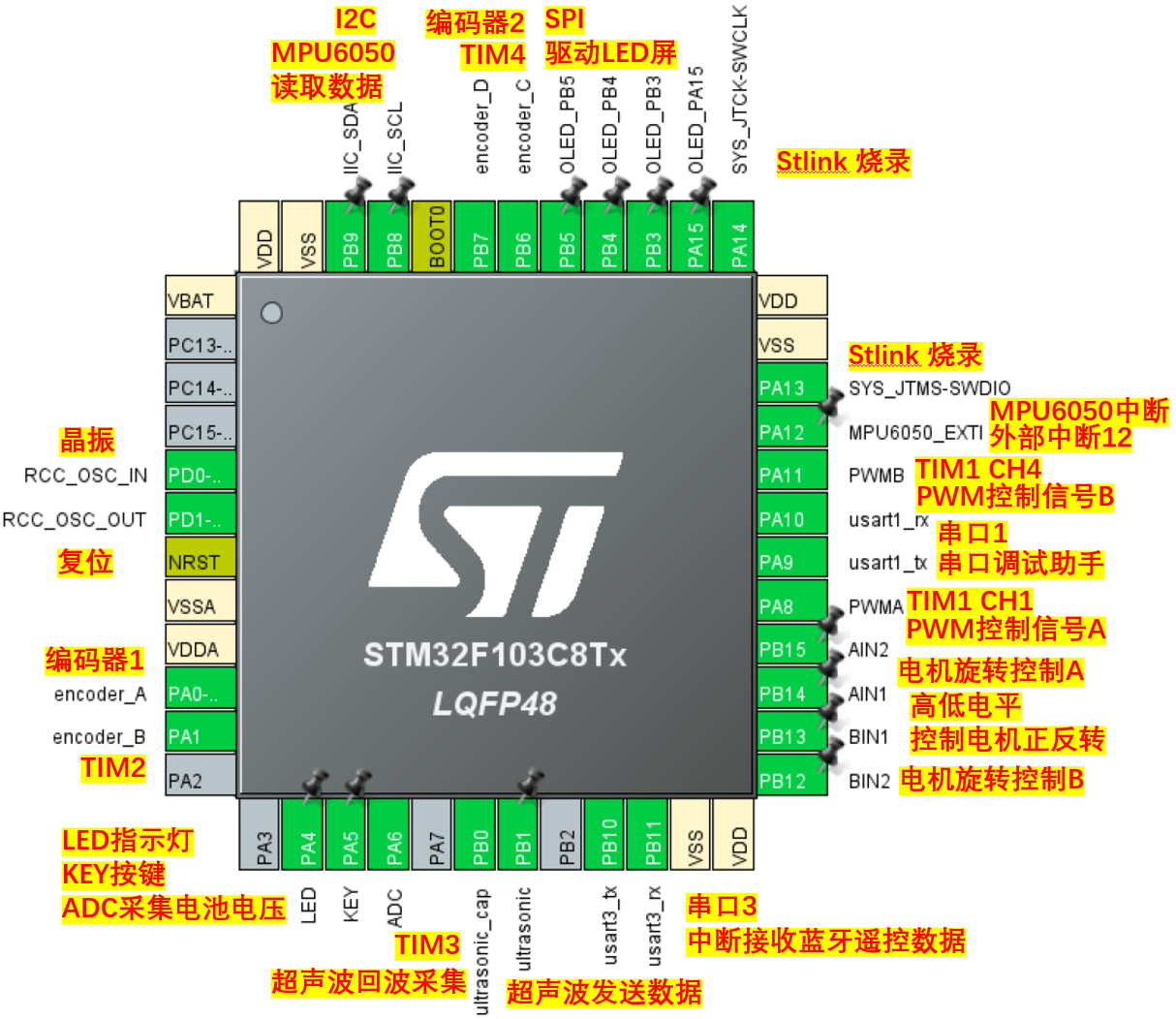
1. 引脚定义：



2. 程序初始化：

1. GPIO配置，没问题；

2. 串口：  
串口1: 115200，串口3: 9600，开启中断接收  
串口发送函数，也不是很理解啊……

3. ADC：

这里ADC是**软件触发单次转换**！转换结束后关闭，无需配置DMA；  
通道为ADC1的Channel6 （PA6引脚）  
采样时间为239.5周期，即21μs  
函数：HAL\_ADC\_PollForConversion();执行单次ADC转换，HAL\_ADC\_GetValue()得到转换结果；

4. 中断优先级：  
中断优先级分组还是4(全 是主优先级，没有子优先级)  
串口3(蓝牙) > TIM3(超声波) > EXIT12(MPU6050时钟)

5. 定时器：

5.1 TIM3 超声波；  
分频器：72-1；计数器：65536-1；配置为输入捕获模式；  
即周期1μs，计数器最大计数: 65535

5.2 TIM1 输出PWM波  
CH1 PWM1；CH4 PWM2；  
分频器：1-1；计数器：7200-1；占空比：0；有效电平为高电平；空闲(刹车)状态为低电平；  
PWM频率：Freq = CK\_PSC / (PSC + 1) / (ARR + 1) = 72M/1/7200 = 10000 (周期：0.1ms)  
PWM占空比：Duty = CCR / (ARR + 1) = 0/1000 = 0  
刹车/死区功能均关闭；

5.3 TIM2/TIM4 编码器计次  
分频器：1-1；计数器：65536-1；   
编码器模式：接口TI1和TI2双线计数；编码器1 2均上升沿计次

6. 其他：  
JTAG\_Set();这个确实不懂，  
delay\_init(); 设置寄存器，与之前相同，但初始化没有懂，  
OLED\_Init(); 我第一次见 SPI 传输使用 SDA/SCL的，这是什么屏啊  
屏幕也没有手册也没有驱动……

3. 软件驱动开发：

1. 编码器：  
程序是直接读取寄存器的，Read\_Encoder(); TIM2 -> CNT;  
编码器测速原理，与例程相同；

2. I2C驱动   
不如江科大的，不如自己写的；

3. 按键KEY

click()里面的flag\_key作用：单次按键也会持续一段时间，第一次按下flag=0;return 1; 后续在未松开前，由于flag=0; 一直return 0;直到松开，flag=1;

Long\_Press()的Long\_Press==0; 意义不明；

click\_N\_Double();其内部操作有些复杂，只要按键按下，Forever\_count++;第一次按键按下，flag\_key=1; double\_key=1; count\_key=1; 之后如果还是持续按下状态，由于count\_key=1参数不变；松开，flag\_key=0,count\_key=0; 然后在按下，flag\_key=1; double\_key=2; return 2(双击)；如果在这期间超时，return 1(单击); 如果发现是一直按着的，就return 0(无动作);

4. LED闪烁  
easy

5. OLED显示屏  
看不懂，不管他，不如看/学习江科大的；

6. 蓝牙(串口3)：  
整个就是一个中断回调函数；（接收到数据执行操作）  
接收到数据后，数据保存至Usart3\_Receive\_buf，之后赋给变量uart\_receive和Usart3\_Receive，

根据uart\_receive，判断档位/前进后退拐弯；

根据Usart3\_Receive，判断采集数据/分析数据，分析数据即PID调整参数，  
数据帧Receive[]格式为：[0]0x7B(开始) , [1]0x3x(PID被设置变量), [2], [3]设置PID Send, [3:end]PID变量设置值Data(LSB高位先行，第i位值\*1，第i-1位值\*10，第i-2位值\*100……), 0x7D(结束)；

7. 超声波模块：  
超声波工作原理见：开发手册4.5节；

模块在发出信号后，自动产生上升沿信号，当模块接收到回波后，产生下降沿信号；  
超声波使用输入捕获：

定时器更新，就会进入HAL\_TIM\_PeriodElapsedCallback();中断，此时(TIM3CH3\_CAPTURE  
\_STA&0X80)==0，无操作; 此时若捕获到上升沿，进入HAL\_TIM\_IC\_CaptureCallback();中断，TIM3CH3\_CAPTURE\_STA=0，TIM3CH3\_CAPTURE\_VAL=0，TIM3CH3\_CAPTURE  
\_STA|=0X40（标记捕捉到了上升沿），之后设置为下降沿捕获； 此时，在HAL\_TIM\_  
PeriodElapsedCallback();中断中，会执行TIM3CH3\_CAPTURE\_STA++操作，之后若此时再次捕获到中断，进入HAL\_TIM\_IC\_CaptureCallback();中断，TIM3CH3\_CAPTURE\_STA|  
=0X80; TIM3CH3\_CAPTURE\_VAL=捕获值; 捕获函数重新配置为上升沿；

若TIM3CH3\_CAPTURE\_STA++过程中==0x3F， 则高电平太长了，此时强制设为捕获到低电平，重新初始化进入主程序；，

主程序中，PB电平跳变，发出信号，TIM3CH3\_CAPTURE\_STA&0X80==1，成功捕获高电平，(TIM3CH3\_CAPTURE\_STA&0X3F)\*65536+ TIM3CH3\_CAPTURE\_VAL = 高电平总时间，之后\*声速即为距离；

8. 屏幕show展示：  
oled\_show(void); easy能看懂；

APP\_Show(void); 需要了解协议，还是可以看懂的；

PS：蓝牙发送数据，直接使用prinf()发送的；

DataScope(void); 没懂！需要首先学会上位机操作！！！

需要结合DataScope\_DP一起看；

4. 控制算法开发：

1. PID算法

2. MPU6050获得角度方法

三种得到准确的角度的算法：1.DMP 算法 2.互补滤波算法 3.卡尔曼滤波算法。

①. 互补滤波：推导看不懂，结论还行？程序还行？

② 卡尔曼滤波：看不懂。

③ DMP 算法？没注释。

5. 主控程序：

1. Balance() 直立PD控制（比例 微分）

入口参数：Angle:角度；Gyro：角速度

Angle\_bias=Middle\_angle-Angle; Gyro\_bias=0-Gyro;

误差 = 目标值 – 实际值

balance=-Balance\_Kp/100\*Angle\_bias-Gyro\_bias\*Balance\_Kd/100;

前：P 后D

/100，应该是细化偏差；

角速度是角度的微分！

P\*角度偏差 + D\*角度微分(角速度)偏差

2. Velocity () 速度PI控制（比例 积分）

入口参数：encoder\_left：左轮编码器读数；encoder\_right：右轮编码器读数

①. 遥控前进后退 & 超声波功能（跟随/避障）

Movement=Target\_Velocity/Flag\_velocity（目标速度/档位？） Flag\_velocity表示低速/高速档

其实就是根据 不同条件判断：Movement的正负！

②. 速度PI控制器

Encoder\_Least =0-(encoder\_left+encoder\_right); //获取最新速度偏差=目标速度（此处为零）-测量速度（左右编码器之和）

低通滤波器：

上一次的Encoder\_bias\*0.86 + 这一次的Encoder\_Least\*0.14 起到平滑滤波作用，

Encoder\_Integral +=Encoder\_bias; 积分呗

Encoder\_Integral=Encoder\_Integral+Movement; 就是除了本身的偏差值，还要加上实际运动情况； 等等等

velocity=-Encoder\_bias\*Velocity\_Kp/100-Encoder\_Integral\*Velocity\_Ki/100;

前：P 后I

P\*速度偏差 + I\*速度偏差积分（这回不是角度了？）

3. Turn () 转向

入口参数：Z轴陀螺仪

Turn\_Target=Turn\_Amplitude/Flag\_velocity;

转向的目标值（是一个常数）

转向的PID控制，和平衡类似

4. Set\_Pwm () / PWM\_Limit()

控制电机正反转 / 限制PWM最大最小值

5. Key () 按键修改小车状态

6. Turn\_Off() 关闭电机

电池电压低/倾角>40/以及Flag置1 关闭

7. Get\_Angle() 获取角度的最终函数

其实就是MPU6050

Read\_DMP();

Angle\_Balance=Pitch; // 实际角度 = Pitch

Gyro\_Balance=gyro[0]; // X角速度

Gyro\_Turn=gyro[2]; // Z角速度

Acceleration\_Z=accel[2]; // 更新Z轴加速度计

这里没有滤波了？？？

8. Pick\_Up () 检测小车是否被拿起

条件1，小车接近静止；

条件2，小车是在0度附近被拿起；

条件3，Z轴加速度>26000(意思就是小车被拿起)；

条件4，小车的轮胎因为正反馈达到最大的转速

9. Put\_Down () 检测小车是否被放下

条件1，小车是在0度附近的

条件2，小车的轮胎在未上电的时候被人为转动

10. Get\_Velocity\_Form\_Encoder() 编码器值转换为速度

电机转速：转速=编码器读数（5ms每次）\*读取频率/倍频数/减速比/编码器精度

这个与自己的编码器不同啊！

11. Choose() 选择小车的运行模式

Easy；

0. EXTI 5ms 外部中断回调，执行所有操作！

得到角度→得到编码器值→确定速度

5ms：读取电压并求均值

10ms：获取距离→按键/普通LED控制→平衡PID控制/速度PID控制/转向PID控制→以上三者相加得到最终PID→检测抬起/放下→选择小车模式→设置PWM