成绩

中国农业大学

课程设计

（2020 -2021 学年夏季学期）

题　　目： 基于 FPGA 的自动往返小车

课程名称： 电子技术综合设计

任课教师： 冯磊

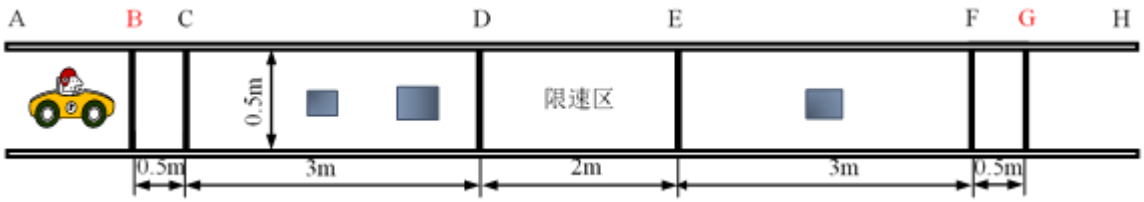
班　　级： 自动 191

学　　号： 2019308130215

姓　　名： 孟令昶

目录

一、课程任务



1. 设计内容

设计制作自动往返于起跑线与终点之间的电动小车。

2. 基本要求

1）小车从起始点 B 出发，到达终点 G 后停留 10s，之后自动返回。

2）路段 DE 为限速区，小车减速通过，车速≤10m/min。

3）行驶时间实时显示，停车后显示全程行驶时间。

4）力求往返时间最短，停车位置距离目标点偏差最小。

3. 提高部分

1）避障功能，前方有障碍时紧急刹车，并发出声光报警，直到障碍解除。

2）声控功能，可以通过声音控制小车的起停等动作。

3）小车行驶路程检测，实时显示车速或里程。11

4）温湿度检测功能，测量并显示环境温度和湿度。

5）自由发挥，其它创新或创意。

二、电路设计

1. 总体方案与子方案比较分析

1.1 总体方案设计

1.1.1 任务要求中的子模块设计

通过分析题目，小车在标有里程的直线赛道上往返，要求在不同区段内完成不同的动作，这首先需要确定小车所处的位置。经过查阅资料和分析题目，赛道上 B-G 各点位置可以由黑线条数来决定，我们的方案是利用红外传感器 TCRT5000 来检测黑线条数，发射的红外线在白色区被接收器接收，输出高电平，黑色区无

接受，输出低电平。在此输出的下降沿或上升沿计数加一，测试时只需要读取寄存器中黑线的计数值，便可对应做出相应的动作。而基础部分小车在不同赛段做出的动作由 Basys3 开发板编程控制两个直流电机来实现。

对于基本要求一，出发和返回动作只需要在黑线数分别为 1-6 和 7-12 时给 L298N 电机驱动模块的输入端输入高低电平 10、低高电平 01，L298N 的输出端 12、34 接到两直流电机的正负极，以实现电机正反转来实现，而 G 点停车 10s 则需要给电机两端输出同样的电平 11 或 00 使电机停止转动，并在程序中用“#10”形式控制延时（#后数字与 10s 的关系由时基 timescale 决定）。由于小车的动作比较复杂，持续改变输出 pwm 方波与否不合理，所以需要利用开关控制 pwm 方波的输出，并通过程序中的变量 pwm_ctrl 控制 pwm 开关的开闭，以实现小车的运动和停止。

对于基本要求二，则需要控制电机的转速，我们采用输出 PWM 方波的形式，利用改变方波占空比实现调速：当占空比为 80%~100%时，电机转速明显加快实现加速，占空比小于 50%时电机转速明显变慢实现减速。通过测量验证，占空比为 60%左右时，车速<=10m/min 通过限速区的实现效果比较好。

对于基本要求三，实时显示时间和停车后显示全程时间则需要通过编程利用开发板上的共阳极数码管实现。实时显示时间，需要利用计数器 t 在小车启动后循环加一直至小车停止。小车停止时计数器的值赋给另一个 reg 变量 t_stop，这便是全程时间，将两个时间的每一位数值用数码管扫频显示即实现了此要求。

对于基本要求四，往返时间最短需要通过调试，在保证前三个基本功能的基础上选择尽可能大的占空比；停车位置距离目标点偏差最小，需要保证两个轮子转速基本一致。由于两个电机本身的制造误差和质量问题，这需要调试两个轮子上输出 pwm 波的占空比，我们采用的是“设置的基础占空比加+调节量”的形式，调节量可以用拨码开关控制。

以上基本要求中，电机转速还受电池电量的影响，在每一次测试时，都要保证电池有充足的电量。

提高部分的电路则需要根据提供的元器件及其特性进行实际电路设计，在焊接前利用仿真软件 proteus 检测电路设计的合理性，并在面包板上调节电路参数利用示波器检测目标点波形。

对于避障功能，我们采用的是 TCT40-16T 超声波传感器。给发射端提供 40MHz 的方波激励，在遇到障碍时，接收端接收信号并产生一个正弦波响应，通过电压比较器 LM393 输出一个方波。将这个方波输入到开发板，利用上升沿循环加一计数的方式检测此方波的脉冲宽度，当方波宽度大于 5us 时判断有障碍，pwm_ctrl

信号控制 pwm 输出开关闭合，小车停止；同时，利用分频函数给蜂鸣器和发光二极管电路输出一个 1kHz 的方波，蜂鸣器报警，二极管闪烁。

对于声控功能，我们采用的是咪头、发光二极管和 LM393 电压比较器电路实现。此部分咪头检测到外界声音时会产生响应，发光二极管闪烁，经过电压比较器后产生一个方波输出，将这个方波输入到开发板，利用上升沿循环加一计数的方式检测此方波的脉冲宽度和达到一定宽度的次数，当方波宽度大于 1ms 时（排除外界短暂杂音干扰），利用 pwm_ctrl_sound 信号控制 pwm 输出，次数为奇数次时，pwm 输出打开，小车启动；次数为偶数次时，pwm 输出关闭，小车停止。此处的 pwm_ctrl_sound 与前面的电机控制 pwm_ctrl 通过逻辑控制总开关 pwm_ctrl_sum，只要有一个为 1，即可以控制 pwm 输出，两个都是 0 时，控制 pwm 无输出。

对于小车行驶路程检测功能，则利用槽型光耦 ITR9606 检测码盘上断连处，经过电压比较器 LM393 在输出端产生一个高低电平的方波信号。根据小车轮子直径 6.6cm 的尺寸计算，一个周期（一个高电平或低电平）对应于实际 1.037cm，因此只需要统计行驶过程中高电平的数目，即可得到行驶的里程。编写除法器程序，用里程除以时间，即可得到速度，并且在数码管上显示即可。

对于温湿度检测功能，我们利用 555 定时器接成多谐振荡器，把湿敏电阻或热敏电阻接入到充放电回路。当环境温湿度变化时，充放电回路的时间常数 τ 随之改变，而三号管脚输出端输出方波的占空比随之改变。通过量化占空比与阻值、阻值与实际温湿度的关系，即可通过占空比来反应实际温湿度。最后的统计量也通过数码管显示。

1.1.2 小车基础组成模块设计

小车基础组成模块包括小车拼装、焊接原件组装、电池选择和供电等。

其中，小车的拼装按照提供的安装教程即可，焊接原件按照仿真电路在洞洞板上焊接，电池使用的是实验室提供两节 3.5v 可充电锂电池组成的 7v 电源与 7805 稳压模块构成 5v 稳压电源供电。

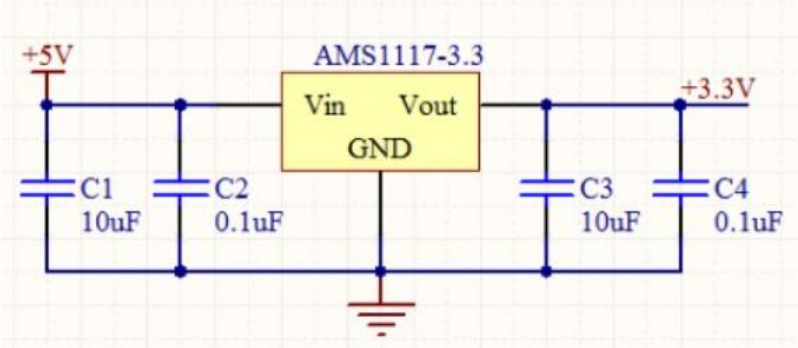
1.2 子模块方案比较分析

1.2.1 供电模块

实验室提供的元件：LM1117-3.3 和 LM7805

(1) 方案一：

利用 LM1117-3.3。电路图如下

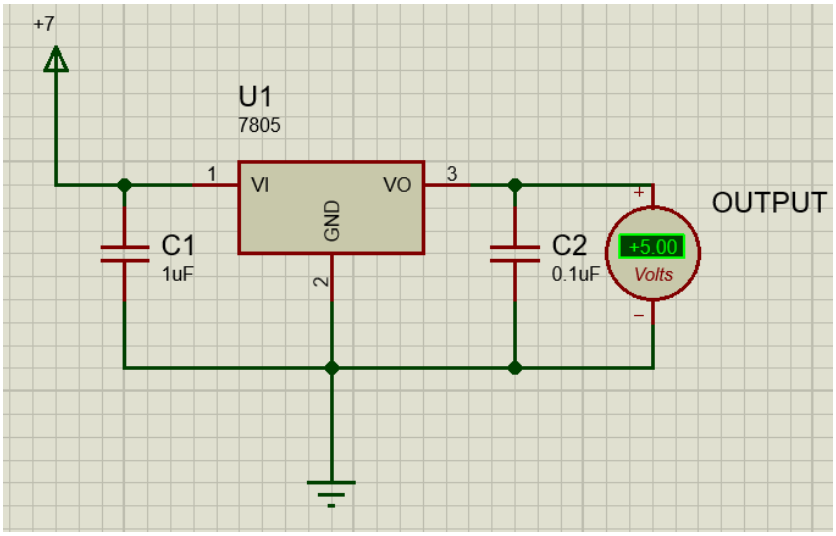


主要问题：

提供的芯片稳定压降在 3.3v 左右，不能提供足够的电压驱动后续模块。

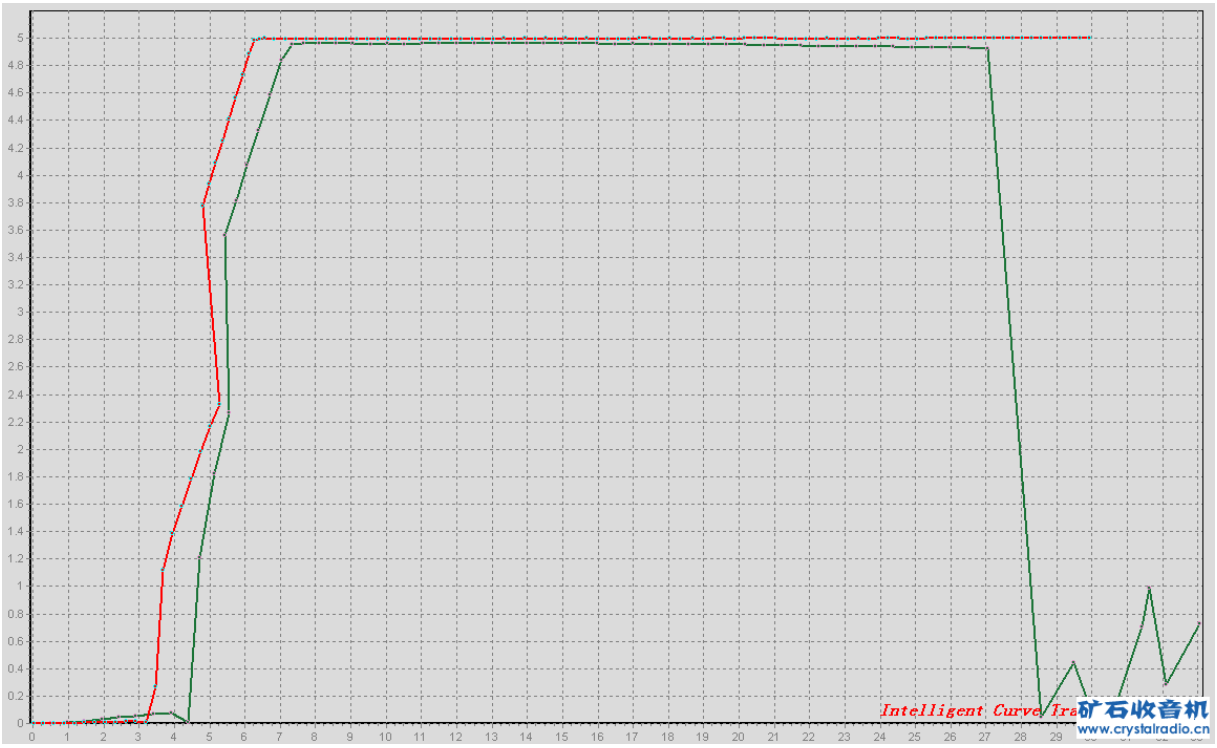
C1 和 C2 是输入电容，对于交流电压整流输入，它们的作用是把单向脉动电压转换成直流电压；而对于直流输入作用是防止断电后出现电压倒置、C3 和 C4 是输出滤波电容，作用是抑制自激振荡，如果不接这两个电容，通常线性稳压器的输出会是个振荡波形。

(2) 方案二：利用 LM7805；电路图如下



主要问题：

根据 LM7805 的输入输出特性曲线：

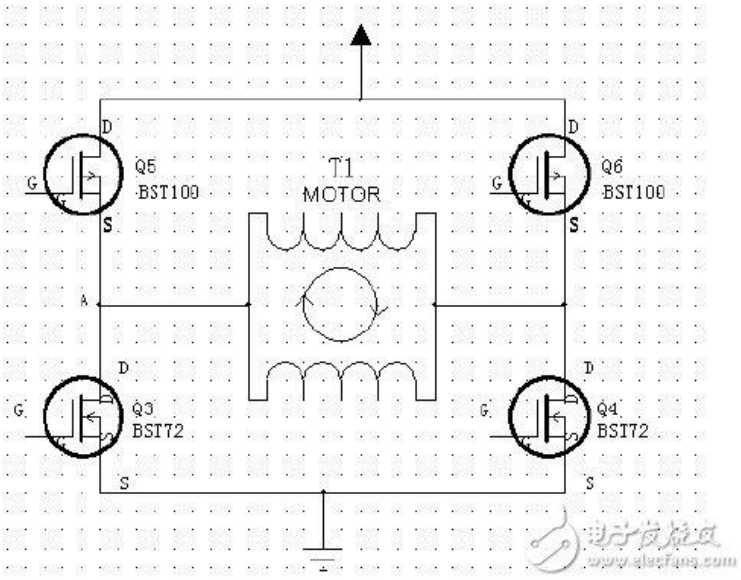


7805 的输入电压为 7~35v，如果电池电压太低将严重影响输出稳压值，如果输入电压过大，则芯片两端压差过大，功耗过大导致输出电压急速下降，引发芯片过功耗保护。[1]

1.2.2 电机驱动模块

实验室提供的芯片：L298N，PMOS，NMOS

(1) 方案一：利用 MOS 管构成 H 桥型驱动电路

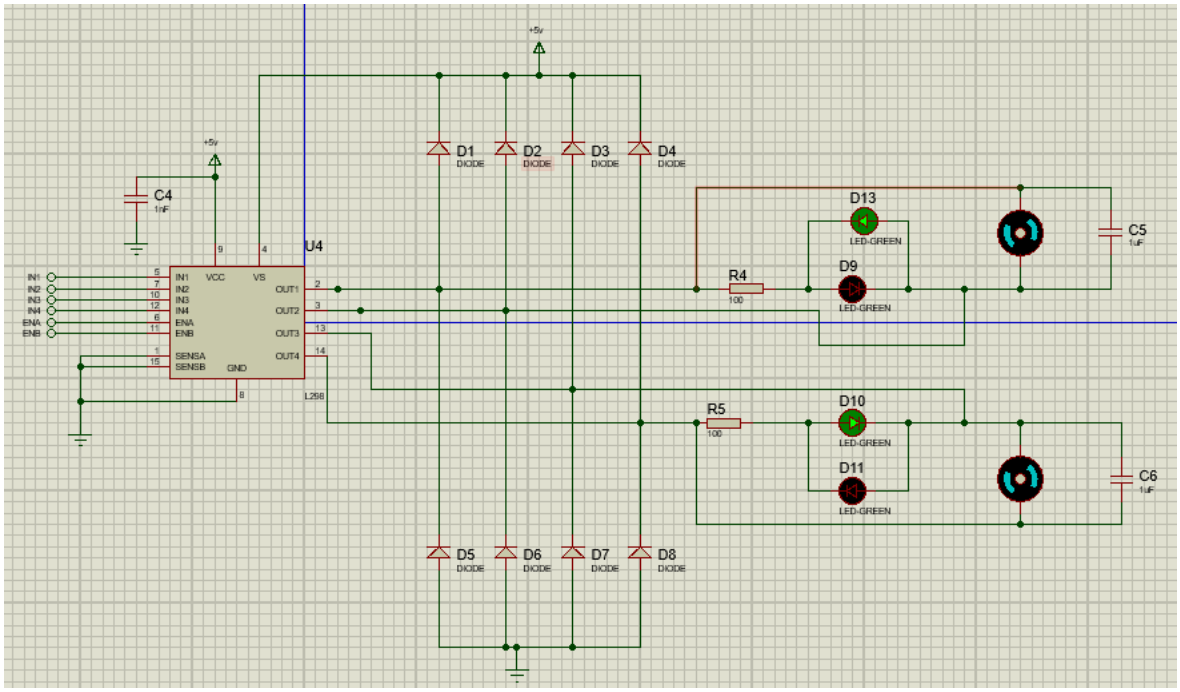


基本原理用四路开关去控制电机两个输入端的电压。当开关 S1 和 S4 闭合时，电流从电机左端流向电机的右端，电机沿一个方向旋转；当开关 S2 和 S3 闭合时，电流从电机右端流向电机左端，电机沿另一个方向旋转。

主要问题：

要使上管 NMOS 打开，必须使上管的 G 极相对于浮地有 10-15V 的电压差，实验室已有电源无法满足此电压值。而且电源受到 MOS 管开关的影响，产生了极大的波动，导致电源产生振铃现象。从而导致了功率管 DS,GS 间的电压时常超过耐压值，导致上下臂直通现象的出现。电源处必须外接 10uF 以上的电容，去除电源干扰。不同的 MOS 管，需要的滤波电容容值也会相应的改变。

(2) 方案二：利用 L298N 模块实现电机驱动



根据 IN 接口的电平逻辑关系（如下表），实现电机正反转控制。D1~D8 二极管的作用是续流和吸收反峰压作用，防止电机线圈在运转过程中两边产生反电势对 L298 形成的冲击。四个发光二极管可以反应电机正反转的方向。

ENA	IN1	IN2	直流电机状态（OUT1和OUT2）
0	任意	任意	停止
1	0	0	刹车
1	0	1	正转
1	1	0	反转
1	1	1	刹车

主要问题：

如果开发板和 L298 的电源不是“共地”的话，在实际调试的过程中可能会遇到接线“正确”、程序正确，但是电机不转的情况。

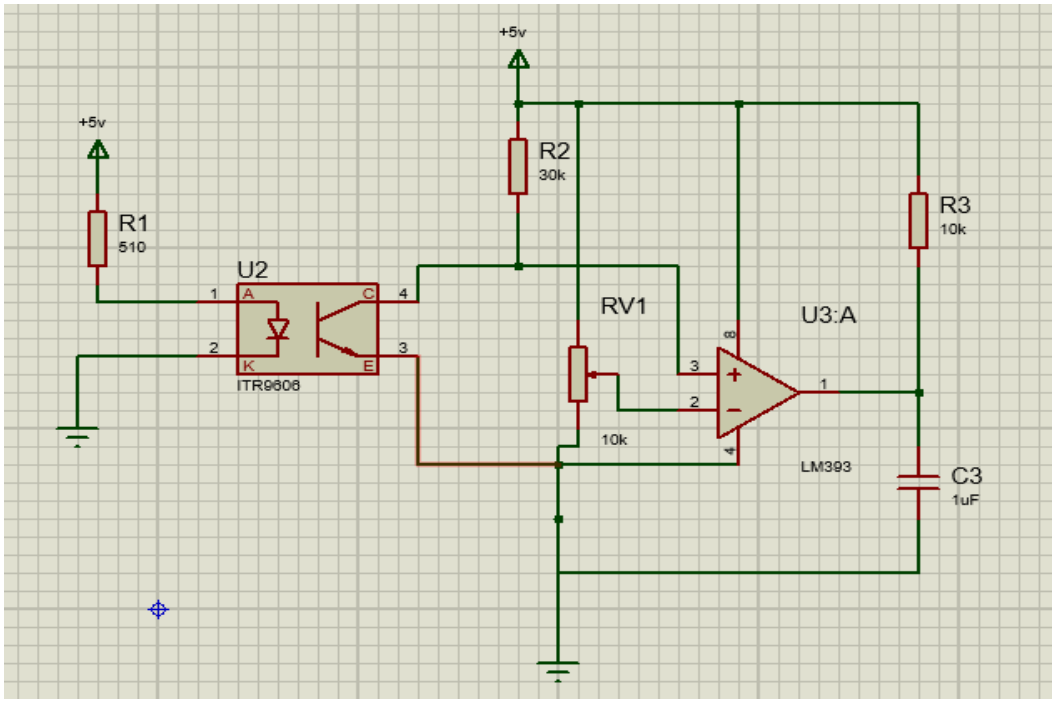
1.2.3 黑线检测模块

实验室提供的芯片 TCRT5000

方案：

利用红外传感器 TCRT5000 与电压比较器 LM393，根据白底时，三极管有光，使三极管饱和，输出低电平；黑底时，三极管无光（光线被吸收），三极管截止，输出高电平。

电路原理图如下：



主要问题：

实际测试时，由于实验室提供的黑线由黑胶带贴成，存在一定的反光现象，实际测试时阈值取舍不当，可能会把白纸检测成无光，而黑线处检测呈有光，导致软硬件逻辑不匹配。

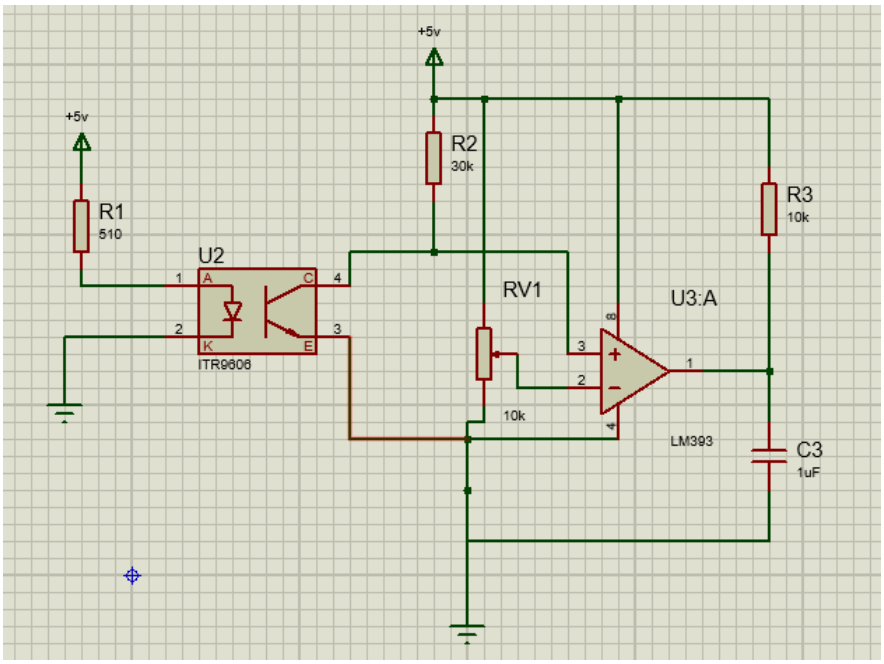
1.2.4 避障模块

实验室提供的元器件：超声波发射器和接收器 TCT40-16T 发 / CT40-16R 收、红外传感器 TCRT5000

(1) 方案一：

利用红外传感器 TCRT5000 与 LM393 构成检测电路，利用光电二极管和蜂鸣器构成报警电路。当有物体遮挡时，电路输出高电平，该高电平可直接控制光电二极管和蜂鸣器报警；无遮挡时，电路输出低电平，报警电路不报警。

电路原理图如下：



主要问题：

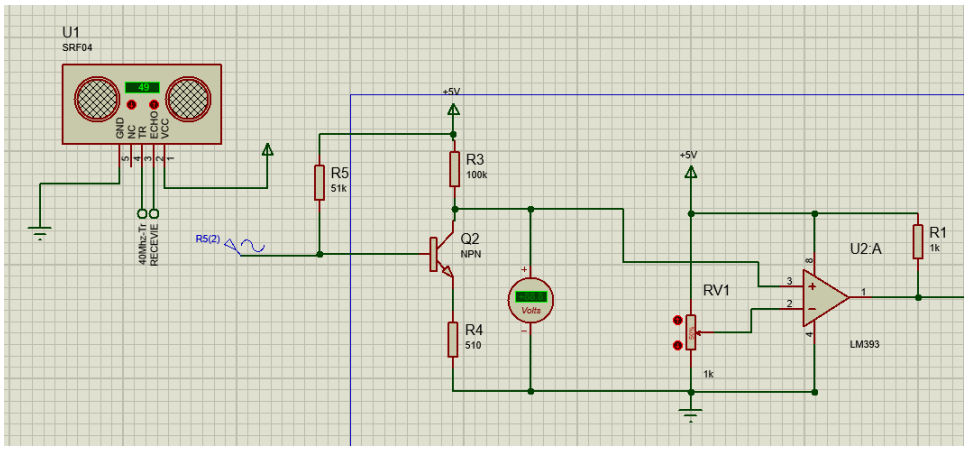
程序在检测到高电平信号时中断电机输出，检测到低电平信号时不中断。该电路实际实现时，如果不加延时控制程序的话，小车在启停时会产生严重的颠簸现象。

(2) 方案二：

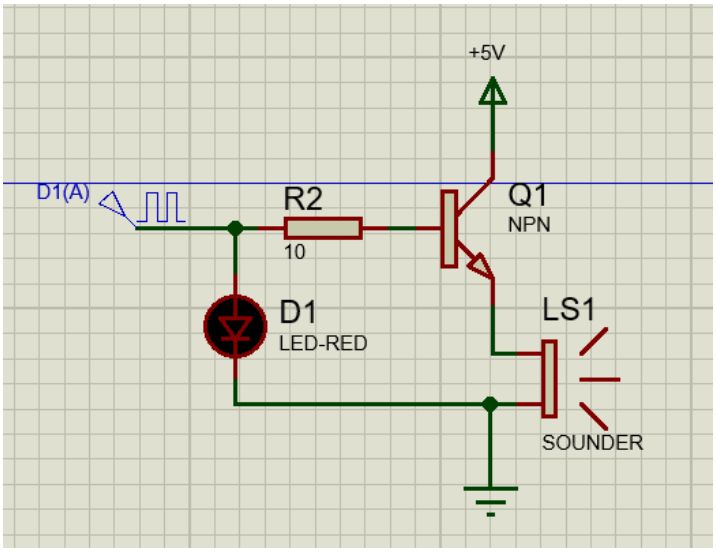
利用超声波发射、接收器（TCT40-16T 发 / CT40-16R 收）和 LM393 构成检测电路，利用发光二极管和蜂鸣器构成报警电路。

基本原理：给发射端提供 40MHz 的方波激励，在遇到障碍时，接收端接收信号并产生一个正弦波响应，通过三极管放大电路和电压比较器 LM393 输出一个方波。将这个方波输入到开发板，利用上升沿循环加一计数的方式检测此方波的脉冲宽度，当方波宽度大于 5us 时判断有障碍，pwm_ctrl 信号控制 pwm 输出开关闭合，小车停止；同时，利用分频函数给蜂鸣器和发光二极管电路输出一个 1kHz 的方波，蜂鸣器报警，二极管闪烁。

检测电路电路图如下：



报警电路电路图如下：



主要问题：

程序设计逻辑比较复杂，且报警电路的 1Khz 方波激励由 basys3 开发板提供，实际检测输出电压过低只有 30mv 左右，需要外接上拉电阻拉高输出电平才能满足报警电路的工作要求。

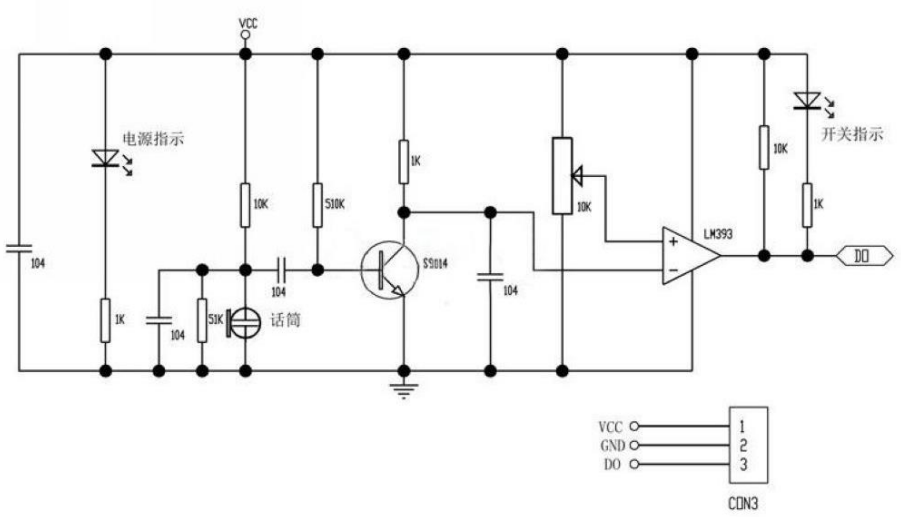
1.2.5 声控运动控制模块

实验室提供的元器件：LM393 和咪头

方案：

基本原理是咪头通过检测空气振动发生电压变化。当没有声音时，经过电压比较器输出低电压，没有输出信号；当有声音时，经过电压比较器输出高电压，输出方波信号。

电路原理图：



利用高电平期间计数的方式进行脉冲宽度检测，脉冲宽度大于 1ms 时，判断为有语音信号，pwm 输出分控制信号 pwm_ctrl_sound 置 1，反之置 0。再跟 pwm_ctrl 做逻辑判断赋值给总控信号 pwm_ctrl_sum，实现利用语音控制启动与停止。

程序实现的 pwm 总控信号 pwm_ctrl_sum 与 pwm 一般分控信号 pwm_ctrl 和声控分信号 pwm_ctrl_sound 的逻辑如下：

Pwm_ctrl	Pwm_ctrl_sound	Pwm_ctrl_sum
0	0	0
0	1	1
1	0	1
1	1	1

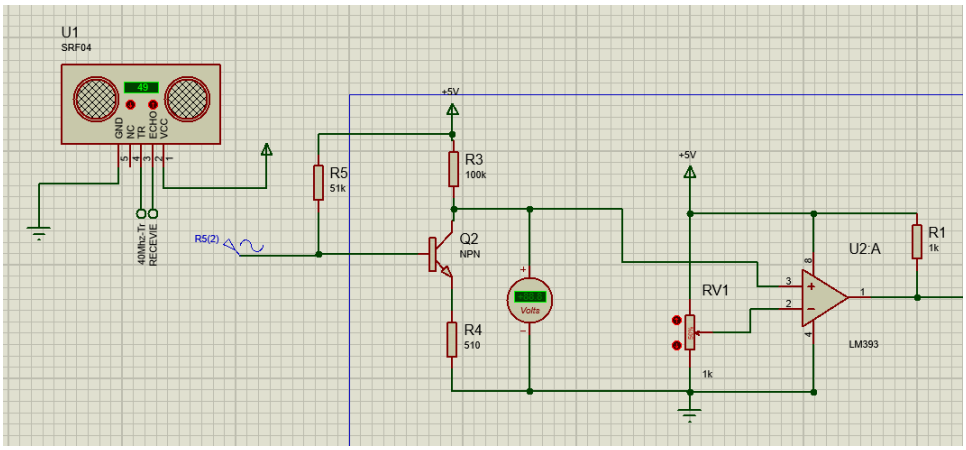
1.2.6 速度检测模块

实验室提供的元器件：ITR9606 槽型光耦、测速码盘、超声波发射器和接收器 TCT40-16T 发 / CT40-16R 收。

(1) 方案一：利用超声波发射器和接收器进行超声波测距

基本原理是：超声波发射器发射一个 40MHz 的方波信号，发射给赛道终点的反射纸板，接收器接收此信号。从发射开始进行计时，知道接收器脉冲方波结束截至，获得时间。根据距离=(时间*340)/2 （m）计算距离。

电路原理图如下：



主要问题：

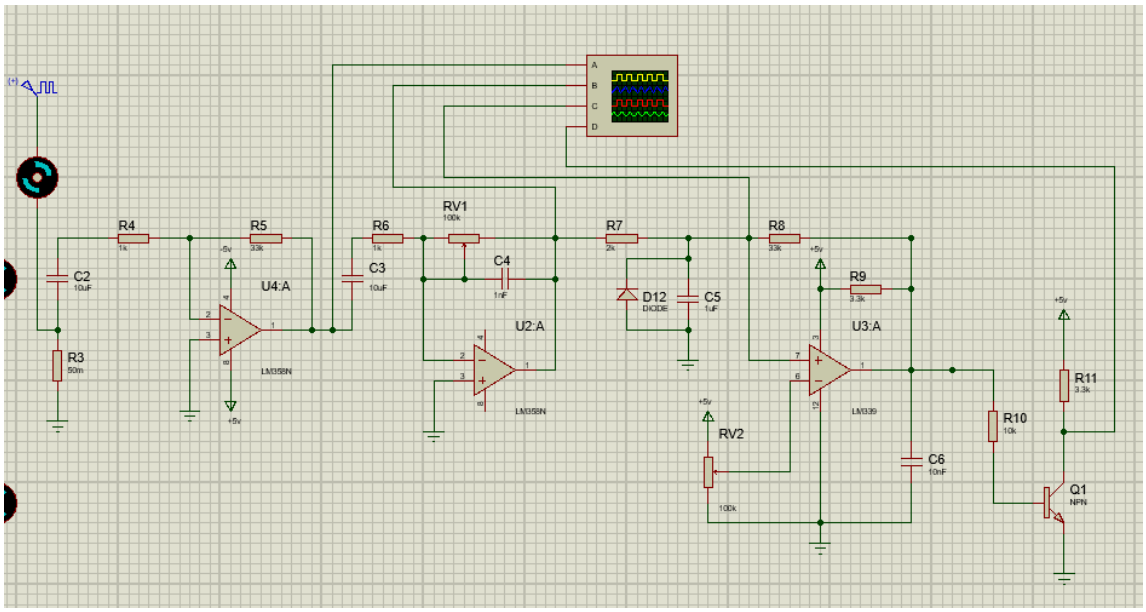
小车运动的路线不是绝对的直线，发射信号和接收信号容易产生比较大的误差。超声波信号容易受到环境的干扰。

(2) 方案二：

利用直接从电机取电压信号，经过滤波和放大后进行分析得到转速。[2]

基本原理：由于采集到的电流信号有直流交流成分，还会产生噪声，高频信号，因此必须对信号进行一定处理。采样的电流经过 C2 电容，直流成份被消除且电容起到一定滤波作用，提取的交流成分经过一级放大器，再经过 C3 电容进入带通滤波器，信号经过 R6 后，低频信号直接经过 RV1，高频信号则要经过 C3。信号二级放大后进入 RC 电路（低通滤波器），此时能通过示波器看到一个相对稳定的正弦信号，正弦信号经过比较器，比较产生的脉冲信号经过三极管放大电路，输出端能输出一个+5v 的脉冲信号。对此脉冲进行计算，则可反应电机转速。

电路原理图：



主要问题：

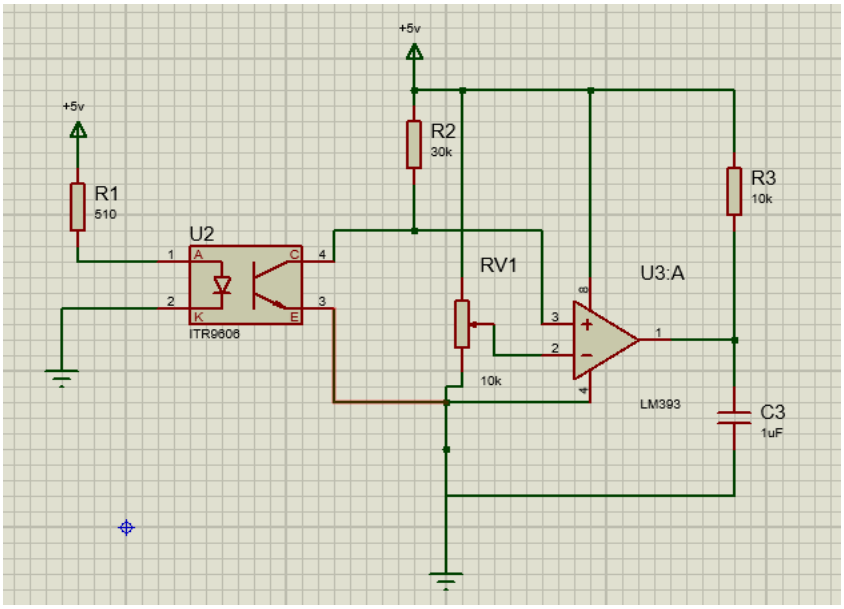
电机连接复杂的载荷，在 5v 电压供电的条件下会使电机转速明显减慢。

(3) 方案三：运用 ITR9606 槽型光耦、测速码盘和电压比较器 LM393。

基本原理是：当缺口转到光耦的凹槽时，三极管有光，使三极管饱和，输出低电平；当缺口离开光耦的凹槽时，三极管无光（光线被挡住），三极管截止，输出高电平。这样就把转速转化为近似的方波脉冲。

根据小车的轮胎的尺寸，r=3.3cm，可得知轮胎一圈长度 l=2πr=20.73cm，齿轮一圈 20 个脉冲周期，相当于一个脉冲运动 1.04cm，统计脉冲个数可得行驶距离，可以近似的把方波的频率看成转速。

电路原理图：



主要问题：

槽型光耦在电机连杆上会随着电机转动而抖动，对采集到的脉冲产生影响，有一定的偏差，最后消抖滤波处理。

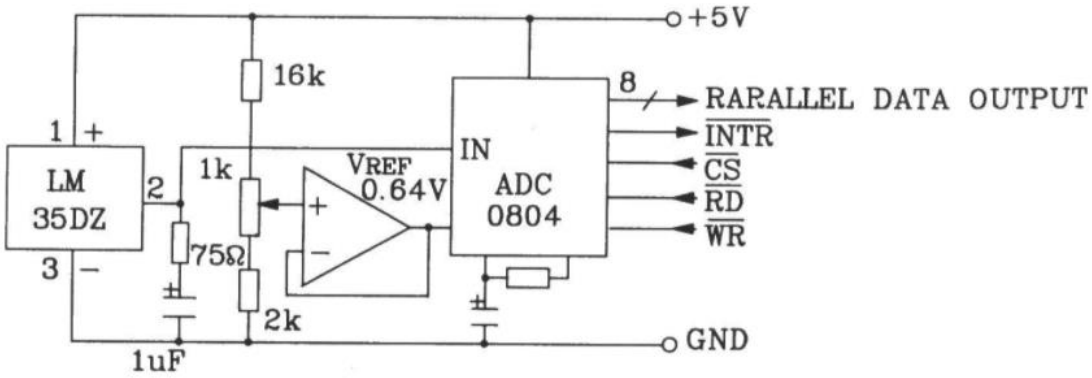
1.2.7 温湿度检测模块

实验室提供的芯片：LM35DZ、18b20、热敏电阻 10D-7、湿敏电阻 CJ-HR31D，NE555 定时器。

(1)**方案一：**利用 LM35DZ 和 AD 转换器构成测温电路

LM35 传感器可以随不同的温度变化而产生不同的电压，为线性关系。0° C 时输出为 0V，每升高 1° C，输出电压增加 10mV。假设 AD 读取出来的值为 val，那么该 AD 值对应的电压为：U=val*5/1023*1000=4.8876*val，单位 mV。对应的温度为 T=4.8876*val/10=0.48876*val，单位为° C

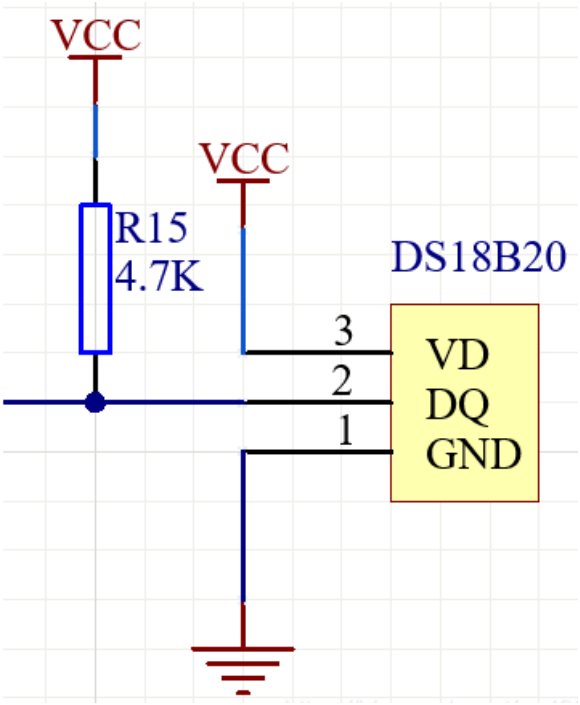
电路原理图：[3]



(2)**方案二：**利用 18b20 构成温度检测电路[4]

基本原理：DS18B20 返回的 16 位二进制数代表此刻探测的温度值，其高五位代表正负。如果高五位全部为 1，则代表返回的温度值为负值。如果高五位全部为 0，则代表返回的温度值为正值。后面的 11 位数据代表温度的绝对值，将其转换为十进制数值之后，再乘以 0.0625 即可获得此时的温度值。

电路原理图：



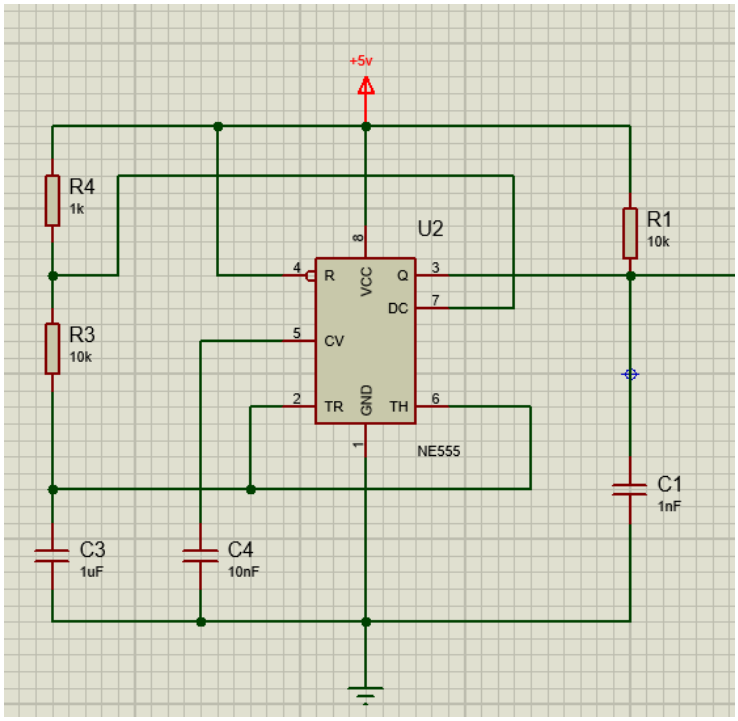
主要问题：

DQ 引脚必须要一个上拉电阻。编程所需代码由于时序编写、初始化等过于庞大，比较耗时。

(3)**方案三：**利用 NE555 定时器和热敏电阻、湿敏电阻构成多谐振荡器。

基本原理：利用 555 定时器接成多谐振荡器，把湿敏电阻或热敏电阻接入到充放电回路。当环境温湿度变化时，充放电回路的时间常数 τ 随之改变，而三号管脚输出端输出方波的占空比随之改变。通过量化占空比与阻值、阻值与实际温湿度的关系，即可通过占空比来反应实际温湿度。

电路原理图：

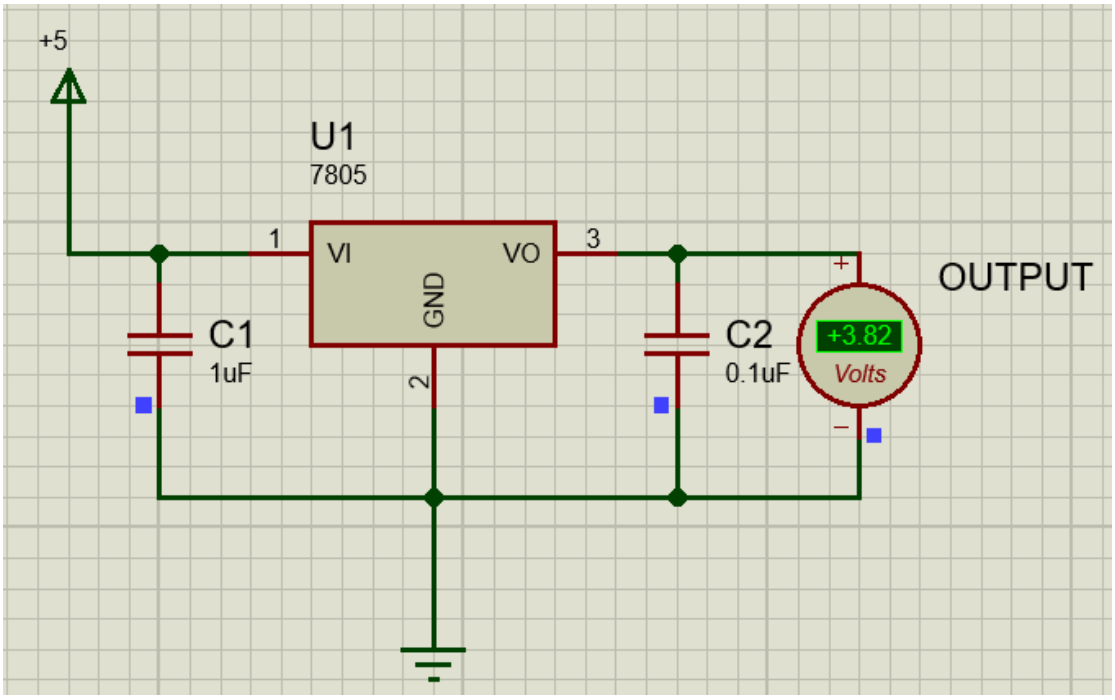


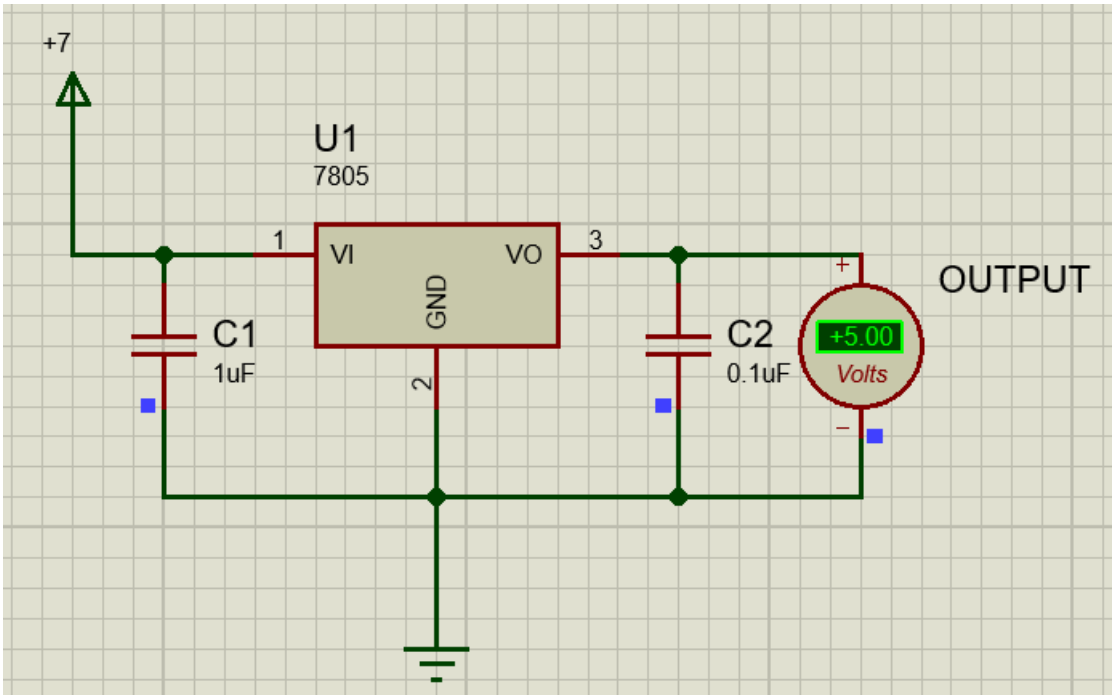
主要问题：

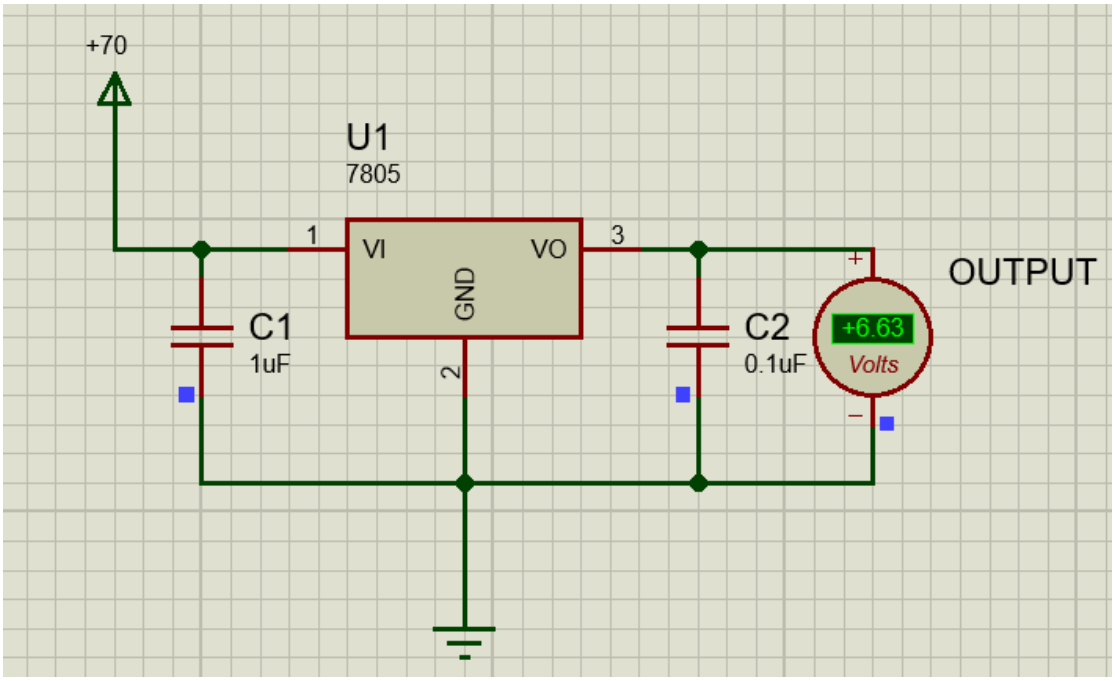
需要对应阻值对占空比进行量化，不能实现连续变化，只能编程实现大概的温度显示。

1.3 选定方案的参数计算及仿真分析

1.3.1 稳压模块







根据 proteus 仿真，外加+5v 电压时，输出电压+3.82v，没达到稳压要求；外加+7v~+35v 时，输出电压+5v；输入电压高于+35v 时，输出电压高于+5.01v，甚至报错。

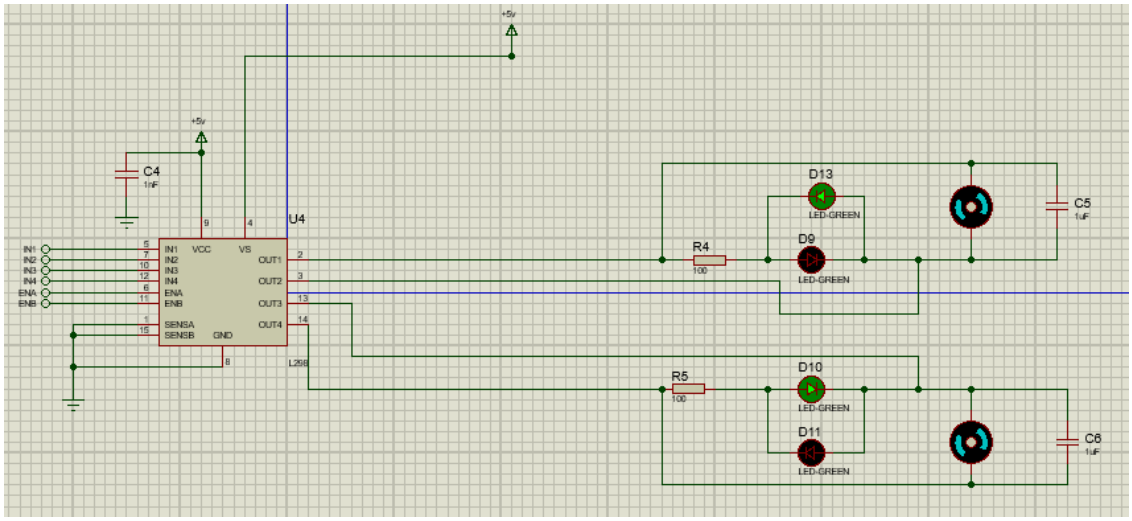
说明稳压模块输入电压最好略高于+7v，输入端大电容滤低频噪声，输出端小电容滤高频噪声。根据波纹系数的要求选择滤波电容，公式为 C=0.289/{f×（U/I）×ACv}。

其中，0.289 是由半波阻性负载整流电路的波纹系数推演来的常数；f 是电路的脉冲频率，单位是 Hz；U 是整流电路最大输出电压，单位是 V；I 是整流电路最大输出电流，单位是 A；ACv，是波纹系数，单位是%。

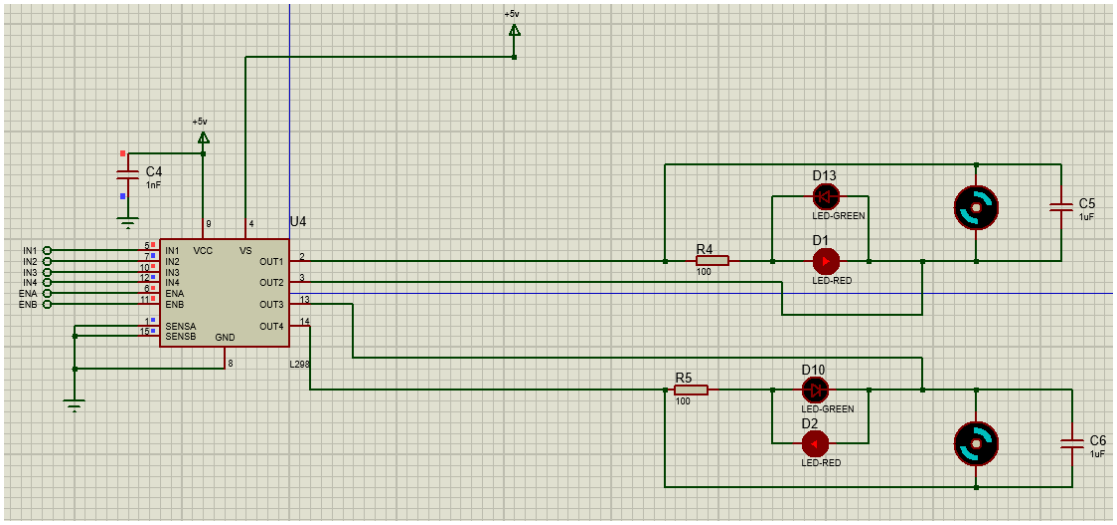
1.3.2 电机驱动模块

仿真电路图如下：

输入端接 1010，电机正转，绿灯亮。



输入端接 0101，电机反转，红灯亮。



此处主要是发光二极管限流电阻的参数计算，发光二极管的压降是比较固定的，通常红色为 1.6V 左右，绿色有 2V 和 3V 两种，黄色和橙色约为 2.2V 左右。对于常用的几毫米大小的二极管，其工作电流一般在 2 毫安至 20 毫安之间，电流越大亮度越高。

用电源电压减去二极管的压降，再除以设定的工作电流，就得出限流电阻的阻值。限流电阻 R 计算式：R=（E－UF）/IF，式中 E 为电源电压，UF 为 LED 的正向压降，IF 为 LED 的正常工作电流。

一个发光二极管的额定电压是 1.5~2.5V，电阻不大于 50 殴姆，他的电流约为 0.04A。使用电源电压+5V，给二极管串联一个电阻，串联电路电流相等，电源两端电压是+5V，那么电阻两端电压 Ur 约为 2.5~4.5V。所以串联电阻的阻值约为 R=U/I=Ur/0.04=62.5~112.5Ω，可以选用 100Ω 电阻。

1.3.3 黑线检测模块、测速模块

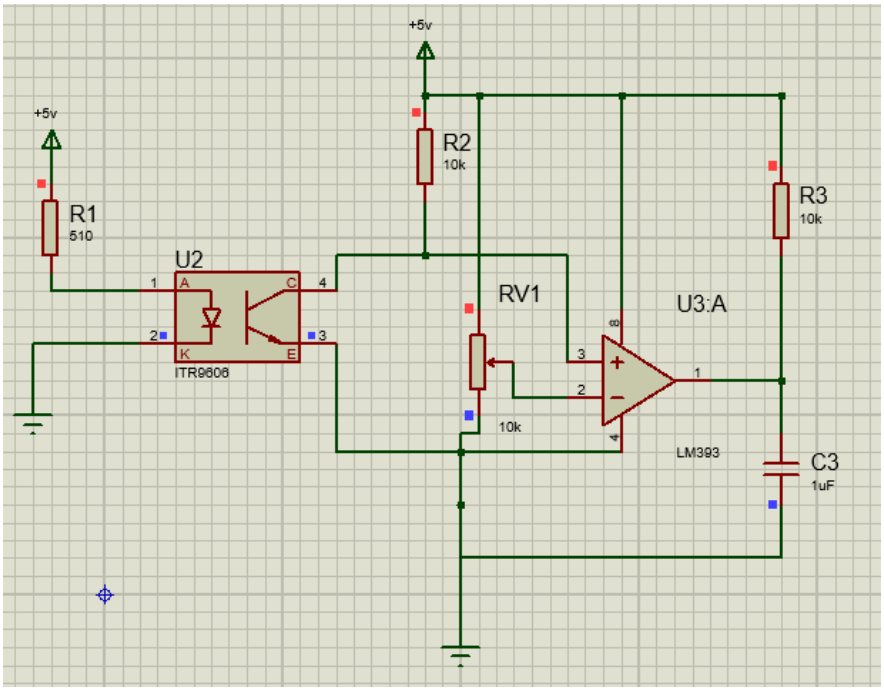
元器件参数：

ITR9606

Absolute Maximum Ratings (Ta=25℃)

Parameter		Symbol	Ratings	Unit
Input	Power Dissipation at(or below) 25℃ Free Air Temperature	Pd	100	mW
	Reverse Voltage	V_R	5	V
	Forward Current	I_F	50	mA
	Peak Forward Current (*1) Pulse width ≦100 μs, Duty cycle=1%	I_{FP}	1	A
Output	Collector Power Dissipation	P_C	75	mW
	Collector Current	I_C	50	mA
	Collector-Emitter Voltage	B V_{CEO}	30	V
	Emitter-Collector Voltage	B V_{ECO}	5	V

仿真电路图如下：



发光二极管回路限流电阻计算与上一模块相似，输入回路额定电流 IF=50mA，所以限流电阻>100 欧姆即可；输出回路额定电流 IC=50mA，由于防止三极管工作在截至区，即 UCE=Vcc-IcRc=5-IcRc 不能太接近 5v，采用 10k 电阻。滑动变阻器考虑到更精细的调节范围选择 1kΩ或 10kΩ均可。LM393 输出端上拉电阻拉高点位，滤波电容稳定电压。

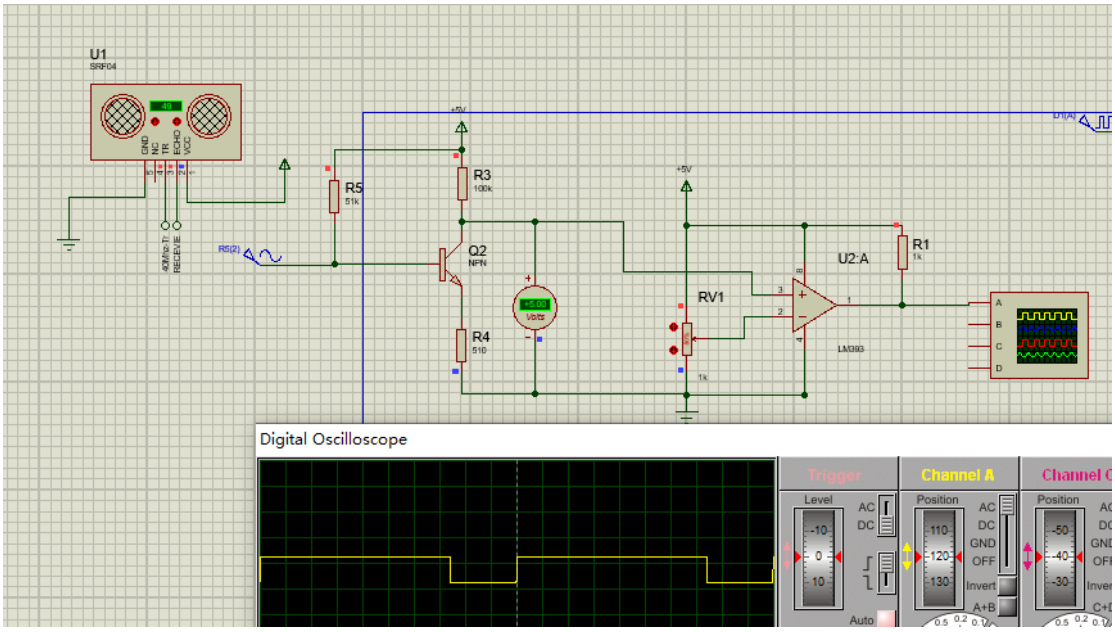
1.3.4 超声波避障模块

仿真电路图如下：

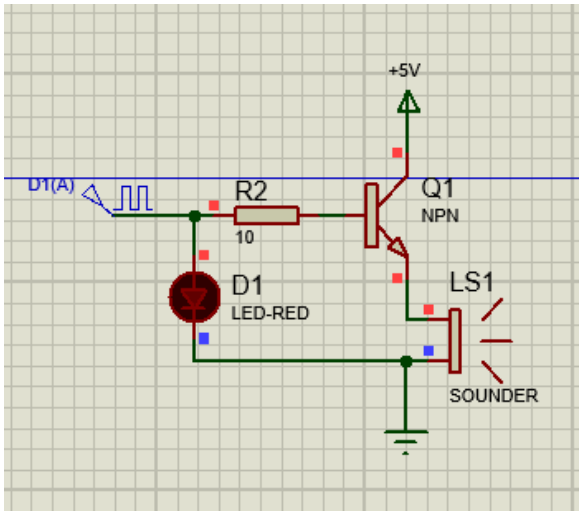
根据 TCT40-16T 发 / CT40-16R 收的参数：

性　　能	发　　射	接　　收
标称频率（KHz）	40	40
发射声压 KHz （0dB=0.02mPa）	117min	——
接收灵敏度 at40KHz (0dB=V/Pa)	——	－67min
静电容量 at1KHz,<1V (PF)	2000±30%	
－6dB 指向角	60°	

发射端提供 40KHz 左右的方波脉冲，输出端产生类正弦响应，最终经过电压比较器输出方波信号。

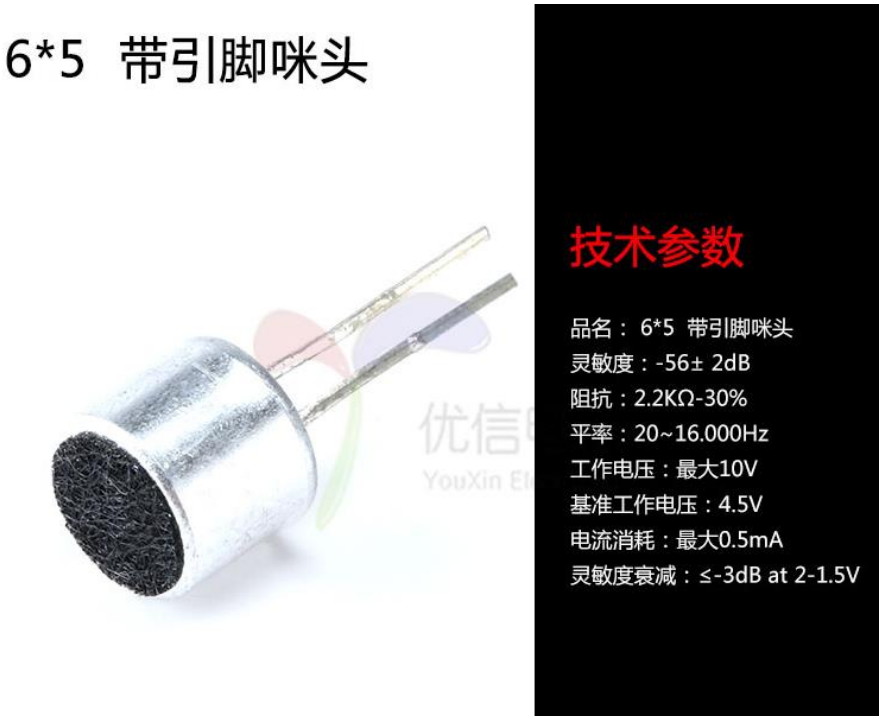


此时，根据蜂鸣器参数，需要 1KHz~5Khz 的方波信号，报警电路报警，红灯亮，蜂鸣器响。

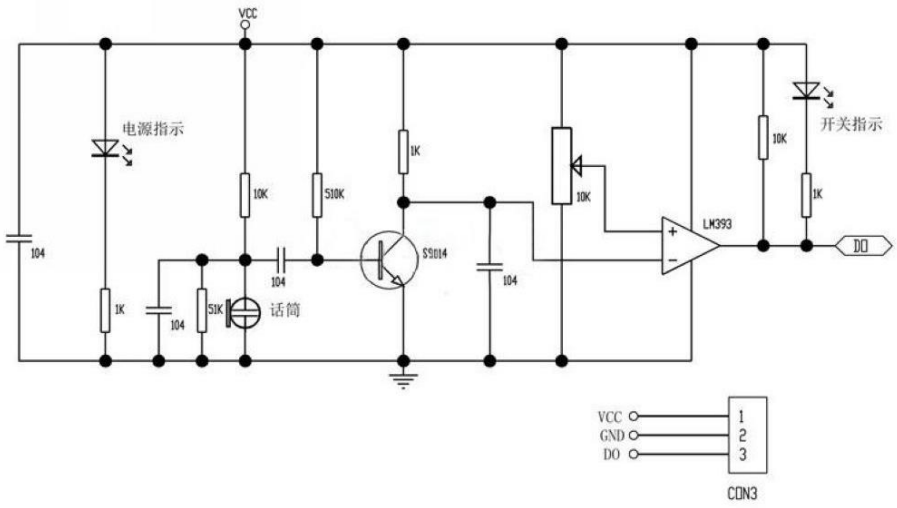


1.3.5 声控模块

根据咪头参数：



根据电路原理图，



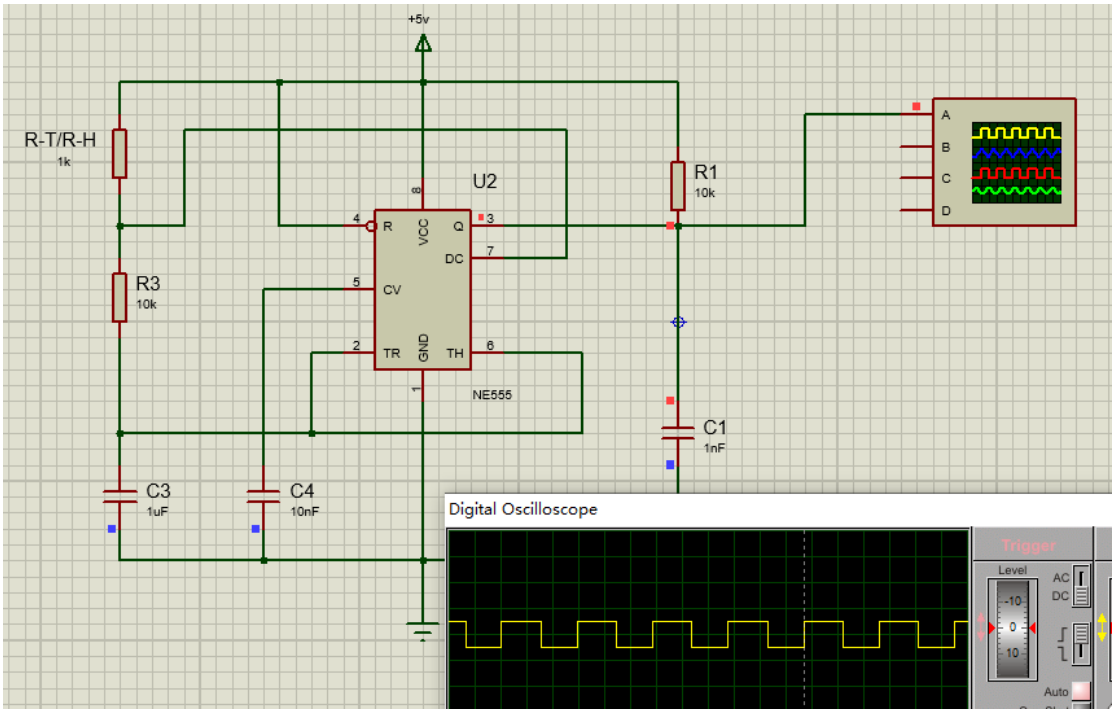
基准工作电压 4.5v，额定电流 0.5mA，因此咪头电路限流电阻选择 10kΩ以上，耦合电容“隔直通交”，旁路电容稳定两端压降。

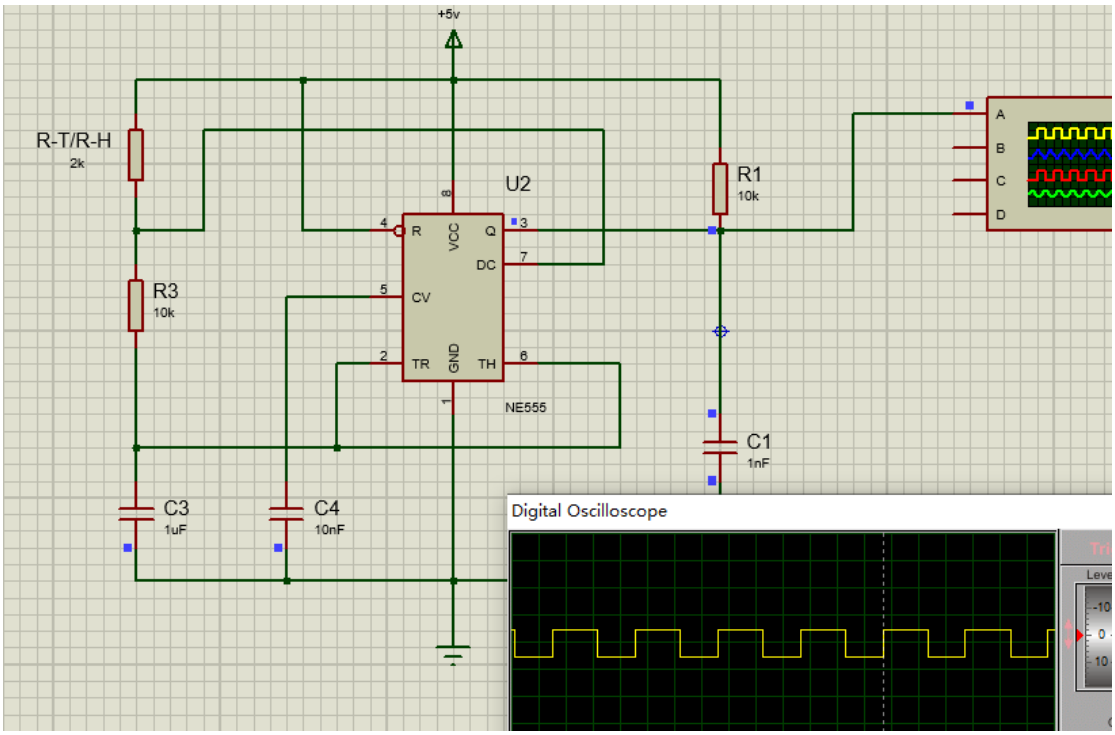
1.3.6 温湿度检测模块

根据 NE555 定时器构成多谐振荡器的原理，充电时间 T1=(RT+R3)C3*ln2，放电时间 T2=R3C3*ln2，所以震荡周期 T=T1+T2≈0.7(RT+2R3)C3。

电路中 C1、C4 主要是滤波的作用防止管脚电平发生变化，三管脚上拉电阻根据经验选择 1k 以上即可。

仿真电路图：





根据公式，改变充放电回路电阻阻值 RT，改变时间常数 RC，改变输出端信号的占空比。因此，可以通过占空比来反应环境温湿度。

2. 所需元器件清单

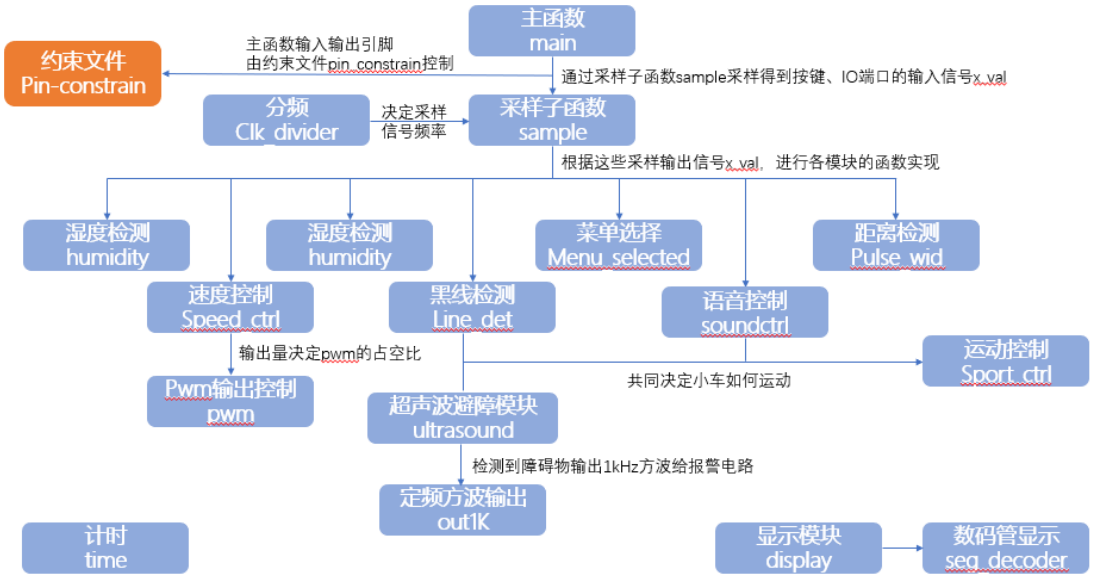
模块	元器件
稳压电源	LM7805*1
驱动电路	瓷片电容 0.01μF*2，电解电容 1μF*1，10μF*1 9V 电池 L298N*1 瓷片电容 330pF*1 1N4001 二极管*8 电阻 100 欧*2 绿色发光二极管*2，黄色发光二极管*2
红外检测(测速测距）	红外传感 TCRT5000*1 LM393*1 滑动变阻器 10K*1 电阻 510 欧*1 电阻 10K*2 瓷片电容 2.2nF*1 槽型光耦 ITR9606 NE555*1
温湿度	湿敏电阻 CJ-HR31D*1 温度传感器 18B20*1 电阻 10K*2 电阻 1K 欧*1 电解电容 1μF*1，瓷片电容 0.01μF*1，2.2nF*1
声控	话筒拾音器（咪头）*1 LM358*2 LM324*2 电阻 510 欧*1，1K*8，2K*1，10K*2 电解电容 1μF*1，瓷片电容 0.1μF*2

避障	1N4001 二极管*2 滑动变阻器 10K*1 超声波传感器*1 LM393*1 滑动变阻器 10K*1 瓷片电容 2.2nF*1 电阻 10K*1，1K*1,100K*1，51K*1，510K*1 蜂鸣器*1 红色发光二极管*1 2N9013 三极管*2 共阴极七段数码管*3
其他	按键（自锁 7*7） 超声波固定架

三、软件设计

1. 系统工作流程

采用模块化的设计方法，主要是利用主函数调用子函数的行驶来实现每个模块的功能。总的流程图如下：



2. 主要模块程序设计

各模块的输入输出变量、引脚、实现的功能详情等都在函数内进行了详细的备注，此处不再赘述。

2.1 主函数

```
//进行湿度检测，输出相应的湿度值给数码管
        wire  [15:0]  hum_wid;
        wire    [15:0]  hum_r;
        humidity  hh1(
        .clk(h_clk),
        .rst(rst),
        .humidity(hum_val),//湿度输出信号
        .hum_wid(hum_wid),//湿度方波的宽度反应阻值
        .hum_r(hum_r)//湿敏电阻阻值反应相对湿度
);

//进行黑线检测，输出黑线数量
        wire[31:0]  lin;//黑线数量的输出值，中间变量，内部使用，不外连；
        line_det  line1(
                .pulse(line_val),
                .rst(rst),
                .num(lin)
        );

//进行计时，输出实时时间和总过程时间
//黑线数小于 11 显示实时时间，大于时间 11 显示截止时间；
        wire  [31:0]tim;//实时时间
        wire  [31:0]  tim_reg;  //总时间时间
        time  t1(             //10Hz 对应的显示 1 是 0.1s
                        .clk(t_clk),
                        .rst(rst),
                        .lin(lin),
                        .tim(tim),
                        .tim_reg(tim_reg)
        );

//超声波避障
        wire[15:0]  wid;
        wire  pwm_ctrl1;//控制电机 pwm
        wire  [15:0]  out11;//给报警电路输出的报警信号
```

```
        ultrasound  u1(
        .clk(clk),
        .rst(rst),
        .usound(usound),//超声波信号经过放大电路的输出信号
        .usclk(usclk),//超声波发射信号的输出管脚
        .wid(wid),//超声波输出信号的宽度
        .out(out11),//蜂鸣器控制信号
        .pwmc(pwm_ctrl1)//pwm 控制信号
);
//蜂鸣器和小灯报警
        outlk  oo1(
        .clk(clk),
        .out(out11),
        .drive(out)//输出 1kHz 方波
);

//进行声音控制
        wire  pwm_ctrl2;//声音控制小车是否运动
        soundctrl  sd1(
        .clk(clk),
        .rst(rst),
        .soundin(soundin),//声音控制输入信号
        .pwmc2(pwm_ctrl2)//控制 pwm 输出的开关
);


//进行脉宽测量，返回脉冲宽度
        wire  [15:0]  length;
        wire  [15:0]  width;
        pulse_wid  dp(
                .pulse(dis_val),
                .rst(rst),
                .clk(clock),
                .distance(length),
                .pul_wid(width)
        );

//进行速度控制，输出调节小车转速
        wire  [31:0]  pwm_l;
        wire  [31:0]  pwm_r;
        speed_ctrl  sc(
                        .lin(lin),
                        .sw_l_adjust_val(sw_l_adjust_val),
                        .sw_r_adjust_val(sw_r_adjust_val),
```

```
                        .pwm_l(pwm_l),
                        .pwm_r(pwm_r)
        );

//小车运动方向及刹车制动控制延时
        wire  pwm_ctrl;             //pwm 开关控制
        sport_ctrl  d1(
                        .rst(rst),
                        .clk(t_clk),
                        .lin(lin),
                        .direction(dir),
                        .pwm_ctrl(pwm_ctrl)
        );

//pwm 输出
        wire  ENA_l_out,ENB_r_out;
        wire  L,R;
        assign  ENA_l=L;
        assign  ENB_r=R;
        PWM  PWML(
                .clk(clock),
                .rst(rst),
                .duty(pwm_l),
                .out(ENA_l_out)
        );
        PWM  PWMR(
                .clk(clock),
                .rst(rst),
                .duty(pwm_r),
                .out(ENB_r_out)
        );
        //L=pwm_c  &&  ENA_l_out  &&  sw_val(pwm 信号)
        and  a1(L,pwm_ctrl,ENA_l_out,sw_val,pwm_ctrl1);
        and  a2(R,pwm_ctrl,ENB_r_out,sw_val,pwm_ctrl1);

//实现语音和避障同时控制小车运动与否的逻辑
        always  @(*clk)
        begin
        if(pwm_ctrl1==0  &&  pwm_ctrl2==0)
                begin
                        ENA_l<=0;
                        ENB_r<=0;
                end
        else  if(pwm_ctrl1==0  &&  pwm_ctrl2==1)
```

```
                begin
                        ENA_l<=1;
                        ENB_r<=1;
                end
        else  if(pwm_ctrl1==1  &&  pwm_ctrl2==0)
                begin
                        ENA_l<=1;
                        ENB_r<=1;
                end
        else
                begin
                        ENA_l<=1;
                        ENB_r<=1;
                end
        end

//数码管显示的菜单选择
//current_show  是 menu_select 函数的输出，
//是被选择的时间、距离等参数，用于数码管 disp 函数的显示；
//menu_show 用于显示第几个模式
        wire  [31:0]  current_show;
        menu_select  ms(
                        .clk(t_clk),
                        .lin(lin),
                        .tim(tim),
                        .tim_reg(tim_reg),
                        .distance(length),
                        .hum_r(hum_r),
                        .select(menu_val),
                        .selected(current_show),
                        .menu_show(menu_show)
        );
//实现数码管显示
//输入待显示量，current_show，就会在四段数码管上循环显示
        disp  p1(
        .clk(clock),
        .val(current_show),
        .bit(bit),
        .seg(seg)
        );
   endmodule
```

2.2 采样子函数

```
module  sample(
            input  clk,
            input  line,
            input  dis,
            input  sw,
            input  menu,
            input  [4:0]sw_l_adjust,
            input  [4:0]  sw_r_adjust,

            input  humidity,
            input  soundin,

            output  line_val,
            output  dis_val,
            output  sw_val,
            output  [4:0]sw_l_adjust_val,
            output  [4:0]sw_r_adjust_val,
            output  menu_val,

            output  hum_val,//湿度采样输出的信号
            output  sound_val//声音信号的采样输出
);

        reg  l,d,s,m,h;//l==线，d==距离，s==pwm 信号，m==菜单，h==湿度,内参数；
        reg  [4:0]  sl,sr;

//借用中间变量赋值  连接输入输出变量
        assign  line_val=l;
        assign  dis_val=d;
        assign  sw_val=s;
        assign  menu_val=m;
        assign  sw_l_adjust_val=sl;
        assign  sw_r_adjust_val=sr;
        assign  hum_val=h;
//检测采样，就是循环用 input 赋值 output，循环输出 output，
        always  @(posedge  clk)
        begin
                s<=sw;
                l<=line;
                d<=dis;
                m<=menu;
                sl<=sw_l_adjust;
```

```
            sr<=sw_r_adjust;
            h<=humidity;
        end

    endmodule
```

2.3 菜单选择子函数

```
    module  menu_select(
                input  clk,
                input  [31:0]  lin,
                input  [31:0]  tim,
                input  [31:0]  tim_reg,
                input  [15:0]  distance,
                input  select,
                input  [15:0]  hum_r,//湿度信号
                output  [31:0]  selected,
                output  [3:0]  menu_show
        );

                reg  [3:0]  mu=3'b001;
                assign  menu_show=mu;

                reg  [31:0]  current_show;
                assign  selected=current_show;
//{}位拼接运算符，实现了 0001，0010，0100,1000  四种模式的切
换；
                always  @(posedge  select)
                begin
                mu<={mu[2:0],mu[3]};
                end

                always  @(posedge  clk)
                begin
                case(mu)
                4'b0001://第一种模式显示时间
                begin
                        if(lin>=11)
                                current_show<=tim_reg;
                        else
                                current_show<=tim;
                end
                4'b0010:current_show<=lin;//第二种模式显示线的数量
```

```
                4'b0100:current_show<=distance;//第三种模式显示距离
                4'b1000:current_show<=hum_r;//第四种模式显示湿度
                default:current_show<=tim;//报错返回显示实时时间
                endcase
                end

    endmodule
```

2.4 速度控制子函数

```
    module  speed_ctrl(
                input  [31:0]  lin,
                input  [4:0]  sw_l_adjust_val,
                input  [4:0]  sw_r_adjust_val,
                output  [31:0]  pwm_l,
                output  [31:0]  pwm_r
                );
                reg  [31:0]  pl;
                reg  [31:0]  pr;
                assign  pwm_l=pl;
                assign  pwm_r=pr;

          always@(lin)
                begin
                //BCD 段  EFG 段  GFE 段  DCB 段全速通过
                if(lin<3||(lin>=4&&lin<9)||(lin>=10&&lin<12))
                begin
                        pl<=2000+(sw_l_adjust_val*15);
                        pr<=2000+(sw_r_adjust_val*15);
                end
                else  if(lin==3||lin==9)
                //DE  ED 段减速通过
                begin
                        pl<=500+(sw_l_adjust_val*5);
                        pr<=500+(sw_r_adjust_val*5);
                end
                end
endmodule

module    sport_ctrl(
                input  rst,
                input  clk,
                input  [31:0]  lin,
```

```
                input  wid,//超声波的测量信号宽度
                output  [3:0]  direction,
                output  pwm_ctrl//pwm 输出的开关
);
        reg  [31:0]  cnt=0;
        reg  [3:0]  direct;//方向变量
        assign  direction=direct;//方向

        reg  pwm_c;//pwm 波的输出控制，==1  输出  ==0 没有输出；
        assign  pwm_ctrl=pwm_c;

        always  @  (posedge  clk)
        begin
                if(rst)//rst 复位，计数器清零
                        begin
                                cnt<=0;
                        end
                if(lin<6)
                        begin
                                direct<=4'b1010;//电机 1010 都正转，前进
                                pwm_c<=1;
                        end
                else  //lin>=6
                        direct<=4'b0101;//0101 反转，后退；

                if(lin==6)//在 G 点停车，并停留一段时间；
                        begin
                                cnt<=cnt+1;
                                if(cnt<100)//延时 100 个单位时间
                                        pwm_c<=0;
                                else
                                        pwm_c<=1;
                        end
                else  if(lin>=12)
                        pwm_c<=0;//返回起点，刹车
                else
                        pwm_c<=1;//一直有输出，运动
        end


//利用方波宽度来控制车的走和停
        always  @(*)
        begin
        if(wid>=5)  //5us 算检测到了
                pwm_c  <=  0;//1?
```

```
        else
                pwm_c  <=  1;
        end
    endmodule
```

2.5 pwm 输出子函数

```
    //利用计数器统计期望占空比维持的时间，没有达到时间持续输出高电平，达到了输出低电平，总的周期用 num 控制；
module  PWM(
        input  clk,
        input  rst,
        input  [31:0]  duty,//设置占空比，相当于高电平维持时间，取决于左右电机需要的占空比时长。
        output  out
        );

        integer  div=20;
        wire  p_clk;
        reg  pwm;
        assign  out=pwm;


        clk_divider  pwm_clk(  //50kHz,f_pwm=50Hz，分频子函数，输出一个 p_clk 信号，
                        .clk(clk),
                        .d_val(div),
                        .d_clk(p_clk)
        );

        reg[15:0]  cnt;
        integer  num=1000;

        always  @(posedge  p_clk)
        begin
                pwm=0;//初始化 pwm 输出波形为低电平
                if(rst||cnt>=num)//  如果  rst 复位或者计数器>给定值，重置计数器
                        cnt<=0;
                else  //否则就计数
                        cnt<=cnt+1;

                if(cnt<=duty)  //如果计数器小于占空比维持的时间，输出高电平
```

```
                pwm=1;
        else
                pwm=0;
    end
endmodule
```

2.6 分频子函数

利用计数器在目标分频值以下一直循环加一计数，到达目标值取反置零再循环加一，实现分频过程。

```
module  clk_divider(
    input  clk,
    input  [32:0]  d_val,//分频器所需的频率；由 integer 常数设置输入
    //这个数是多少，分频后的数值就是 100MHz/这个数。
    //调用子函数之前在主函数定义这个值，integer  clk_1khz=1000;
    output  d_clk
    );

        reg  [32:1]  cnt=0;//计数器

        reg  fp=1'b0;//输出初始化为 0;
        assign  d_clk=fp;//输出产生的是一个 clk 时钟信号！！！

        always  @(posedge  clk)
        begin
            if(cnt==((d_val/2)-1))
                begin
                    cnt<=0;
                    fp=~fp;
                end
            else
                cnt<=cnt+1;
        end
endmodule
```

2.7 显示子函数

```
module  disp(
        input  [31:0]  val,//输入的需要显示的数值，时间，温度..
        input  clk,
```

```
            output  [3:0]  bit,//输出位控制和段控制
            output  [7:0]  seg
);
wire  r_clk;
integer  clk_1khz=1000;//时钟频率整数
reg  [31:0]  register;//寄存器保存输入的数值
reg  [3:0]  bit4;//第四个数码管
reg  [3:0]  bit3;//第三个数码管
reg  [3:0]  bit2;//第二个数码管
reg  [3:0]  bit1;//第一个数码管
reg  [3:0]  bit0=0;//初始化全为 0;
reg  [3:0]  num_select;
wire  [7:0]  seg_1;
clk_divider  refresh_clk(
            .clk(clk),
            .d_val(clk_1khz),
            .d_clk(r_clk)
);
//输出是 1khz 的 r_clk 信号

always@(*)
    begin
        register<=val;//输入值进行数位划分，个十百千
        bit4<=register/1000;                    //千位
        bit3<=register%1000/100;            //百位
        bit2<=register%100/10;                //十位
        bit1<=register%10;                        //个位
    end

reg[3:0]  bit_ctrl=4'b1110;//位控制初始化，共阴极数码管 4 ‘b0001?

always@(posedge  r_clk)
    begin
            bit_ctrl<={bit_ctrl[2:0],bit_ctrl[3]};//拼接移位，循环调用;
    end

    assign  bit=bit_ctrl;//设置哪个数码管被选用

    always  @(bit_ctrl)
        begin
          case(bit_ctrl)
            4'b1110:num_select<=bit1;//调用第一个数码管
            4'b1101:num_select<=bit2;
            4'b1011:num_select<=bit3;
```

```
                4'b0111:num_select<=bit4;
                default:num_select<=bit0;//出错全部显示
                endcase
            end

        //调用数码管
        seg_decoder  p1(
                        .val(num_select),
                        .out(seg)//哪个数码管显示，显示什么
        );
    endmodule
```

2.8 数码管显示子函数

```
    //7 段数码管译码器无小数点（共阳）
module  seg_decoder(
                input  [3:0]  val,
                output  [7:0]  out
);
        reg  [7:0]  code;
        assign  out=code;//输出就是八位的二进制 01 数组
        always@(val)
        begin
        case(val)
                4'd0:code=8'b1000_0001;
                4'd1:code=8'b1100_1111;
                4'd2:code=8'b10010010;
                4'd3:code=8'b10000110;
                4'd4:code=8'b11001100;
                4'd5:code=8'b10100100;
                4'd6:code=8'b10100000;
                4'd7:code=8'b10001111;
                4'd8:code=8'b10000000;
                4'd9:code=8'b10000100;
                default:code=8'b11111111;
        endcase
end
    endmodule
```

2.9 黑线检测子函数

```
module  line_det(
     input  pulse,
     input  rst,
     output  [31:0]  num
     );

     reg  [31:0]  cnt;
     assign  num=cnt;
     always@(posedge  pulse,posedge  rst)
     begin
             if(rst)
                     cnt<=0;
             else
                     cnt<=cnt+1;
     end
endmodule
```

2.10 距离脉冲宽度检测子函数

```
module  pulse_wid(
     input  pulse,
     input  rst,
     input  clk,//输入的是 1MHz 信号
     output  [15:0]  distance,
     output  [15:0]  pul_wid                    //ms
     );

     integer  div_count=100;          //1ms
     wire  cou_clk;


     clk_divider(
                     .clk(clk),
                     .d_val(div_count),
                     .d_clk(cou_clk)
     );
     //把输出 cou_clk 作为时钟信号;

     reg  [15:0]  dt=0;//初始化计数器
     reg  [15:0]  pw=0;//初始化输出
```

```
        reg  [15:0]  pw2;
        assign  pul_wid=pw;//输出脉冲宽度

        always@(posedge  cou_clk,  posedge  rst)
        begin
                if(rst)
                        begin
                                dt<=0;
                                pw<=0;
                                pw2<=0;
                        end//rst 复位信号全部清零。
                else
                        begin
                                if(pulse)
                                        begin
                                                dt<=dt+1;
                                                pw2<=dt;
                                        end
                                        //pulse 高电平期间计数，pw2 中间寄存器；
                                  else
                                        begin
                                                pw<=pw2;//pulse 下降沿则把 pw2
的保存值赋给 pw 输出；
                                                dt<=0;//同时计数器清零等待下一
次高电平的到来；
                                        end
                        end
        end

        reg[15:0]  dis;
        assign  distance=dis;//输出距离检测值

//脉冲进入高电平，一个周期相当于走了 1.034cm
        always  @(posedge  pulse,posedge  rst)
        begin
                if(rst)
                        dis<=0;//rst 复位则 dis 清零重新计数
                else
                        dis<=dis+1;//1.034
        end
   endmodule
```

2.11 计时子函数

```
module  time(
      input  clk,
      input  rst,
      input  [31:0]  lin,
      output  [31:0]  tim_reg,
      output  [31:0]  tim
      );

      reg[31:0]  t;
      assign  tim=t;//实时时间
      reg[31:0]  tim_stop=0;
      always@(posedge  clk,negedge  rst)
            begin
                  if(rst)
                        t<=0;
                  else
                        t<=t+1;
            end
      //计数加一，表示时间在计时
      assign  tim_reg=tim_stop;//截至总时间
      always  @(t,lin)
            begin
                  if(lin==11)
                        tim_stop<=t;
            end
endmodule
```

2.12 超声波控制函数

2.12.1 检测电路

```
   `timescale  10ns/1ps
module  ultrasound  (
      input  clk,
      input  rst,
      input  usound,//超声波信号经过放大电路的输入信号
      output  usclk,//超声波发射信号的输出管脚
      output  [15:0]  wid,//超声波输出信号的宽度
      output  out,//蜂鸣器控制信号
      output  pwmc//电机控制信号
);
```

```
        integer  div=2500;
        integer  div1=100;

//分频生成 40Mhz 方波
        clk_divider  us(
        .clk(clk),
        .d_val(div),
        .d_clk(usclk)
);

//分频生成 1MHz 方波
        clk_divider  us2(
        .clk(clk),
        .d_val(div1),
        .d_clk(usclock1)
);

        wire  [15:0]  dis;
//计算每一个方波的宽度
        pulse_wid  uswid1(
        .pulse(usound),
        .rst(rst),
        .clk(usclock1),//输入的是 1MHz 信号
        .distance(dis),
        .pul_wid(wid)                        //ms
        );

        reg  [15:0]  out1;
        assign  out  =  out1;
        reg  pwmc1;
        assign  pwmc=pwmc1;
//检测到 5us 脉宽，输出一个信号让蜂鸣器报警
//蜂鸣器输出 30mv 有点小，驱动不了
        always  @(*)
        begin
                if(rst)
                begin
                      out1  <=  0;
                end
                if(wid>=5)
                begin
                        out1  <=  1;
                        pwmc1  <=  0;
                end
```

```
                else
                begin
                        out1  <=  0;
                        pwmc1  <=  1;
                end
        end
    endmodule
```

2.12.2 报警电路

检测电路有输出时，报警电路输出 1kHz 方波给报警电路

```
    module  out1k  (
        input  clk,
        input  out,
        output  drive//1k 方波
);

        reg  [32:1]  cnt=0;//计数器
        reg  fp=1'b0;//输出初始化为 0;
        assign  drive=fp;//输出产生的是一个 clk 时钟信号！！！

always  @(posedge  clk)
if(out)
        begin
                if(cnt==((100000/2)-1))
                        begin
                                cnt<=0;
                                fp=~fp;
                        end
                else
                        cnt<=cnt+1;
        end
else
        fp  <=  0;
    endmodule
```

2.13 语音控制子函数

```
    module  soundctrl  (
        input  clk,
        input  rst,
        input  soundin,//声音控制输入信号
        output  pwmc2//控制 pwm 输出的开关
```

```
);
        reg  pwm_c2=0;
        assign  pwmc2=pwm_c2;
always  @(posedge  soundin)
begin
        if(rst)
                pwm_c2  <=0;
        else
                pwm_c2  <=  1;

end
   endmodule
```

2.14 温湿度检测子函数

```
   module  humidity  (
        input  clk,//工作频率 1kHz
        input  rst,
        input  humidity,//湿度输出信号
        output  [15:0]  hum_wid,//湿度方波的宽度反应阻值
        output  [15:0]  hum_r//湿敏电阻阻值反应相对湿度
);
        wire  [15:0]  dis;

        pulse_wid  hum1(
        .pulse(humidity),
        .rst(rst),
        .clk(clk),//输入的是 1MHz 信号
        .distance(dis),
        .pul_wid(hum_wid)                    //ms
        );

        reg[15:0]  r;
        assign  hum_r=r;
        //量化湿度百分比
        always  @(posedge  clk  )
        begin
        if(hum_wid<=8)
                r<=30;
        else  if(hum_wid>=8  ||  hum_wid<=10)
                r<=31;
        else  if(hum_wid>=10  ||  hum_wid<=12)
                r<=32;
```

```
        else  if(hum_wid>=12  ||  hum_wid<=14)
                r<=33;
        else  if(hum_wid>=14  ||  hum_wid<=16)
                r<=34;
        else  if(hum_wid>=16  ||  hum_wid<=18)
                r<=35;
        else  if(hum_wid>=18  ||  hum_wid<=20)
                r<=36;
        else  if(hum_wid>=20  ||  hum_wid<=22)
                r<=37;
        else  if(hum_wid>=22  ||  hum_wid<=24)
                r<=38;
        else  if(hum_wid>=24  ||  hum_wid<=26)
                r<=39;
        else  if(hum_wid>=26  ||  hum_wid<=100)
                r<=40;
        else
                r<=60;
        end

endmodule
```

3.约束文件

```
##数码管段选位选
set_property  PACKAGE_PIN  W4  [get_ports  {bit[3]}]
set_property  PACKAGE_PIN  V4  [get_ports  {bit[2]}]
set_property  PACKAGE_PIN  U4  [get_ports  {bit[1]}]
set_property  PACKAGE_PIN  U2  [get_ports  {bit[0]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {bit[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {bit[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {bit[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {bit[0]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[7]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[6]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[5]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[4]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {seg[0]}]
set_property  PACKAGE_PIN  V7  [get_ports  {seg[7]}]
set_property  PACKAGE_PIN  W7  [get_ports  {seg[6]}]
```

```
set_property  PACKAGE_PIN  W6  [get_ports  {seg[5]}]
set_property  PACKAGE_PIN  U8  [get_ports  {seg[4]}]
set_property  PACKAGE_PIN  V8  [get_ports  {seg[3]}]
set_property  PACKAGE_PIN  U5  [get_ports  {seg[2]}]
set_property  PACKAGE_PIN  V5  [get_ports  {seg[1]}]
set_property  PACKAGE_PIN  U7  [get_ports  {seg[0]}]
##l298N 输入 IN 四个信号
set_property  PACKAGE_PIN  H2  [get_ports  {dir[0]}]
set_property  PACKAGE_PIN  K2  [get_ports  {dir[1]}]
set_property  PACKAGE_PIN  J2  [get_ports  {dir[2]}]
set_property  PACKAGE_PIN  L2  [get_ports  {dir[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {dir[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {dir[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {dir[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {dir[0]}]
##时钟及采样信号
set_property  PACKAGE_PIN  W5  [get_ports  clk]
set_property  IOSTANDARD  LVCMOS33  [get_ports  clk]
set_property  PACKAGE_PIN  G2  [get_ports  dis]
set_property  IOSTANDARD  LVCMOS33  [get_ports  dis]
set_property  PACKAGE_PIN  A14  [get_ports  line]
set_property  IOSTANDARD  LVCMOS33  [get_ports  line]
##pwm 左右轮输出开关
set_property  PACKAGE_PIN  J1  [get_ports  ENA_l]
set_property  IOSTANDARD  LVCMOS33  [get_ports  ENA_l]
set_property  PACKAGE_PIN  H1  [get_ports  ENB_r]
set_property  IOSTANDARD  LVCMOS33  [get_ports  ENB_r]
##复位信号
set_property  PACKAGE_PIN  T18  [get_ports  rst]
set_property  IOSTANDARD  LVCMOS33  [get_ports  rst]
set_property  PACKAGE_PIN  V17  [get_ports  sw]
set_property  IOSTANDARD  LVCMOS33  [get_ports  sw]
set_property  PACKAGE_PIN  U18  [get_ports  menu]
set_property  IOSTANDARD  LVCMOS33  [get_ports  menu]
##菜单选择信号
set_property  PACKAGE_PIN  L1  [get_ports  {menu_show[3]}]
set_property  PACKAGE_PIN  P1  [get_ports  {menu_show[2]}]
set_property  PACKAGE_PIN  N3  [get_ports  {menu_show[1]}]
set_property  PACKAGE_PIN  P3  [get_ports  {menu_show[0]}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {menu_show[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {menu_show[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {menu_show[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {menu_show[0]}]
```

```
##左右轮占空比调节量
set_property  PACKAGE_PIN  U1  [get_ports  {sw_r_adjust[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_r_adjust[4]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_r_adjust[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_r_adjust[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_r_adjust[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_r_adjust[0]}]
set_property  PACKAGE_PIN  R2  [get_ports  {sw_r_adjust[0]}]
set_property  PACKAGE_PIN  T1  [get_ports  {sw_r_adjust[1]}]
set_property  PACKAGE_PIN  W2  [get_ports  {sw_r_adjust[3]}]
set_property  PACKAGE_PIN  R3  [get_ports  {sw_r_adjust[4]}]
set_property  PACKAGE_PIN  T2  [get_ports  {sw_l_adjust[0]}]
set_property  PACKAGE_PIN  T3  [get_ports  {sw_l_adjust[1]}]
set_property  PACKAGE_PIN  V2  [get_ports  {sw_l_adjust[2]}]
set_property  PACKAGE_PIN  W13  [get_ports  {sw_l_adjust[3]}]
set_property  PACKAGE_PIN  W14  [get_ports  {sw_l_adjust[4]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_l_adjust[4]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_l_adjust[3]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_l_adjust[2]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_l_adjust[1]}]
set_property  IOSTANDARD  LVCMOS33  [get_ports  {sw_l_adjust[0]}]
#超声波信号的输入
set_property  PACKAGE_PIN  A16  [get_ports  {usound}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {usound}]
set_property  PACKAGE_PIN  B15  [get_ports  {usclk}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {usclk}]
set_property  PACKAGE_PIN  B16  [get_ports  {out}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {out}]
##湿度检测输入
set_property  PACKAGE_PIN  A15  [get_ports  {humidity}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {humidity}]
##语音控制
set_property  PACKAGE_PIN  K17  [get_ports  {soundin}]

set_property  IOSTANDARD  LVCMOS33  [get_ports  {soundin}]
```

四、电路的组装、编程和调试

1. 使用的主要仪器和仪表

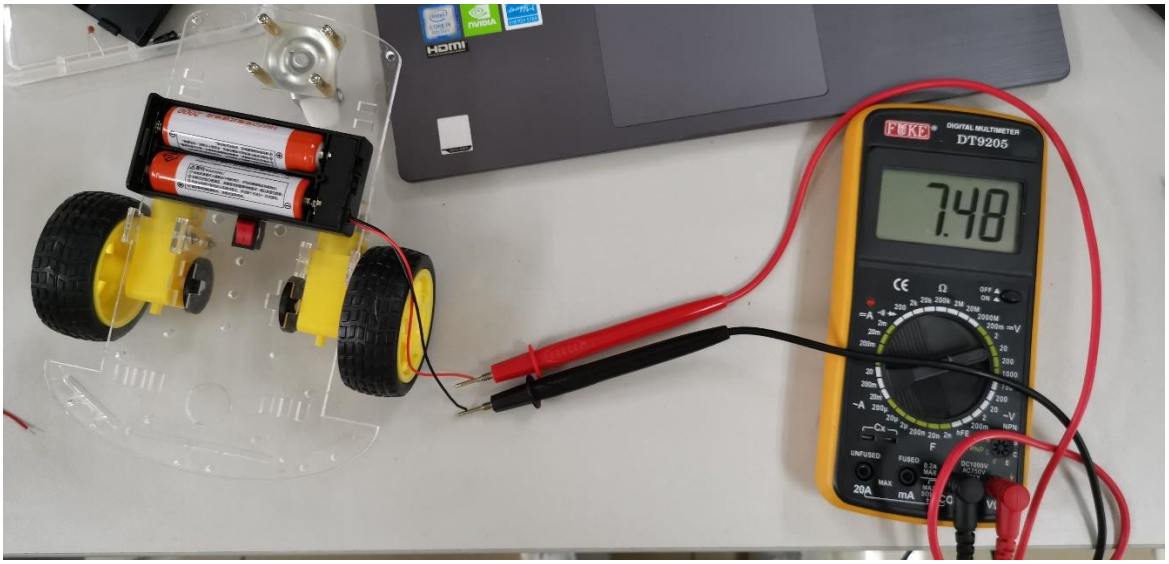
使用的主要仪器和仪表有信号发生器、示波器、数字万用表、稳压电源、热熔胶枪、电烙铁、烙铁架、螺丝刀、剥线钳、面包板等。

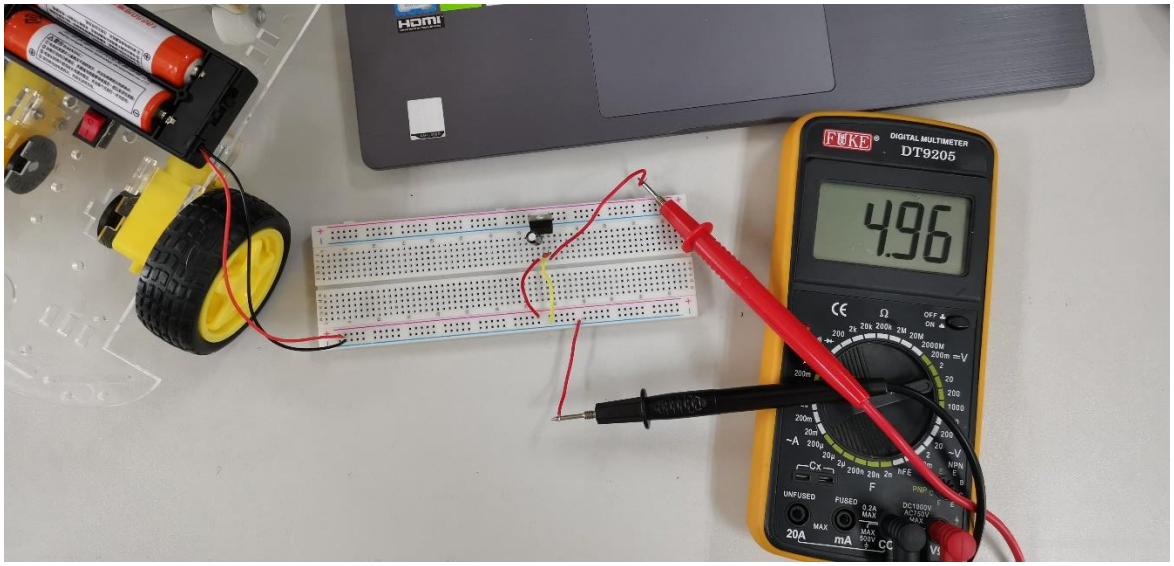
2. 调试电路的方法和技巧

(1) 仿真结束后，在实际焊接之前一定要在面包板上测试实际效果，否则焊接错误再焊很耗时。
(2) 在洞洞板上焊接电路要严谨的控制走线贴板，否则走线杂乱，调试电路检验错误都会非常费力。
(3) 各元器件精密度足够，再调试阈值电压、电路参数时一点点拧即可。
(4) 不要一出现问题就怀疑元件坏了，要耐心检查每一点的电压跟仿真结果的差别，细心找寻错误。
(5) 不同类型的管口用不同颜色的线，比如输入用绿色，输出用白色，接地用黑色，接 vcc 用红色等。

3. 测试数据的整理与结果分析

3.1 稳压模块

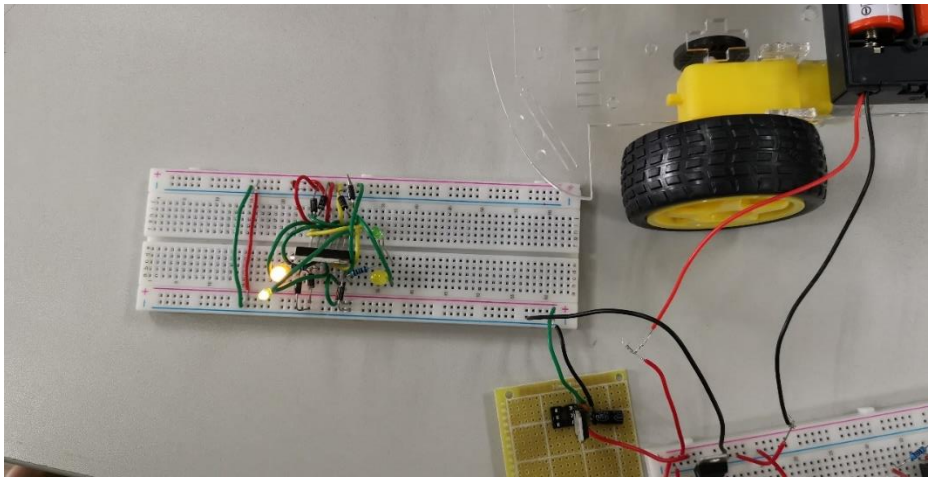




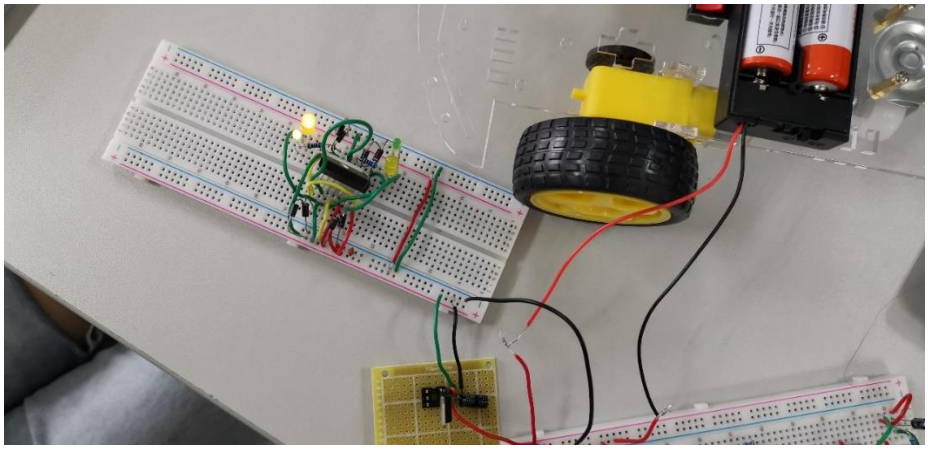
调试时，电源电压 7.48v，稳压输出 4.96v。7805 芯片有点烫手，需要后期外加散热片。

3.2 电机驱动模块

1010 正转时，一边（靠近红绿线）二极管亮

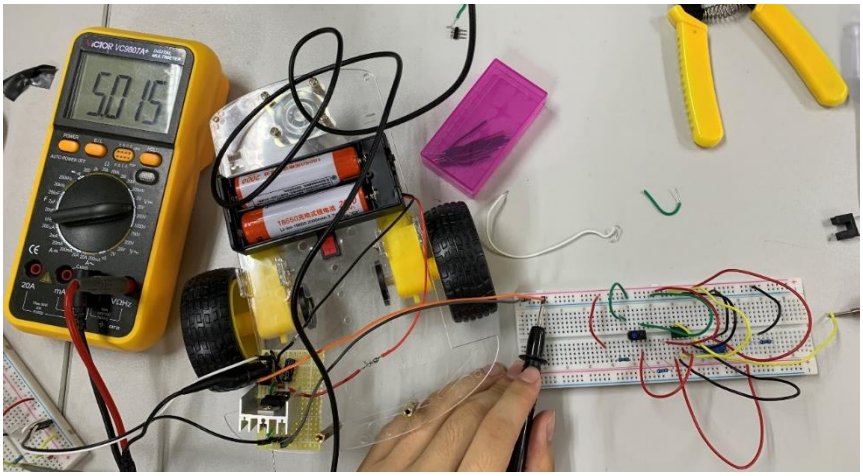


0101 反转时，另一边（远离红绿线）二极管亮

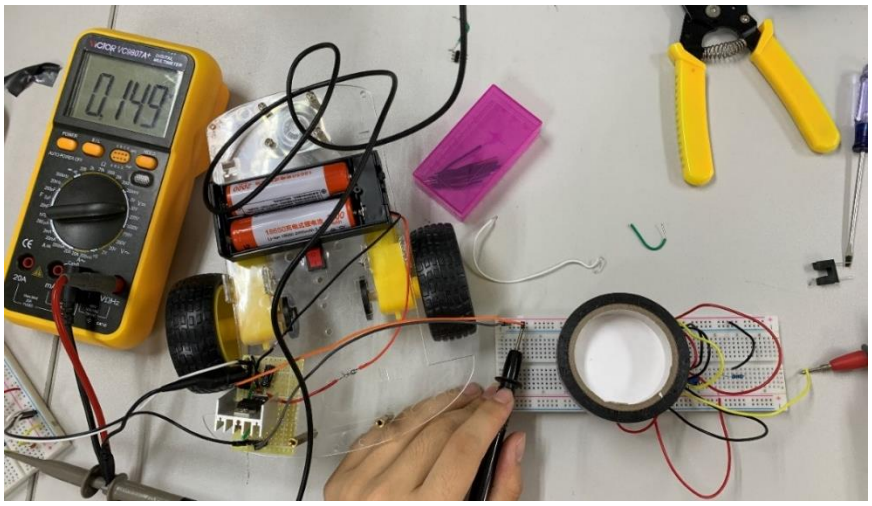


3.3 黑线检测模块

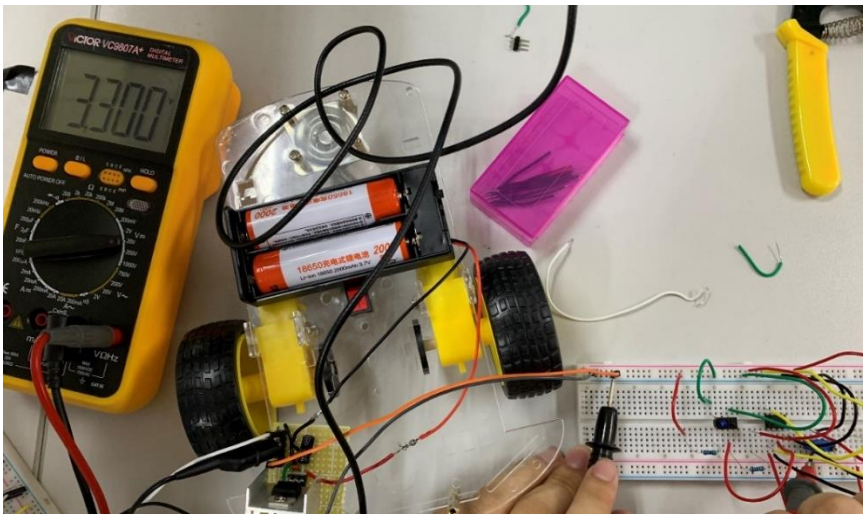
无黑线遮挡输出：



有黑线遮挡输出：



阈值电压：

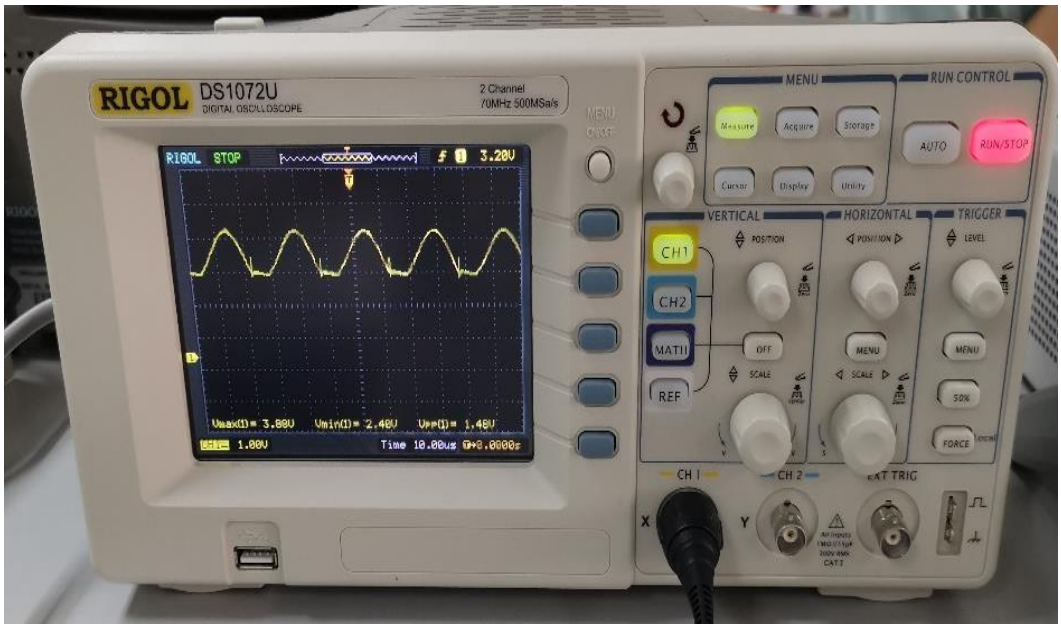


结果与理论和仿真结果很贴合，效果比较好

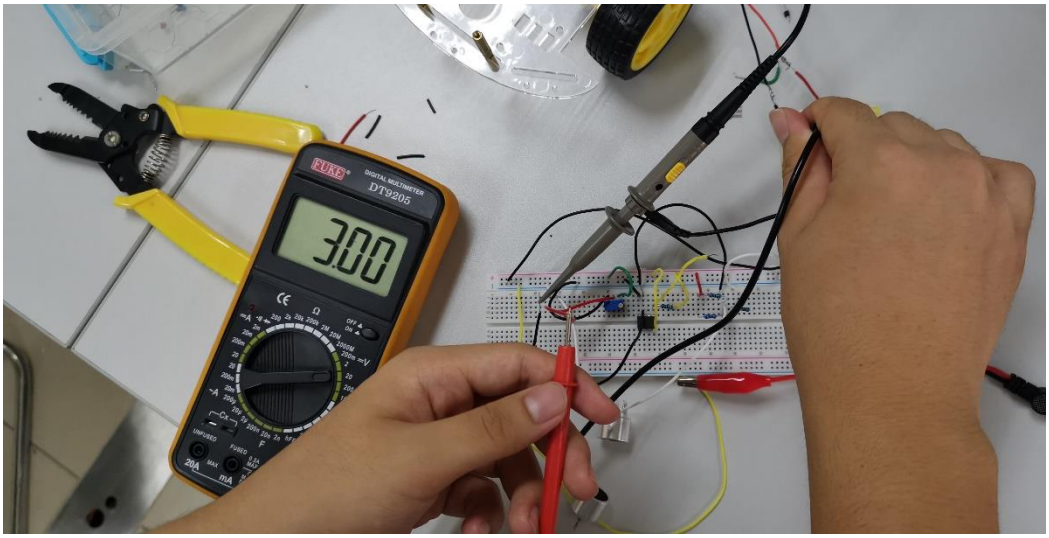
3.4 超声波避障电路：

Basys3 开发板 IO 接口输出检测电路 40kHz 方波、报警电路 1Khz 方波（外接上拉电阻）时：

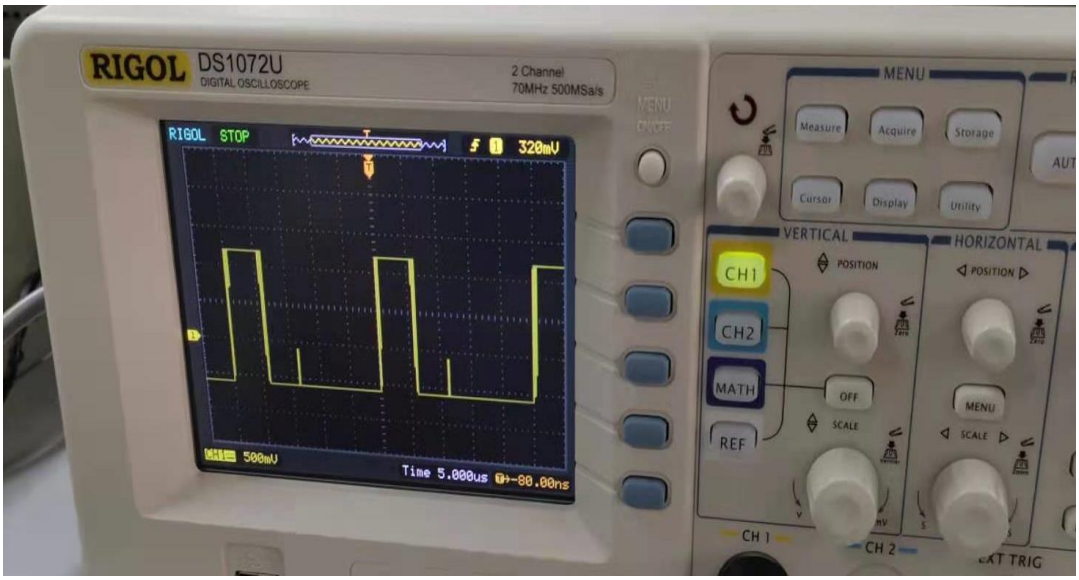
接收端波形为类正弦波：



阈值电压设置为 3v 时



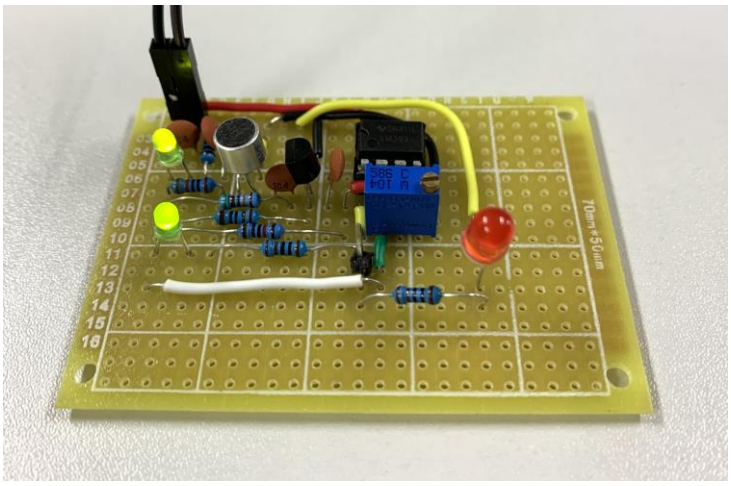
有障碍物遮挡时，输出端波形为比较标准的方波，脉宽为 5us：



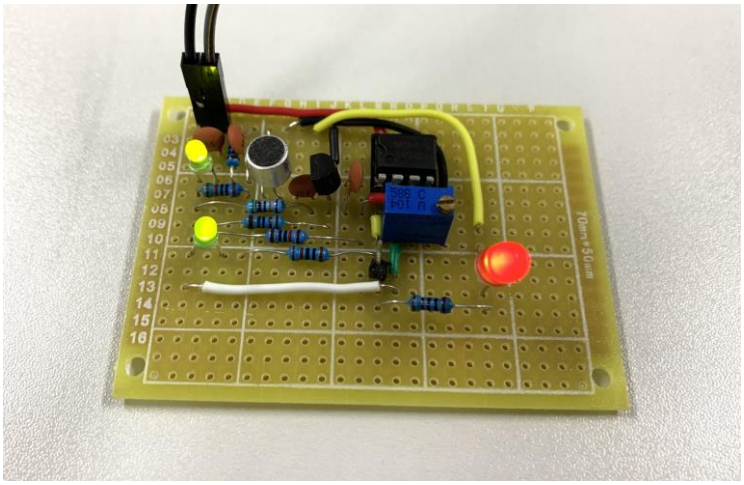
这跟仿真时的结果比较类似，实际效果比较好

3.5 声控模块

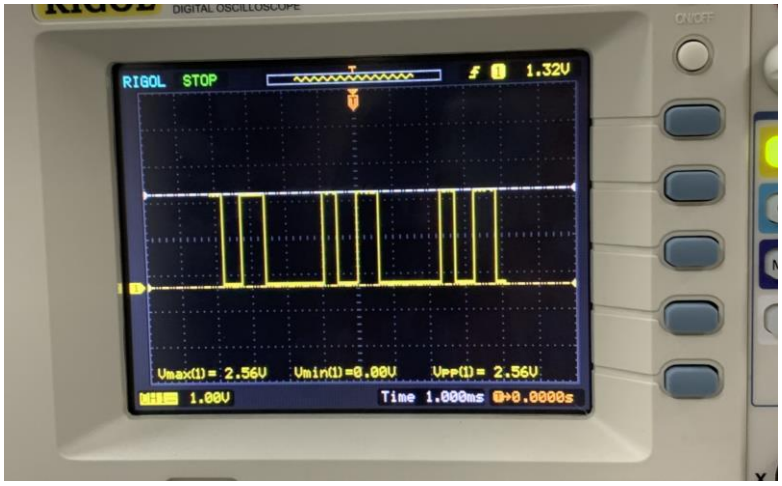
无声环境下，声控灯（红色）不亮，无输出高电平；



有声环境下，声控灯亮，有高电平输出。

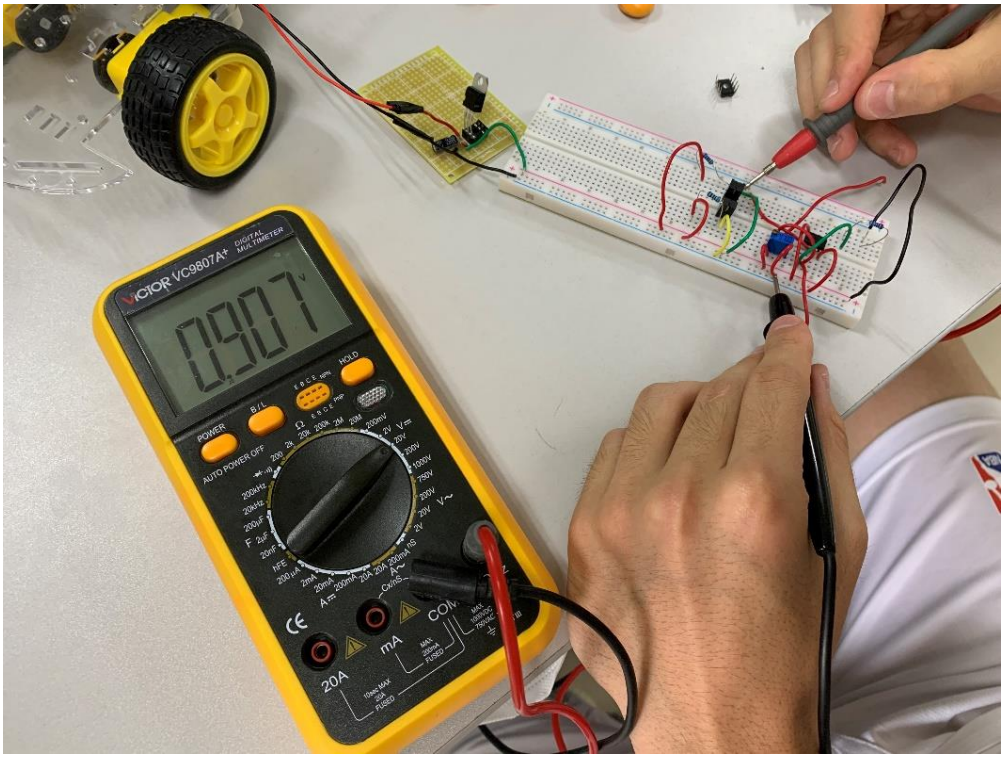


输出波形：

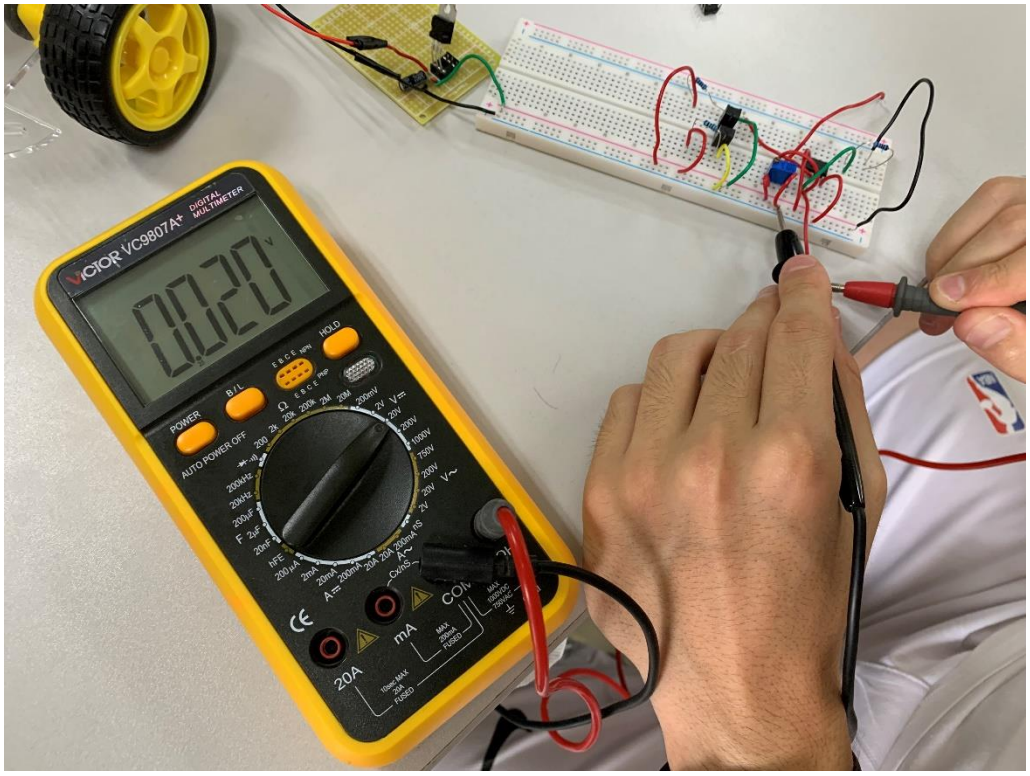


3.6 速度检测模块：

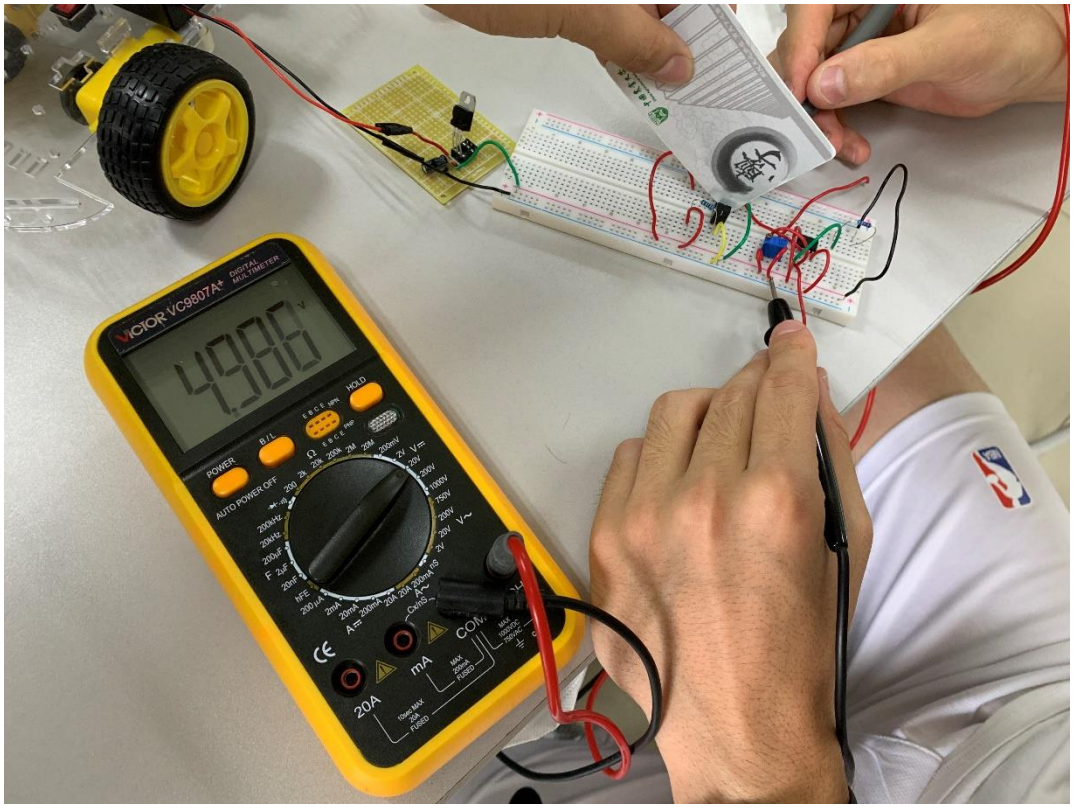
无遮挡时，光耦三极管输出：



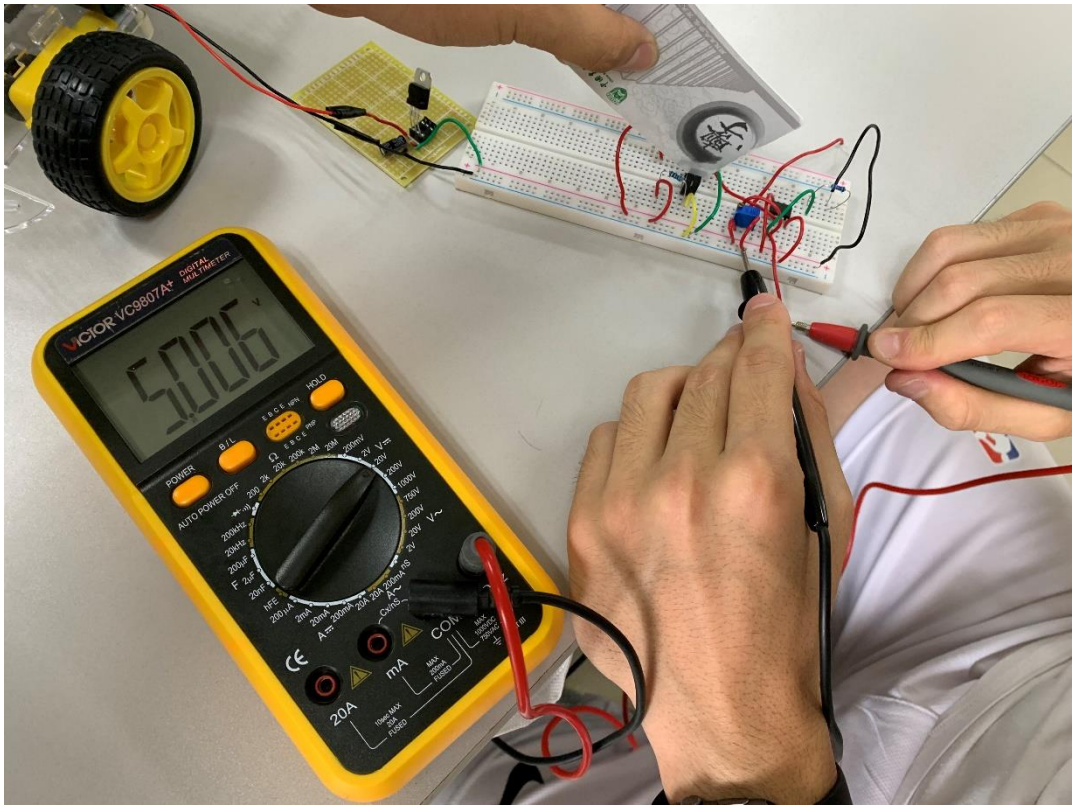
经过电压比较器后输出低电平



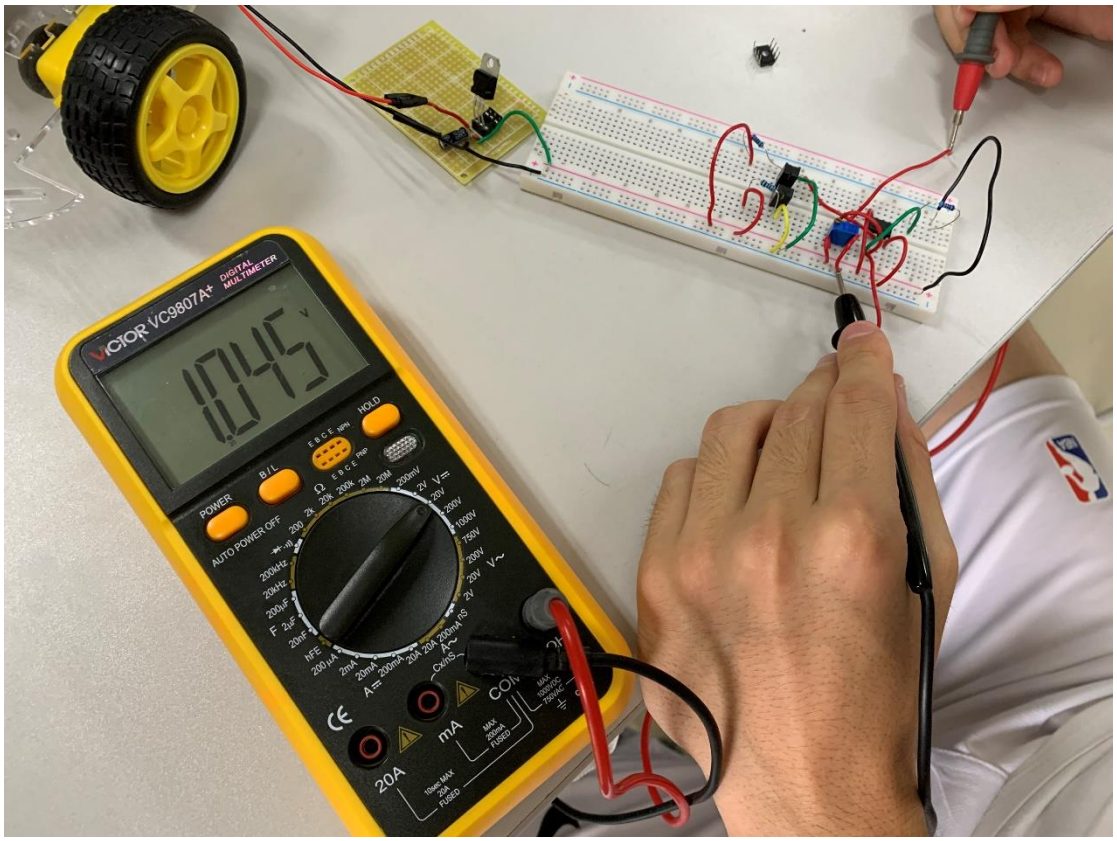
遮挡时，光耦三极管输出：



有遮挡时，经过电压比较器，输出：


VICTOR VC9807A+
DIGITAL MULTIMETER
5.006
POWER
AUTO POWER OFF
HOLD
20A
mA
COM

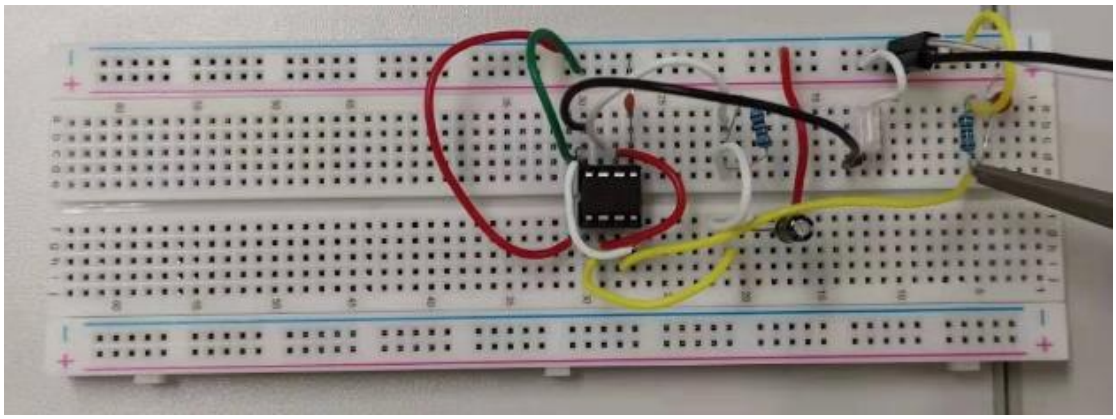
设置的阈值电压：


VICTOR VC9807A+
DIGITAL MULTIMETER
1.045
POWER
AUTO POWER OFF
HOLD
20A
mA
COM

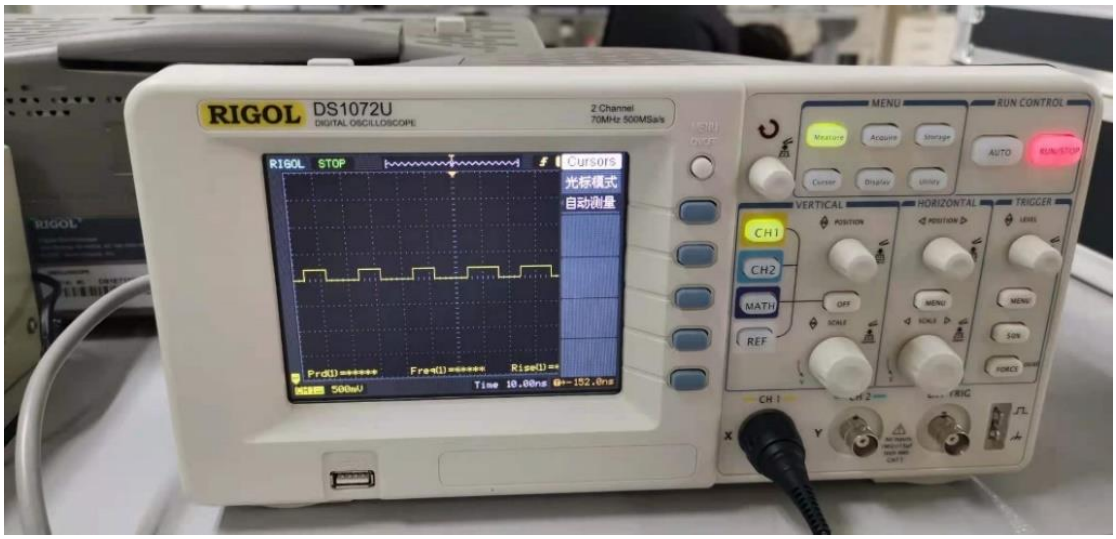
在码盘上断连处转动过程中，实际出现的波形也是高低电平交替的方波，效果很好。

3.7 温湿度检测模块

测试电路：



输出的波形：在改变温湿度后，可以看出起占空比发生了变化



实际效果与仿真预测比较接近，效果较好。

4. 调试中出现的问题、原因和排除方法

(1) 编程过程中出现逻辑错误。

原因是：verilog 语言中||、&&双写是逻辑运算，单写|、&是位运算，与一般编程中不太一样。

排除方法：进行逻辑运算时双写&&、||

(2) 数码管显示错乱

原因是：没有弄清数码管接线，八段数码管段由低到高位为（abcdefgh），h为小数点位，也有可能是数码管频率过高。

排除方法：按数码管每一根管子与每一位对应接，将数码管频率设置在人眼可视范围左右，位选信号扫描频率大概在1000hz左右，显示正常

(3) 断电之后，basys3再供电没有程序

原因是：烧写bit流文件断电后擦除

解决方法：烧写bin文件

(4) Vivado综合时出现[Synth 8-91] ambiguous clock in event control

原因是：错误定位在这个always块，我们通常习惯边缘触发，时钟上升沿和复位下降沿写到一起，但是，在always块里面并没有初始化的一些变量，也就是我们并没有用到rst_n，综合的时候就会报出上面的错误。

解决方法：将触发模式更改为always@(posedge clk)即可

(5) 电机转速问题

原因：有多个方面，有可能时电源电压不够，导致稳压输出低于5v了；也有可能是pwm频率的问题，pwm频率太低时，电机不能带动车轮转动，只是震动。

解决方法：电池充好电，pwm输出设置的频率要适当高一点。

(6) 编程过程中报错

原因是：中英文标点、语句后少分号等

解决方法：按照vivado综合后Message栏的指示对应更改即可。

五、总结收获、存在的问题和进一步的改进意见（包括课程目标的达成情况）

1. 总结收获：

怎么说呢，一分耕耘一份收获，考完期末考试就跟组员们投入到了小学期的准备中，画电路、想方案、尝试着写代码。每一次合作后的成果都会增强你的信心和期望，同时也会增进友谊，总体说来还是大学到目前为止最有收获和感触的体验了。

走进实验室，一天半的时间焊完所有模块的电路，检测完毕，看似顺利，但真正的挑战还在后头。因为没有发现电池盒的问题，一直怀疑是代码逻辑的问题，

在红外检测黑线上花了将近三天的工夫，好在崩溃的边缘，朋友们的互相支持与鼓励，让我们顺利解决了这个问题。

后面几天，属于提高部分的发挥，因为这部分没有代码参考，所以基本都是自己按照逻辑自己编写的，写完虽然有错，但是都经过调试得到了解决。当看到超声波电路每个点的波形跟仿真完全一致时，那种一层一层破冰的感觉，真的很像是在绿茵场上行云流水的配合后球飞入球网的感觉。同时，经过速成 verilog 后自己能看懂代码到能编写出完整的程序，这个过程也是对代码能力的一种提升，我感觉学习语言这个过程，还是要参照例程，在与同学的讨论中才会发现问题，增加收获。给别人讲解代码的实现，就像给别人讲解习题一样，一举两得。

稍有遗憾是最后有点着急，看着大家都交了，也确实有点累。语音控制的 pwm_ctrl_sound 和分控 pwm_ctrl 跟总控制 pwm_ctrl_sum 的逻辑当时用脑子想以为单纯是个或的关系，后来回来在纸上写才知道是个除 00 外都是 1 的逻辑，这块当时如果写一下会更好，好脑子不如烂笔头啊。

总的来说，收获满满，不仅在实际中理解了控制，还能动手编程改变控制的方式，真的非常有成就感。同时，虽然感觉平时成绩还凑合，但是实际操作中可能真的有不如别的同学理解精妙的地方，在交流中认识到了别人身上的有点和别人努力的态度，这让我能更安静下来反思自己的不足，这种体验也是平时上课体验不到的。在实验闲下来时，大家在一起聊天，谈起过往、谈起课堂，青春仿佛就在眼前，触手可及，那种逃脱学习压力的解脱感，让人觉得温暖与感动。

看到所有人走出实验室，遍地都是线头，器件凌乱，我们三个人和冯老师把教室从头到尾收拾干净，仪器摆放整齐。上一次这样的扫除还是在高考前，熟悉的感觉，不同的人，真的很有成就感，感谢小学期的相遇，未来继续努力。

2. 课程目标的达成情况、存在的问题和进一步改进意见

课程目标的达成情况，基础功能全部实现，提高部分除了语音控制那块代码没写完整（后来回来写完了，编译也没啥问题），最后着急交，只实现了语音控制小灯闪烁，其他的提高部分基本都实现出来了。

存在的问题和改进意见如下：

那个小车由于只有三个轮子中的两个驱动，所以不是能跑得特别直，当时想设计反馈来形成闭环控制，但是最终没有实现，之后 urp 用 python 编程的时候会尝试着试试能不能自己实现这个闭环控制过程。

温湿度检测模块最后着急只用了个热敏电阻显示，没用 18b20 和 lm35dz，当时觉得这俩太难了，后来看看代码感觉太方便了，如果下次再选温度这块一定用 lm35dz。

语音控制那个逻辑没写完，导致小车只能语音控制跑，不能语音控制停，后来改了逻辑，编译对了，没来得及烧进去，但是有自信那个肯定能实现出来，之后会在自己的小车上试试这个语音控制功能。

六、参考文献和资料

[1]. 智能图示仪测 7805 的 Vi-Vo 曲线——http://www.crystalradio.cn/thread-171961-1-1.html

[2]. 电赛模拟训练日志总结一（直流电机测速装置项目）——https://blog.csdn.net/weixin_43184208/article/details/89680023?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216257304181678026254385 2%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=162573041816780262543852&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_v2~rank_v29-13-89680023.pc_search_result_cache&utm_term=%E7%94%B5%E6%9C%BA%E6%B5%8B%E9%80%9F&spm=1018.2226.3001.4187

[3]. LM35DZ 摄氏温度传感器构成温度量电路设计与分析

[4]. 【常用传感器】DS18B20 温度传感器原理详解及例程代码——CSDN

[5]刘允峰.基于 Multisim12 的电机驱动模块设计与仿真[J].电子设计工程,2014,22(08):153-155.

[6]朱春华,顾雪亮.基于红外反射式传感器 TCRT5000 的循迹小车设计[J].现代电子技术,2018,41(18):143-146.

[7]杨伟光, 基于湿敏电阻实现湿度测量电路的设计,中北大学.