Week4 study notes

超声测距模块

模块引脚

超声波模块有4个引脚，分别为Vcc、 Trig（控制端）、 Echo（接收端）、 GND；其中VCC、GND接上5V电源， Trig（控制端）控制发出的超声波信号，Echo（接收端）接收反射回来的超声波信号。模块如图：

控制原理

通过Trig引脚发一个 10US 以上的高电平,就可以在Echo接收口等待高电平输出；一有输出就可以开定时器计时,当此口变为低电平时就可以读定时器的值,此时就为此次测距的时间,方可算出距离.如此不断的周期测,就可以达到你移动测量的值了。

工作流程

a.单片机引脚触发Trig测距，给至少 10us 的高电平信号;
b.模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回；
c.有信号返回，通过 IO 输出一高电平，并单片机定时器计算高电平持续的时间;
d.超声波从发射到返回的时间.
计算公式：测试距离=(高电平时间*声速(340M/S))/2;

在学习之后，我将代码改编了一下加入了第三周小车控制系统之中。我首先在第三周小车的右侧加装了超声波测距仪HC-HR04，然后在主控板上加入了如下代码

```
boolean AIdrive;

void loop(){
if(ps2x.ButtonPressed(PSB_TRIANGLE)){//按下三角后转变AIdrive的状态
      AIdrive = !AIdrive;
     }

if(!AIdrive){  //非AI状态下手动控制
     read(); //电机驱动
     if(ps2x.ButtonPressed(PSB_CIRCLE) && turnOnMusic ) {  //音乐驱动
      musicOpen();
      turnOnMusic = !turnOnMusic;
     }
     if(ps2x.ButtonPressed(PSB_CIRCLE) && !turnOnMusic) {
```

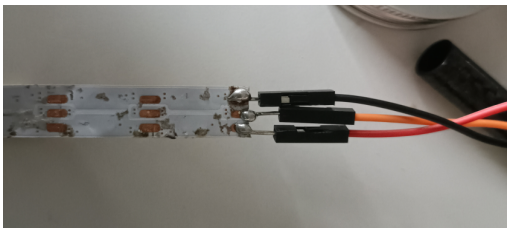
```
       musicClose();
       turnOnMusic = !turnOnMusic;
      }
      turn();  //手动转向
      }
      if(AIdrive){ //AI状态下自动控制
       read();
       AIturn();
      }
}


void AIturn(){
 digitalWrite(trig,LOW);
 delayMicroseconds(20);
 digitalWrite(trig,HIGH);
 delayMicroseconds(20);
 digitalWrite(trig,LOW);                      //发一个20US的高脉冲去触发Trig
 distance  = pulseIn(echo,HIGH);              //计数接收高电平时间
 distance  = distance*340/2/10000;             //计算距离 1：声速：340M/S  2：实际距离1/2声速距离 3：计数时钟为1US
 Serial.print("distance:");
 Serial.println(distance);
 if(PS2_LY <= 126){  //前进状态
 if(distance > 16){  //自动右转
  myservo.write(70);
 }
 else if(distance < 6){
  myservo.write(20);  //自动左转
 }
 else if(6 <= distance <= 16){
  myservo.write(45);  //不转
 }
 delay(20);
  }
 else if(PS2_LY >= 130)  {  //后退状态
   if(distance > 16){  //左转倒车
  myservo.write(20);
 }
 else if(distance < 6){  //右转倒车
  myservo.write(70);
 }
 else if(6 <= distance <= 16){  //直线倒车
  myservo.write(45);
 }
 delay(20);
 }
}
```

LED灯带显示当前操作模式

这周我还学习了LED灯带有关的代码与接线方式

我从垃圾堆中翻出了长六十多节的WS2812B LED灯带，将其剪短至32节，焊好线后开始了测试。



LED灯带有三个接口，vcc、din和、gnd，其中din是用于接信号端。

通过相关学习以及询问老师，当灯带长度大于10时，最好不要用arduino板供电，以免因电流过大导致板子损坏，所以我的灯带使用电池或充电宝供电。

调控代码

由于灯带内部调控从底层入手较难，且我们也没有自制灯带样式的需求，因此选择使用FastLED库来控制LED灯带。

FastLED库是专门控制LED灯带的一个非官方库，由于其强大的功能与简单的操作收到大家的好评。

只需在程序中调用

```
#include <FastLED.h>                          //LED灯带设置
#define LED_PIN 1
#define NUM_LEDS 26
CRGB leds[NUM_LEDS];
```

即可。

以下是网上原代码，可以实现LED流水灯切换颜色

```
#include <FastLED.h>
#define LED_PIN 1
#define NUM_LEDS 26
CRGB leds[NUM_LEDS];

void setup() {
FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
FastLED.setMaxPowerInVoltsAndMilliamps(5, 500);
FastLED.clear();
FastLED.show();
}

void loop() {
 //RED Green Blue
for (int i=0; i<NUM_LEDS; i++ )
{
leds[i] = CRGB(0, 255-2*i, 10*i );
FastLED.setBrightness(6*i);
FastLED.show();
delay (100);
```

```
}

for (int i=NUM_LEDS; i> 0; i-- )
{
leds[i] = CRGB(10*i, 0, 255-10*i );
FastLED.setBrightness(60-2*i);
FastLED.show();
delay (100);
}
}
```

效果如上

于是，我将代码修改了一下，准备变成方法加入到主控板的loop里，其中的两种方法分别对应了主动驾驶和辅助驾驶模式的LED状态。

```
void LED_1(){
  for(int i=0; i<NUM_LEDS; i++){
  leds[i] = CRGB(10*i, 0, 255-10*i );
  FastLED.setBrightness(5*i);
  FastLED.show();
  delay(50);
}
for(int i=0; i<NUM_LEDS; i++){
  leds[i] = CRGB(0,0,0);
  FastLED.setBrightness(5*i);
  FastLED.show();
  delay(50);
}
}

void LED_2() {
 //RED Green Blue
for (int i=0; i<NUM_LEDS; i++ )
{
leds[i] = CRGB(0, 255-2*i, 10*i );
FastLED.setBrightness(6*i);
FastLED.show();
delay (100);
}
```

```
for (int i=NUM_LEDS; i> 0; i-- )
{
leds[i] = CRGB(10*i, 0, 255-10*i );
FastLED.setBrightness(60-2*i);
FastLED.show();
delay (100);
}
```

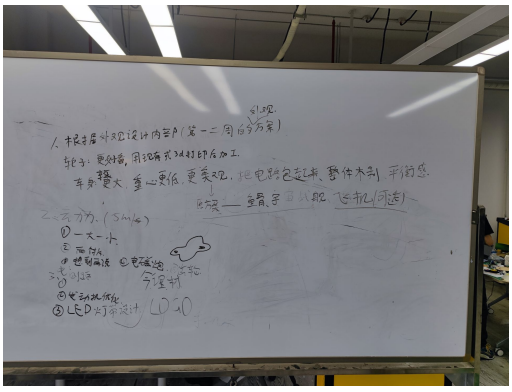
虽然看起来很完美无缺，但当我烧录到arduino板后，发现舵机变得不太对劲，具体提现是没过几秒他就会自动转一下，而且不受控制。在搜索原因后，我发现可能是因为delay函数的频繁使用导致出现了看门狗现象。

看门狗是一个定时器电路, 一般有一个输入,叫喂狗,一个输出到MCU的RST端,MCU正常工作的时候,每隔一段时间输出一个信号到喂狗端,给 WDT 清零,如果超过规定的时间不喂狗,(一般在程序跑飞时),WDT 定时超过,就回给出一个复位信号到MCU,使MCU复位。防止MCU死机，看门狗的作用就是防止程序发生死循环，或者说程序跑飞。

在思考过后，我认为一块Arduino开发板不足以同时驱动电机舵机同时驱动LED，因此，我决定再买一块板子Arduino nano板（比较小）当做副控制板，专门控制LED灯。目前板子还没到，因此下周我将把这个计划实现。

电磁炮发射功能的学习

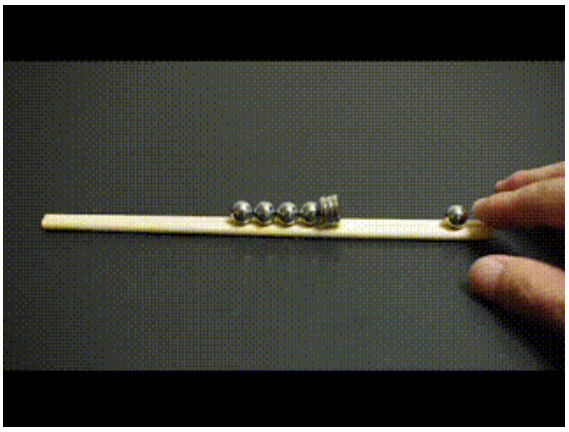
在本周开始时的会议上，我们讨论了关于小车到达5m/s的方案，其中我们特地讨论了一下电磁炮或者高斯炮的可能性。



在接下来一天中，我学习了许多有关电磁炮的相关知识，并对可行性进行了计算

已知B＝μ0nI，其中n是匝密度，μ0=4π×10-7，假设螺线管长10cm，电流1A，那么1T需要80000圈（每跟管子），然后管子上的线圈要闭合，车子上需要一个长10cm通2A电流的导线，产生F=BIL=0.2N。这点力太小了，而我们需要做的努力却太多了，因此，在计算后我们放弃了电磁炮这个方案。

关于高斯炮，我们认为这个方案可行



但从原理上他需要外力援助或者在跑道上丢杂物（否则就是内力了，无法提供动力），因此也被pass掉了。最终我们决定用机械结构里的弹簧来发射小车达到5m/s。

虽然学习的电磁炮相关的知识暂时看来没什么用，但我相信日积月累下来的“无用”知识终究会在不知道某天帮助到我。

使用micro()函数实现Arduino板的多线程研究

周日下午，我们小组在位置上学习。我网上搜索到可以用micro函数替代delay函数的方法实现多线程，并且突发奇想能不能用这个方法不用第二块板做到同时驱动音乐和LED灯呢。于是，我花了一个下午来研究。

millis（）使用方法：millis函数是在Arduino上电后会持续记录开机时间的函数，使用时仅需

```
unsigned long currentTime=millis();
```

可以通过确定一个周期，然后记录开始时间，在到达周期之后执行程序，然后将执行程序的时间记录，达到连续delay函数的效果，代码如下：

```
const int ledPin=13;
int ledState=LOW;
long previousTime=0;
long interval= 2000;

void setup(){
  pinMode(ledPin,OUTPUT);
}

void loop(){
  unsigned long currentTime=millis();
  if(currentTime - previousTime > interval){
  previousTime=currentTime;
  if (ledstate=LOW)
  ledState= HIGH;
  else
  LedState = LOW;
```

```
  digitalWrite(ledPin, ledstate);
  }
}
```

根据这个方法，我实现了超声测距功能的优化：

```
void AIturn(){
  unsigned long currentTime = micros();
  if(currentTime - previousTime > 20000){
//里面是一些方法
previousTime = currentTime;
//每过20000us执行一次
}
```

但在调控LED运行的时候，出现了一些问题。

LED灯带是由leds[]数组控制的，通过对其中每一个进行参数（亮度、颜色等）确定完成的，因此需要用到for循环。但在for循环下，如果使用if方法，还没有等到相应的时间，程序就会continue到下一个，以至于一下子运行完。然后我考虑能不能用switch或者if来做，但是三十多个灯需要三十多次判断，会非常阻碍程序的运行，而且工作量很大，因此放弃了这个方案。**后续我会再考虑一下关于能否用for循环来完成，但目前我的能力好像想不出来。**

其他事项

本周末我们小组进行了一次还不错的答辩，在答辩前大家提前演练相互磨合，让我懂得了团队协作的真谛。周四小组全体加班到11点，也体现出Sdimer的活力与热情。相比于上周方案考虑的简单，这周更加注重方案的实用性和可实现性，争取一遍完成原定计划，提前想好会出现的问题。相比于上周的小车，这周小车的许多地方都翻新提升，因此需要耗费更多的时间。我想，这可能就是迭代的意义吧。