**景德镇陶瓷大学**

**课 程 设 计 说 明 书**

**题目： stm32蓝牙智能小车**

**学院（系）： 机械电子工程学院**

**年级专业： 15级自动化1班**

**学 号： 115040200135**

**学生姓名： 叶剑武**

**指导教师：**

**摘要**

本文的主要工作是基于STM32蓝牙智能小车的设计，随着越来越多的高科技产品逐渐融入了日常生活中，步进电机控制系统发生了巨大的变化。单片机、C语言等前言学科的技术日趋成熟与实用化，使得步进电机的控制系统有了新的研究方向与意义。本文描述了一个由STM32微处理器、步进电机、蓝牙，L298N等模块构成的。该系统采用STM32微处理器为核心，在MDK环境下进行编程，通知控制L298N模块IN1~IN4引脚的高低电平进而控制电机的转动方向，使小车产生不同的转动模式。控制小车的转动模式通过蓝牙模块来实现，利用手机蓝牙助手和蓝牙模块相通，进而控制小车的各种运转模式。结果表明该系统具有结构简单、工作可靠、精度高等特点。

**关键词**：STM32微处理器；蓝牙智能小车；

1. **蓝牙智能小车简介**

小车可以分为三部分——传感器部分、控制器部分、执行器部分。

控制器部分：接收传感器部分传递过来的信号，并根据事前写入的决策系统（软件程序），来决定智能小车对外部信号的反应，将控制信号发给执行器部分。好比人的大脑。

执行器部分：驱动智能小车做出各种行为，包括发出各种信号（点亮发光二极管、发出声音）的部分，并且可以根据控制器部分的信号调整自己的状态。对智能小车来说，最基本的就是轮子。这部分就好比人的四肢一样。

传感器部分：机器人用来读取各种外部信号的传感器，以及控制机器人行动的各种开关。好比人的眼睛、耳朵等感觉器官。

无线部分：用蓝牙从机的TTL接口接收信号反馈给单片机，还可以通过电脑的无线局域网网卡来控制路由器的TTL串口来控制单片机的状态。由于蓝牙模块容易得到且接口方便，电压可以用+5V控制，携带方便，便于在小车上安装。

电机驱动：电机驱动需要大电流，较高的电压，考虑到单片机的负载能力，用L298N驱动模块来驱动电机，单片机输出逻辑信号控制L298N模块输出正反向电压，由此控制电机的正反转。

1. **系统硬件设计**

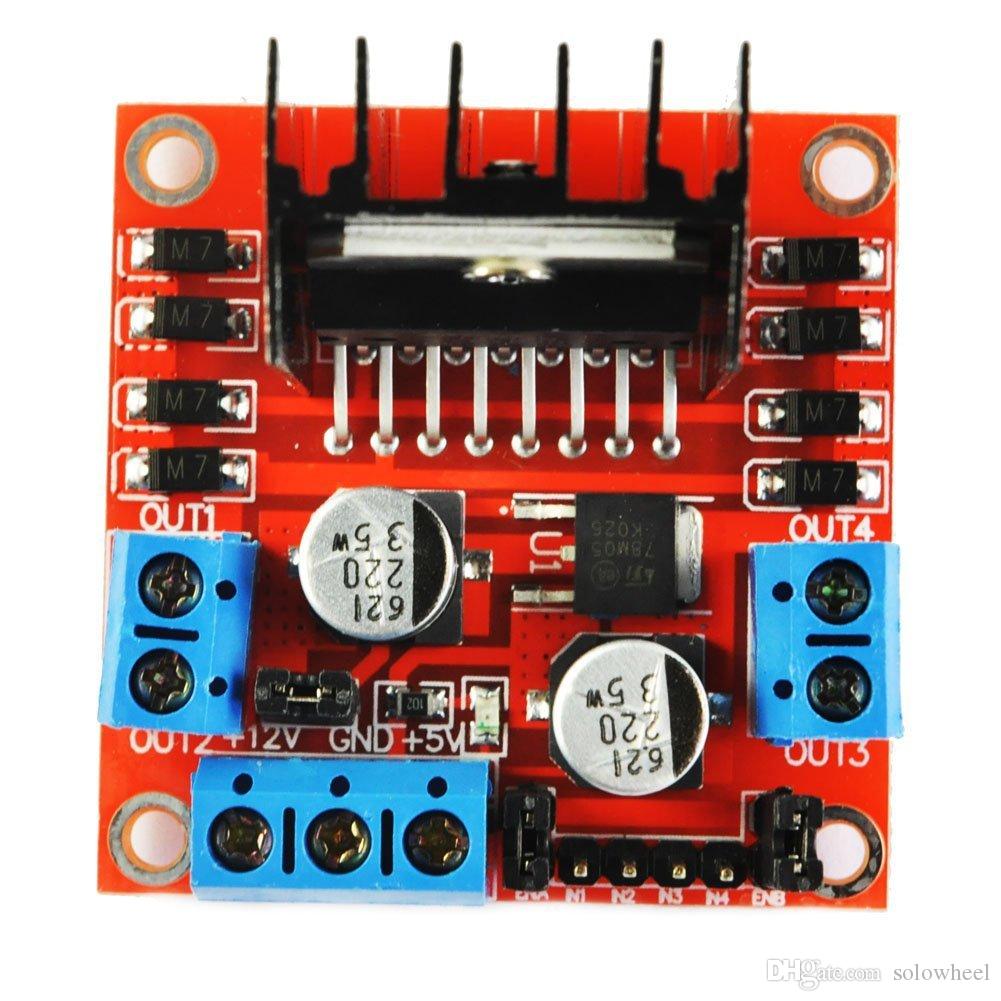
**2.1. 小车底盘**

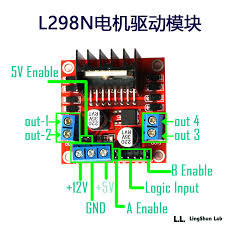
小车底盘是机器人最重要的载体，相当于人体的驱干，智能小车平台采用差速转弯非常灵活可以实现原地打转。小车平台大小刚好，可以承载一些如驱动器，控制器，电池，传感器等。



**2.2．驱动器**

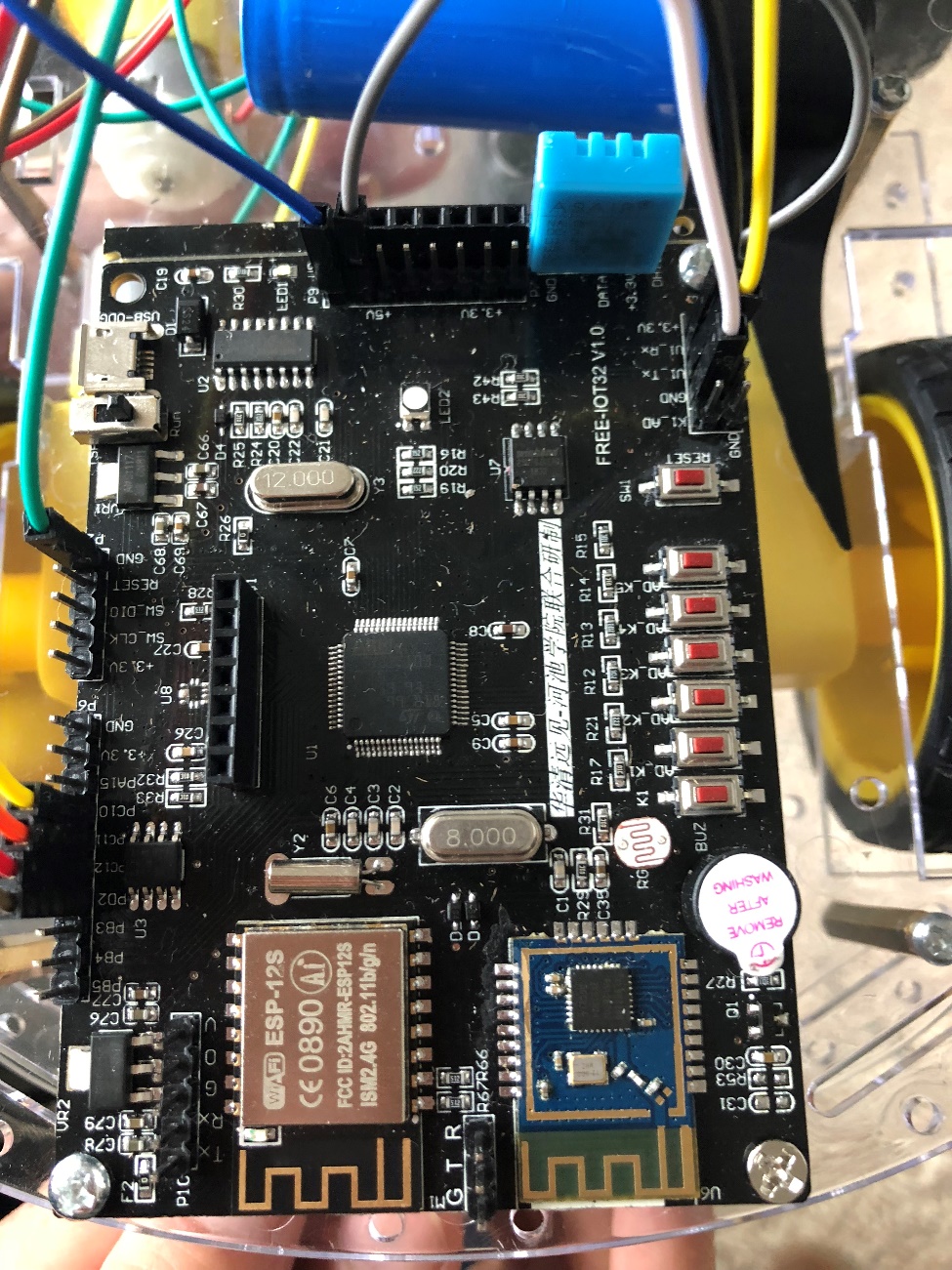
小车直流电机工作电流一般是200-400MA 有些更大，如果一个小车是二个轮子，那么总的电流在400-800MA 左右，这些电机轮子都是要接受单片机指令执行相应的动作，而市面有的单片机IO 口一般只能提供5MA 到10MA 的电流，直接驱动不了电机，所以需要搞一个驱动模块。放大电流。驱动采用专业的L298N驱动模块





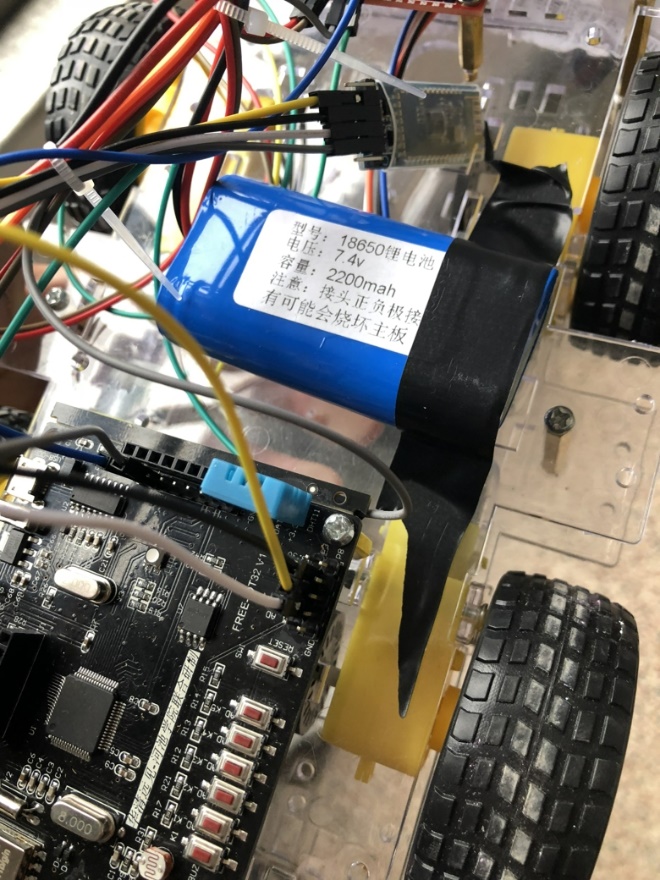
**2.3.控制器**

STM32单片机控制器：这款控制器主控芯片是STM32F103ZET6属于入门型的单片机，内带PWM，AD ，1T 运行速度（主流），集成USB转串可以直接用USB 下载程序。集成WIFI，蜂鸣器，晶振，外避障，12864，1602，无线模块等接口，板载输入按键。控制板休积小，非常适用于小车。在不用的时候可作用学习板使用。



**2.4.小车所需的能源**

可以用普通的AA5 号电池，但一般采用低内阻的充电电池，充电电池套装。



**2.5.蓝牙无线传感器**

可以用Android 智能手机控制小车运行（我们有配套的Android 软件），Android 手机用内置蓝牙与蓝牙无线模块配对，发出指令，蓝牙无线模块接收其指令。当单片机接收到蓝牙无线模块传来的指令，执行相应解码动作，从而控制小车。



1. **系统软件设计**

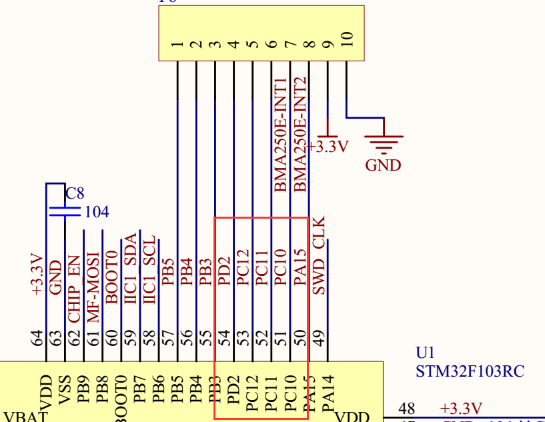
**3.1.电机驱动部分代码**

电机的驱动是通过L298N来实现的，L298N上IN1和IN2控制左边的电机转动，IN3和IN4控制右边的电机转动，只要单片机给L298N上的IN1~IN4高低电平就能驱动电机转动。

所以我们先要使能单片机上的GPIO口，使其和L298N的IN1~IN4端口相通，在这里我使能了stm32单片机上的PC10，PC11，PC12，PD2端口，且依次接到了L298N上的IN1~IN4端口上。通过下面的电机转动表编写了电机驱动代码



初始化GPIO电路引脚图：



电机驱动头文件motor.h ：

#ifndef \_MOTOR\_H

#define \_MOTOR\_H

void MotorInit(void);

void Turnleft(void);

void Turnright(void)

void Turnback(void);

void Turnfront(void);

void Stop(void);

void Leftaround(void);

void Rightaround(void);

#endif

电机驱动代码 motor.c ：

#include "STM32F10X.h" //官方库

#include "motor.h"

void MotorInit(void)

{

//初始化和使能单片机上的GPIO端口，保证可以正常给端口高低电平

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOC , ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10 | GPIO\_Pin\_11 | GPIO\_Pin\_12 | GPIO\_Pin\_6 | GPIO\_Pin\_7 | GPIO\_Pin\_8;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOC, GPIO\_Pin\_10 | GPIO\_Pin\_11 | GPIO\_Pin\_12 | GPIO\_Pin\_6);

// MOTOR\_APB\_PERIPH\_FUNG(MOTOr\_APB\_PORT , ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOD , ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2;

GPIO\_Init(GPIOD, &GPIO\_InitStructure);

GPIO\_SetBits(GPIOD, GPIO\_Pin\_2);

}

//停止模式

void Stop(void)

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_2);

}

//左转模式 -> 左电机正转，右电机反转

void Turnleft(void)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_SetBits(GPIOD,GPIO\_Pin\_2);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_6);

}

//右转模式 -> 左电机反转，右电机正转

void Turnright(void)

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_2);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_6);

}

//后退模式 -> 左电机反转，右电机反转

void Turnback(void)

{

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_SetBits(GPIOD,GPIO\_Pin\_2);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_8);

}

//前进模式 -> 左电机正转，右电机正转

void Turnfront(void)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_2);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_7);

}

//以左电机为中心原地左转

void Leftaround(void)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_11);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_SetBits(GPIOD,GPIO\_Pin\_2);

}

//以右电机为中心原地右转

void Rightaround(void)

{

GPIO\_SetBits(GPIOC,GPIO\_Pin\_10);

GPIO\_SetBits(GPIOC,GPIO\_Pin\_11);

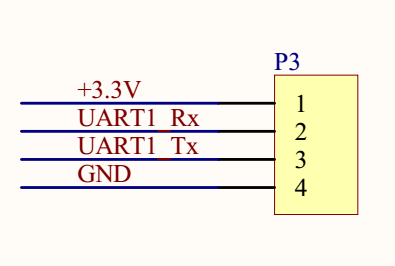
GPIO\_SetBits(GPIOC,GPIO\_Pin\_12);

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_2);

}

**3.2.蓝牙驱动部分代码**

蓝牙模块因为要和手机进行收发，所以需要用到中断，需要有中断的相关配置，以及中断处理函数。本来是需要把中断相关函数写在一个单独的函数中的，为了简单起见我把中断配置和中断服务函数全都写在了main函数中，方便查看。还有蓝牙模块和单片机连接的时候一定要注意的是蓝牙模块的Tx必须和单片机上的Rx相连，Rx和Tx相连。



代码部分：

#include "MOTOR.h"

#include "stm32f10x.h" //官方库

#include "stm32f10x\_gpio.h"

#include "stm32f10x\_rcc.h"

#include "stm32f10x\_it.h"

#include <stdio.h>

/\*

#ifdef->endif的内容部分和PUTCHAR\_PROTOTYPE{}的内容属于C语言的语法规则引入，

所以头部必须引入#include<stdio.h>,这样子的话C语言的语法就可以完美引用

\*/

#ifdef \_\_GNUC\_\_

/\* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf

set to 'Yes') calls \_\_io\_putchar() \*/

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif /\* \_\_GNUC\_\_ \*/

//中断服务函数

u8 i;

u8 flag; //定义一个标志位

void USART1\_IRQHandler(void)

{

if(USART\_GetITStatus(USART1, USART\_IT\_RXNE) == SET) //检查指定的 USART1 中断发生与否

{

USART\_ClearITPendingBit(USART1, USART\_IT\_RXNE); //清除 USART1 的中断待处理位

GPIO\_ResetBits(GPIOC,GPIO\_Pin\_6); //设置LED灯作为中断发生与否的指示灯

i= USART\_ReceiveData(USART1); //返回 USART1 最近接收到的数据

//这里if语句的内容部分可以保证蓝牙接收到的flag数据是正确的，如果不加的话会导致接收不到正确的数据

if(i=='0')

{

flag=0;

}

if(i=='1')

{

flag=1;

}

if(i=='2')

{

flag=2;

}

if(i=='3')

{

flag=3;

}

if(i=='4')

{

flag=4;

}

if(i=='5')

{

flag=5;

}

if(i=='6')

{

flag=6;

}

}

// USART\_ClearITPendingBit(USART1, USART\_IT\_RXNE);

}

//main函数部分

int main(void)

{

//利用中断进行蓝牙的收发

GPIO\_InitTypeDef GPIO\_InitStructure;

USART\_InitTypeDef USART\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1|RCC\_APB2Periph\_GPIOA, ENABLE); //使能USART1，GPIOA时钟

// USART\_DeInit(USART1); //复位串口1

// USART1\_TX PA.9

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9; //PA.9

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //初始化PA9

//USART1\_RX PA.10

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;//浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //初始化PA10

/\* USARTx configured as follow:

- BaudRate = 9600 baud 波特率

- Word Length = 8 Bits 数据长度

- One Stop Bit 停止位

- No parity 校验方式

- Hardware flow control disabled (RTS and CTS signals) 硬件控制流

- Receive and transmit enabled 使能发送和接收

\*/

USART\_InitStructure.USART\_BaudRate = 9600;

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

USART\_Init(USART1, &USART\_InitStructure);

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//开启中断

//Usart1 NVIC 配置

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 3; //子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

USART\_Cmd(USART1, ENABLE); //使能串口

MotorInit();

//蓝牙数据接收判断函数

while(1)

{

switch(flag)

{

case 0: Stop(); break ;

case 1: Turnleft(); break ;

case 2: Turnright(); break ;

case 3: Turnback(); break ;

case 4: Turnfront(); break ;

case 5: Leftaround(); break ;

case 6: Rightaround(); break ;

default: Stop(); break ;

}

}

}

PUTCHAR\_PROTOTYPE

{

/\* Place your implementation of fputc here \*/

/\* e.g. write a character to the USART \*/

USART\_SendData(USART1, (uint8\_t) ch);

/\* 循环等待直到发送结束\*/

while (USART\_GetFlagStatus(USART1, USART\_FLAG\_TC) == RESET)

{}

return ch;

}

**第四章、心得与体会**

为期一周的stm32培训已经结束了，此次实训我做的题目是蓝牙智能小车。简单地说就是通过STM32微控制器通过蓝牙控制小车运动。

实现这个功能主要是通过编写单片机程序然后传送到STM32芯片中，控制电机的运动。所以此次实训的主要目的就是编写程序然后调试程序。

刚开始的时候，老师就带着我们熟悉编程序的软件，并且告诉我们一定要养成良好的代码风格，并且让我们熟悉掌握外设的初始化和设置，这个是编写程序的基础吧，如果不按照外设初始化和设置的步骤来编写程序，程序就无法正常运行，就算运行了也无法实现正确的功能，所以对于我这个初学者来说，这是很好的经历，可以在调试程序或者是修改程序的时候少走弯路。老师给我们讲解了流水灯，中断，串口等等程序之后，我对编写程序也比较熟练了，然后就是自己把程序下载到单片机上，驱动小车车轮前后转动，根据小车出现的问题去改变参数，调试程序，最后让小车可以稳定的运行。

此次课设基本要求实现了，并且锻炼了自己的动手能力，养成了良好的代码风格，对于STM32也有了一定的了解，可以编写一些简单的程序了，对于我来说这是很大的一个突破。在编写程序时，也学会了看参考手册，以及查找相应参数的方法。这次实训学到了不少东西，这和我自己的努力有关，这个更离不开老师们的细心指导和教学。

**第五章、参考资料**

1.《STM32库开发实战指南》 刘火良 杨森编著，机械工业出版社

2.《STM32快速上手》 陈志旺，机械工业出版社

3.《STM32F10xxx参考手册中文版》