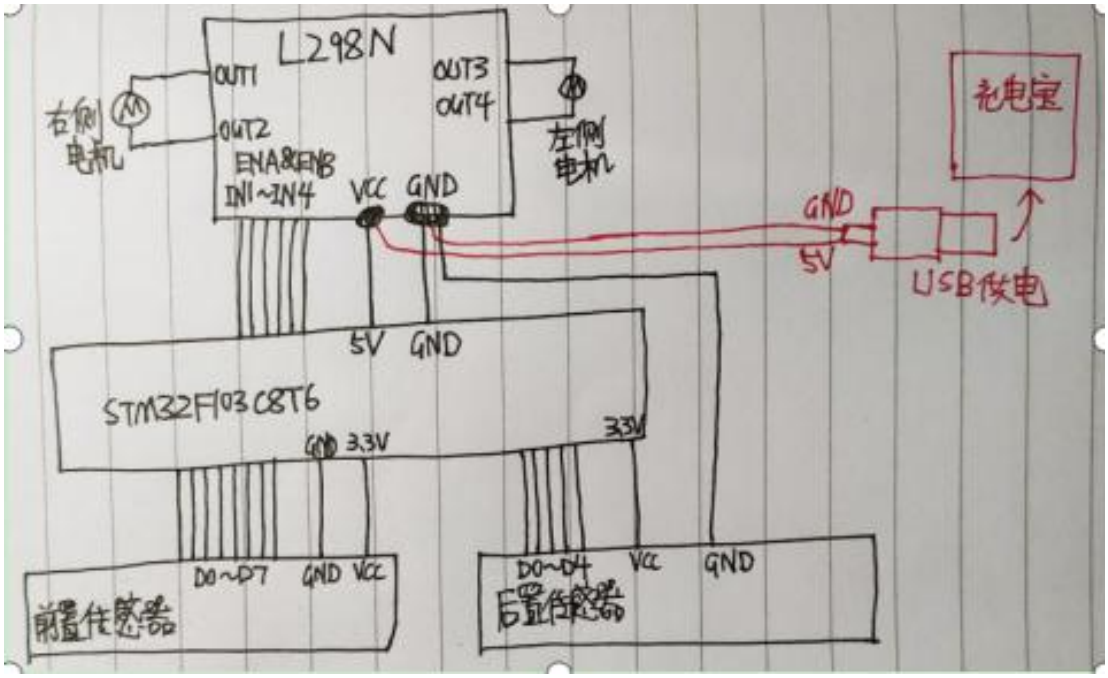
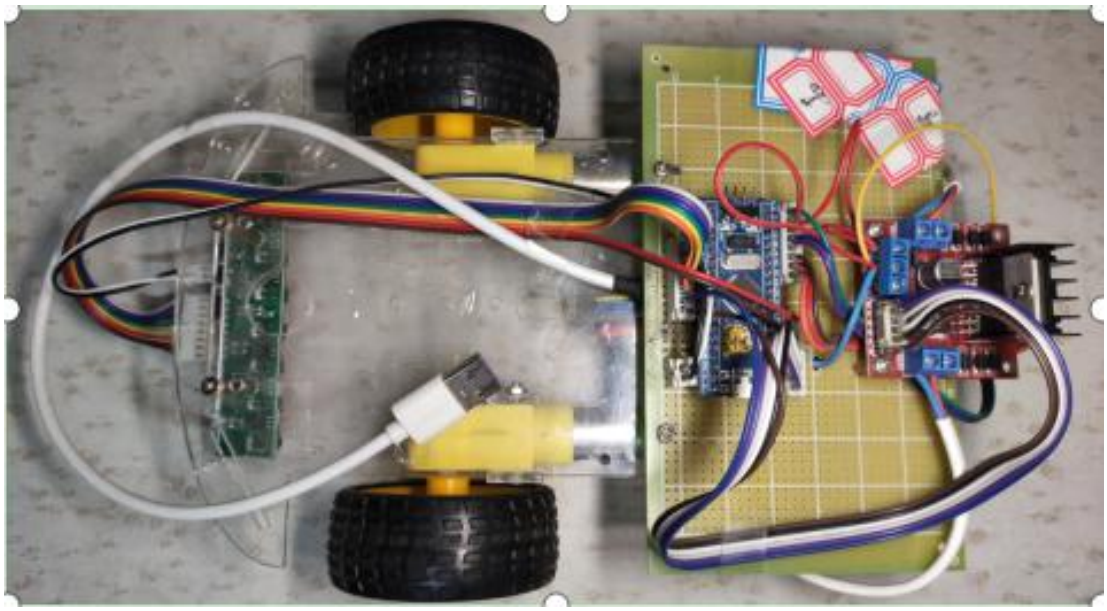
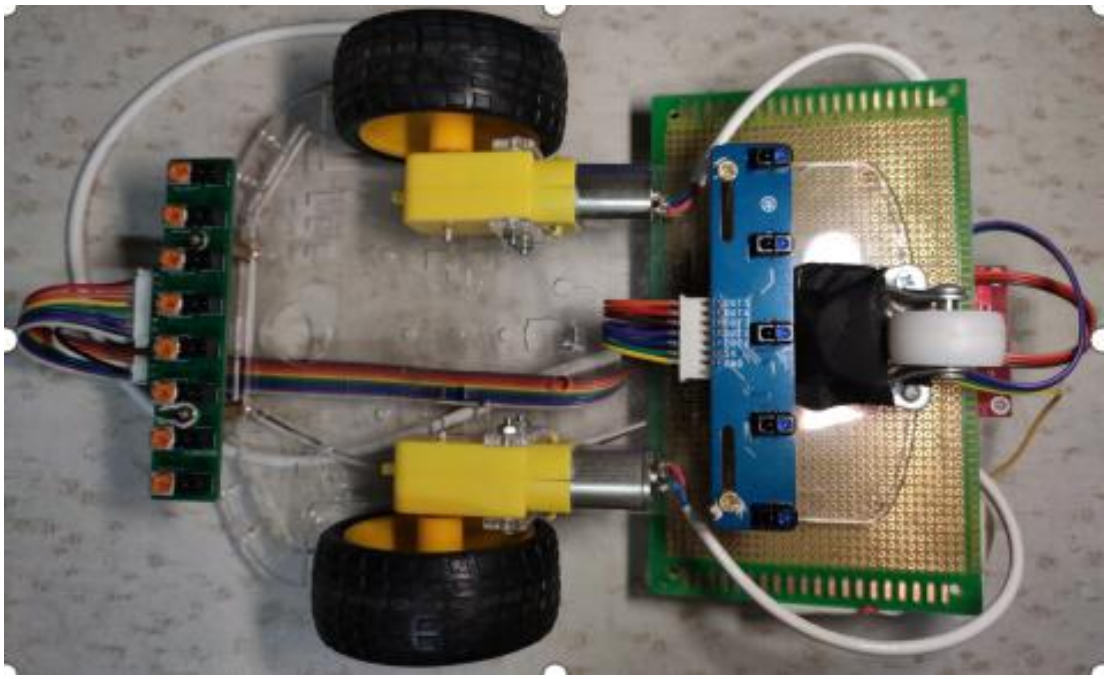
目录

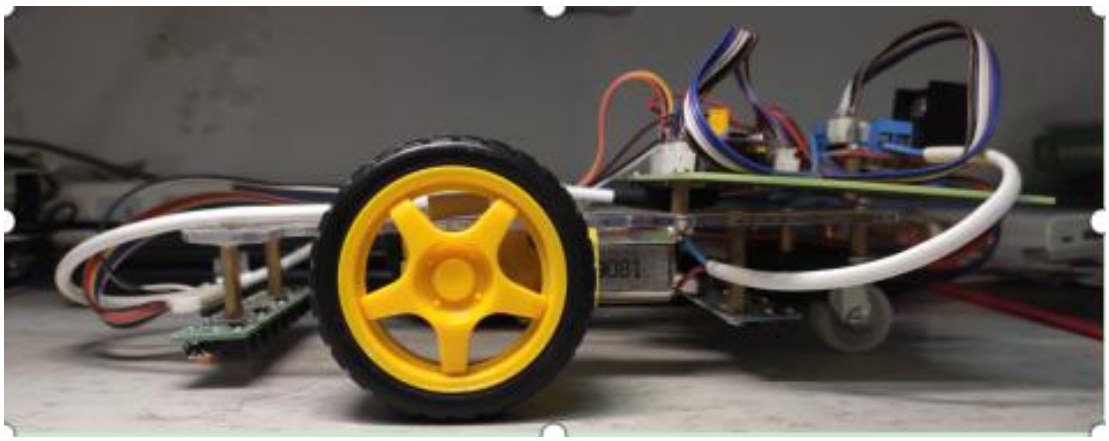
1. 小车介绍



上图是本小车的电路图，并不复杂。核心采用 STM32F103C8T6，电机驱动采用 L298N，充电宝供电，传感器在接收不到反射回来的红外信号时会将相应引脚置高电平，这种情况可以是小车悬空或传感器下面是黑线。







上面的三张图片分别是从小车顶端、底端、前端拍摄的小车图片。

在电脑上运行 CMD，通过 CD 命令进入想要存放程序的路径后，使用 git clone git@github.com:lu1198373615/myTrackingCar.git 就可以将程序连同 git 一起克隆到本地了，程序在 Keil_v5 环境下编译并下载至 STM32F103C8T6 上面，下载方式不唯一。

2. 程序讲解

我知道程序对于读者来说并不复杂，但本人代码习惯很差，因此还是应该对程序做一定的介绍。

stm32f10x_it.c 中设置了 TIM1 的定时器中断，中断每毫秒发生一次，中断时会对跨文件的全局变量 FrontSig 做出更新，同时决定是否对跨文件的全局变量

T_Flag 和 L_Flag 置位处理，这两个 FLAG 用于 mode=jiqiao 时判断左转和右转。

```
void  ADVANCE_TIM_IRQHandler (void)
{
  static uint8_t FrontSig_buf = 0;
  if ( TIM_GetITStatus( ADVANCE_TIM, TIM_IT_Update) != RESET )
  {
    FrontSig=ReadFrontSig();
    //BackSig=ReadBackSig();后置传感器后来没用到
    TIM_ClearITPendingBit(ADVANCE_TIM , TIM_FLAG_Update);
  }

  FrontSig_zero = FrontSig;
  //
  if((FrontSig&240)==240) T_Flag = 1;//小车左边发现直角
  if((FrontSig&15) == 15) L_Flag = 1;//小车右边发现直角

  if(FrontSig!=0) FrontSig_buf = FrontSig;
  if(FrontSig==0) FrontSig = FrontSig_buf;
}
```

main.c 中宏定义了 jingsu,jiqiao,mode 三个变量，main 函数中将 mode 值作为了 if 判断的条件，决定在循线前进时是否对直角的标记做出处理。

```
#define  jingsu 0
#define  jiqiao 1
#define  mode jiqiao
```

mode 值为 jingsu 时，不会对直角信号做出处理，mode 值为 jiqiao 时会对直角信号做出处理，因此只需修改宏定义值就可以决定小车工作的模式。

在检测到直角信号时，会将相应的 FLAG 置 1，此时主函数会进入处理这些情况的分支函数。此时小车会由于惯性继续向前走一段距离，因此应该让小车后退至直角线前，启动硬件定时器控制的 PWM 波并向左/右转至需要去的路上，然后关掉硬件 PWM 波定时器，继续采用软件控制 PWM 波实现转弯。软件 PWM 指通过对 GPIO 置 0/1 来实现 PWM 波，硬件 PWM 波指定时器自动产生 PWM 波。

main.c 只添加一个头文件：bspmethod.h

```

#include "bspmethod.h"
```

这个头文件中定义有 12 函数，如下图所示

```
//下面的5个函数是通过对L298方向输入引脚置位实现
void allStop(void);//俩轮都停
void allGoFront(void);//俩轮都向前
void allGoBack(void);//俩轮都向后
void goLeftFront(void);//向左前方走，左轮听，右轮向前
void goRightFront(void);//向右前方走，右轮停，左轮向前
//下面的两个函数也是通过对L298方向输入引脚置位实现
void volveLeft(int aaa);//右轮停止，左轮后退，达到向左旋转的效果，用于对方向矫正过度时的刹车
void volveRight(int aaa);//左轮停止，右轮后退，达到向右旋转的效果，用于对方向矫正过度时的刹车
//下面这个是延迟函数，但是在传感器信号改变时会被提前终止睡眠
void FlexibleDelay(__IO uint32_t ms,uint8_t ppp);
//下面的4个函数是通过对L298的PWM引脚置位实现，需要调用volveLeft和volveRight来刹车
void dTurnLeft(void);//左轮停止，右轮慢速，从而向左转
void dTurnRight(void);//右轮停止，左轮慢速，从而向右转
void xTurnLeft(void);//左轮减速，右轮全速，从而向左转
void xTurnRight(void);//右轮减速，左轮慢速，从而向右转
```

这个头文件中还引入了一些.h 头文件

```
1
2  #include "stm32f10x.h"
3  #include <stdio.h>
4
5  #include "bsp_key.h"
6  #include "bsp_led.h"
7
8  #include "bsp_SysTick.h"
9  #include "bsp_GeneralTim.h"
10 #include "bsp_AdvanceTim.h"
```

bsp_key.h 定义了定义了传感器输入的标准

```
57 void Key_GPIO_Config(void);//将所有宏定义了的传感器输入引脚设为浮空输入模式
58 uint8_t Key_Scan(GPIO_TypeDef* GPIOx,uint16_t GPIO_Pin);//检查特定传感器输入引脚的电平高低，返回0或1
59 uint8_t ReadFrontSig(void);//通过调用Key_Scan函数来检查前置传感器的8个引脚，并移位计算得到无符号8位整型数据
60 int32_t FrontIndex(uint8_t value);//通过查看8位整型字符变量FrontSig哪一位为1来确定小车位置
61 uint8_t ReadBackSig(void);//和ReadFrontSig类似，但是这个函数后来并没有用到
```

这个头文件还宏定义了 GPIO 引脚，因此想更换传感器输入引脚十分容易，为了节省篇幅，就不截图了。

bsp_led.h 中宏定义了引脚，方便我们改变 GPIO 与 L298N 的连接。在 42-72 行定义了 6 个简单的函数使我们能方便的操纵 L298N，76 行的函数则用于将与 L298N 连接的引脚初始化为推挽输出。

```
76 void LED_GPIO_Config(void);//将连接L298的6个引脚设为推挽输出
```

"bsp_SysTick.h"用于实现简单的延迟，"bsp_GeneralTim.h"用于技巧赛产生硬件 PWM 波（虽然竞速循迹时采用软件 PWM 波）。"bsp_AdvanceTim.h" 用于实现 1 毫秒 1 次的定时器中断。