课程设计报告

基于 Arduino Uno 的示波器影音系统

课程名称：	电子系统专题设计与制作
姓名：	武钰淞 马啸阳 伍雨心
专业：	电子信息
组号：	233

2019 年 8 月 31 日

目录

一、 构思与项目综述

1. 项目背景与构思

该项目源自一些 B 站视频带来的灵感和想基于其继续开发的构思，如 SSTA **出品：Bad Apple 示波器版**[1]、**教你把实验室仪器变成小电视**[2] 等优秀的作品。并希望最终给出一套完整的从原视频到可以在示波器上播放影音的解决方案。

2. 项目综述

这个项目从功能上可以分为三个部分，预处理、播放图像与播放声音，同时要保证预处理尽量的快速，图像连贯细节相对完整，同时图像和第三个声音同步。

预处理部分综述

预处理部分主要负责将视频帧经过灰度化、滤波、边缘提取成线条。之后按照帧保存在特殊的“视频文件”中，同时保留视频的帧数、帧率和名称等信息。灰度化、滤波、边缘提取通过 OpenCV 在树莓派这个嵌入式平台上运行，并输出至可被 Arduino 接受的 sd 卡上。

图像播放部分综述

图像播放部分主要关注如何让已经预处理好的边缘图像输出为示波器 X-Y 的模拟信号。在这部分中主要完成了 Arduino 的程序的编写和外部电路的设计与实现以达成上述的目的。读取 SD 卡相关文件，通过 DAC 数模转换芯片将帧逐线逐点输出至示波器上，同时利用示波器的余晖和视觉暂留效应显示出完整的图像。

声音部分综述

声音部分主要关注如何将声音与图像一同输出。通过 SD 卡读取相关音乐，同时通过 PWM 输出到声音模组中，经过滤波和功放最终转化为音响中的声音信号。

3. 项目难点与创新点

图像的处理与转换

除与 Bad Apple [3]类似的黑白图像外，其它的视频图像均为彩色构成，需要进行边缘检测。且边缘检测的处理结果必须是单宽度边，需要对每条边进行检查。检测完毕后要保证点在示波器上显示的有序性和清洁，不相邻点之间的连线尽量少，就需要进行一系列路径搜索、最短回路等算法，处理起来较为缓慢。

视频文件的读取与播放

Arduino 和 SD 卡与 DAC 芯片均采用了 SPI 总线通讯，通讯速率相对较慢。同时，Arduino 的内存仅有 2KB，至多能存储 512 个点，对文件的读取与播放造成了较大的限制。不同帧渲染一

[1] 哔哩哔哩 av360665

[2] 哔哩哔哩 av34298971

[3] 哔哩哔哩 av706

遍的时间差距很大，需要恰当的重复渲染以保证帧率。因此，MCU 需要重复地读取文件的某一段，这就对文件读取效率提出了更为严苛的要求。Arduino 的 IO 封装运行速率较慢，但与 DAC 通讯需要大量的电平改变，影响输出效率。进一步，我们希望这个项目能够播放不同的文档，这就需要能够打开不同的文件，以及遍历文件夹的功能。

音频播放与同步

由于处于对性能的担忧，我们采用了两个 Arduino 协作的方式，分别播放图像和音频。这就提出了一个如何保证开始的同时性的问题。由于在设想中图像播放部分需要播放不同的文档，音频部分也有相似的需求，以及，进一步的，与图像部分的统一。

二、 预处理部分的设计与实现

1. Edge Drawing[4]算法

Edge Drawing 算法的原理

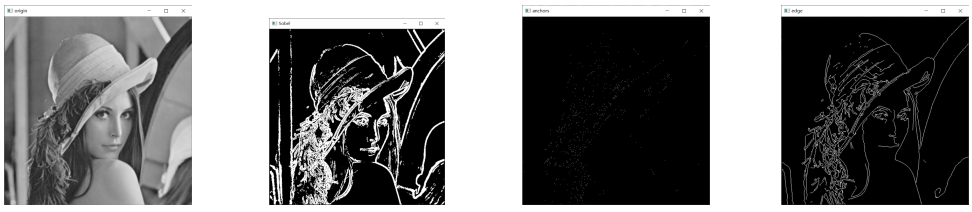
该算法的算法流程如下

1. 利用 OpenCV 的相关函数实现读入视频帧，转为灰度图，并进行高斯滤波。

2. 利用 Sobel 算子，计算各像素处梯度，总梯度为横纵两个方向上 Sobel 梯度绝对值之和，并设定阈值，舍弃总梯度较低的点。边缘的方向仅考虑横向与纵向两个方向，根据横纵梯度大小决定，较小梯度的方向即为边缘方向。

3. 利用已有的梯度信息提取锚点，锚点为梯度图沿边缘方向的极大值，且此处可设定扫描区间和锚阈值 (锚点较相邻边缘点梯度高出的下界) 调整锚点的密度，改变生成图像特征清晰度与准确度。

4. 利用启发式搜索算法连接所有锚点。在这一步中我们选定锚点，并按梯度图边缘方向沿左右或上下遍历。以向左遍历边缘为例，每次考察该点左上、左侧、左下三个点，选取梯度最大的点作为该边的下一个点，向其他方向类似。如此遍历，直至我们遇到一个已被遍历过的点，或遇到梯度为 0 的点。如此我们得到一段连续的边，点按顺序排列，若此边长度较短则忽略。随后再从其它锚点出发遍历边缘，最终我们就获得边缘点的坐标链表，且其尽可能连续的按照边缘排列，以使后续示波器显示不出现拖线的现象。



(a) 原图　(b) 梯度　(c) 锚点图　(d) 边缘图

图 1: Edge Drawing 算法分步效果展示

[4]C. Topal, C. Akinlar, and Y. Genc, Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection, Proceedings of the ICPR, pp. 2424-2427, August 2010.

2. Edge Drawing 算法的优势

Edge Drawing 算法的具有如下优势

第一，Edge Drawing 算法保证了边界均为宽度为 1 的边界。其采用的启发式搜索算法，每个处于边界上的点最多与其 8 连通邻点中的 2 个相连，保证了宽度至多为 1。

第二，Edge Drawing 原生地保证了每一条边界上的点都是相邻的，保证了每帧点之间的相对连贯，降低了拖线的数量。

第三，Edge Drawing 的时间复杂度是 $O(mn)$ 的, 其中 m,n 为帧的长与宽，优于采用其他边界算法后计算欧拉回路的 $O(m^2n^2)$，保证了处理数据的速度。

3. 其他细节实现

提取完边缘特征后，将其存入 sd 卡中，设计如下数据格式方便读取。以下图片视为仅 1 帧的视频。存储的文件按 512 字节对齐，不足部分用 0 补齐。每部分以 0xffff 开头，0xf00f 结尾。第一部分先存储视频帧率，再存储总帧数。随后每部分存储一帧，先存储总字节数，再按每点横纵坐标依次存储。

为标明视频文件名以供选择，每个视频开始添加 20 帧的视频文件名显示。视频帧率实际设置在每秒 15 帧，每秒 30 帧的原视频中可跳取帧，边缘点过多的情况下，按顺序每 5 个点取一个可保证视频效果。

三、 播放系统的设计与实现

1. 系统的设计与架构

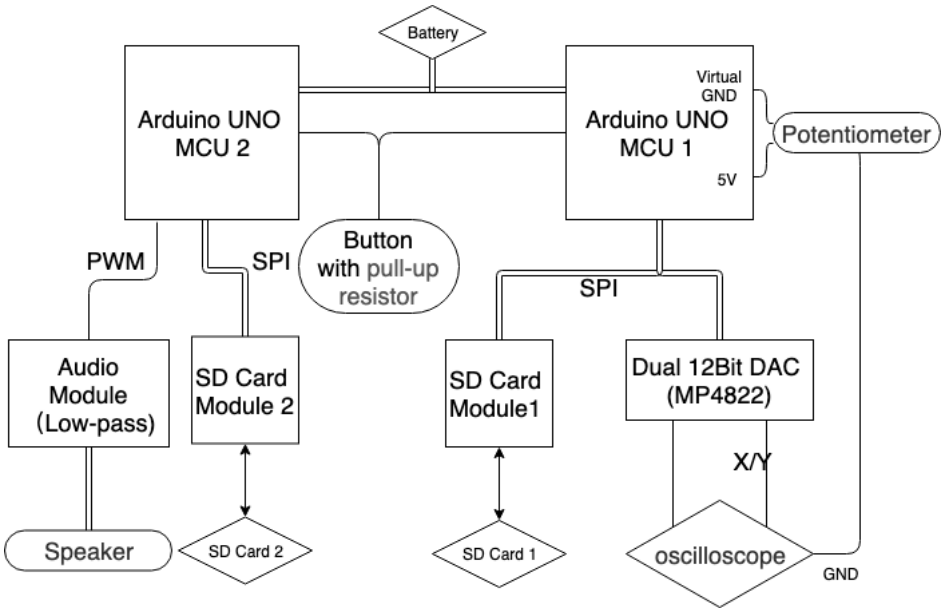


图 2: 整体架构

2. 图像播放系统的实现

文件读取的实现与优化

正如前文所言，该流程的瓶颈之一在于内存过小所带来的文件**重复**读取。我们采用了更为底层的优化进行文件读取相关操作。

事实上，SD 卡内的存储是分成 512 个字节一个块 (Block) 或者簇 (Clust) 来存储的。读 SD 卡的协议内保证了每次读取都会读取一整个块的所有数据。Fat 文件系统保证了每个文件的头都是块对齐的。所以只要我们保证我们的帧也是块对齐的，就可以在读取的时候每次读取一整块并存储下来。使用完成再进行下一个块的读取或者文件指针返回之前的块进行读取。这样每个文件块所访问量的都是最少的，防止因为相邻的字节分属不同的文件块而来回读取。

IO 的底层优化

Arduino 对于 digitalWrite 的实现是较为缓慢的，但经过对于 AVR 与 Arduino 文档的阅读，我们了解到，Arduino 的 Atmega 芯片使用了 9 个寄存器来对 IO 接口进行控制。分别是 DDRX,PINX 和 PORTX，其中 X 为 B,C 或 D，这些寄存器分别对应着不同接口的不同状态。简单的变换这些寄存器内部 8-bit 数的位值，就可以快速的进行 IO 的输出，指令周期数从几十个周期降低为几个周期，效率提高了十数倍。对于控制器，我们采用了外部中断的方法而不是轮询等待，这样也会相对的提高效率。

控制电路的实现

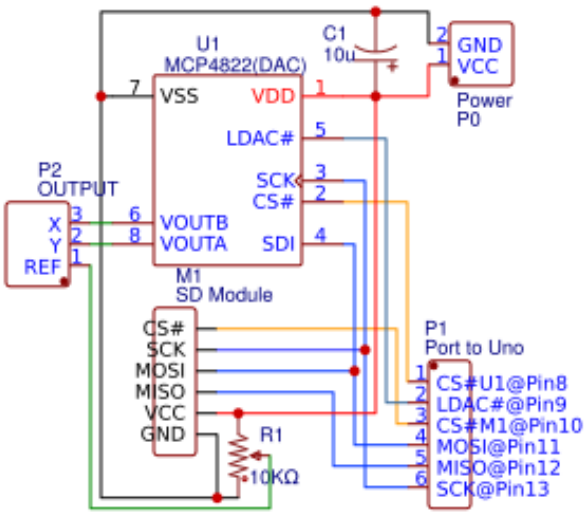


图 3: 电路原理图

由于 DAC 芯片输出的是 $0 \sim 2.048\mathbf{V}$ 的电压。所以需要一个虚拟地对示波器进行调节，使示波器能够完全显示下，我们采用了用电位器进行分压的方法使得示波器的地在这个电路中不是 0 电位。

令人遗憾的是，由于时间原因，我们仅完成了 PCB 的设计而未能进行 PCB 的印刷、制作与焊接。替代的仅万用板自制了一个简单的电路。

3. 音频播放系统的实现

外部电路的实现

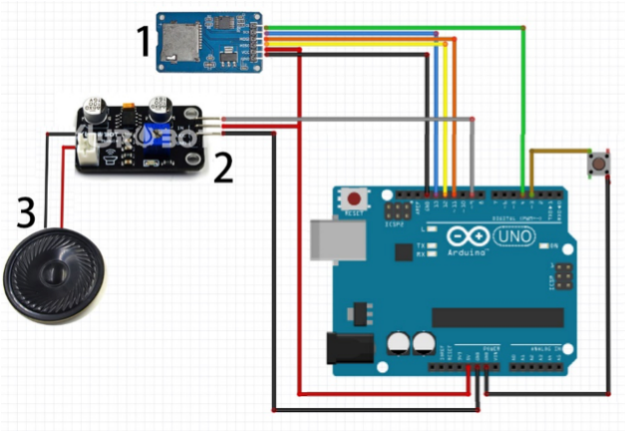


图 4: 电路原理图

其中音频模块自带了功率放大与低通滤波功能。

软件的实现

音乐播放模块的代码使用了 SimpleSDAudio 库。这个库对于采用 PWM 播放音频文档进行了封装，但前提为用相应的工具进行格式的转换以满足要求。

在 arduino 主板重置后，音乐播放模块自动播放第一首音乐。按下按钮，音乐播放模块将顺序播放下一首音乐。一首音乐播放结束后，如果没有按下按钮，音乐播放模块将从头播放当前音乐。通过调节功率放大模块上的电位器，可以改变音量的大小。

四、 调试与问题解决

在调试的过程中，我们遇到了以下几个问题并得出了相应的解决方案。

1. 画面偏置问题

在播放的过程中，我们发现在一些情况下，图像的偏置超出了示波器可以调节的最大值，于是我们采用了控制器与示波器不共地的解决方案。利用电位器调节图像的偏置。

2. 按钮信号失稳问题

在加入“下一个视频”按钮来控制播放后我们发现，当按下和松开是都会出现一个或若干个上升沿，严重干扰 MCU 对于外部中断的识别。我们的解决方法是将下拉电阻改为上拉电阻，然后按钮一段接地。外部中断改为下降沿，这样可以利用 Arduino 自身的硬件，（自带的上拉电阻）。

3. **不规则亮点问题**

在播放的过程中，我们发现边缘上每个一段距离会有一个亮度远高于其它点的亮点。初步判断为文件 IO 阻塞时示波器在该点停留时间过长，后通过观察相似帧之间亮点的分布确定该结论成立，属于该项目的实现所带来的特性，而不是实质性的问题。

4. **音频功放问题**

在音乐播放模块电路的搭建过程中，我们开始打算用 uA741 运算放大器进行音频功率放大电路的搭建。但经过调研与计算后，我们发现 uA741 并不适合用于搭建功率放大电路，于是我们改为尝试用 LM386 进行音频功放模块的搭建。但由于自行搭建的 LM386 音频功放电路中产生的噪声太大，加上时间有限，来不及深入研究，我们最终决定用已经封装好的音频功率放大模块代替自己搭建的模块。

五、 总结

整体项目全部完成，在测试中顺利完成了以下操作：

- 原视频格式转换（树莓派上完成）
- 播放视频
- 切换视频

在整个项目中，我们学习了图像处理与机器视觉的相关知识与算法。对于嵌入式开发的相关局限性与底层优化的知识有所了解，同时通过自己的学习与思考完成了一个完整的项目。在这个过程中，我们对于电子系统有了更深的了解，将课内的电路知识与程序设计知识运用到电子系统的设计与制作中去，同时对于硬件相关的设计与调试有了更深刻的理解和体验。

六、 参考文献

(1) C. Topal, C. Akinlar, and Y. Genc, Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection, Proceedings of the ICPR, pp. 2424-2427, August 2010.

(2) Arduino AG,PortManipulation,https://www.arduino.cc/en/Reference/PortManipulation,2019

七、 附件

文件名	文件描述
OpenCV.h	预处理代码的头文件
OpenCV.cpp	预处理代码的实现文件
musicPlay.cpp	音乐播放代码的实现文件
DumpFrame.cpp	视频播放代码的实现文件
PCB.pdf	PCB 原理图
清华网盘	效果视频和素材