智能小车回顾总结

1. 主要源文件（模块）

在头文件中定义结构体（首字母大写）、枚举（全大写）、宏定义（全大写）、函数声明；分模块，结构较清晰；注意物理层（bsp、isr、flash等，直接操控外设，为接口层提供自定义函数）、接口层（op、action；给应用成提供接口，操作物理层，不直接操作库函数和寄存器）、应用层（process直接操作接口层），注意让三者层层上传，

* common\_defines.h：共用定义头文件，程序共享的结构体
* sfr.h：special function register，定义板上外设
* parameter.h：参数头文件，机器人主要参数
* share\_defines.h：共享头文件，与提升机、控制界面共享的头文件，方便协作
* abnormal.c：异常检测模块，处理机器运行过程中异常
* bluetooth.c：蓝牙模块，与树莓派之间通信
* bsp.c：外设模块，直接操作板上外设，如：定时器、gpio等
* cmd.c：命令模块，主要用于解析WiFi命令
* common.c：共用模块，如一些通用asc-十进制转换函数
* control.c：控制模块，直接控制机器人动作，如推杆、毛刷等
* flash.c：flash模块，保存、读取一些设置数据
* flow.c：流程模块，控制流程及切换函数
* isr.c：中断服务函数模块
* mpu6050.c：姿态传感器模块
* period.c：不同周期检测模块
* sensor.c：传感器模块
* spi\_ex.c：spi通信模块
* system.c：系统模块，包括初始化参数等函数
* WiFi.c：WiFi模块
* wire.c：有线模块
* data\_debug.c：调试数据模块，后期添加，为了方便调试

1. 主要结构体

* Machine：机器人，Motion\_Component（动作的部件，包含提升机、推杆等结构体）

1. 主函数循环

while(1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

if(control\_flag\_4ms!= flag\_4ms) //3.58msÑ­»·Ò»´Î

{

control\_flag\_4ms = flag\_4ms; //翻转，用于固定循环周期

wifi\_task(); //主要与上位机通信，进行相应处理

wire\_task(); //有线通信，与提升机之间通信

bluetooth\_task(); //与树莓派之间通信，图像处理，绳子偏差

update\_sensor\_data(); //更新所有传感器

period\_update\_run\_status(); //周期检测更新

abnormal\_detect(); //异常检测

flow\_switch(); //切换开关，定义相关切换条件

flow\_process(); //切换流程

motion\_control(); //动作

#if 1

daemon\_aging(); //老化测试

#endif

}

}

1. 问题定位方法

* 用示波器测波形确定运行时间，程序自带调试时显示时间不准确

1. 重要细节

* 流程切换：何时切换？怎么切换？
* 通信实现：标志位？处理流程？
* 机器动作：operation（多做过程）+action（具体动作），注意实现框架

1. LoRa学习：<https://blog.csdn.net/HowieXue/article/details/78028881>
2. 华为射频培训：

<https://wenku.baidu.com/view/1d8d791c6bd97f192279e989.html>

1. 快速平方根倒数算法 - CSDN博客 <https://blog.csdn.net/w450468524/article/details/52529901>
2. 1

公司总结

1. 数据结构：
2. 调用函数放在被调函数后；
3. 学会使用指针回调函数和列表；
4. 利用j = （j + 1） & 0xff形式循环；
5. 使用#define do(...;...;)while（0）；
6. 善用i++，注意数据范围，不确定时用更小范围0xff或111推算；
7. 修改程序后出现错误，可通过注释函数的方法找错误；
8. 多看stm32源码和Linux源码，单条细看，仔细观察里面的数据结构：可读性：看注释、命名等；可移植性：函数封装、架构方式等；面向对象的思路；
9. y = \*(int32\_t \*)&x可借助：int32\_t \*x；y = \*x理解，其实等价于：y = （int32\_t） ，这样做的好处是：先把形参x地址确定了，减轻编译系统负担。
10. 快速平方根倒数算法（【学习笔记】快速平方根倒数算法 - CSDN博客 https://blog.csdn.net/w450468524/article/details/52529901）：

float invSqrt(float x)

{

float xhalf = 0.5f \* x;

int32\_t i = \*(int32\_t \*) &x; //以无符号形式读取

i = 0x5f375a86 - (i >> 1); //计算

x = \*(float \*) &i; //以浮点数读取，此时即为根号近似值

x = x \* (1.5f - xhalf \* x \* x); //使用牛顿方法一遍提高精度

return x;

}

1. 宏定义函数和函数用法（C语言宏定义和宏定义函数\_klugzhong\_新浪博客 http://blog.sina.com.cn/s/blog\_861912cd0100tc94.html.）： 宏定义直接编译插入程序，函数有额外开销，需记录地址、压栈和释放栈；函数需要定义类型，宏定义不用；若实现内容短宜采用宏，较长宜采用函数，因为宏是直接编译的，会造成代码过长，占用内存，通常在多个地方出现宜采用函数；
2. 一阶滤波算法（算法学习笔记之一阶低通滤波算法 - CSDN博客 https://blog.csdn.net/sinat\_23338865/article/details/52672721.）：y = y\*x + y0\*（1-x），x为滤波系数，越小越稳定；
3. 窗口均值滤波：

int16\_t window\_average\_filter(struct Window\_Filter\_Struct \*filter\_data, uint16\_t new\_data)

{

filter\_data->acc = filter\_data->acc + new\_data - filter\_data->data[filter\_data->counter];

filter\_data->data[filter\_data->counter++] = new\_data;

filter\_data->counter &= 0x0f; //利用主函数循环作循环

filter\_data->result = filter\_data->acc >> 4; //求均值 return (int16\_t)(filter\_data->acc >> 4);

}

1. （C标准函数库）拷贝函数：void \*memcpy(void \*dest, const void \*src, size\_t n);
2. 初始化函数：void \*memset(void \*s, int ch, [size\_t](https://baike.so.com/doc/6847447-7064872.html) n); size：字节长度
3. 在头文件中声明函数；
4. 在头文件中定义函数；
5. Ppc : PowerPC Figure – PPC入门与优化 - CSDN博客 <https://blog.csdn.net/xxxl/article/details/43502203.>
6. 有很多C/C++静态代码检查工具，其中Logiscope RuleChecker和PC-Lint 是应用比较广泛的两个工具。
7. Pc\_lint代码检查，百度pclint错误大全；
8. 433（433M）能传15km，频率最小相差0.5M不会相互干扰；WiFi（2.4G）能传2km。
9. 无线通讯培训资料：https://wenku.baidu.com/view/1d8d791c6bd97f192279e989.html

；lora模块相关知识：https://blog.csdn.net/HowieXue/article/details/78028881。

1. STM开发总结：
2. 在配置定时器、串口等外设CR、SR时需先失能配置使能；
3. STM开发问题总结
4. STM32F1库函数\_\_HAL\_AFIO\_REMAP\_TIM2\_PARTIAL\_1()会将复用映射寄存器MAPR时配置成了JTAG下载;需紧接其后加：

AFIO->MAPR &= 0x0ffffff;//avoid debuger disconnect

AFIO->MAPR |= 0x2000000;

1. DMA读取ADC数据时个一段时间会丢掉前几位数据，读到ADC缓存数组的数据整体前移。

解决方法：DMA 读取数据前清空缓存数组，然后在读取缓存数组前判断数组最后一位内部电压是否为0，为0就丢弃数组返回；void get\_us\_data(void)由于偶尔还是会出错，就在异常判断时，当出现连续2次电压过低时才上报异常。start\_adc\_dma();放在函数最后.注意，在判断数组前要留有足够时间给DMA读数。

1. 使用激光超声波时打印出数据用图画出来，选择合适的算法滤波，测试数据主要
2. 激光测距精度2mm，但与光强关系较大，只在一定光强范围测得距离是准确的。测量玻璃距离时可能会偏小（实际2m，测得0.525m）；玻璃后有障碍物会测到障碍物距离；从障碍物到玻璃会出现测不到数据的情况；激光测到边缘时，测量值不会突变，会缓慢变化，5、6次左右。
3. 超声波测距精度1.5cm；有发散角，可通过加挡板的方法减小发散角，四面都加会造成信号过强；超声波盲区8mm。
4. 电磁干扰问题：机器运行时，仿真器不能正常工作，自动设断点；解决方法：a、改变仿真器位置和方向；b、加金属套筒屏蔽干扰；c、在下载线SWCLK和SWDIO加100R电阻（最终解决方法，因为干扰通信线）。
5. 使用stm8模拟串口通信，在逻辑没问题的情况下，可能会因为定时器等硬件问题造成错误，需增加错误检测，防止死循环；
6. Stm32F1库函数问题：STM32F1的串口dma接收库函数接收存在问题，造成代码接收数据出错；
7. 算法总结
8. 增量式PID：
9. 一阶滤波（y=x\*y + （1-x）\*y\_latest）：可通过调节滤波系数x改变数据的稳定性。
10. 窗口滤波：可以通过增加窗口数据个数增加数据稳定性，可通过判断窗口数据与窗口均值的差值判断数据是否异常。