

2019.07.01 上午

电子设计小学期工作日的第一个上午，首先我们较为顺利地通过了预习验收，鼓舞了项目开始时的士气。

此外，在等待验收的前前后后的过程中，我们主要使用W3Cschool 的 arduino 教程，对我们所选的主控模块进行了简单的上手热身。主要了解了其整体的程序语法，控制流，IO 功能和串口通信调试功能。

接近上午调试结束时，我们还盘点了已有的一些模块。我们现有的模块有 LCD 显示屏，蓝牙通信模块，基本可以实现数字部分的功能。而模拟部分的模块，大部分传感器仍在配送，电源管理模块可先根据已有的备选芯片进行一定的调试。故而我们敲定了之后的计划，按照电源管理、arduino 并行的方法进行调试。而 LCD 的调试相蓝牙模块调试而言比较简单，故先进行调试；并且另一路对电源管理模块调试的结束后，可以分人手去提前学习一下蓝牙模块的使用。得到近期调试优先度大纲如下：

1. arduino,LCD, 串口联调；电源管理模块参数测试
2. 蓝牙模块学习调试。
3. 传感器模块的参数调试与联调。
4. 其他基于分立元件（如光敏电阻）的外围传感电路设计
5. 写数字系统整体代码框架

2019.07.01 下午

在经过上午的验收以及上手热身后，我们在下午正式开启了设计与调试，由于大多数传感器还没有送达，我们手中已有的模块是 arduino uno 主控模块和 LCD1602 液晶显示模块，为了减少 IO 的使用，我们特地前往中发电子大厦购买了 I2C 转接板，将 16 引脚方便的减少为 4 引脚和 arduino 相连接。

在购置回转接板后，我们一方面开始学习 LCD1602 与 arduino 的硬件连接方法，以及其各个引脚的说明，并且在利用已有的 LiquidCrystal 库函数的情况下，尝试进行了字符数据的显示，成果如下：



图 1: LCD 字符显示

起初并不能显示，后来我们很快发现是转接板电位器的问题，转接板电位器直接控制了 LCD 显示的亮度，因此在使用镊子改变电位到合适的亮度后便能观察到字符。在能够显示字符后，我们进一步结合上午的学习进行了串口 LCD 通信联调，使得在键盘上实时输入字符在 LCD 上进行显示，这是我们之后显示模块的重要基础，我们拍摄成果的照片如下：

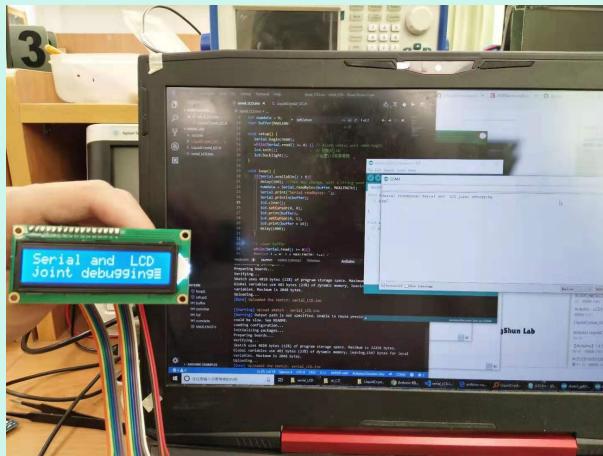


图 2: LCD 串口通信联调

另一方面，我们组在调试 LCD 的同时，对电源管理电路进行了实际的检测，我们计划使用 9V 的干电池，而恰好在实验室中找到一块电源管理的模块，能够在小于 12V 输入的情况下，输出 5V/3.3V 的直流电压，我们类比于电网 10% 的波动，使用 8V ~ 10V 的 50Hz 正弦波作为输出，观察两输出的电压情况，结果十分令人满意，根据示波器的显示，以及自动测量的结果，能够得到纹波非常小的直流电压，并十分接近其标称的输出，记录如下图所示：

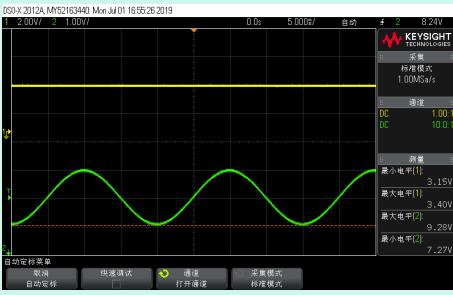


图 3: 3.3V 稳压输出

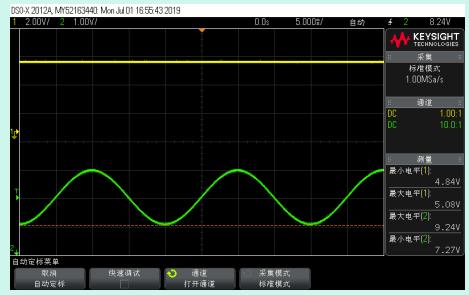


图 4: 5V 稳压输出

其中黄色线为稳压输出，绿色线为输入，验证了该电源管理模块能够提供理想的供电电压，在 TI 公司的样片到来之前为我们的电源管理提供了替代。

在下午收工以后我们另外找到一片蓝牙模块，计划于明天开启蓝牙模块的调试以及 DHT11 的湿度模块的调试，并通过 LiquidCrystal 库编写代码，实现自己需要的函数的头文件。

2019.07.02 上午 DHT11 温湿模块与蓝牙模块调试

电子设计小学期工作日的第二个上午，我们明显加快了调试的进度，在昨天初步调试 LCD1602 后，今天我们进一步同时开始调试蓝牙模块和 DHT11 温湿模块。

首先，上午蓝牙模块的调试并不是十分顺利，中间遇到了一些连接的问题，进度稍慢，而温度湿度传感器模块借助于 Arduino 官网可查找的 dht11 的库，可以很快的进行实现。我们设置每两秒更新并发送一次数据，可以在串口接收到数据。之后便很方便地将数据

为了验证其正确性，我们将传感器分别在室内与室外进行了测量，结果表明室外比室内温度大约高 3°C 左右，湿度也稍高于室内，符合实际情况，测量结果如下图5记录：

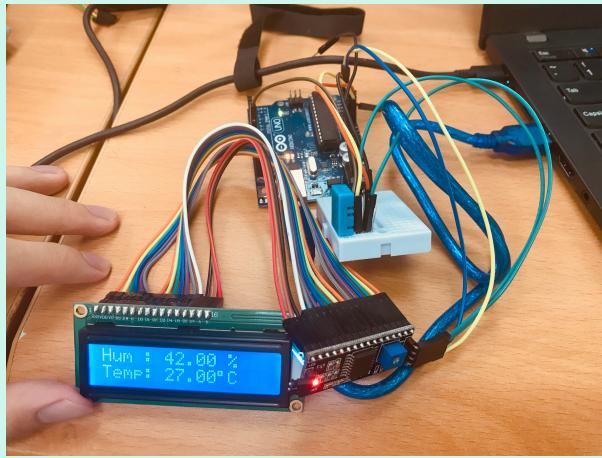


图 5: 温度湿度测量与 LCD 显示

该 LCD 显示了当前教室的室温和室内湿度。整体上温度湿度模块由于能够直接输出校准后的数字信号，所以调试过程比较顺利，只是在单位的输出时 $^{\circ}\text{C}$ 的符号难以直接输

出，库中的 `print()` 函数无法输出该字符，需要去自定义字符，否则会输出响应的日文，因此我最终采用了 `print((char)233)` 在其中一格的 5×7 矩阵先输出“度”的上标，再输出 C 来完成单位的显示。

此外，我们修改了 `LiquidCrystal_I2C` 库的内容，将其精简并增加了和串口联调的功能，成为我们可以使用的 `serial_lcd` 库，其中的函数实现原理大致与官方库相同。此后该模块恰好能够利用该屏幕进行显示，因此我们考虑了一下多模块测量的显示方法，我们选用 TTP226 电容触摸开关进行显示内容的选择，不同的开关控制不同内容的显示，因此我们目前手头只有 TTP224，并用其进行测试其触摸效果，结果十分令人满意。

而另一模块，蓝牙模块的调试，能够初步和设备进行连接，但基于 Arduino 进行数据通讯仍具有一定的问题，有待下午进一步调试。

2019.07.02 下午 蓝牙模块调试和系统架构细节讨论

在下午我们完成了蓝牙模块的调试，经过排查，发现是使用的蓝牙模块已经进行了配置，和出厂设置不同，厂方给出的设置方案不能直接使用。在简单了解 AT 指令集后，我们迂回使用 USB 转 TTL 的芯片，先对蓝牙模块进行恢复出厂设置，便可按照厂方文档进行调试。按照应用情景，我们将蓝牙模块设置为从机，波特率设置为 9600（之后可能会调节到更低以节省功耗），简单测试了手机通过蓝牙模块、电脑通过蓝牙模块与 arduino 的通信，顺利完成数据的收发并能做出简单的响应，可以预料到能与后续的模块完成连接。

此外，气压传感器 BMP180 的特性测试和简单实用调试也在今天下午完成。此外在考虑微波雷达探测人体存在的实际情况，我们认为可能需要做一定的角度的扫描检测，故而在实验末尾简单阅读了舵机的工作原理和操作例程，这是一个控制模块，可以预料到调试环节不一定顺利。但是这是一个数字驱动的模块，可以在今晚先将大部分功能点实现，其他的预想功能如根据雷达信号的反旋转舵机，需要之后联调实现。

最后，已经看到的情况是，由于外设较多，arduino 的 IO 资源已经有些紧张，而且各外设的使用方式不一，有直接用 IO 功能进行读写，有用 I2C 进行通信。我们敲定方案是用 IO 转 I2C 和 I2C 扩展板，实现一个统一的数据总线结构，方便整体系统调试。另外我们还需要一个数据存储媒介给仪器记录实验数据。所以在今天下午实验结束后，我们还去中关村中发市场进行了相关器件的采购。

2019.07.03 上午 舵机、SD 卡、LCD12864、红外收发调试

电子设计小学期工作日的第三个上午，我们继续对已有的模块进行测试。我们在昨天的采购时重新购置了一块较大的液晶显示屏，型号为 LCD12864，是一块分辨率为 128*64 的显示屏，并带有中文字库，相比与 1602 具有更大的优势，首先我们先去将其 20 引脚焊好排针，之后我们利用了已有的一个简单库函数，并且对其进行函数的补充与丰富，例如清除光标等函数，并对其进行了显示的测试，测试结果能够清楚的显示如下图6，但是发现该屏幕对电磁干扰非常敏感，在硬件接线有扰动或者其他电磁干扰的情况下，会出现乱码的情况，这是我们进一步需要解决的问题。

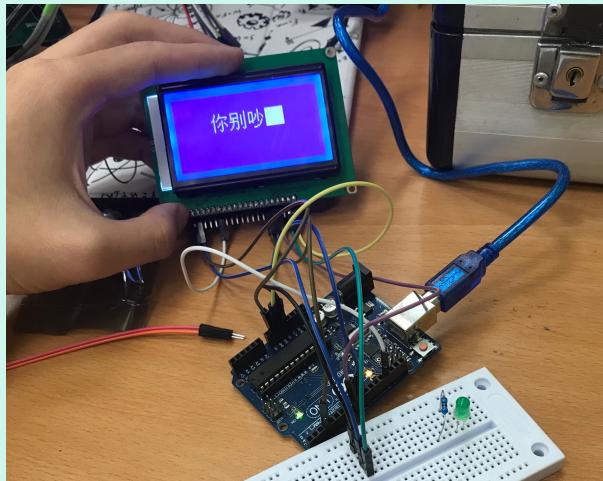


图 6: 12864 显示

除此之外，我们调试使得舵机能够正常运转，我们希望借助 180° 旋转的舵机能够帮助微波雷达模块对室内的人体进行感应与检测，扩大其检测范围。在调通舵机后，我们开始了对 SD 卡存储的调试，初步能够实现通过 Arduino 将数据发送的 SD 卡进行存储。并且在舵机模块和 SD 卡模块的类库的使用中，我们注意到从类库生成的对象较大（一个舵机大约占用动态内存的 20%、一个 SD 文件对象大约占用动态内存的 50%），故而我们讨论了两种解决方案，第一是精简类库，减少内存消耗，二是使用 PROGMEM 后缀，将变量存储在 flashrom 中。第二个解决方案用于对读写实时性的要求不高的场景，初步确认 SD 卡文件对象内存占用大的问题可以使用这个方式得以解决。这是因为将测量数据转存到 SD 卡的操作本身就是外存文件操作，速度较低。而舵机对象的问题推迟到最后联调时视情况解决。此外我们还进行了红外收发模块的调试和标定：得到编码如下：

```
//result
/*
    CH-      CH      CH+
FFA25D  FF629D  FFE21D
|<<     >>|     >|||
FF22DD  FF02FD  FFC23D
-       +       EQ
FFE01F  FFA857  FF906F
0       100+    200+
FF6897  FF9867  FFB04F
1       2       3
FF30CF  FF18E7  FF7A85
4       5       6
FF10EF  FF38C7  FF5AAS
7       8       9
FF42BD  FF4AB5  FF52AD
*/
*/
```

图 7: 红外遥控编码

并且我们进一步讨论总线的管理方法，由于 IO 资源较少，而待测量变化频率一般较低，我们希望能够在每次轮询周期内利用一根总线分别选通各传感器进行数据传输，因此我们需要搭建数据选择电路，为此我们在午饭时再次前往中发市场购置数据选择器等器件。

2019.07.03 下午 元件增购与总线结构讨论

根据课上所学的知识，我们在中发市场增购了八输入数据选择器 74151，用做输入总线的设计。此外采购了 GPIO 转 I2C 的芯片 PCF8574。I2C 在 LCD1602 调试时我们接触到的一个串行通信协议。在下午的正是调试中，我们两路并行进行，一方面对购得的元件进行逻辑测试，另一方面学习理解 I2C 协议。以下是在 PCF8574 的数据手册中截图得到的关于 I2C 的介绍。大约分为主机发送起止命令、选择地址和读写模式，以及从机被读或被写时按 8 位数据位一位应答位的串行数据传输形式。

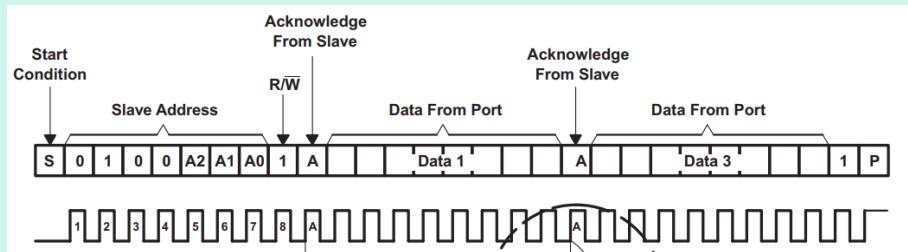


图 8: I2C 数据传输举例（从数据手册中摘录）

在简单讨论后，我们决定先试用 I2C 进行总线管理。主要出于以下几个目的：

1. I2C 适用于通信速率低的情景，在轮询传感器时，符合这个特点。
2. I2C 是相对主流的通信协议，可以方便后续拓展。
3. I2C 协议读写均支持，是更完整意义上的总线形式，比起我们用数据选择器设计的只读总线结构更为合理，也可以把读传感器值和内容显示等读写操作统一在一个时序中，让程序更为统一。
4. I2C 协议要求我们对时序电路有更好的理解，我们也想在这方面挑战一下自己。

2019.07.04 上午 I2C 总线协议理解与讨论

在网购元件到来前的最后一个上午，我们首先进一步研究 I2C 总线协议，为我们之后的总线架构做出准备与铺垫。我们阅读并参透了 LiquidCrystal 的库函数，进一步参透了解 I2C 传输协议。

除此之外，我们另一路开始为下午的模块调试进行准备，计划在下午能够完成所有模块的调试使得明天可以进行各模块的联调。首先查阅了 GP2Y1050AU0F 的模拟输出特性，以此为基础先编写好程序，并进一步连接其接线，由于我们已知其为模拟输出，自然考虑到传感器的输出会受到噪声的影响，为了改进输出特性，我们提前设计好了 VCVS 低通滤波电路，滤波特性如下：

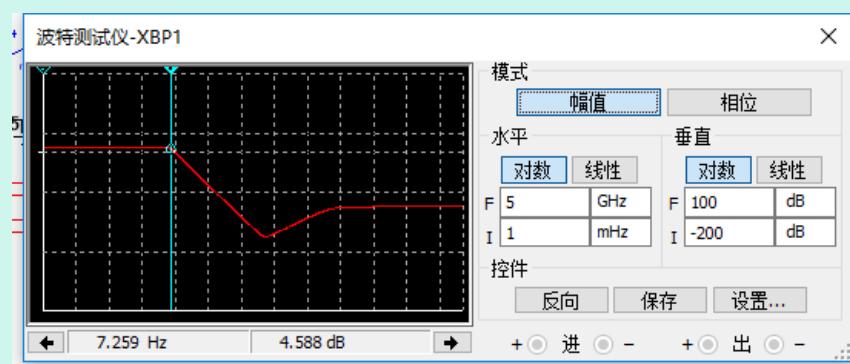


图 9: 低通滤波特性

可见我们已经能够实现低通滤波的仿真，使得该模块的模拟输出稳定，并且在之后综合比较了 LCD1602 和 LCD12864 的显示效果以及调试难度，由于 12864 的显示常常因电磁干扰而显示乱码，我们仍决定采用 1602。在上午的工作接近末尾时，我们大概完整的阅读并理解完 I2C 协议与 LCD 显示的库函数，我们在网上购置的模块也终于到来，为我们开发工作的进度有进一步推动。

2019.07.04 下午 PM2.5 传感器、微波雷达、光强测量电路的调试

下午模块一到，我们便首先对 PM2.5 检测的 GP2Y1050AU0F 传感器的特性进行标定。室内空气质量好，故传感器的模拟输出较低，且用示波器看不到明显变化。在回忆了传感器的原理，是通过对管对自己发射的光信号进行放大，而收集槽中的灰尘会部分挡光导致输出示数变化。故而我们简单用铅笔伸入收集槽的方法对传感器特性进行测试。下图为伸入铅笔后的，滤波前和滤波后的输出：

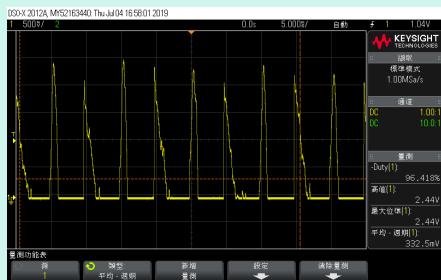


图 10: 灰尘传感器滤波前输出



图 11: 灰尘传感器滤波后输出

可以看到，滤波后输出质量明显提高。并且可以想到的是，在本次实验中的噪声主要来自铅笔的晃动，而实际测量中的噪声可能来自空气颗粒的随机运动，应该会是更大的噪声，故而进行这次滤波是很有必要的。

微波雷达的生物探测功能已经进行了验证，在所需范围内有生物移动时会得到高电平输出，并且其穿透探测的功能也得到了验证。但是由于小组中一人工作台的示波器工作异常，故没有记录详细波形。

最后是光强测量电路的设计、仿真和调试。手边拥有的光敏元件是光敏电阻，在查阅了光敏电阻的光敏特性曲线，其理想的关系应当是

$$\ln\left(\frac{R}{R_0}\right) \propto \ln\left(\frac{I}{I_0}\right)$$

最直接的想法是用分压关系将反映成电阻变化的光强变化作为电压信号，接入后级的放大和滤波电路后得到检测信号。但分压网络的缺点是会在分子和分母都出现变化的电阻，故而电压信号和待测物理量不是线性关系，需要后级的数字系统用计算处理。而在本例中，问题更为复杂了，因为待测物理量和电阻值的变化并非线性关系，导致了一个超越方程，难以求解。若使用计算机上常用的方式，即基于试根的方法会比较占用 MCU 本就不多的计算资源，很得不偿失。故而转换思路，使用 555 压控振荡器做电阻-频率转换。首先进行仿真验证，搭接如下电路。

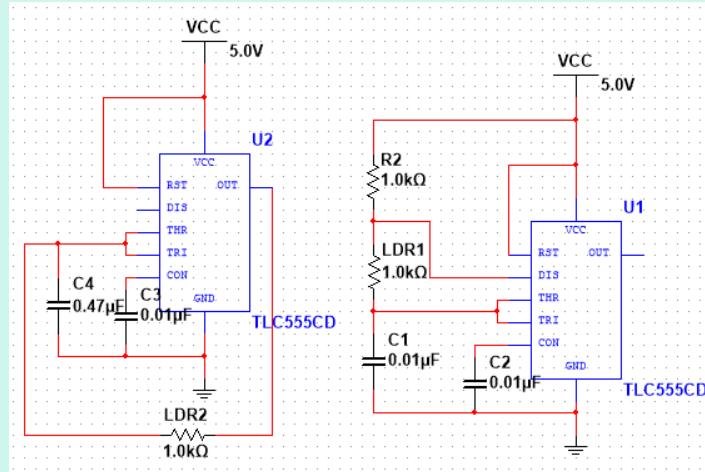


图 12: 基于 555 多谐振荡电路的光强检测计

图中标注 LDR 的即为光敏电阻，左侧方案更为简洁，并且得到的是占空比 50% 的方波，更好进行检测。进行优先的仿真和调试。简单的数学推导后得到输出波形振荡周期 T 与光敏电阻值 R 成正比，是一个简单的关系。故而可以避免解超越方程的尴尬。仿真振荡波形如下，但由于实际电路需要对光强进行标定，实际测量的调试在明天携带进行，仿真波形仅用于确认电路可正常工作。此外，基于 555 的测量电路还有在光敏电阻上电压值较为恒定，输出电平标准稳定不容易损害后级电路和低功耗的特点，是我们认为较之于基于分压电路的设计方案更好更简洁的解决方案。

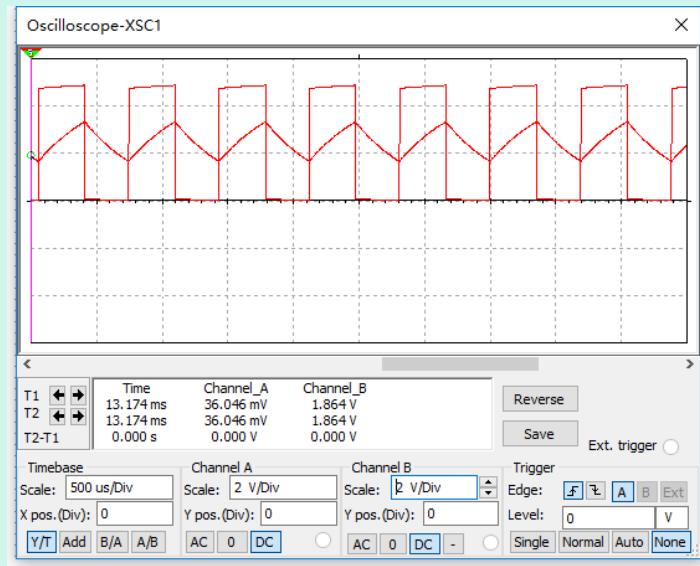


图 13: 555 多谐振荡光强测量电阻仿真波形图

2019.07.05 上午 模块联调初步搭建与电源管理模块 PCB 设计

在工作日的第五个上午我们在调试通过各种模块串口数据收发的基础上，开始了所有模块的联合调试，此外，我们在昨晚收到了 TI 公司的样片，电源管理模块使用的芯片 TPSM84，也进一步对电源管理模块进行测试。

首先在面包板上搭建电源管理模块的电路，输入 $8 \sim 10V$ 的正弦波，观察稳定输出的结果，如下图所示：

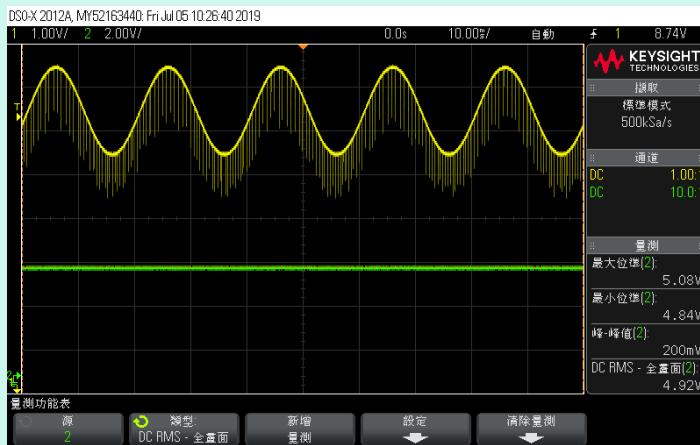


图 14: 电源管理模块

在输入正弦波存在毛刺的情况下，输出电压非常稳定，波动在毫伏级别，因此我们也验证了该模块能够正常工作，因此，我们计划将电源管理模块单独设计出一块 PCB 板进行系统的供电，由于本身的电源管理模块部分比较简单，因此也便于我们去添加额外的开

关和电源指示灯，我们在上午绘制好了电源管理模块的 PCB 图并送去制版，如下图所示：

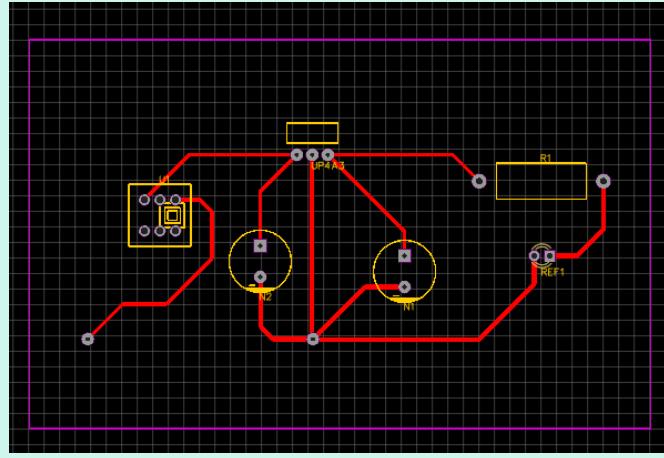


图 15: 电源管理模块 PCB 绘制

在 PCB 绘制的过程中，由于我们无法从元件库找到 TPSM84 芯片，因此查阅其 datasheet 换算单位后自己制作出了 TPSM84 的 PCB 模块，并且后来由于添加开关我们现找了自锁的按动开关，查阅其引脚和内部连接后进行接线，初次绘制 PCB 耗费了一定的时间。

另一路工作首先对基于 555 的光强检测电路进行了简单的实物搭接和收尾验证，得到波形如下：

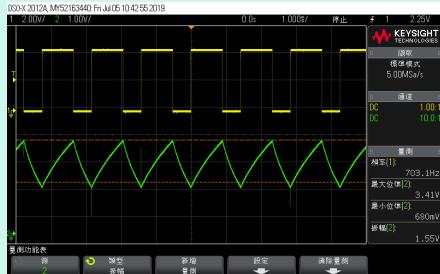


图 16: 光强检测电路波形 (室内正常环境)

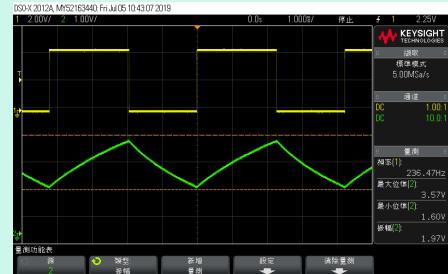


图 17: 光强检测电路波形 (遮挡光敏电阻)

可以从波形可以定性看到，由于光强减弱，光敏电阻阻值的增加，振荡周期增加，频率减小，符合定性规律。另外一方面，参考室内 300lx，而遮挡时，近似看为野外室外环境光强 1lx 扣除偏移量，验证其对数值的近似线性关系。

之后的工作计划于面包板上先实现硬件的连接，随着硬件的搭建我们同时开始软件方面的引脚分配，初步在无总线的情况下在 Arduino 串口轮询读取数据，若其能够联合正常工作并读取数据，则可进一步设计 PCB 和添加总线方案。我们已经在面包板上完成了基于 555 的光强检测计，DHT 温度湿度模块，BMP180 气压检测和 PM2.5 模块的低通滤波部分，以及红外遥控模块，如下图所示：

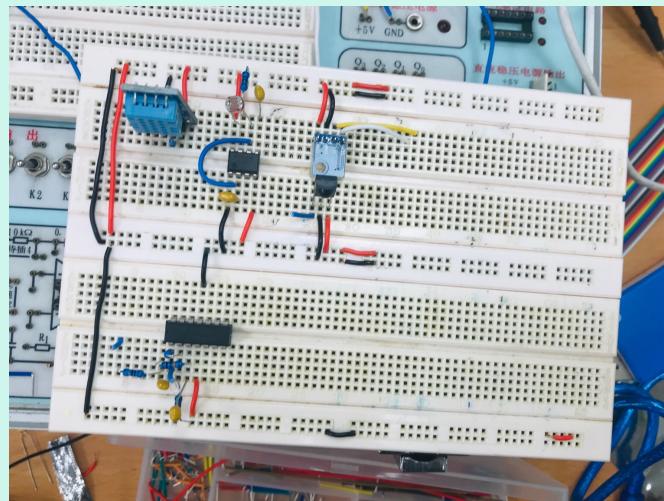


图 18: 联合调试部分硬件

在下午的工作中我们将进一步进行搭建以及编写 ino 进行串口数据读取，如果 PCB 制作完成则会进一步焊接与测试。

2019.07.05 下午 模块联调

下午实验开始后，我们打印的 PCB 很快就做好了，所以我们兵分两路，一路进行 PCB 的焊接，一路继续进行联调电路的搭接。最后的搭接完成后，我们在面包板上放置了基于 555 的光强检测计，DHT 温度湿度模块，BMP180 气压检测和 PM2.5 模块和其配套的低通滤波器，红外接收管、微波雷达和按键及其对应总线管理的数字模块。如下图（图片为实验结束后拍摄，当时为了携带方便，体积较大的 PM2.5 传感器和疑似烧坏的按键没有连接）

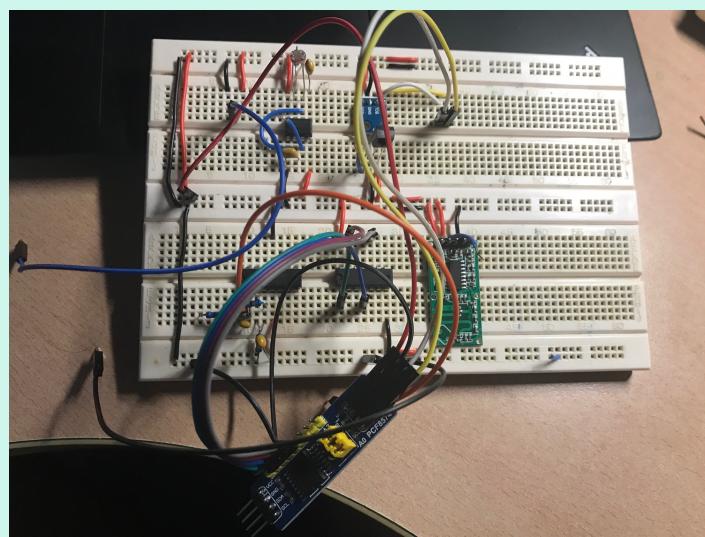


图 19: 外围电路联调

另一路在进行 PCB 板焊接时，由于按钮插孔的焊盘过小，我们失手将其焊落，难以进行修补。这给了我们一定的教训，PCB 印制结束后，并不意味着工序的结束，在实验的任何一个阶段都要小心操作。为了保证进度，达成本周实验结束后将外围硬件部分联调通过，方便周末写软件部分的目的，我们选择换用万能板做电源管理模块，迅速搭接如下电路，并将电源管理模块加入联调。

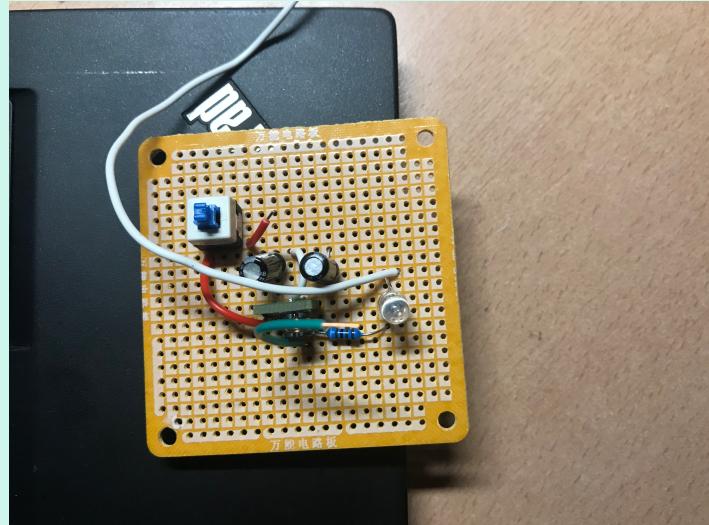


图 20: 电源管理模块实物

最后在用示波器简单验证各模块输出，再辅助简单的串口打印检测量的程序，我们验证了外围电路联调的基本通过，给周末进行软件部分调试打好了坚实的基础。

2019.07.06 上午 联合检测代码与元件采购

在经过周五的电源的准备，为我们的联合调试提供了基础，我们进一步推动了工作的进度，开始想办法在节省动态内存的情况下编写代码，首先我们在检测部分进行联合调试，我们结合了温度湿度气压与灰尘和基于 555 的光强检测在串口显示数据。

而在上午的调试过程中我们遇到了 DHT11 模块显示不准的问题，我们在测试的过程中发现了串口显示的温度要明显的高于室温 5°C 左右的，并且会在 checksum 的时候报告 checksum error，据查阅是串口数据传输的校验位错误，而我们在仔细研究了 dht11.h 的库后发现旧版本的 dht 库函数在检验 sum 和中将小数部分当做零处理，而我们现在使用的传感器小数部分并不等于 0，将其进行如下所示的修改便可：

```

humidity_int = bits[0];
humidity_dec = bits[1];
temperature_int = bits[2];
temperature_dec = bits[4];

if (bits[4] != sum) return DHTLIB_ERROR_CHECKSUM;
return DHTLIB_OK;

```

图 21: 库函数修改

在修改后明显测量数据在正常的室温范围内，并且相较于起初的单元模块调试，能够精确的小数点后两位，有较大的精度上的改善，除此之外，我们前往中发市场采购了中型的洞洞板，为检测模块的固定进行了进一步的准备。

2019.07.06 下午 通信部分代码编写与焊接设计

在所有的检测模块都能够正常工作后，我们一路进行代码的完善补充另一路进行焊接的设置，首先是代码层面，开始了将蓝牙部分以及舵机和雷达部分接口的设计，并将蓝牙与 SD card 部分的库与代码加入我们总的 debug 代码中，进行了编译以及调试，效果良好，并且，我们为了节省内存，查阅了一些 arduino 编程的技巧以及变量运用的技巧，尽量减少内存的占用。

而另一路则将面包板的检测部分，设计在洞洞板上进行焊接，由于在周五的焊接中发现洞洞板在接线方面不太方便，因此首先查阅了洞洞板焊接的技巧，其中接线主要分为两种方法，其中一种为连焊走锡式焊法，通过焊锡来使电气连接，另一种方法为背面飞线式，通过导线在洞洞板背面实现电气连接，由于走锡式焊法极易造成错误，难以修改，因此我们决定采用飞线式焊法，并且做了以下简单的设计：

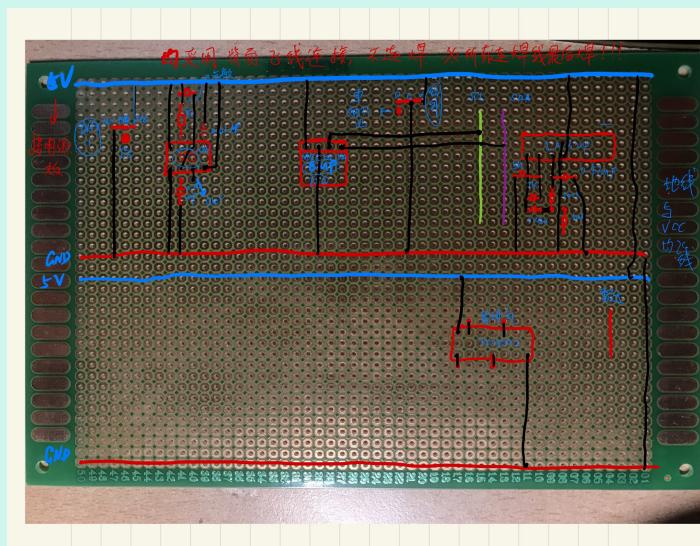


图 22: 焊接简单设计

我们此板打算只焊接检测部分，并且电源管理部分单独用一块小板，显示部分另用一

块板进行共地处理。

2019.07.07 上午 显示方法设计讨论与代码完善

在电子设计小学期第六天的上午，我们在其他模块的代码完成后，开始了显示部分的讨论，首先是硬件上对显示的控制，我们列举了需要显示的内容：温度，湿度，气压，灰尘，雷达监测、光强这六项内容，初步设计通过四个独立按键来控制，由于 LCD1602 仅为 16*2 的显示，因此每一个测量指标需要占用一行，我们采用翻页设置，分别利用两个按键进行上翻和下翻，另一个按键为滚屏的开始/停止键，滚动屏幕每次滚动一行，另一按键为灵敏度调节，开启此按键后，通过另一电位器调节检测周期，我们设计基准的周期为 1s。而另一方面利用红外遥控进行同样的控制，另外单独通过按键来改变显示的顺序，同样通过蓝牙进行串口的数据传输，类似于电脑的串口监视器。这是我们对于显示部分的讨论结果。

在讨论结果设计好后，便开始将显示部分加入我们的主代码，具体使用的函数见于报告附的代码中，基本的实现方法和我们模块调试大同小异，只是在翻页设置我们添加了状态与按键编码的练习，通过逻辑运算来切换状态，至此我们已经完成了整体的所有部分的代码，动态内存也达到了 73%，即将到的 75% 的阈值。只需在周一进行代码的测试。

2019.07.07 下午 混沌电路的设计以及独立按键硬件设计

在完成了整体代码的设计后，我们便需要在硬件部分进行设计，也即我们设计的独立按键的四个功能：滚屏、上下翻页和灵敏度调节。我们所需要的是在按下时得到其高电平，因此我们可以通过对四个键进行编码，经过 IO 转 I2C 接口后直接以 I2C 形式给到 Arduino 的 SDA 和 SCL，在内部对其进行地址分配即可，我们通过四个独立按键给出的编码来进行状态的切换，而设计以简单为原则，参照 FPGA 独立按键的原理，采用上拉电阻和下拉电阻便可简单的解决问题。

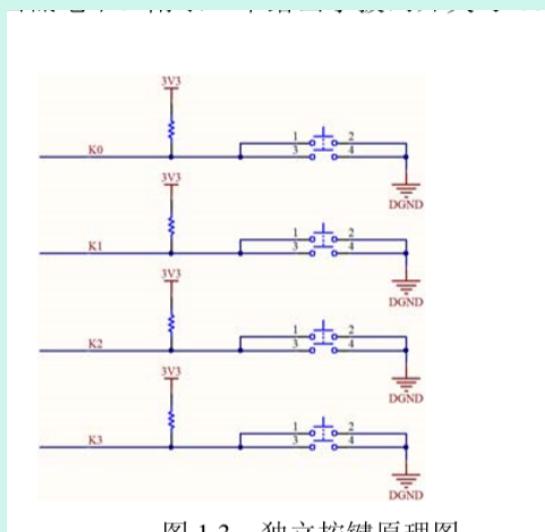


图 23 独立按键原理图

图 23: 独立按键设计

这部分设计比较简单，因此还需明天在实验室进行显示、通信、检测三大部分的联合调试，再具体的发现问题。

除此之外，由于雷达需要进行旋转扫描，而针对于室内情况，随机角度的扫描更符合对环境检测的需求，而不是定时的进行旋转，因此舵机旋转角度的设置便可有产生随机数来进行确定，我们初步的思路为产生电压值随机的模拟电压，通过 Arduino 的模拟接口来读入模拟电压在程序内部进行进一步的随机角度设置。

但设计的关键之处在于随机电压的产生，我们在查找文献后找到了一种混沌电路的发生方法，其原理为使用 RC 振荡电路但使其震荡点不稳定从而使振荡频率非周期变化，来产生混沌。其原理电路如下：

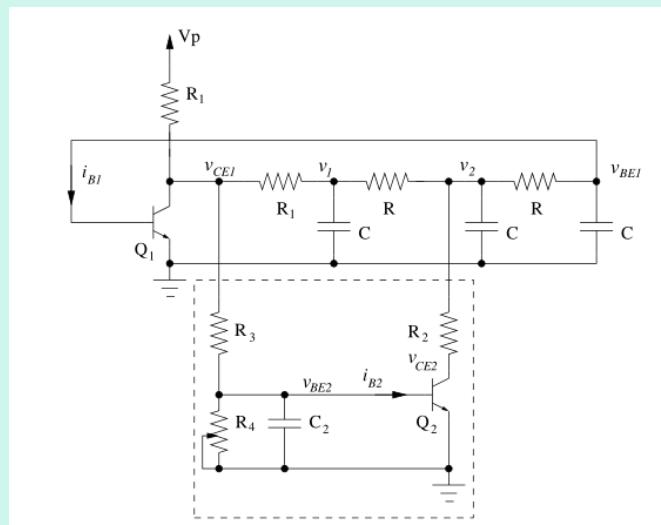


图 24: 混沌电路

在修改其参数便于实验室中找到已有元件后，得到稍规律化后的混沌电路，但观察其相图仍为混沌状态，仿真结果如下：

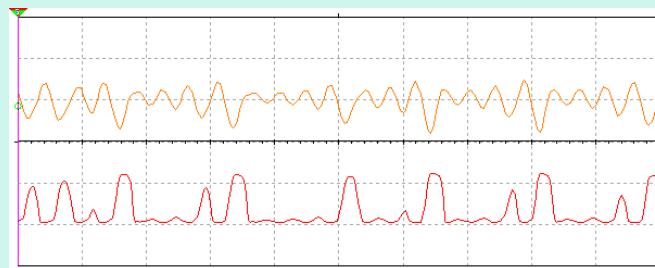


图 25: 混沌电路仿真

我们进一步在面包板上搭建了混沌电路，计划周一上工后利用电位器进行调节到混沌状态，至此我们周末的调试基本结束，总体的代码也已基本完成，若在周一所有模块联调后正常工作，我们便进行焊接与外围电路位置分配的设计。