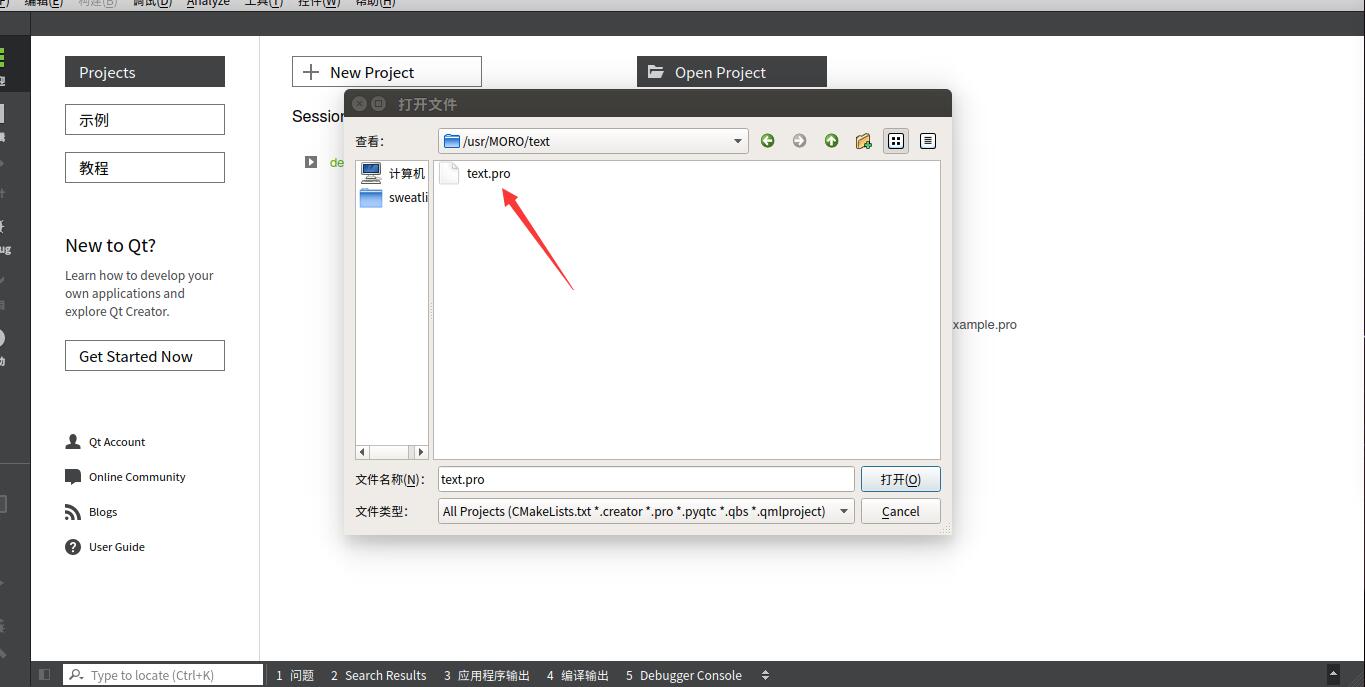
打开文件夹过后就会出现如下图所示的4个文件。



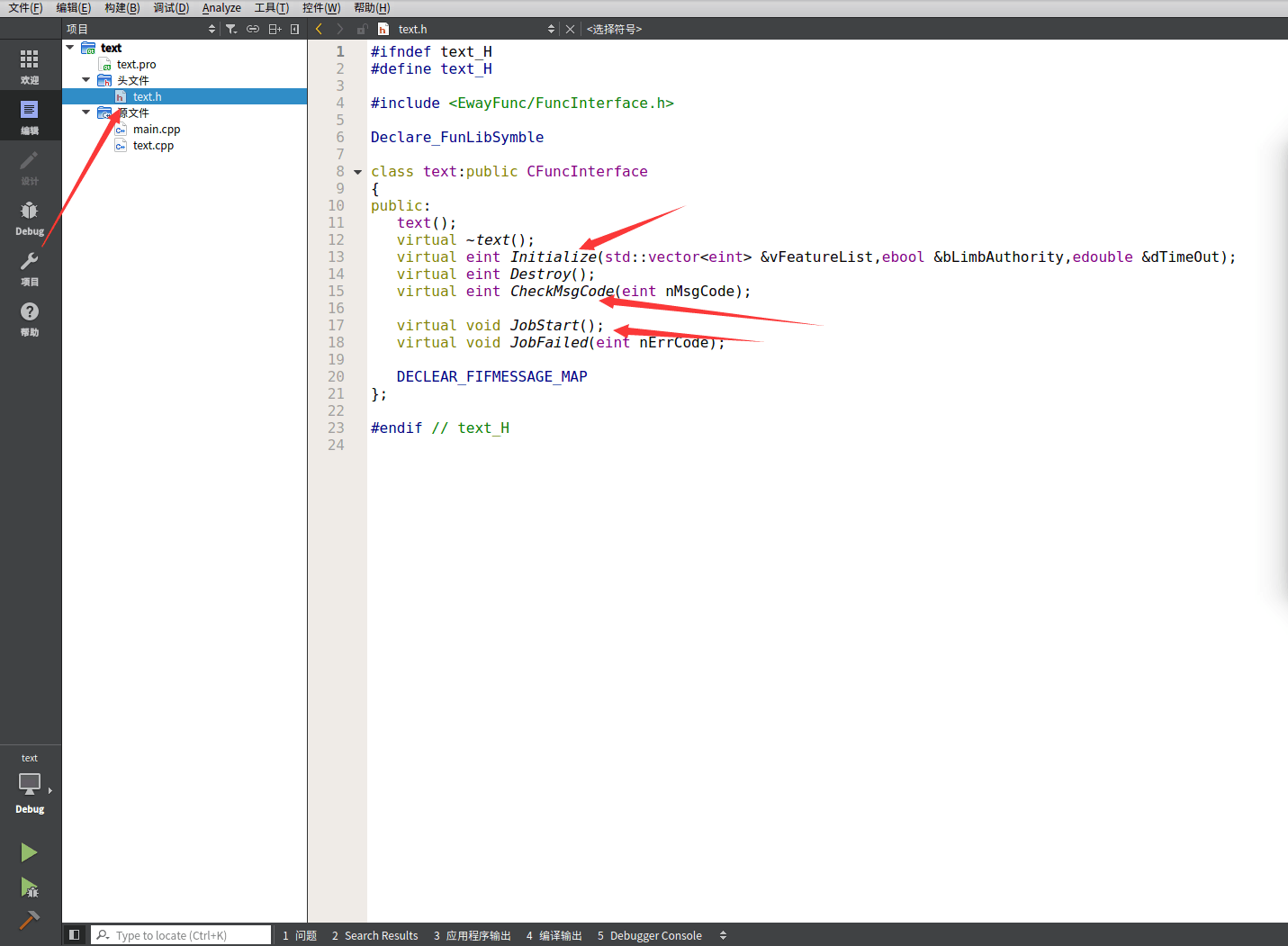
2.打开qt软件，点击文件，选择打开工程。



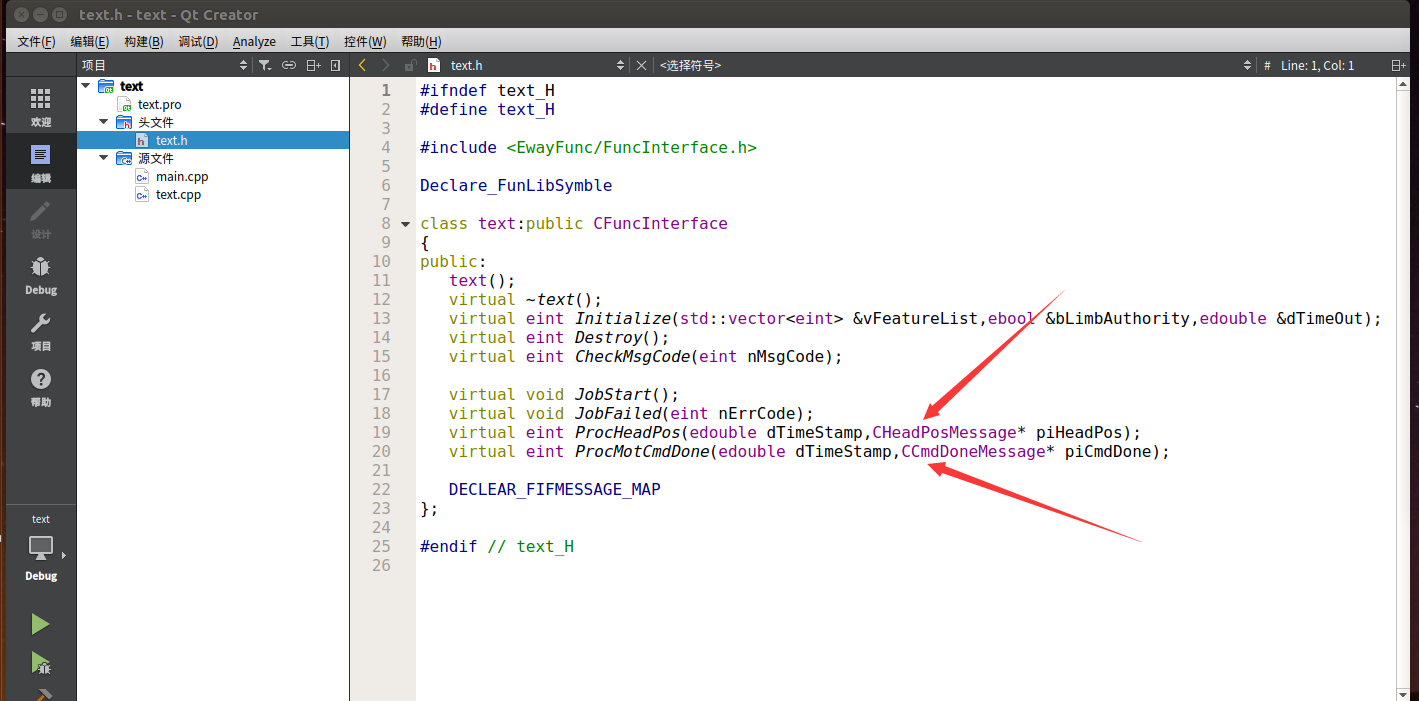
找到刚才所创建的工程的位置，选择text.pro的文件 打开，如图。



3.打开工程过后可以 点开text.h文件，如图所示。

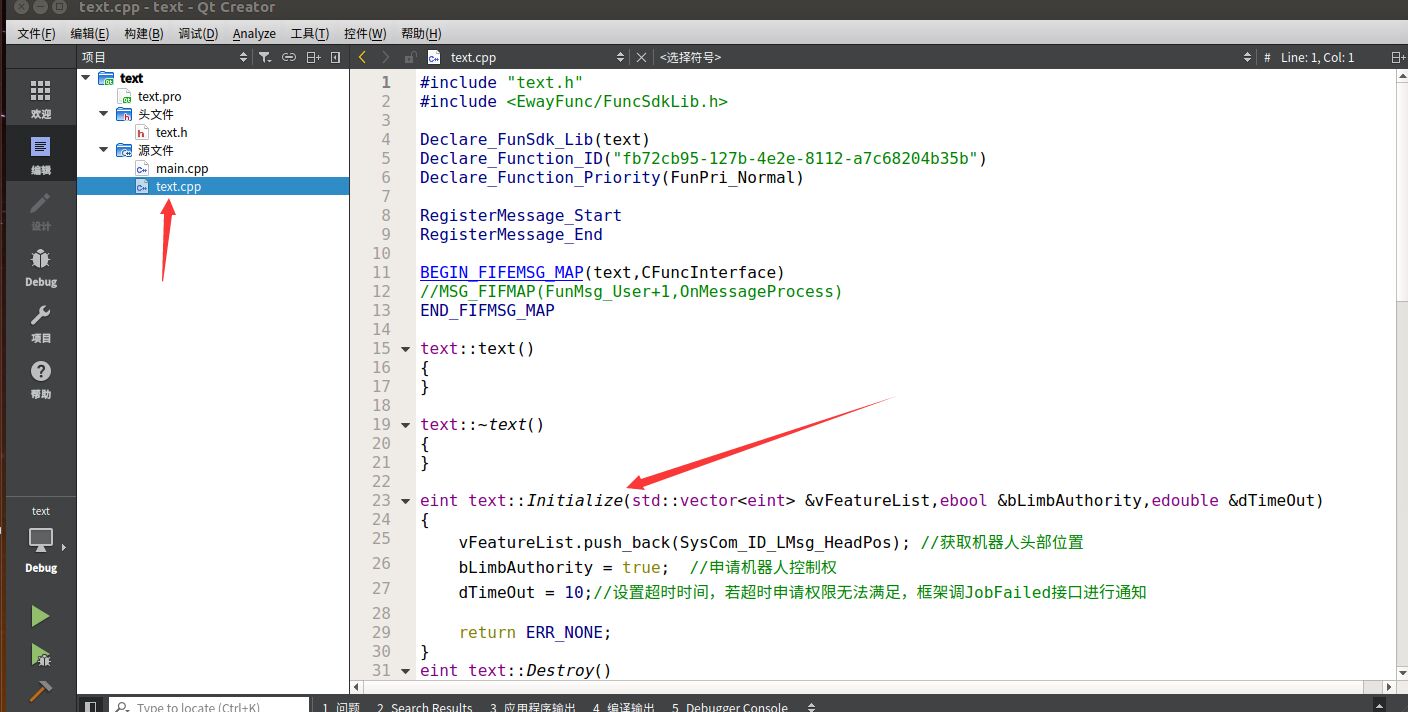


可以看到这里只有我们之前讲到的一部分函数，我们需要将剩下的函数补充完整。如图所示，红色箭头指向的函数为我们在头文件中添加的。

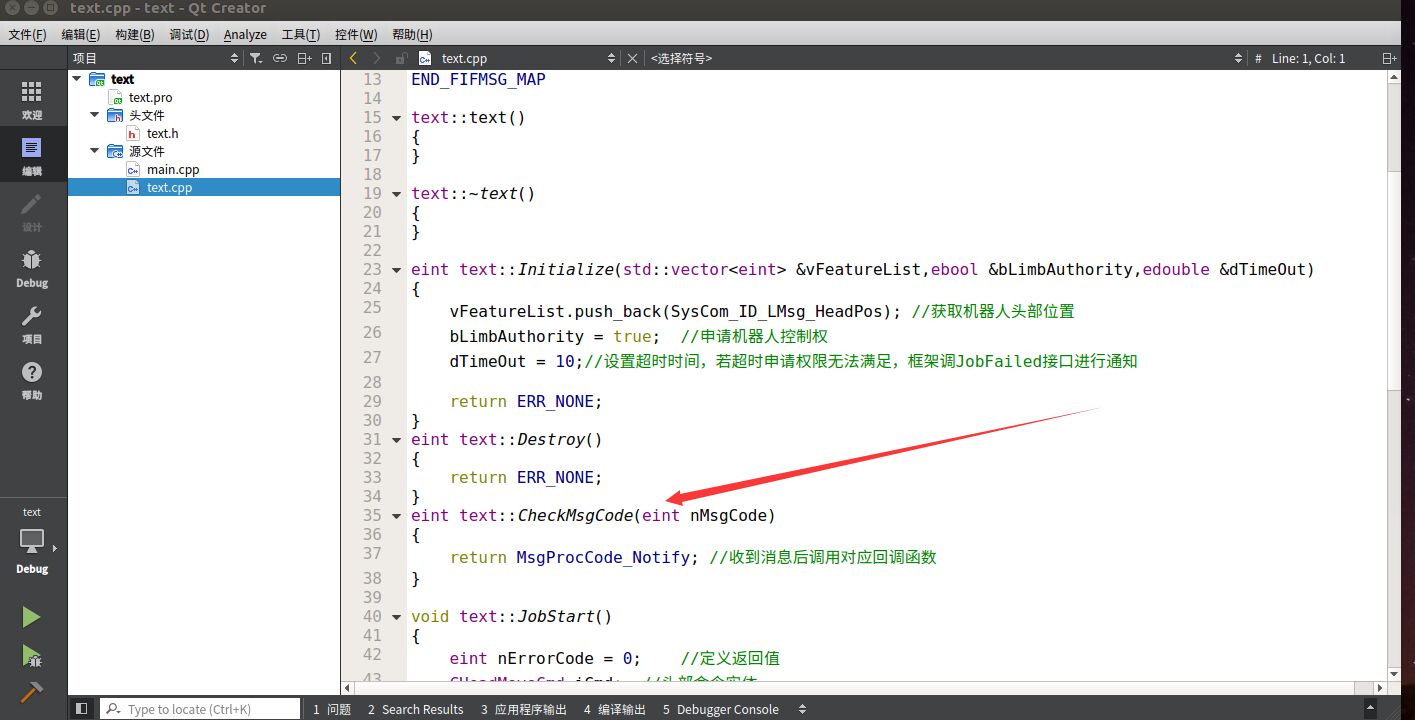


4.打开text.c文件依次写入函数。

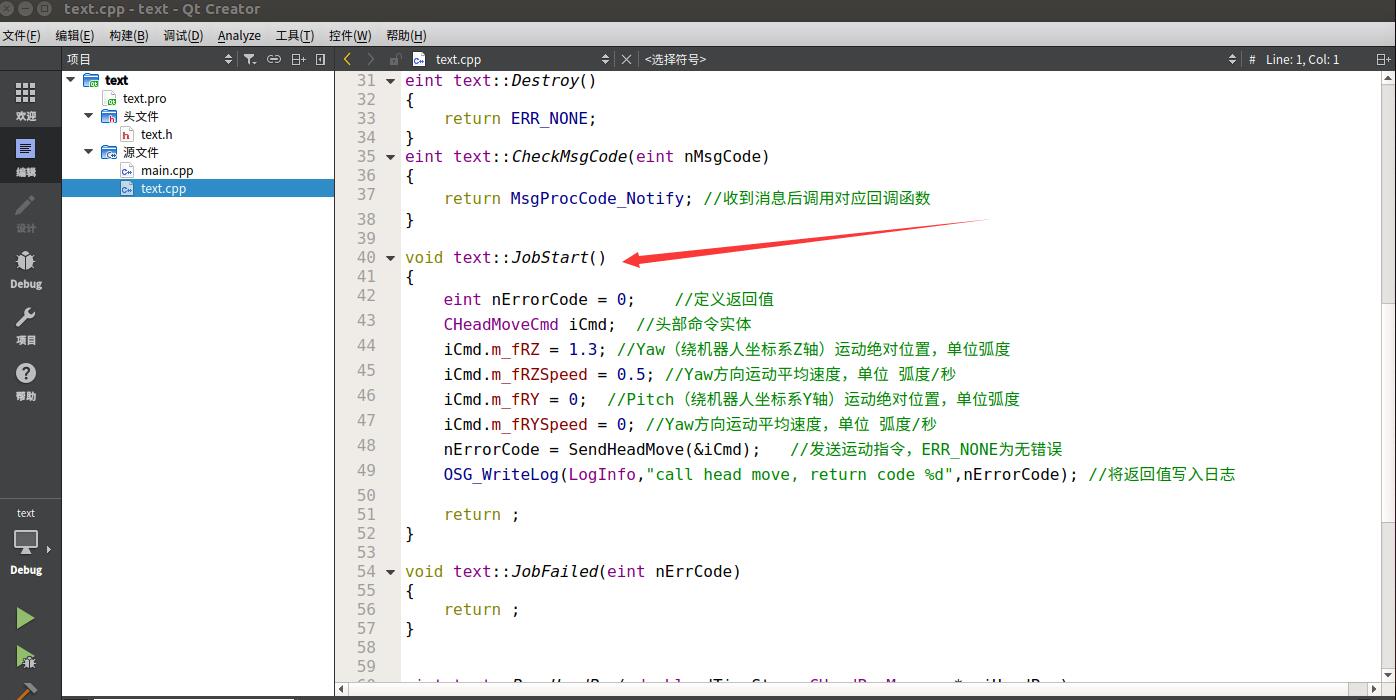
① Initialize()函数



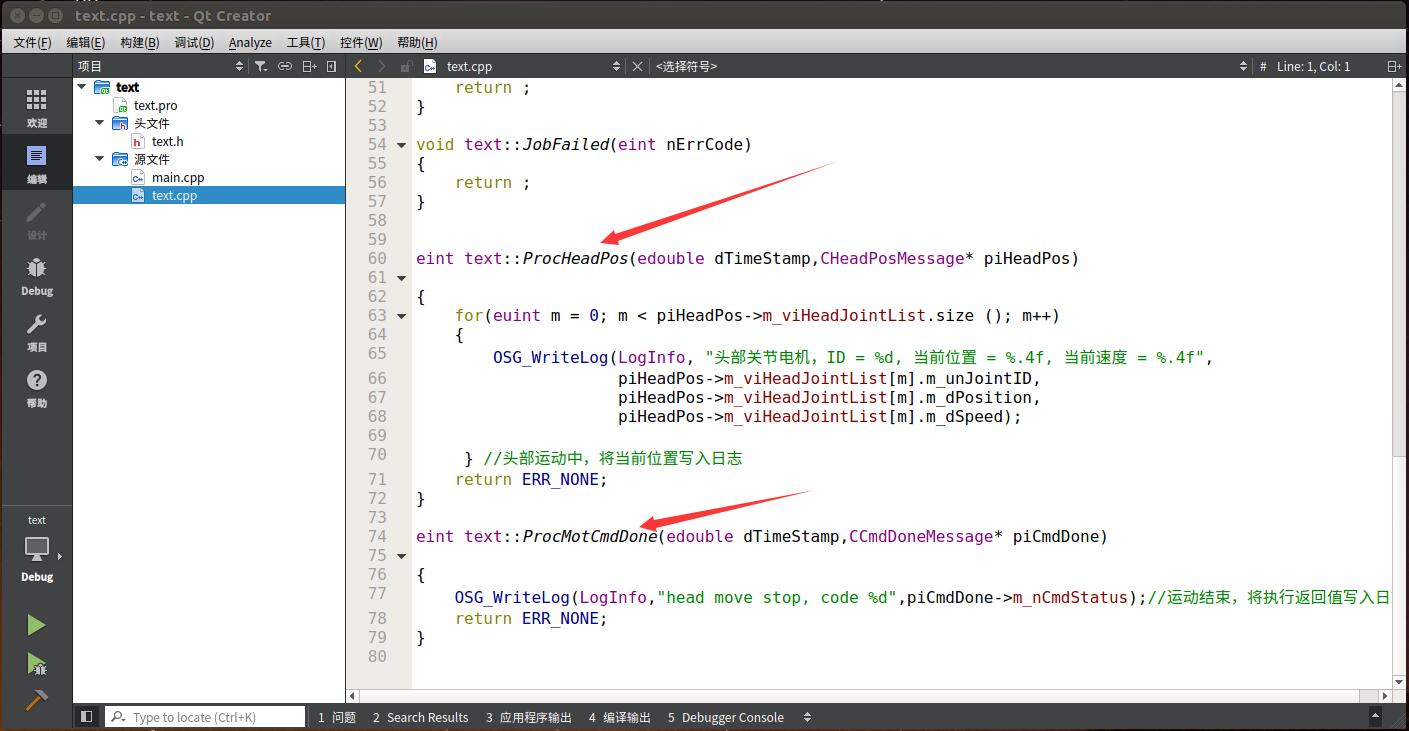
② CheckMsgCode()函数



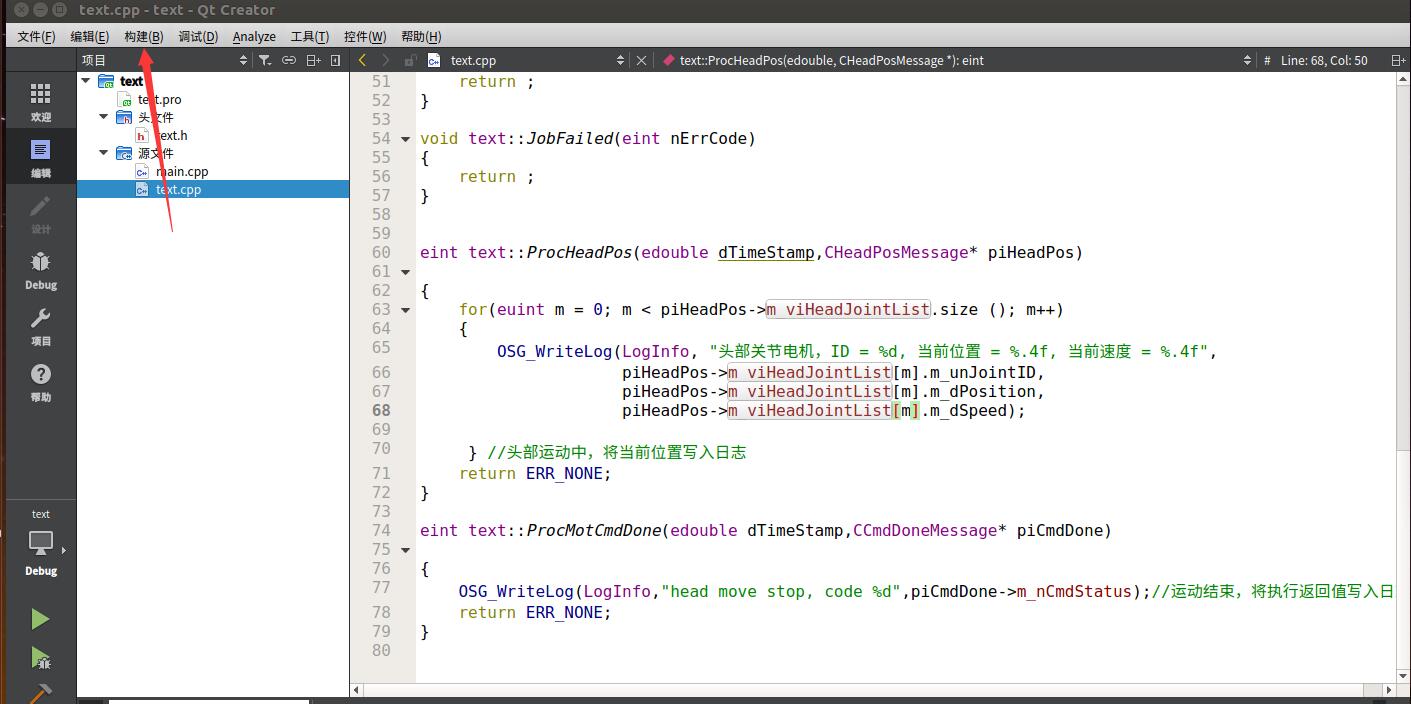
③ JobStar()函数



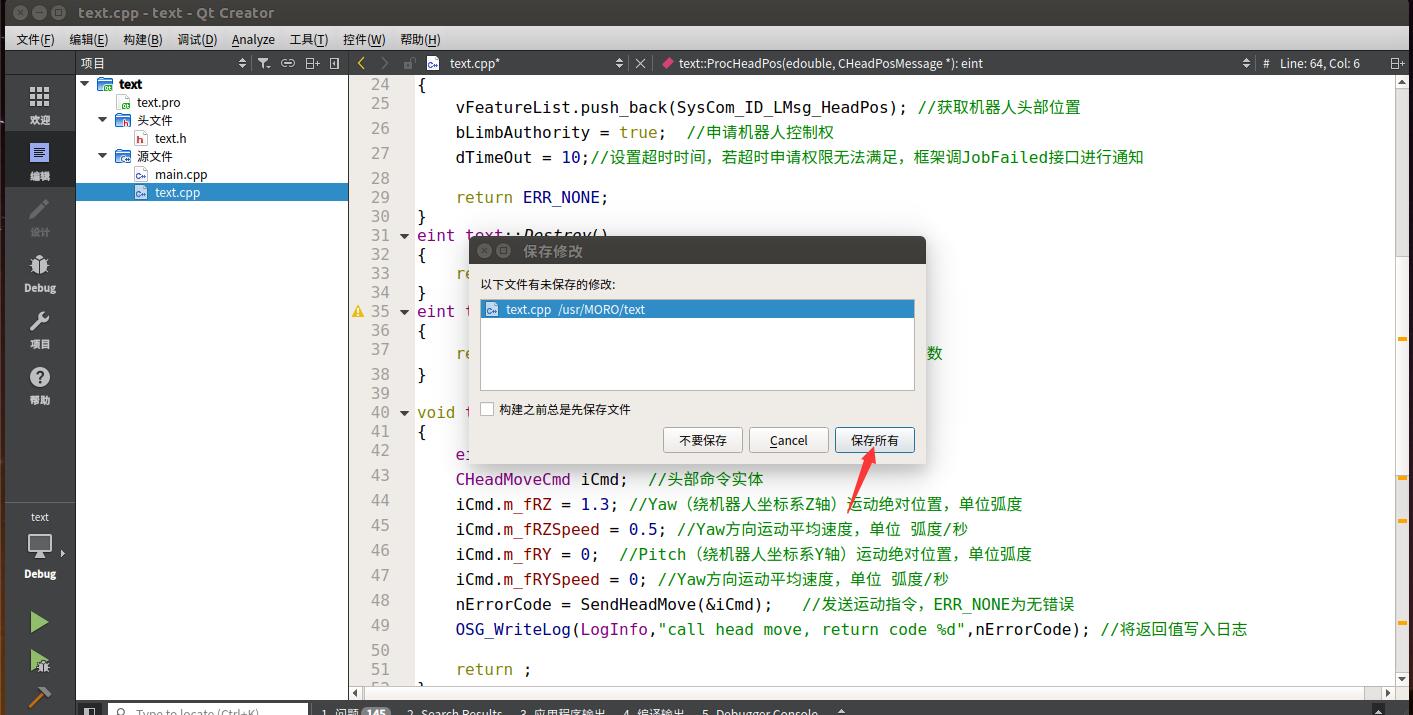
④ ProcHeadPos()函数和ProcMotCmdDone()函数



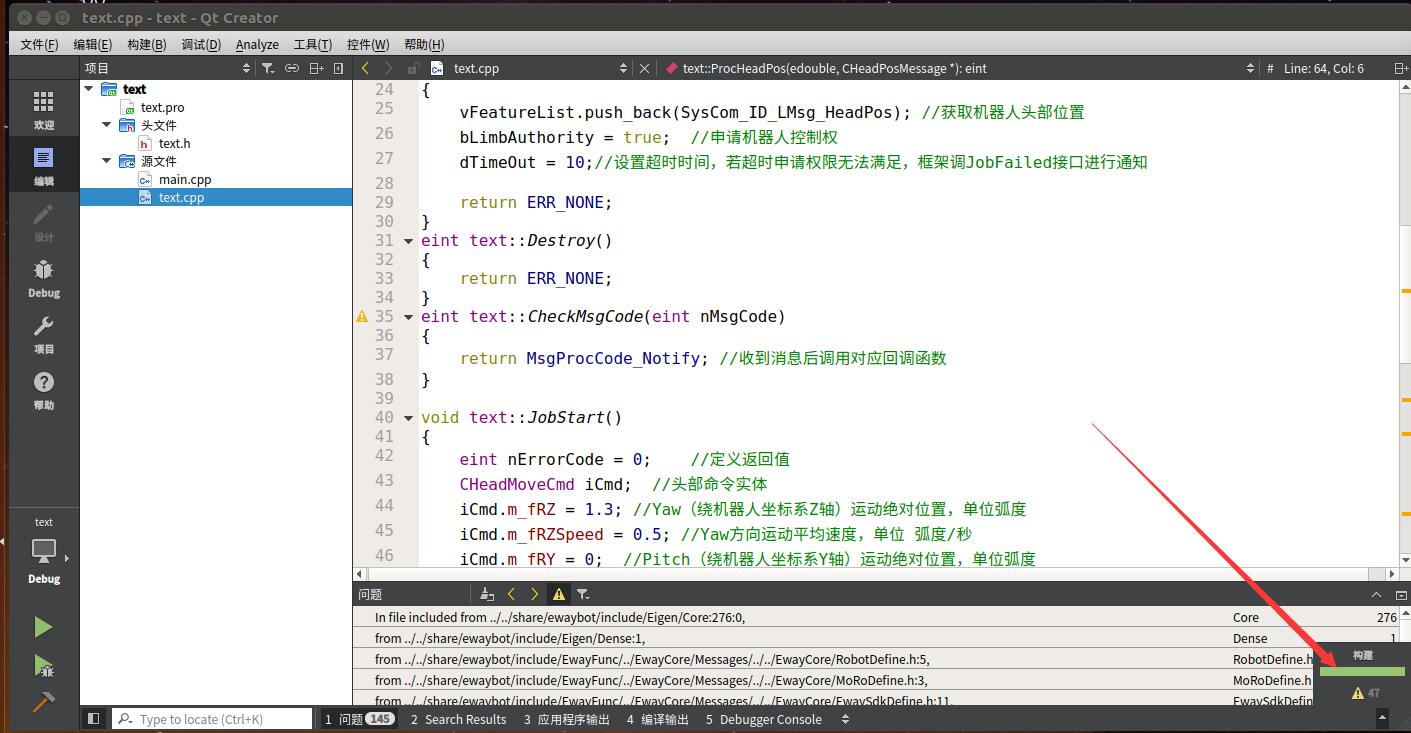
5.所有函数均写完之后就可以构建工程了。



构建点击了之后，选择构建工程，然后会有一个弹窗，选择保留所有。如图所示。

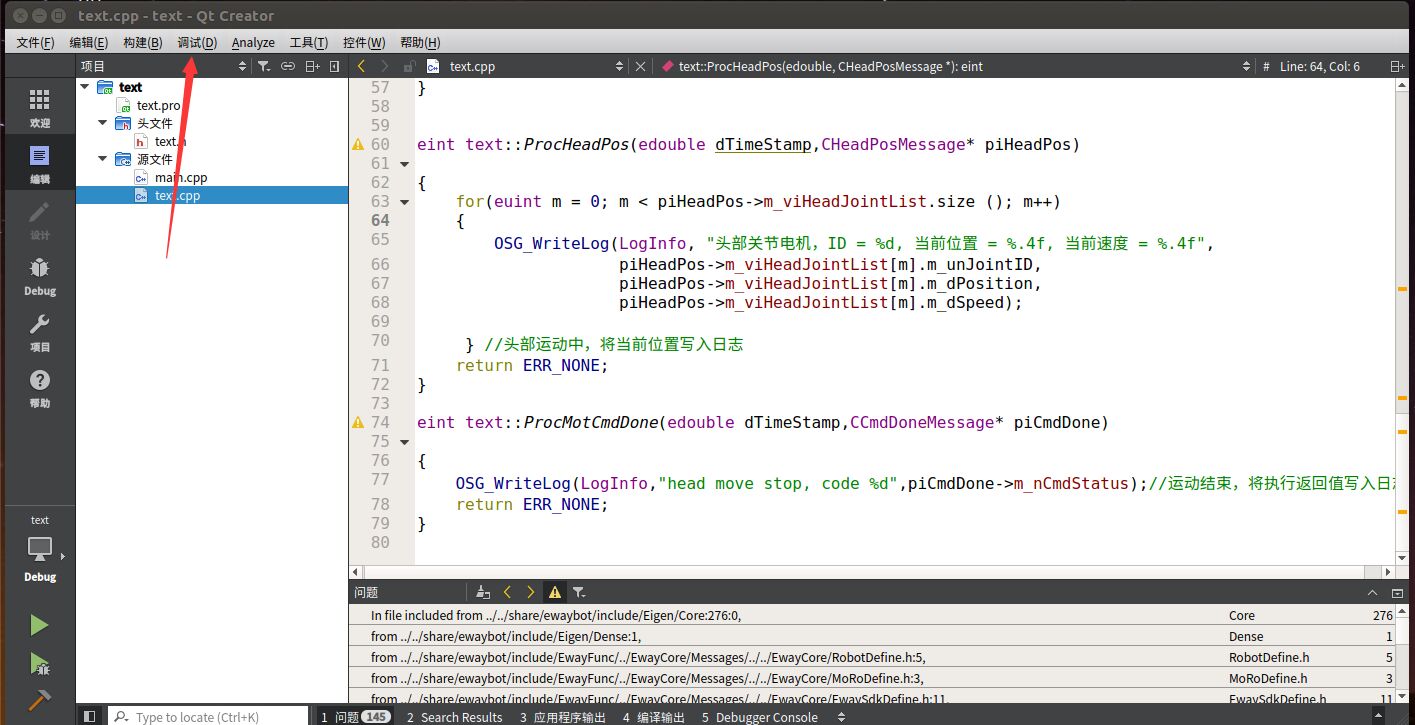


右下角会有一些检测提示，如果有红色小三角就代表编写程序语法有错误，如果没有就代表构建成功没有错误，就可以进行下一步调试了。

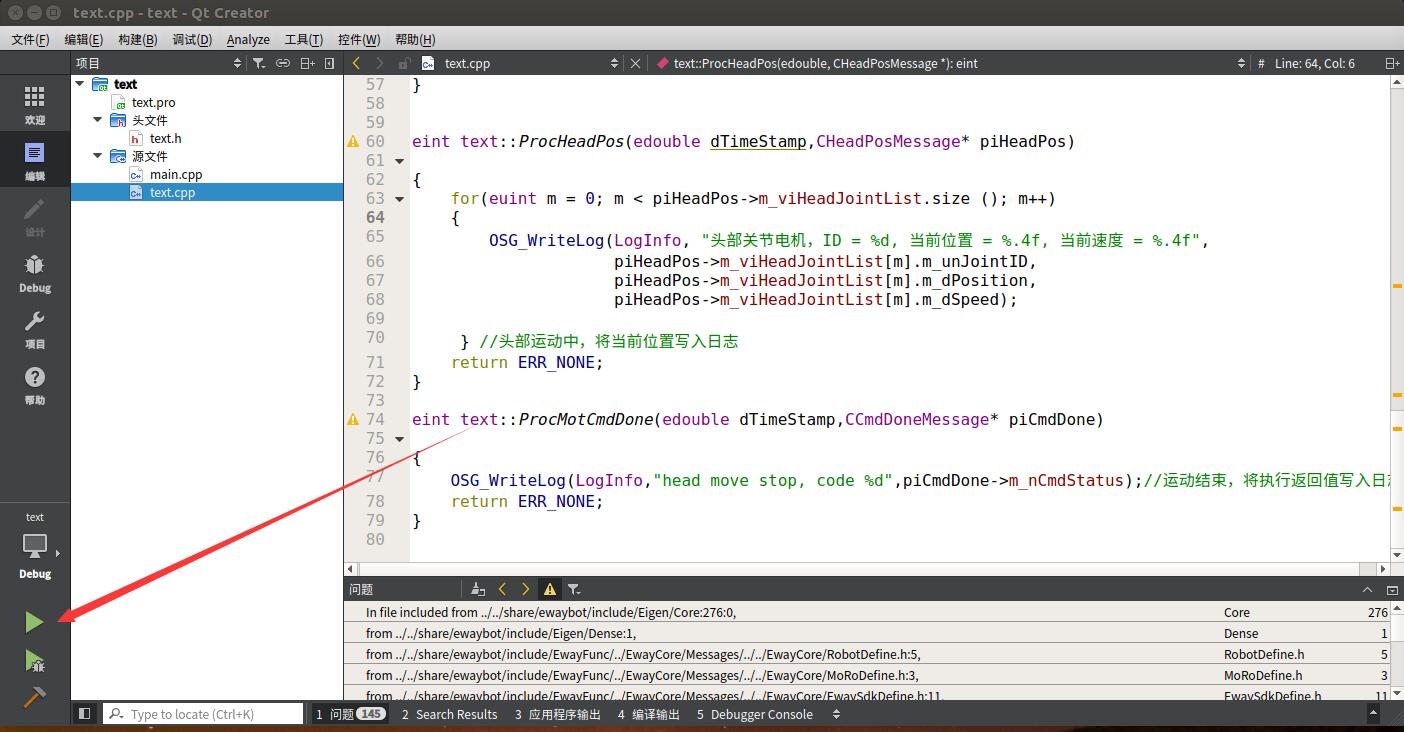


6.调试。

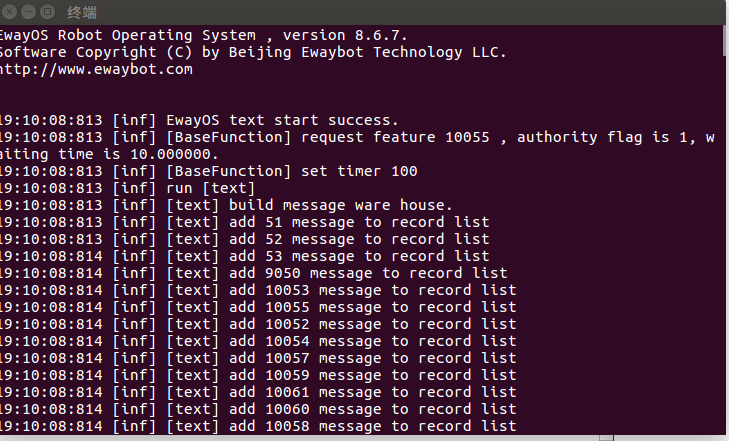
工程构建好了之后，就将电脑连接上moro机器人的WIFI。然后再选择调试，如下图所示



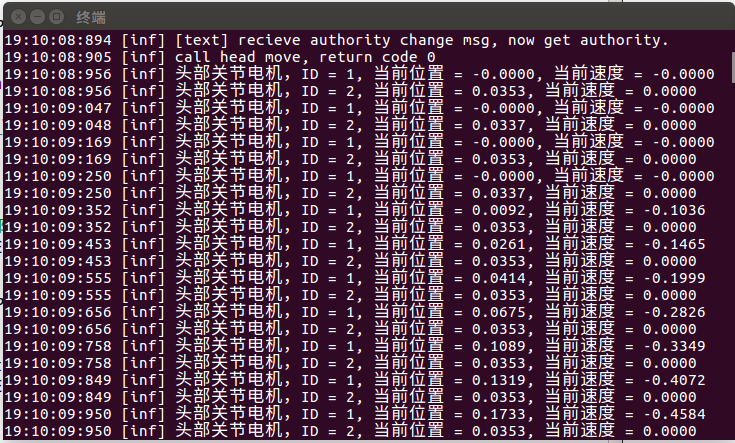
或者点击左下角的播放按钮，如图。



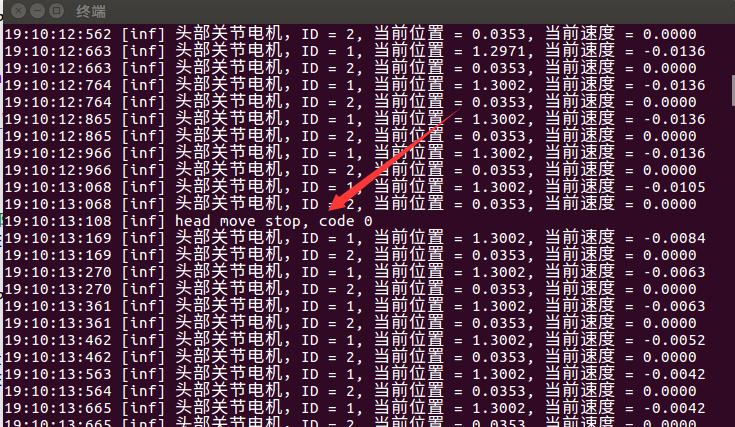
开始调试之后，程序会生成一个控制台程序，他会显示我们执行程序时的一些返回信息。



下图为机器人头部关节的反馈情况，可以看出机器人的头部是从0运动到了1.3 ，速度在这个过程中也有相应的变化。



下图的红色箭头所指的地方则表示我们的机器人的头部运动已经到达了指定位置，头部的运动就结束了。



好了，到现在为止，你已经学会了如何自己写一个机器人应用程序并且对其进行调试了。

# 机器人开发与调试

* 机载PC内部预装了机器人操作系统Eway OS以及开发环境Eway SDK，用常用的PC连接线，将显示器、键盘、鼠标连接至机载PC，即可进行在线开发与调试。
* 非机载PC开发，请联系我公司获取授权的操作系统与开发调试环境
* 具体的开发资料到一维弦技术论坛<http://forum.ewaybot.com/>获取