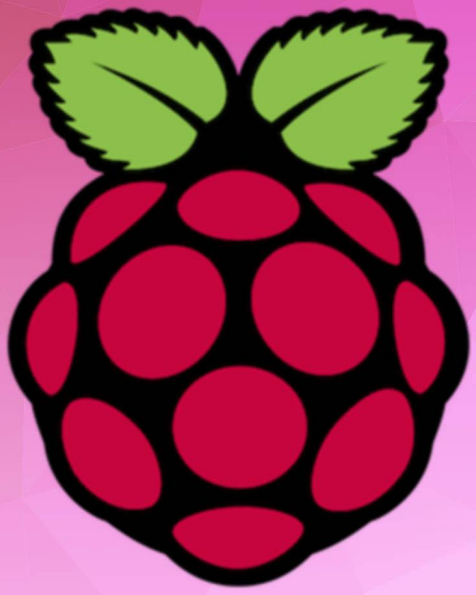


SESSDSA2017
树莓派创意编程活动
实习报告集

SESSDSA2017

树莓派创意编程活动
实习报告集

2017 年 7 月

指导教师：陈斌
编辑：郭浩
特约封面设计：罗哲楷

谨以此书献给

SESSDSA2017 全体成员

前言

由陈斌老师主讲的地空学院 2017 年数据结构与算法课，举办了两项主要的课程活动：“漂移乒乓算法竞赛”是计入期末成绩的必做大作业，而“树莓派创意编程活动”则是选做项目。本书就是后一活动的报告集。

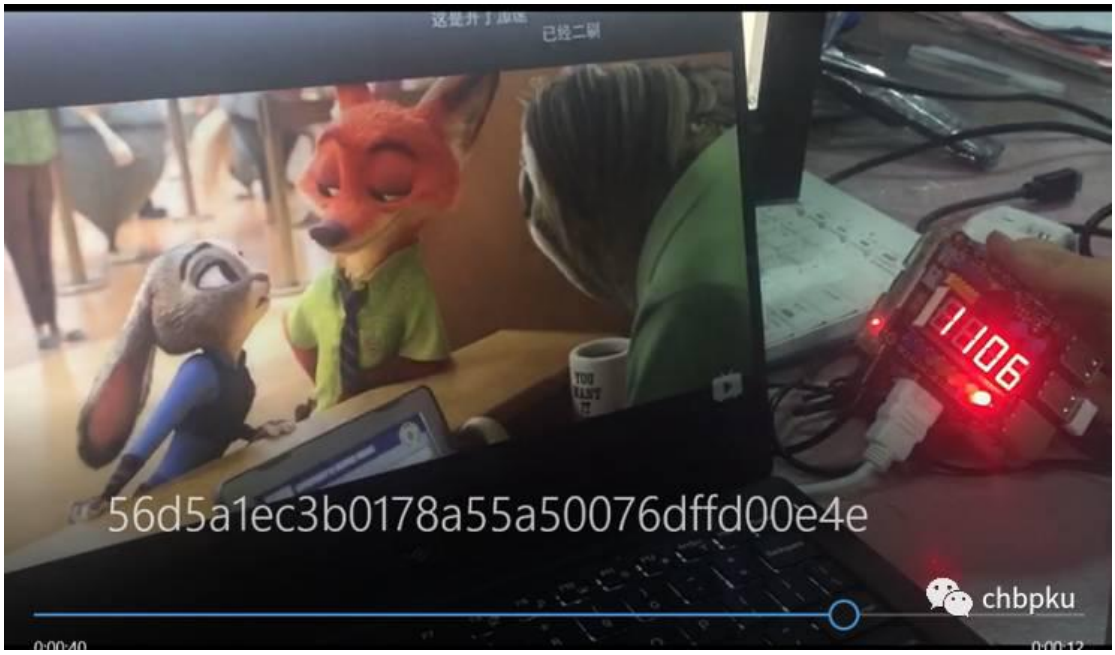
“树莓派创意编程活动”是计算机课堂中少见的硬件编程活动。参加的同学 1-3 人结为一组，并在报名表单中陈述自己的创意；通过选拔的小组将领到课程团队提供的树莓派单片机、瑞士军刀扩展板、显示器等设备，在两周时间内用 Python 语言完成开发，最后在课堂上向全班同学展示。参与本活动的同学可以获得 2-5 分的课程加分。

这一活动先后举办了两期，共有 15 组、38 名同学完成作品的开发，不少作品得到同学的认可和欢迎。这一活动的开发者需要自学树莓派使用与编程，并解决实际工作中遇到的种种软硬件问题，这是对同学能力的良好锻炼。而他们在活动中表现出的热情和创造力，在编者看来是更为可贵的。

2017 年 8 月 2 日

作品巡礼

1. 节奏投喂



作品简介：运用树莓派及其扩展硬件，通过按键状态的改变人工输入节奏，使 SAKS 复述输入节奏自我踩点，实现如“自我投喂”的树莓派拓展功能。

小组分工：冯禄：创意来源，代码编写及加注，后续优化。贾昊凝：测试，代码校验，作品展示。

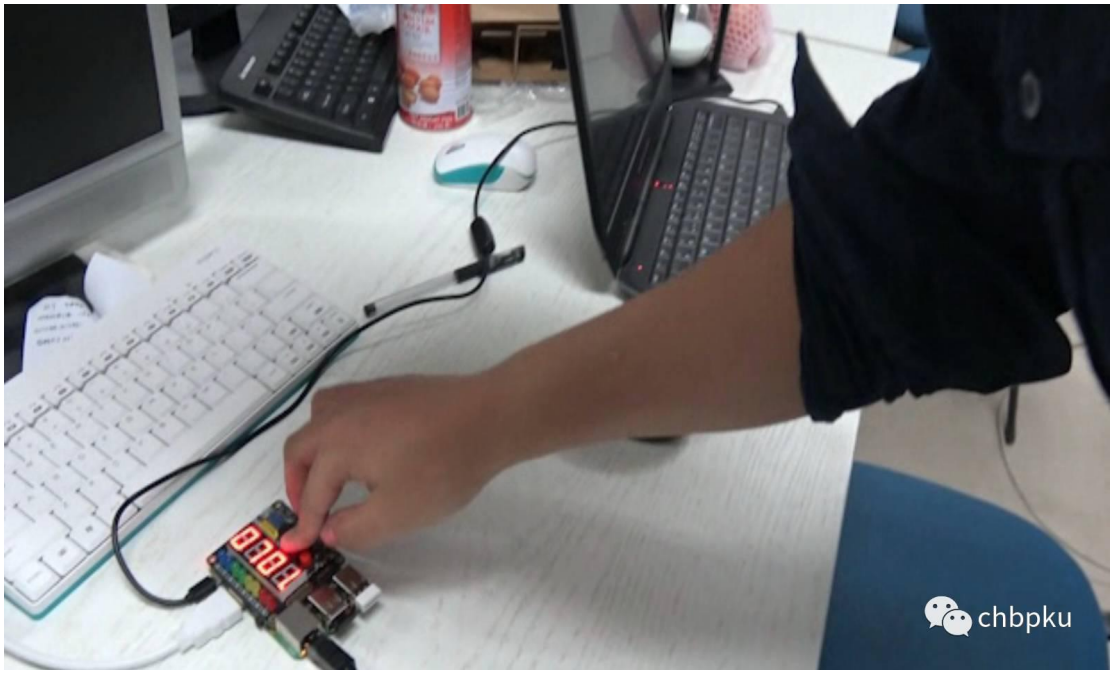
后续工作：

（1）由于数字输出不够明晰，可以通过引入新的代码模块，实现数码管单线段掉落显示。通过多线程操作，四位数字处有短横线掉落，触底亮灯，蜂鸣器按照节拍工作。

（2）本次试验仅实现了节奏的变化，可以增加拓展硬件来反映音调的变化，使之更接近节奏大师的游戏模式。

（3）可以尝试使用其他硬件作为待出场节奏的表示。

2. 多模式唤醒系统



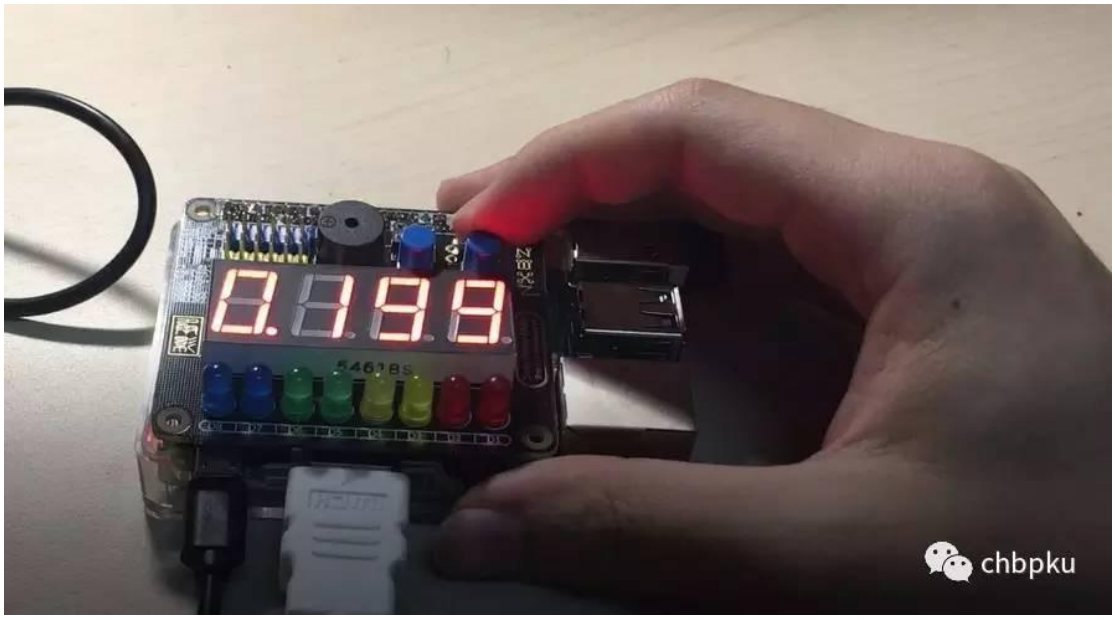
作品简介：利用树莓派可播放声音的功能和扩展板已有的八个小灯、LED 显示屏和两个按键，实现闹钟的时间显示、灯光和声音结合作为闹铃、通过按键组合来实现比较复杂的闹铃关闭的功能。

小组分工：寻找资料、创意和构思、编程实现均为小组成员共同完成。

后续工作：

1. 实现通过树莓派按键设定闹钟时间的功能
2. 闹钟播放音效或者歌曲
3. 添加其他硬件（如拳击手套）丰富唤醒方式

3.以触控开关为输入端的简易游戏



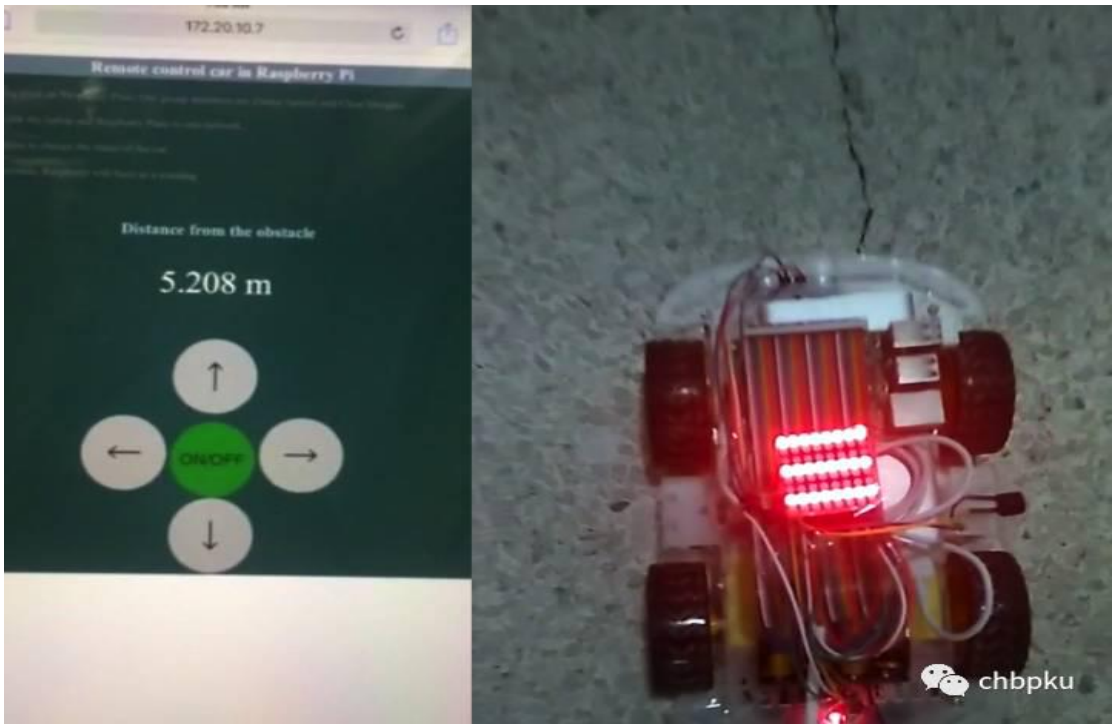
作品简介：以树莓派单片机（RaspberryPi）和瑞士军刀扩展板（SAKS）为硬件设备，SAKSSDK 开发包为编程工具，实现反应时间测试、计时能力测试、LED 的全部点亮等游戏.

小组分工：组长郭浩主要负责树莓派开发包代码的研究、主要代码的编写以及实验报告的书写。组员罗哲楷主要负责树莓派的开机与使用的探索、代码与报告的完善、PPT 和演示视频的制作。

后续工作：对于反应时间测试游戏，我们可以在单一的听觉反应测试的基础上添加视觉反应测试，这样游戏就能锻炼玩家全方位的注意力。对于 LED 解密游戏，我们可以在数学上将游戏由 8 个 LED 灯推广到 n 个 LED 灯的情况，探究其内在性质。

活动感想：本次活动是我们对单片机、硬件编程的初次接触. 这一过程中我们自己学习树莓派和 SAKSSDK 开发包的使用，并把最初的设想变成了现实. 对我们而言，这是非常宝贵的经历。

4.Web 遥控小车



作品简介：本设计基于树莓派 3 代 B 型开发板，搭配 L298N 系统小车，构成主体硬件架构。利用客户端/服务器（C/S）模型，通过移动终端远程控制树莓派 GPIO 接口输出高低电平，来控制小车的运动。LED 显示屏可显示小车的行进状态，并和蜂鸣报警器一同具有紧急报警功能。超声波测距探测前方障碍物的距离，实现紧急制动。

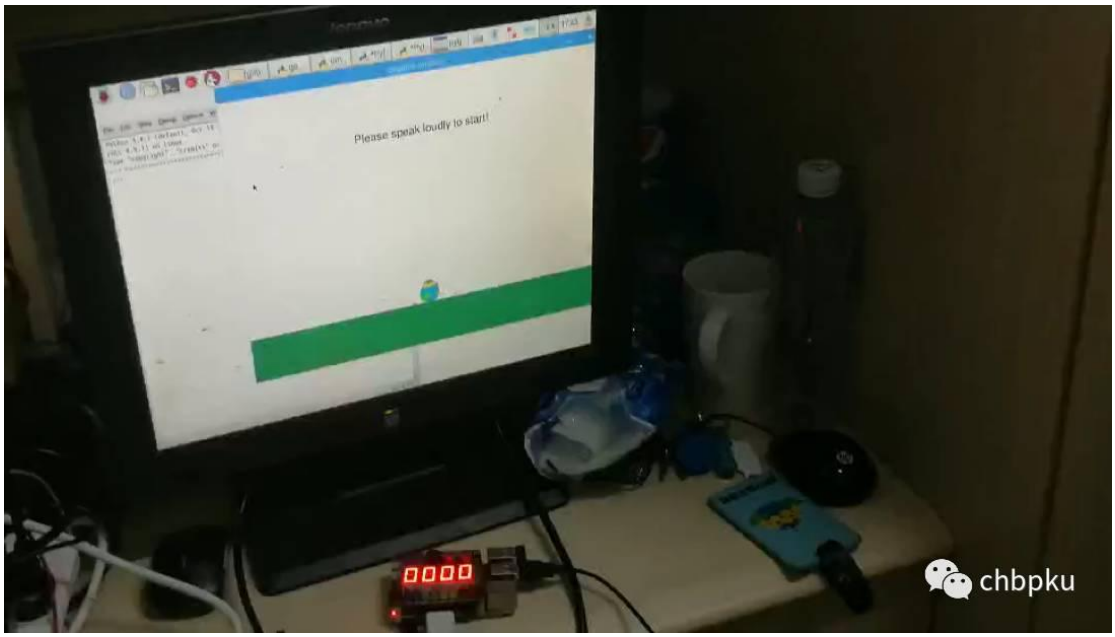
小组分工：张峻伟：项目设计、服务器搭建、代码实现与调试

陈梦珂：网页前端的开发、背景调研、物资采购

后续工作：自动规避障碍物；小车的无人驾驶功能；

产品设计优化；基于树莓派的车载无线传输视频系统。

5.树莓派八分音符酱



作品简介：用树莓派实现八分音符酱游戏；主体上是一个声控游戏，通过声音大小控制游戏主人公的走与跳，并在瑞士军刀扩展板上通过亮灯个数表示音量、数字表示分数来实现这个游戏。

小组分工：

张颢丹	技术担当	负责最主要的游戏程序部分
李佳益	创意担当	提出创意，丰富创意，收集资料
刘小辉	组长	负责树莓派的基本操作（GPIO 的实现）

后续工作：游戏背景与美观性；游戏地图去单一化与难度设计；充实游戏内容，增强可玩性；代码更加面向对象化；优化窗口界面；优化合作。

6. 音乐可视化

作品简介：

用音乐控制灯珠产生绚丽的闪光效果。对一首乐曲，以手动录入的方式将它的乐谱输入至树莓派通过程序对乐谱进行读取、转换、输出操作，并根据其中的音符，按照既定的音符配色方案控制灯阵列的亮灭。

小组分工：

孙景南：烧录系统、采购外设、初期摸索（大量 BUG）、设计和接线路（电工活儿）

王泽鑫：寻找音频素材 、提供思路、多次找到问题的突破口。

杨德鼎：图像点阵化处理、乐谱录入、编写主要代码、设计灯泡样式。

后续工作：

1.使整个设计更加美观；

2.使音乐的展示更加多样化；

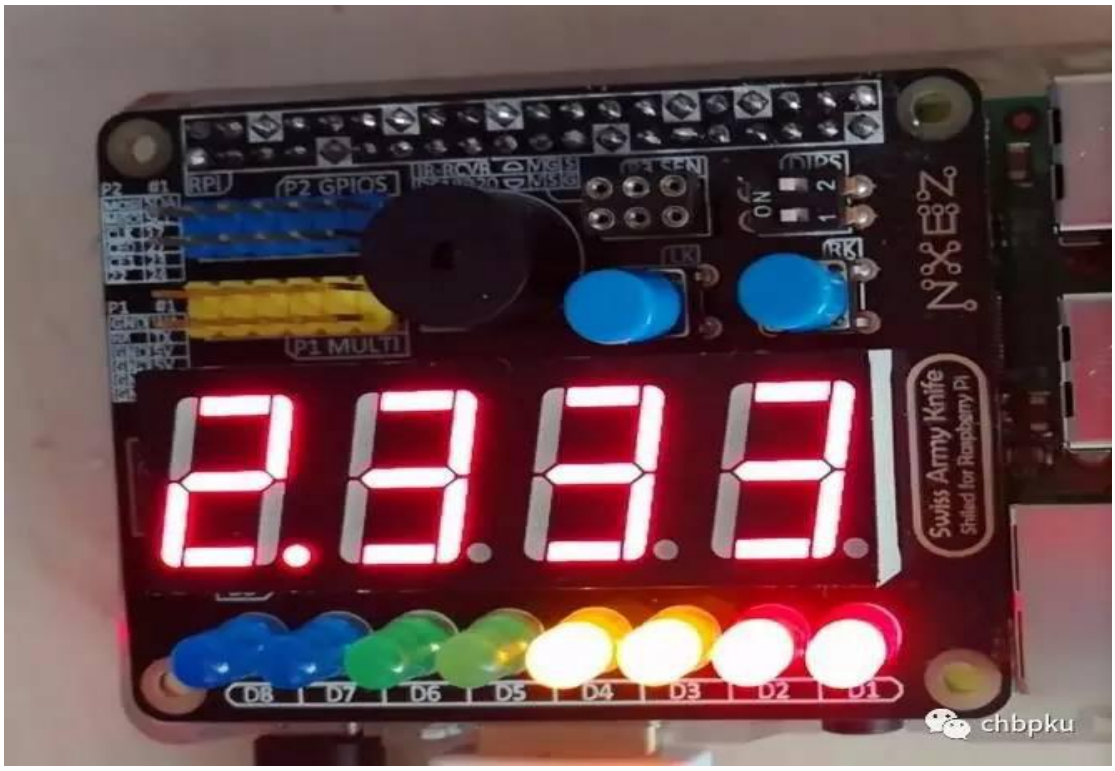
3.未来进行基于音乐进行歌曲展现的研究，即通过读取乐曲本身，让程序自动获得音乐节奏的信息；

4.实现与用户的交互，甚至可以录入使用者本身的声音，伴随着他的声音进行灯光的闪烁。

活动感想：

在完成这个项目中，我们三人合作，共同学到了很多，享受到了学习的乐趣，有过为一个难题纠结数小时的苦闷，也有终于搞好的拍手叫好。在这个过程中，我们各有各的收获。本次树莓派实习项目真的很棒。

7. 摩尔斯电码



作品简介：实现有关于摩尔斯电码的三种功能：

1）英文转摩尔斯电码

2）声音输入摩尔斯电码，输出英文

3）键盘输入摩尔斯电码，输出英文

小组分工：

戴琪：原始代码（只有两种模式且较为冗长），PPT， 实验报告

黄岭贝：大幅度简化组长原始代码...增加声音输入功能

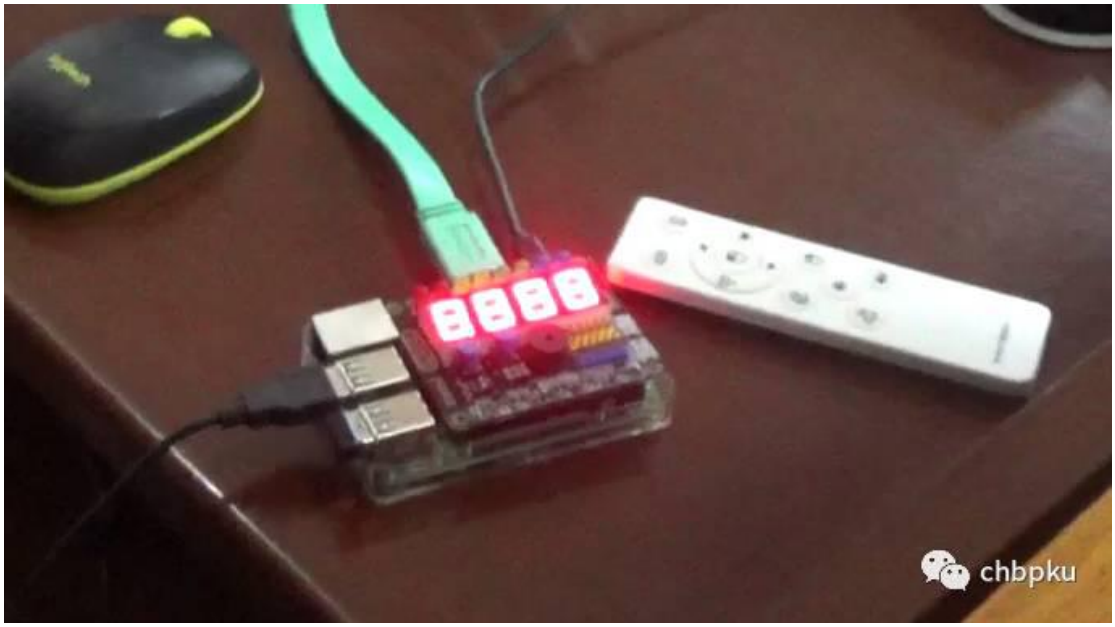
后续工作：

1）用树莓派的两个按键实现点，划，空格的输入

2）提高声音输入的精准度

3）实现外界面包板，显示输出内容

8. 可视化电子钢琴

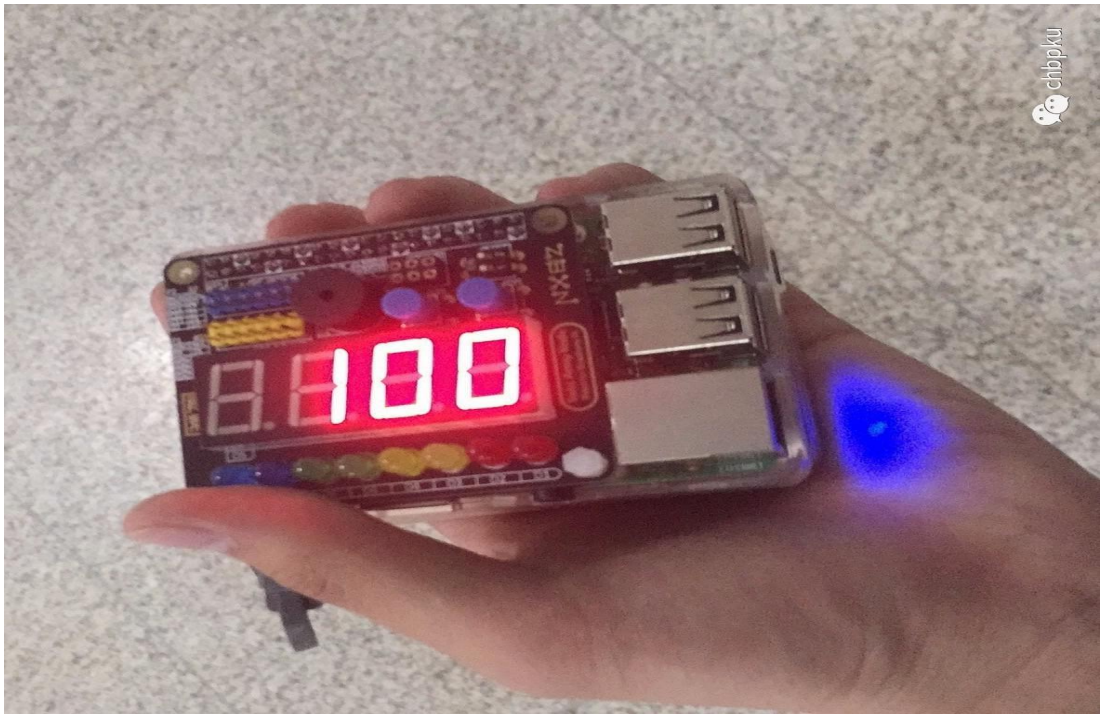


作品简介：具有电子琴最基本的功能，并将音乐同树莓派的 LED 灯结合，打造出良好的展示效果。使用 pygame 为基础的多线程编程实现。

小组分工：杨帆负责树莓派编程环境配置，主体代码的编写。张昊负责测试 SAKS 板的性能，树莓派部分代码的编写。孙唯一负责寻找播放的单音音源，转换格式，整理命名归类。

后续工作：录放音、叠加音轨；节奏伴奏；GUI；延迟；复杂和弦处理；常见格式导入导出；蜂鸣器使用。

9.老虎机



作品简介：充分利用树莓派瑞士军刀扩展版上的开关、数码管、LED 灯等组件，在只外接充电宝的情况下独立完成老虎机功能。

小组分工：刘启航：买读卡器，尝试开机，归还仪器；沈泽盛：把显示屏搬上楼，尝试开机，尝试固定树莓派，输入 ppt 上部分代码，试玩游戏，视频摄像；陶天阳：其他工作。

后续工作：

1.下注时可以多选几个灯，增加游戏性。

2.外接投币机，把数学期望调至 0 以下即可赚钱。

10.防止狗扒垃圾桶报警装置

作品简介：一个能够阻止宠物乱翻垃圾的装置。运用蜂鸣器，红外和温度传感器以及开关，在宠物出现的时候报警。在不影响人类活动的基础之上，防止宠物扒垃圾桶。

小组分工：两个传感器的创意来源：端韵成；软件和硬件的实现：谢冠旖，端韵成；视频制作：谢冠旖；ppt 制作：端韵成；撰写实验报告：谢冠旖，端韵成。

后续工作：

1.通过按按钮来调整反应时间并显示；

2.优化硬件，使硬件更美观；

3.录制好主人的声音，当狗狗接近装置时，通过蓝牙音箱播放。

11.篮球计分计时器

作品简介：遵循“计分计时电子化，进球自动化”的原则，即进球就有提示功能，在篮框内置红外线传感器，进球即发出“滴~~”一声；为了方便裁判计分，将把纸质版的比分电子化，顺带加上计时功能。同时通过拍摄直播给不能到场的同学们提供服务。

小组分工：创意想法和编程实现均为小组成员合作实现

后续工作：解决以下问题

（1）视频直播时无法同时在视频下方显示比分和时间

（2）视频直播时无法播放声音

（3）无法多角度观看比赛

12.面部签到系统

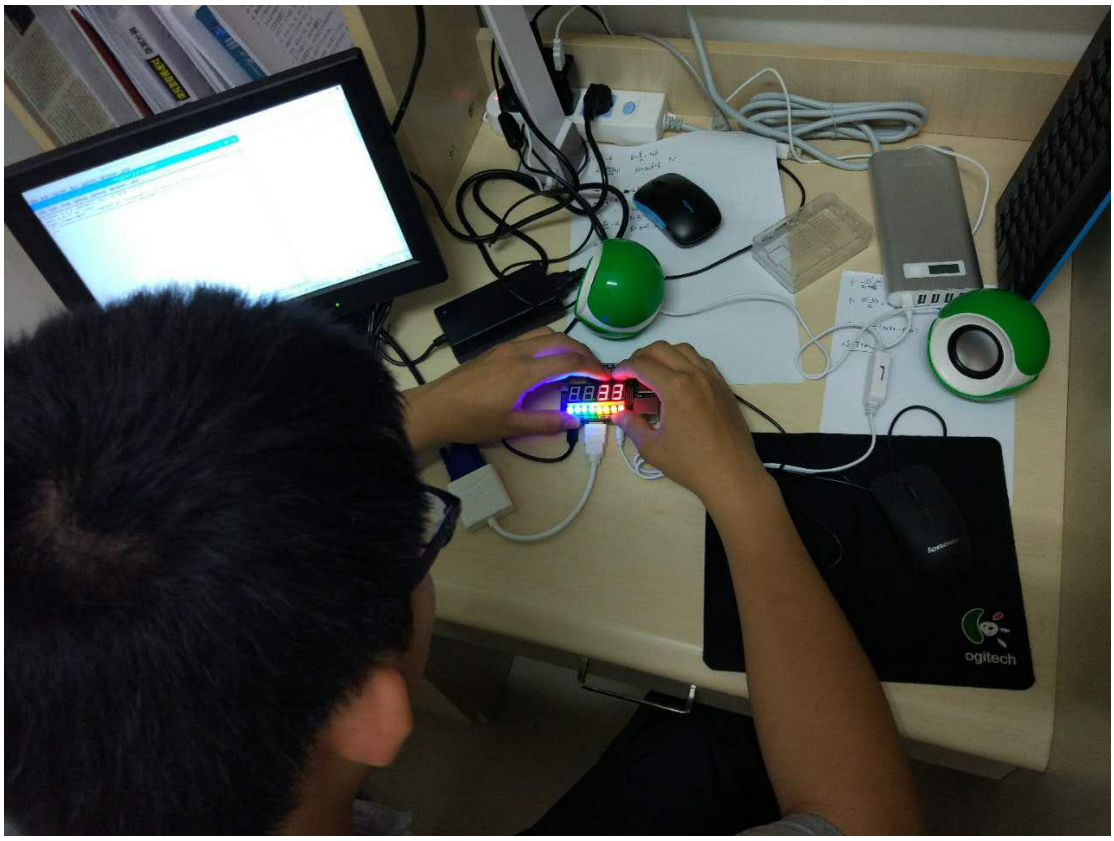
作品简介：通过面部识别和面部对比，实现简单的自动化签到系统。

小组分工：主要内容为两人合作共同学习完成。大致上，王梦瑶负责了主程序的书写，任庆杰负责了代码在树莓派上的调试与实现。活动展示内容与活动报告的撰写等其它工作由二人共同完成。

后续工作：将加载的数据库从内存中的静态数据库转向服务器或者云端的动态数据库；将字典型结果进一步处理，如上传到服务器中、和历史数据进行合并与比较等；提高识别准确度。

活动感言：感谢在本次活动中一直提供帮助的助教陈旭。因为显示器和其它的各种问题，多次打扰助教；而助教总是耐心地给予答复和鼓励，使我们得以顺利完成本次作业。同时感谢陈斌老师在本学期数据结构与算法课程上的付出，使我们度过了一个快乐而充实的学期的同时，从编程小白逐渐熟悉灵活强大的 *python* 语言与神奇的程序算法。

13. 树莓派小游戏开发

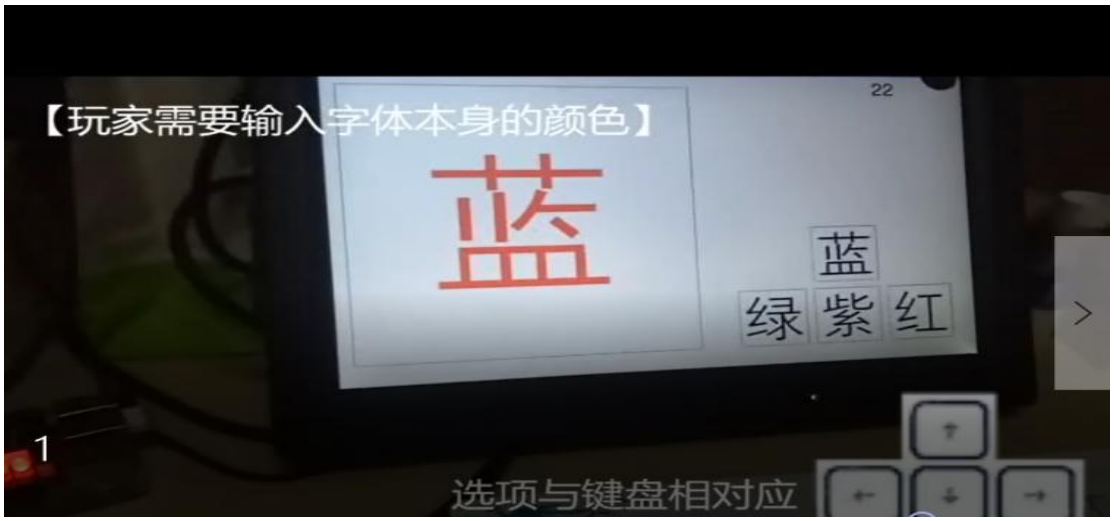


作品简介：“连携技”的想法非常简单，每次在数字屏上随机显示由“1”或“2”组成的 4 个数字，其中“1”对应左键，“2”对应右键。要求玩家在规定时间内按照完全正确的顺序按下对应的按键；“别踩白块儿”基本的想法是用数字 1 到 7 代替 7 个音高，在数字屏上从右往左滚动显示简谱数字；当某个数字移动到最左端时，需要在这时按下对应的按键（奇数对应左键，偶数对应右键）；最后显示按对数字的比例来表示玩家对这首音乐的完成度。

小组分工：祝奇文主要负责实现树莓派 IO 功能部分的代码，傅昊博主要负责逻辑部分的代码。

后续工作：稳定“别踩白块儿”游戏的运行速度；完善音乐效果；实现树莓派版“2048”

14.“最强大佬”游戏



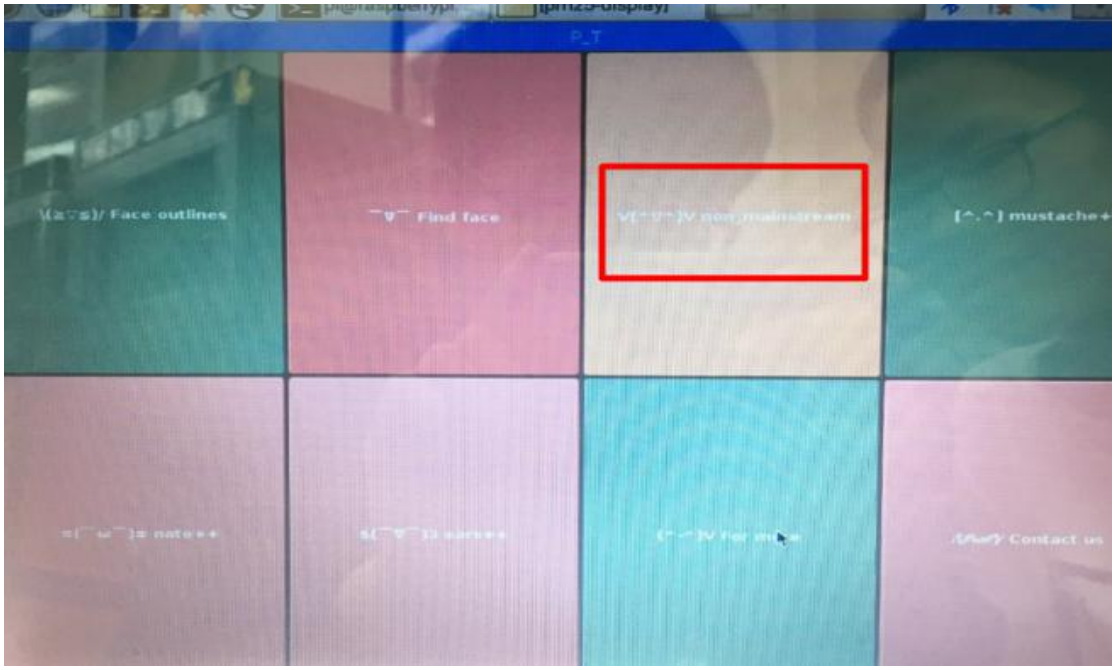
作品简介：“数字闪现”：根据难度级别显示 2-4 位的数字，数字闪现结束后，玩家需要按照提示音要求（“请输入最大数”“请输入最小数”“请输入中位数”），尽可能快地输入正确答案。“颜色辨别”：游戏中屏幕会出现相应的颜色和干扰汉字，比如“紫”这个汉字它却是黄色的，而玩家应该快速排除“紫”字的干扰，输入黄色。

小组分工：滕沅建：编写代码；陶韵竹：创意来源、游戏语音来源、制作视频；王瑞敏：录制视频、制作展示 ppt、撰写实习报告。

后续工作：引入更多的游戏语音；对于“颜色辨别”游戏，加入颜色背景提高难度；开发立体积木的识别、“石头剪刀布高阶版”等游戏。

活动感想：我们这个小组成员是一个寝室的室友，在宿舍讨论开发这个游戏时，从宿舍里会时常传出我们“魔性”的笑声。其实在此之前，我们三个对树莓派甚至编程都不是特别精通，但是在组长滕沅建的带领下，不断探索，不断学习，终于实现了我们开发小游戏的设想，当看到数码管和屏幕出现我们预想中的游戏界面时，三个人都很有成就感。如此欢乐的合作气氛，以后想来也是会骄傲地扬起嘴角的事。

15.定制美图秀秀



作品简介：通过树莓派，实现美图秀秀中的一部分功能：对图片进行铅笔画操作（滤波），以及进行人脸识别，再在识别出来的人脸上加胡须、耳朵等饰品。

小组分工：叶勃：组织小组讨论、进行分工，完成“饰品”功能，进行展示视频的后期处理。金恬：完成在瑞士军刀扩展板上的功能实现，制作 PPT。曹寒冰：完成人脸识别功能实现，完成 GUI，代码整合。

后续工作：增加人脸识别功能的应用性，如建立一个数据库；通过机器学习等等方法来实现人脸识别，提高人脸识别的精度；提供更多的饰品选择；更友好的图形界面。

活动感想：我们小组三人齐心合作，从零开始学习树莓派，在经历各种克服困难的同时感受到学习的乐趣。在这个过程中，我们成功实现了很多曾经觉得特别高大上的功能，从实战中感受到编程的力量，大大激发了进一步深入学习的热情和信心。

目录

树莓派创意作品实习报告——节奏投喂

冯禄、贾昊凝

摘要：

受到风靡一时的节奏游戏的启发，本组运用树莓派及其扩展硬件，通过按键状态的改变人工输入节奏，使 SAKS 复述输入节奏自我踩点，实现如“自我投喂”的树莓派拓展功能。

一、 选题及创意介绍

节奏类游戏曾风靡一时，在游戏过程中，需要玩家观察掉落的光标配合音乐节奏敲击屏幕下方相应区域以得分。对于许多“手残党”而言，节奏类游戏让其望而生畏，为了给这一类玩家训练手眼节奏协调的机会，锻炼在节奏游戏背景下的反应速度，我们想到了节奏投喂的创意。即使用树莓派及 SAKS 板，通过简单的代码实现手动输入节奏，而后四位数码管滚动显示待亮起 LED 灯的编码，蜂鸣器与 LED 灯在节奏点上工作的过程。

在改进代码的过程中，我们发现调用四位数码管亮起的处理时间过长，当必须要满足快节奏输入输出时，只能选择放弃数码管的数字显示，程序精简为投喂节奏-按照节奏蜂鸣及闪烁 LED 灯。

在代码实际完成之后，我们还发现经过简单的修改它可以充当简单的测量多点时间间隔的计时器，随机密码的生成器，学习某一段舞蹈或演奏时的打点节奏器，等等。总之，我们希望通过简单的代码展现出多可能性的功能，让更多的同学看到不仅是知识储备丰厚的大神，只知道基本法的新手也可以按照自己需要的功能按图索骥找到相应的函数和模块，以实现自编程把玩树莓派的目的。

二、 设计方案

一）输入

1.重复（左轻触按键按下：获取当前时间），用列表储存各时间点。并且生成一个在该节奏点对应亮起的 LED 灯序号

2.右轻触按键按下：结束节奏输入部分

二）处理

3.处理每一时间点相对前一时间点的差，得到间隔时间列表，复制出输入的节奏。

三）输出

4.延时处理，每个节奏点：数码管显示上表四位切片（eg.DCBA），蜂鸣器工作,A 数字对应的 LED 亮

三、实现方案及代码分析

为了实现上述方案，我们通过网上搜索和编译器内查询，获得了相应的函数，有些可能不是最优选择，但基本实现了我们需求的功能。这里对主要使用的函数和模块进行说明，详见代码附件。

一）输入#使用按键反应函数 tact_event_handler(pin,status)重设按键按下时的对应工作，判断是否为按下操作可以使用 SAKS 内置函数实现（pin == PINS.TACT_RIGHT and status == True）。

1.（左轻触按键按下：获取当前时间使用 datetime 实现），用列表储存各时间点。利用 random 生成一个在该节奏点对应亮起的 LED 灯序号

2.右轻触按键按下：结束节奏输入部分

二）处理

3.处理每一时间点相对前一时间点的差，（datetime 可以直接做减法得到 timedelta 格式，再由 timedelta 转化为数字格式，以秒为单位）得到间隔时间列表，复制出输入的节奏。

三）输出

利用 threading.Timer 及 threads 实现多线程工作，每一节奏点的蜂鸣器+LED 反应作为一个线程，四位数码管显示作为另一线程。同时开启各线程的工作

综上所述，在实现过程中我们并没有选择 RPi.GPIO 模块来作为我们的语言模块，而是使用了 sakshat，对于本程序而言，代码可读性更好。总体代码长度没有超出 60 行，算是树莓派入门级代码，供各位同学参考。

四、后续工作展望

1.存在问题及处理

（1）当节奏密集时，输出节奏明显变慢：经过调试，发现节奏变慢的主要原因是四位数码管显示字符串切片的实现需要相对长的时间，在将数码管的显示单独拆分出线程之后，蜂鸣器和 LED 灯的工作基本无延迟，而由于 Timer 的同时性，数码管显示会发生一定情况的混乱。同时我们尝试使用 sched 替代 Timer，当输入间隔为秒尺度时，数码管显示良好，但当输入间隔为 0.1 秒尺度时，同样会发生延迟。

（2）硬件按钮灵敏度：在人工输入节奏时，由于按键灵敏度过高，单次节奏输入常被识别为连续的 2~4 次。

（3）输出不够清晰：数码管生成数字 KEY 按照 LED 灯亮起的节奏不断向右滚动，但每一位有 0—7 八种数字可能，实际情况下“手残党”在读取时目不暇接。

2.后续展望

（1）由于数字输出不够明晰，可以通过引入新的代码模块，实现数码管单线段掉落显示。通过多线程操作，四位数字处有短横线掉落，触底亮灯，蜂鸣器按照节拍工作。

（2）本次试验仅实现了节奏的变化，可以增加拓展硬件来反映音调的变化，使之更接近节奏大师的游戏模式。

（3）可以尝试使用其他硬件作为待出场节奏的表示。

五、小组分工合作

1.冯禄：创意来源，代码编写及加注，后续优化。

2.贾昊凝：测试，代码校验，作品展示。

附程序源码如下：

```
from sakshat import SAKSHAT
from sakshat.sakspins import SAKSPins as PINS
from datetime import timedelta
import threading,time,sched,random,datetime
from threading import Timer #导入各种需要用到的模块
SAKS=SAKSHAT()
timelist=[]
ledlist=[] #建立基本的用于存储的列表和字符串
sled=''

def tact_event_handler(pin,status):
#该函数表示当轻触按钮状态改变时，对应要完成的指令
    global timelist
    global ledlist
    global tlist
    global sled #这里我们把所有在引用函数里用到的存储列表全局化
####输入阶段####
    if pin == PINS.TACT_LEFT and status == True:#当左轻触按钮被按下（需要注意的是，对于轻触按钮来讲，按下和松开都是状态改变，我们只以按下计次）
        timelist.append(datetime.datetime.now())#使用 datetime 记录当前时间点
        SAKS.buzzer.beep(0.05)                    #蜂鸣器在轻触按钮按下时鸣叫 0.05 秒以与体验者互动
        sled=str(random.randrange(0,8))+sled    #与此同时，生成与按下次数相同长度的 LED 亮起随机顺序表,以字符串形式存储
####处理阶段####
    elif pin == PINS.TACT_RIGHT and status == True:#当右轻触按钮被按下，输入过程结束
        tem=timelist[0]
        timelist=[(i-tem) for i in timelist]  #首先将时间点列表中的时间点转换为 timedelta 格式，表示与第一下按下的时间间隔
```

```
        tlist=[(i.seconds + i.microseconds/ 1000000) for i in timelist]#转化为秒为单位的数字
        sled='---'+sled+'0'                          #将 led 随机亮序补位，前面的---使得轮到最后一位数字到最右一位数码管时表示方便，末尾补位 0 方便切片
####输出阶段####
        time.sleep(2)
        threads=[]  #创建 threads 列表
        for i in range(len(tlist)):
            # threads.append(threading.Timer(tlist[i],nums,(i,)))
            threads.append(threading.Timer(tlist[i],bling,(i,)))#创建线程并添加进上述列表，括号内线程用 Timer 实现，三个参数依次代表延长的时间、任务、任务所需参数
            threads.append(threading.Timer((tlist[i]+0.5),over,))#添加最后一个线程，关闭所有的元件，清空所有列表和字符串
        for t in threads:
            t.setDaemon(True)
            t.start()          #同时启动 threads 列表中的线程

#################
def bling(n):
    SAKS.ledrow.off_for_index(int(sled[(-1-n)]))     #关闭上一次亮起的 led 灯
    SAKS.buzzer.beep(0.025)                          #蜂鸣器工作
    SAKS.ledrow.on_for_index(int(sled[(-2-n)]))      #点亮当前轮到的 LED 灯

def nums(n):
    SAKS.digital_display.show(sled[(-n-5):(-n-1)]) #在四位数码管显示当前及紧随其后的三个即将亮起的 LED 灯编码

def over():
    SAKS.ledrow.off()
    SAKS.digital_display.off()                       #关闭各种显示，清空各种列表和字符串
    tlist[:]=[]
    timelist[:]=[]
    sled=''
#################
SAKS.tact_event_handler =tact_event_handler          #将原始的按钮反应函数修改为我们所定义的新按钮反应函数
```

多模式唤醒系统实验报告

信息管理系　蒋天骥 杨家鑫 王宇萱

1 选题及创意介绍

当下，起床已成为众多都市青年每天都在面对的难题之一。听见闹钟起不了，停了闹钟继续睡，这样的场景常常发生在我们的日常生活中。甚至很多时候，因未能及时起床而影响了我们当天的工作学习安排。正因如此，寻找各种各样的创意，去帮助人们更容易、更轻松清醒地起床已成为当下许多创意设计师们共同参与的命题。现如今已诞生了一定量的创意成果。我们也注意到了这一问题，通过参考国内外对树莓派的已有方案，并结合实际应用的需求，我们计划利用树莓派及军刀扩展版也设计一种趣味、有效的闹钟，希望能利用树莓派可播放声音的功能和扩展板已有的八个小灯、LED 显示屏和两个按键，实现闹钟的时间显示、灯光和声音结合作为闹铃、通过按键组合来实现比较复杂的闹铃关闭的功能，还设想外接机械（如拳击手套等）来实现比较好的叫醒效果。结合技术实现的实际情况，我们最终设计的多模式唤醒系统（MMCS）基本实现了上述功能。设计成果的创意点在于闹钟每一次响铃都会随机生成四位“0”和“1”的组合数字并显示在显示屏上，用户需要利用两个按键正确按出显示的数字，才能关闭闹钟。

2 设计方案

我们对多模式唤醒系统（MMCS）的功能设计如下：

1. 时钟功能：该系统具有一般时钟具有的基本功能，即能够显示准确的系统时间。

2. 可变速流水灯：在到达设定的闹钟时间前 70s 时，8 个灯泡开始以流水灯的形式逐个闪烁，且流水灯闪烁频率逐渐加快

3. 变节奏提示音：在到达设定的闹钟时间前 15s 左右，蜂鸣器开始作用，变节奏的伴随着流水灯的频率发出声音。

4. 声光唤醒：到达设定的闹钟时间时，8 个小灯以及蜂鸣器以固定频率同时作用。小灯亮起时，蜂鸣器发出声音。

5. 深度唤醒：到达设定的闹钟时间后，显示屏不再显示时间，而是随机显示“0000”、“1100”、“0011”、“1111”四个数字中的一个。左按钮对应数字“0”，右按钮对应数字“1”。通过对按钮的操作，输入显示屏上对应的数字，闹钟才能成功能关掉。

3 实现方案及代码分析

我们将我们的设计方案分为四个具体模块加以实现，下面，我将具体分模块介绍我们的代码模块并分别做分析：

第一个模块是时钟模块，时钟模块通过灵活运用 python 中自带的 time 模块函数分别输出当前时间（即 localtime），并将具体的时、分、秒分别输出，再将获得的时间通过数字显示屏显示出来，输出时间的代码如下：

```
    t = time.localtime()
    h = t.tm_hour
    m = t.tm_min
    s = t.tm_sec
    w = time.strftime('%w',t)
    print "%02d:%02d:%02d" % (h, m, s)
```

第二个模块是 LED 彩灯模块，本模块实现这样的功能：首先，当时钟时间到达预定闹钟时间前 70s 钟时，LED 彩灯按照从左往右的顺序依次亮起并熄灭，形成流水灯效果。我们将八盏小彩灯全部亮起并熄灭一次称为一个循环，每经过一个循环，流水灯的速度就加快一倍。在经过了约 15 次循环，即系统时间到达了预定的闹钟时间时，小彩灯停止以流水灯的形式明灭运转，而转而配合闹钟同亮同灭，以最大化一明一灭的灯光效果。LED 彩灯模块的代码大体如下：

```
    if ("%02d:%02d:%02d" % (h, m, s)) <= __alarm_time and ("%02d:%02d:%02d" % (h, m, s)) >=
__firstalarm_time:
    for i in range(15):
        for j in range(8):
            SAKS.ledrow.on_for_index(j)
            if j == 0:
                SAKS.ledrow.off_for_index(7)
            else:
                SAKS.ledrow.off_for_index(j-1)
            time.sleep(3.0/(float(i)+1.0))
        if i > 8:
            beepAction(0.05,0,i-7)
        print(i)
        print "%02d:%02d:%02d" % (h, m, s)
        t = time.localtime()
```

第三个模块是蜂鸣器模块。在我们的设计中，我们以军刀扩展版自带的蜂鸣器作为多模式唤醒系统的声源部分，我们通过操纵蜂鸣器来实现闹钟的闹铃功能。蜂鸣器模块主要实现三部分功能：当系统时间到达预设闹钟时间前约 15 秒时，蜂鸣器配合小灯开始作用，每当小灯完成了一个流水灯循环，蜂鸣器就会短鸣一次。随着系统时间接近预设闹钟时间，蜂鸣器鸣叫的时间延长，两次鸣叫的时间间隔缩短；当系统时间到达预设闹钟事件时，蜂鸣器转为停 0.5 秒长鸣 0.5 秒的稳定工作模式，试图将用户直接唤醒；而若蜂鸣器鸣叫了 30 声依然无人理会，我们的唤醒系统将判定现有闹钟无法唤醒用户，因此，蜂鸣器将再次改变自己的工作状态，取消两次鸣叫间的间隔而改变为持久长鸣，以最大化的音量和音效来唤醒用户。
蜂鸣器模块的代码大致如下：

```
def beep(seconds):
    GPIO.output(PIN_NO_BEEP, GPIO.LOW)
    GPIO.output(PIN_NO_LED, GPIO.LOW)
```

```
    time.sleep(seconds)
    GPIO.output(PIN_NO_BEEP, GPIO.HIGH)
    GPIO.output(PIN_NO_LED, GPIO.HIGH)
def beepAction(secs, sleepsecs, times):
    for i in range(times):
        beep(secs)
        time.sleep(sleepsecs)
if ("%02d:%02d:%02d" % (h, m, s)) >= __alarm_time and judge == 1:
    __alarm_beep_status = True
    __alarm_beep_times = 0
    judge = 0
if __alarm_beep_times > 30:
    __alarm_beep_status = False
    __alarm_beep_times = 0
```

第四个模块则是深度唤醒模块。我们的深度唤醒模块用这样的措施实现，即：我们首先定义军刀扩展版上的两个轻触按键分别代表 0 和 1，之后，当系统时间到达闹钟预设时间后，我们操作树莓派在军刀扩展板的数字显示屏上随机生成一个由 0 与 1 两个数组成的四位数，用户需自行判读显示屏上的四位数，并正确地通过轻触按钮输入显示屏上的四位数才能关闭闹钟。我们的深度唤醒模块的实质在于通过改变关闭闹钟的机制，迫使用户在关闭闹钟时必须保持相对清醒，从而到达更好地唤醒用户，使用户在关闭闹钟的过程中从睡眠状态恢复清醒，以此达到我们的设计初衷。
以下代码为在显示屏上生成 4 位随机二进制数的函数:

```
flag = 0
abc = 0
global abc
flag = random.randint(1,4)
if flag == 1:
    SAKS.digital_display.show("0000")
    abc = 0
elif flag == 2:
    SAKS.digital_display.show("1100")
    abc = 11110000
elif flag == 3:
    SAKS.digital_display.show("0011")
    abc = 1111
elif flag == 4:
    SAKS.digital_display.show("1111")
    abc = 11111111
```

4 后续工作展望

我们的多模式唤醒系统仍有一定的改进空间，我们拟在以下方面进一步完善：

1. 实现通过树莓派按键设定闹钟时间的功能

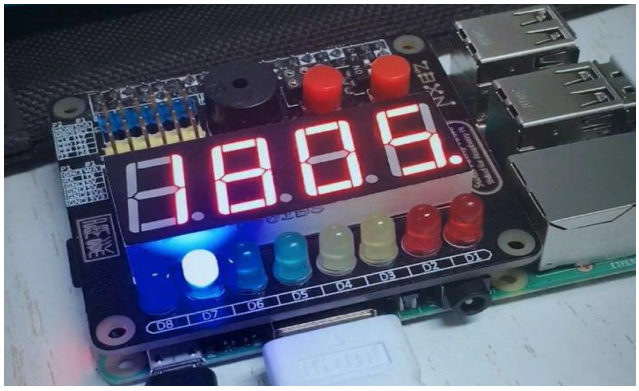
目前设计方案中时间设定只能在代码中实现，可能给实际应用带来不便。我们希望在后续改进中实现在树莓派上利用按键设定闹钟小时和分钟的功能。

2. 闹钟播放音效或者歌曲

我们的设计中闹铃声音是通过蜂鸣器实现的，未来改进中可以考虑将其他音乐或者特殊音效设置为闹铃。

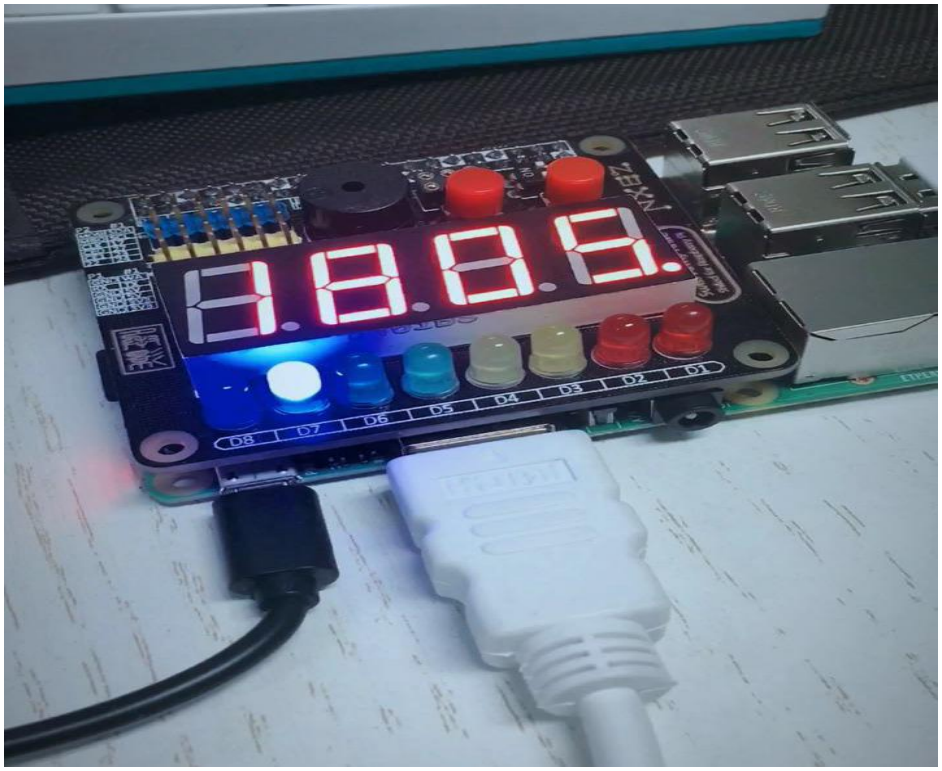
3. 添加其他硬件（如拳击手套）丰富唤醒方式

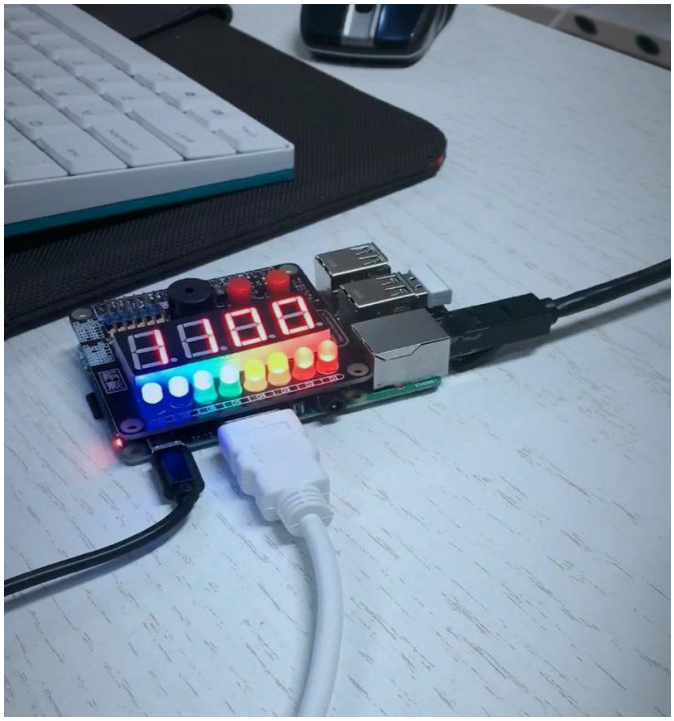
添加其他硬件如外接拳击手套来丰富唤醒方式是我们的设想之一，以后的改进中我们也会继续在这方面努力。



5 小组分工合作

寻找资料、创意和构思、编程实现均为小组成员共同完成





以触控开关为输入端的简易游戏设计

——"树莓派创意编程活动"实验报告

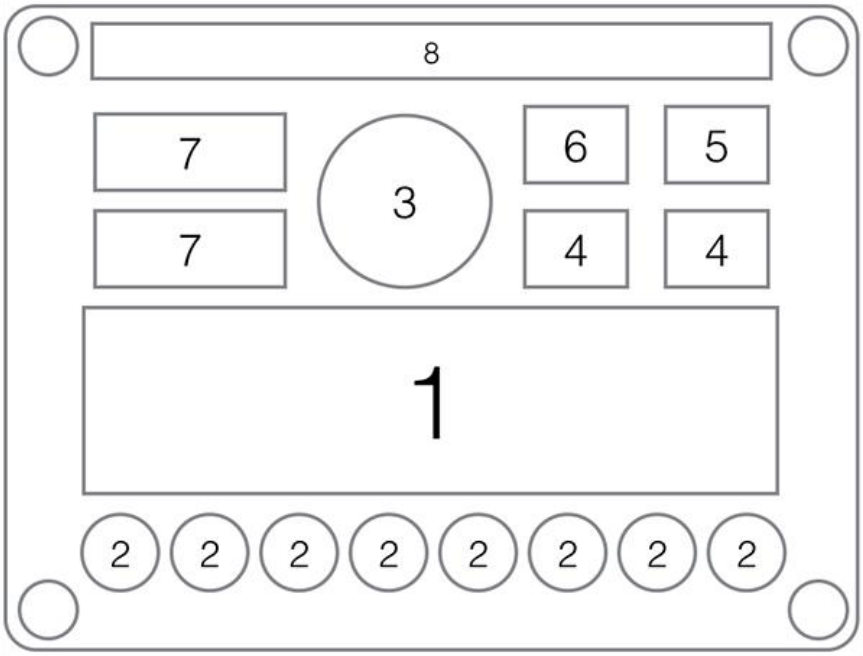
地球与空间科学学院
郭浩 罗哲楷

摘要：以树莓派单片机（RaspberryPi）和瑞士军刀扩展板（SAKS）为硬件设备，SAKSSDK 开发包为编程工具，实现反应时间测试、计时能力测试、LED 的全部点亮等游戏.
关键词：瑞士军刀扩展板；python 应用；树莓派

一、背景材料

1.瑞士军刀扩展板

树莓派瑞士军刀扩展板（Swiss Army Knife Shield for Raspberry Pi）由 NXEZ 设计，集成了多种常用电气元件.示意图如下：（来自 www.nxez.com）



本项目涉及的元件有：1—4 位数码管；2—LED 灯；3—有源蜂鸣器；4—轻触开关 2 个。

2. SAKSSDK

SAKSSDK 对用于树莓派硬件编程的 RPi.GPIO 模块作出了较好封装，可以方便地调用元件对象的方法来实现所期望的功能.

本项目涉及的类主要有 LedRow、TactRow、Buzzer 和 Digital_Display.对于所调用方法的介绍，可参见 wiki.nxez.com/saks:sdk.

特别要提到的是触控开关操作事件的捕捉.SAKSHAT 类中定义了 tact_event_handler 这一函数实现对触控开关状态改变的响应.它有两个参数：pin 代表操作的开关编号，status 代表改变后开关的的状态.程序中对这一函数进行定义，即可实现对玩家操作的处理.

3. python 模块

本项目还用到 python 的以下模块：

Time 模块：time()方法获取系统时间，用于计时；sleep()方法实现灯和数码管的延时控

制.

Random 模块：randrange()方法产生随机数.

Math 模块：floor()方法取整.

二、反应时间测试游戏（Beepcatch）

1.创意来源

小时候在科普展览中体验过反应时间测试游戏. 联想到 SAKS 的蜂鸣器可给出鲜明的提示信号，数码管可起到较好的显示效果，于是希望在树莓派机上实现这一简单的游戏.

2.设计方案

共三轮测试，每轮在随机等待若干秒后触发蜂鸣器，并记下开始时间；tact 处理函数收到开关被按下的信息后记下截止时间，立即将时间差显示在数码管上. 经过测试，tact 函数与主函数是双线并行执行的，所以主函数等待 3 秒，这足够 tact 函数完成处理，之后进入下一轮.

三轮结束后，根据评分公式统计出三次平均分，显示在数码管上. 程序还会根据分数等级点亮相应的灯.

经过初步测试，我的成绩大多在 0.22 秒上下，最好为 0.185 秒，而只要精神专注，反应时间不会超过 0.3 秒. 故评分函数 mark 选取 0.5 秒（及以上）0 分、0.3 秒 60 分、0.18 秒 90 分为控制点，中间用分段线性函数连接.

最终程序由主程序、tact 函数、mark 函数三部分构成.

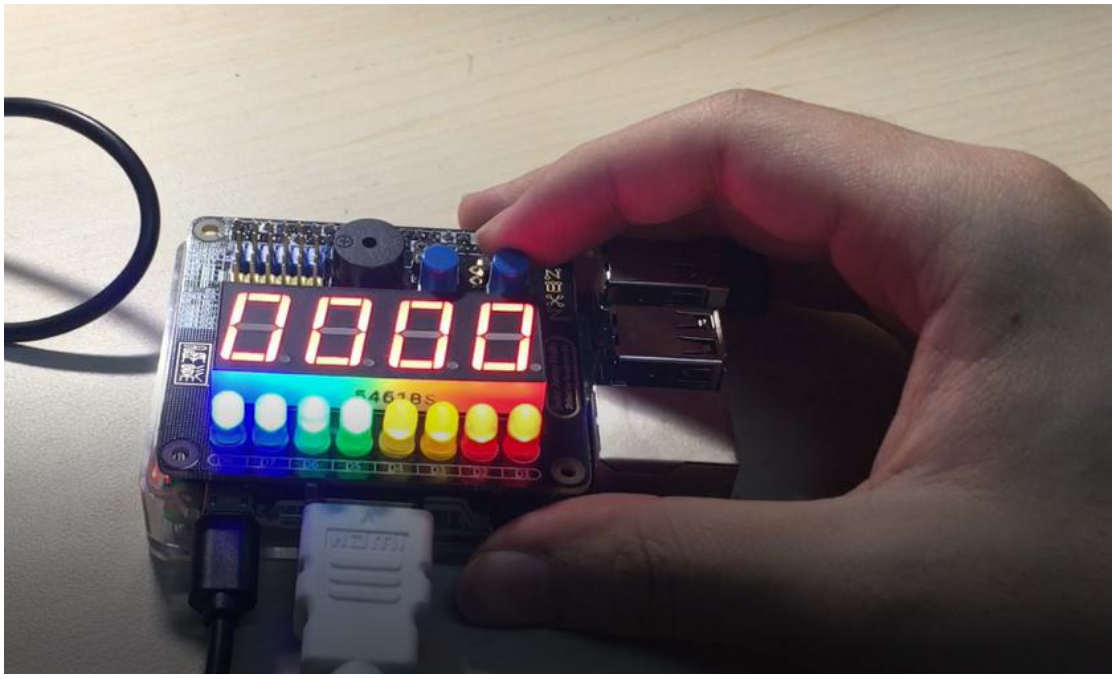
3.代码实现

```
from sakshat import SAKSHAT
import time
import random
import math

SAKS=SAKSHAT()
buz=SAKS.buzzer
ledr=SAKS.ledrow
digr=SAKS.digital_display
tactr=SAKS.tactrow

#蜂鸣器响五声，LED 灯全亮，数码管显示 0000,3 秒后一起熄灭，提醒玩家游戏开始
buz.beepAction(0.2,0.2,5)
ledr.on()
digr.show("0000")
time.sleep(3)
ledr.off()
digr.off()


def tact(pin,status):#设置按键动作
    if status:
```

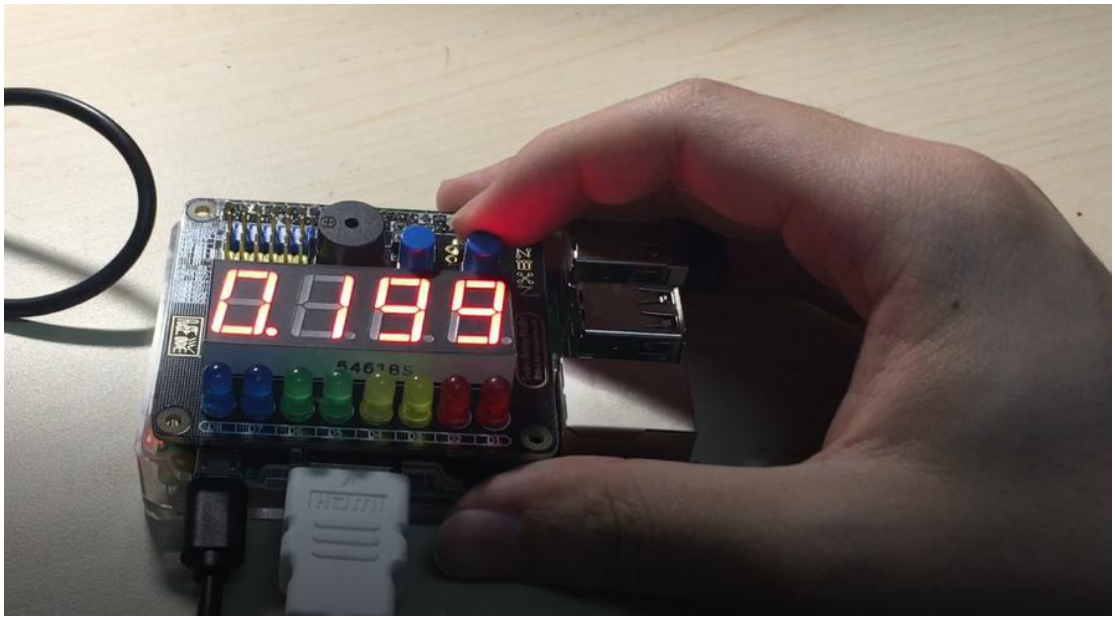
```
            * ct=time.time()                          #记录按下按键的时刻
            gap=* ct-prest                            #用按下按键的时间减去蜂鸣器响起的时间作
为反应时间
            print(gap)
            a=(math.floor(gap))%10                    #算出反应时间各位数字
            b=(math.floor(10*gap))%10
            c=(math.floor(100*gap))%10
            d=(math.floor(1000*gap))%10
            digr.show("%d.%d%d%d"%(a,b,c,d,))         #在数码管上显示反应时间
            time.sleep(1)
            result=a+0.1*b+0.01*c+0.001*d
            resultlst.append(mark(result))            #将本次游戏的结果计入结果列表
            digr.off()                                #熄灭数码管


def mark(gap):#设置计分方式
    if gap>=0.5:
        return 0
    elif gap>=0.3:
        return 150-300*gap
    elif gap>=0.18:
        return 135-250*gap
    else:
        return 100-55*gap


SAKS.tact_event_handler=tact                          #开始按键操作

resultlst=[]                                          #创建记录结果的列表
for round in range(3): #进行 3 次游戏
    beept=3+random.randrange(7)                       #随机设置一个大于 3 秒的等待时间
    time.sleep(beept)
    prest=time.time()                                 #记录下开始时刻
    buz.beep(0.03)                                    #蜂鸣器响起
time.sleep(3)

SAKS.tact_event_handler=None                          #停止按键操作
print(resultlst)
average=(resultlst[0]+resultlst[1]+resultlst[2])/3 #计算 3 次结果的平均值
a=(average//10)%10                                    #计算平均结果的各位数字
b=(math.floor(average))%10
c=(math.floor(10*average))%10
d=(math.floor(100*average))%10
digr.show("%d%d.%d%d"%(a,b,c,d,))                     #在数码管显示平均结果
```

```
if average>=60:                                    #不同位置的 LED 灯根据结果的好坏亮起
    ledr.on_for_index((math.floor(average)-60)//5)
time.sleep(3)                                      #停止 3 秒后全部熄灭游戏结束
ledr.off()
digr.off()
```

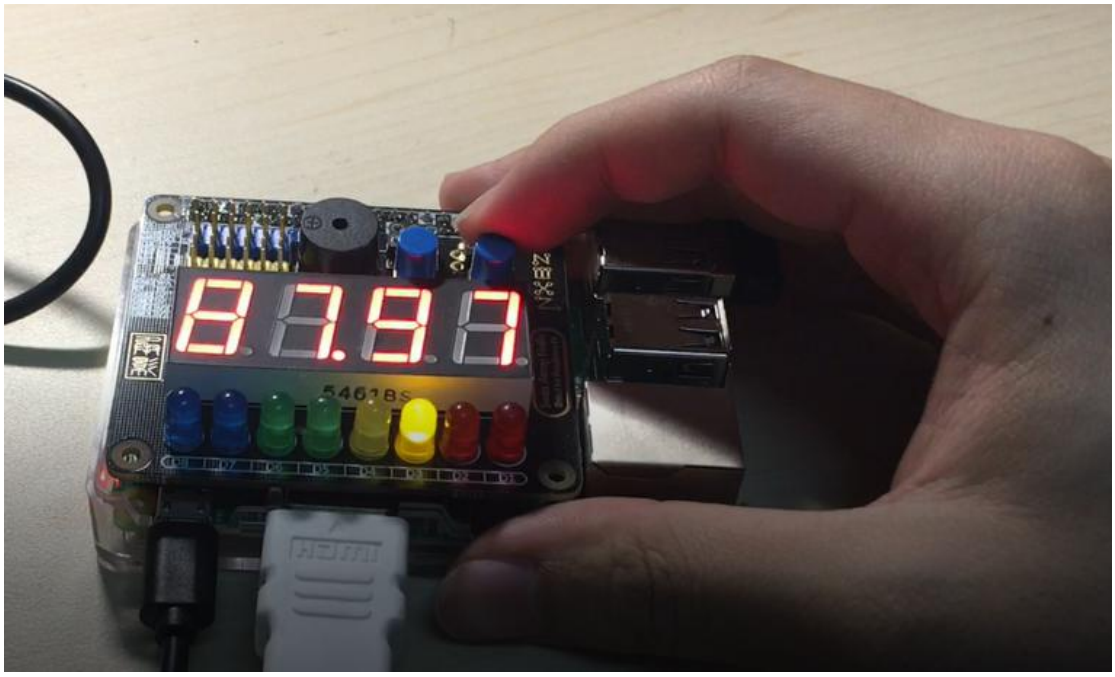
4.实现效果



游戏开始提示



单次反应时间显示



最终成绩

三、计时游戏（Timer Test）

1.创意来源

请你在不看手表的条件下，按照自己对 1 秒长度的概念计时. 当你认为 30 秒过去，看看表，实际过了多少秒？

这是郭浩的一家人一起走路时为打发时间曾玩过的游戏，虽然简单，但确实可以检测和训练玩家对时间的感觉.

2.设计方案

整体设计几乎与 Beepcatch 相同，只是要测量的时间变长了.

同样是三轮测试，要求计的时间从一个 5 到 30（秒）之间的长为 10 的列表中选出. 选出的时间在数码管上显示 2 秒后蜂鸣器响，计时开始. 同样由 tact 函数处理按下事件. 主程序等待要求时间的 1.5 倍后进入下一轮.

评分函数 mark 沿用线性函数，分数是根据相对误差，即时间差与要求时间的比值计算的.

3.代码实现

```
from sakshat import SAKSHAT
import time
import random
import math

def mark(errorpc):#根据估计误差计算分数
    if errorpc>0.5:
        return 0
    elif errorpc>0.25:
```

```
        return 120-240*errorpc
    elif errorpc>0.1:
        return 85-100*errorpc
    else:
        return 100-150*errorpc

def tact(pin,status):#设置按键动作
    if status:
        endtime=time.time()                          #记录按下按键的时刻
        yourtime=endtime-starttime      #用按下按键的时间减去蜂鸣器响起的时间作为估计时间
        print(yourtime)
        a=(math.floor(yourtime))//10                 #算出估计时间各位数字
        b=(math.floor(yourtime))%10
        c=(math.floor(10*yourtime))%10
        d=(math.floor(100*yourtime))%10
        digr.show("%d%d.%d%d"%(a,b,c,d,))            #在数码管上显示估计时间
        marklist.append(mark(abs(yourtime-acqtime)/acqtime))    #将本次游戏的结果计入结果列表
        time.sleep(1)
        digr.off()

SAKS=SAKSHAT()
ledr=SAKS.ledrow
tactr=SAKS.tactrow
digr=SAKS.digital_display
buz=SAKS.buzzer

#蜂鸣器响五声，LED 灯全亮，数码管显示 0000,3 秒后一起熄灭，提醒玩家游戏开始
buz.beepAction(0.2,0.2,5)
ledr.on()
digr.show("0000")
time.sleep(3)
ledr.off()

acqlist=[5,6,7,8,10,12,15,18,24,30]                  #设置要求时间的列表
marklist=[]                                          #创建记录结果的列表
for round in range(3):#进行 3 次游戏
    acqtime=acqlist[random.randrange(10)]            #在要求时间的列表里随机选取一个作为本
次游戏的要求时间
    digr.show("%d%d.00"%(acqtime//10,acqtime%10,))   #显示本次游戏要求的时间
    time.sleep(2)                                    #2 秒后蜂鸣器响起
    buz.beep(0.3)
    starttime=time.time()                            #记录开始时间
    SAKS.tact_event_handler=tact                     #开始按键操作
    time.sleep(acqtime*1.5)                          #等待玩家按下按键
```

```
    SAKS.tact_event_handler=None                    #停止按键操作

while len(marklist)<3:                              #若无操作则自动记为0分
    marklist.append(0)
print(marklist)
average=(marklist[0]+marklist[1]+marklist[2])/3     #计算3次结果的平均值
a=(math.floor(average))//10                         #计算平均结果的各位数字
b=(math.floor(average))%10
c=(math.floor(10*average))%10
d=(math.floor(100*average))%10
digr.show("%d%d.%d%d"%(a,b,c,d,))                   #在数码管显示平均结果
if average>60:                                      #不同位置的LED灯根据结果的好坏亮起
    ledr.on_for_index((math.floor(average)-60)//5)
time.sleep(5)                                       #停止3秒后全部熄灭游戏结束
digr.off()
ledr.off()
```

四、LED 灯的全部点亮（LED Puzzle）

1.创意来源

郭浩是冒险解谜游戏的爱好者，此类游戏中常有各部件连锁的机关，如一排灯，当操作一盏灯时其它某些灯的状态也随之改变，这时一个原本简单的任务就变成了一个小型智力游戏.

我们利用 8 个 LED 灯实现了一个此类游戏. 规则如下：一排 8 个灯依次编为 0-7 号，每个灯只可以与相邻的两盏灯（0 号和 7 号为相邻的一盏灯）同时改变状态，即开变为关，关变为开，这记为一次操作. 在给定的初始状态下，玩家需要用尽可能少的操作，在尽可能短的时间内把灯全部点亮.

2.设计方案

程序可分为主程序、tact 函数、end 函数（结束处理）、最优解模块（最优解列表及其调用）和一系列辅助函数（switch，lightcount，rowstat_to_int）.

为保证问题可解，主程序先从全亮状态打乱 15-30 步，得到一个 puzzle.

游戏过程中，数码管显示已进行的操作数和当前的操作灯位. 左触控开关用于移动操作灯位，右触控开关用于给出改变状态的指令. 这一部分由 tact 实现.

每次改变状态时，检查灯是否已全部点亮，若已点亮，游戏胜利;若操作数超过上限 99 步，游戏失败. 以上两种情况均触发 end 函数进行最终评分.评分时只要成功即得基本分 60 分，时间分 20 分，步数分 20 分.

3.数学描述与最小值求解

添加最优解模块之前，我们发现此游戏过于简单，一般不足 10 步即可完成，于是有必要在评分上严格要求，即以可能的最短步数为标准，每多用一步都要扣分. 这就有必要探究所有可能的 256 种状态是否都可以全部点亮，然后用广度优先搜索算法求出可以点亮的状态的最少步数，作为已知数据存在程序中.

李宇轩与郭浩在讨论中揭示了 256 个状态在给定操作下的关系，简要叙述如下：

1）第 0、1、3、4、6、7 六盏灯中点亮的灯数的奇偶性是给定操作下的不变量；

2）进一步，全部的 256 种状态可归为给定操作下的两个等价类，每类所包含的状态可相互转换.第 0、1、3、4、6、7 六盏灯中点亮的灯数为偶数的 128 种状态为其中一类，都可以转化为全部点亮的状态.

接下来，郭浩以此为依据编写程序，实现了对全部 256 种状态的可行性判定和最小步数求解.代码如下：

```
# coding=utf-8
# Led Puzzle-最少步数计算

# 二进制数 i 的从右往左第 k 位代表第 k 盏灯的开闭（这里 k 取 1-8）
# 1 为开，0 为关

leaststeps=[]   # 记录最少步数值的空列表

# 获取二进制数 n 的第 k 位
def getdig(n,k):
    return (n//(2**(k-1))) % 2

# 反转二进制数 n 的第 k 位
def revdig(n,k):
    if getdig(n,k) == 1:
        return n-2**(k-1)
    else:
        return n+2**(k-1)

# 根据已证明的数学结论，判断一个初始序列可否到达全亮状态
def able(n):
    sum = getdig(n,1)+getdig(n,2)+getdig(n,4)+getdig(n,5)+getdig(n,7)+getdig(n,8)
    if sum % 2 == 1:
        return 0
    return 1

# 实现游戏规定的操作
def alter(n,k):
    n=revdig(n,k)
    if k!=1:
        n=revdig(n,k-1)
    if k!=8:
        n=revdig(n,k+1)
    return int(n)

# 计算二进制数 n 确定的最小序列
def LeastStep(n):
    if not able(n):
        return -1
```

```
    dis = [-1 for j in range(256)]
    dis[n] = 0
    get = False
    round = 0
    # 广度优先搜索
    while not get:
        for i in range(256):
            if dis[i] == round:
                for k in range(1,9):
                    if dis[alter(i, k)] == -1:
                        dis[alter(i, k)] = round+1   # 新到达的状态标记最小距离
        round += 1
        if dis[255]>0:
            get=True   # 到达全亮状态，停止
    return dis[255]

for i in range(255):
    leaststeps.append(LeastStep(i))
leaststeps.append(0)
print(leaststeps)
```

4.代码实现

```
# coding=utf-8
# Led Puzzle

from sakshat import SAKSHAT
from sakspins import SAKSPins as PINS
import time
import random
import math

# 预先求好的各状态最少步数，-1 表示不可到达
Leaststeps=[3, -1, -1, 2, 3, -1, -1, 2, -1, 3, 4, -1, -1, 3, 4, -1, -1, 2, 3, -1, -1, 4,
3, -1, \
            2, -1, -1, 3, 2, -1, -1, 1, 3, -1, -1, 2, 3, -1, -1, 2, -1, 3, 4, -1, -1, 3,
4, -1, \
            -1, 2, 3, -1, -1, 4, 3, -1, 2, -1, -1, 3, 2, -1, -1, 1, -1, 3, 4, -1, -1, 5,
4, -1, \
            3, -1, -1, 4, 3, -1, -1, 2, 4, -1, -1, 3, 4, -1, -1, 3, -1, 4, 5, -1, -1, 4,
5, -1, \
            -1, 3, 4, -1, -1, 5, 4, -1, 3, -1, -1, 4, 3, -1, -1, 2, 4, -1, -1, 3, 4, -1,
-1, 3, \
            -1, 4, 5, -1, -1, 4, 5, -1, -1, 2, 3, -1, -1, 4, 3, -1, 2, -1, -1, 3, 2, -1,
-1, 1, \
```

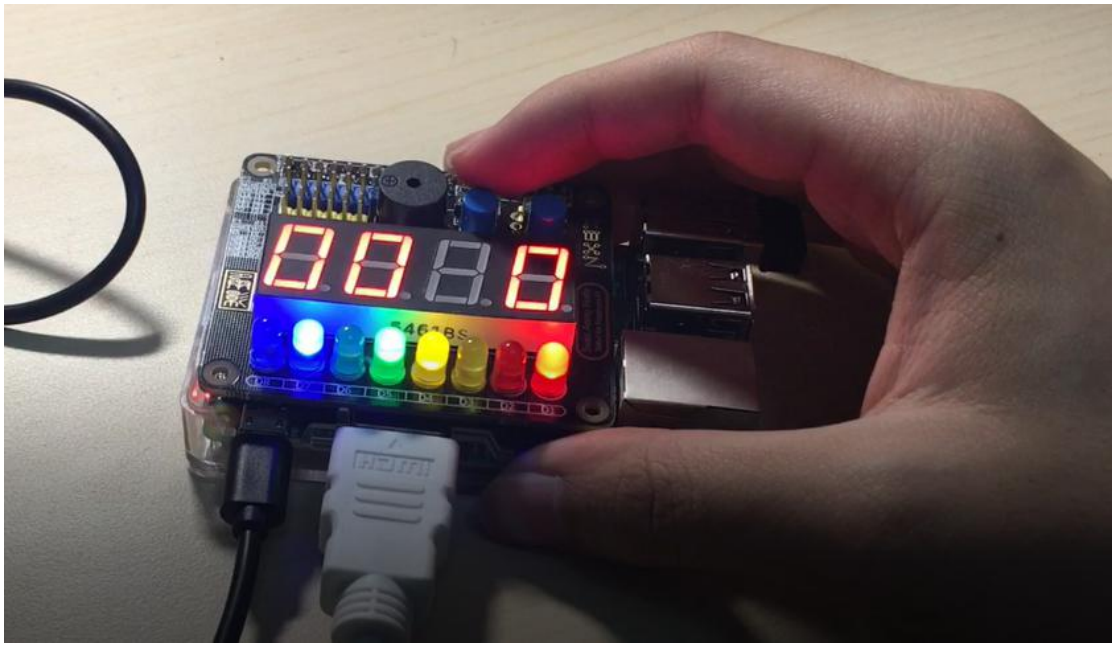
```
            3, -1, -1, 2, 3, -1, -1, 4, -1, 5, 4, -1, -1, 3, 4, -1, -1, 4, 5, -1, -1, 4,
5, -1, \
            4, -1, -1, 3, 4, -1, -1, 3, 3, -1, -1, 4, 3, -1, -1, 2, -1, 3, 4, -1, -1, 5,
4, -1, \
            2, -1, -1, 3, 2, -1, -1, 1, -1, 2, 3, -1, -1, 4, 3, -1, -1, 3, 4, -1, -1, 3,
4, -1, \
            3, -1, -1, 2, 3, -1, -1, 2, 2, -1, -1, 1, 2, -1, -1, 3, -1, 4, 3, -1, -1, 2,
3, -1, \
            -1, 1, 2, -1, -1, 3, 2, -1, 1, -1, -1, 2, 1, -1, -1, 0]

# bool 值状态转换
def switch(stat):
    if stat==False:
        return True
    else:
        return False

# 统计当前点亮的灯数
def lightcounter():
    lighted=0
    for index in range(8):
        if ledr.is_on(index):
            lighted+=1
    return lighted

# 将 bool 列表转化为整数，以便调用最少步数
def rowstat_to_int(rowstat):
    res=0
    for i in rowstat:
        res*=2
        if i==True:
            res+=1
    print(res)
    print(Leaststeps[res])
    return res

# 程序结束处理（计分与显示）
def end(status):
    SAKS.tact_event_handler=None
    if status==0:  # 失败
        lights=lightcounter()
        mark=5*lights
        digr.show("%d%d.00"%(mark//10,mark%10,))
        ledr.off()
```

```
    buz.beep(3)
    digr.off()
    input("Press 'ENTER' to exit")
    exit()
elif status==1:  # 成功
    finishtime=time.time()
    usetime=finishtime-begintime
    stepdiff=actstep-best
    print(usetime)
    print(stepdiff)
    # 分段计算时间分
    if usetime>130:
        timemark=0
    elif usetime>30:
        timemark=130-usetime/10
    else:
        timemark=20-usetime/3
    print(timemark)
    # 计算步数分
    if stepdiff<=10:
        stepmark=20-stepdiff*2
    elif stepdiff>10:
        stepmark=0
    else:  # 步数比最小步数更少，将报错
        digr.show("9999")
        time.sleep(3)
        exit()
    print(stepmark)
    result =60+timemark+stepmark
    print(result)

    a=(math.floor(result))//10
    b=(math.floor(result))%10
    c=(math.floor(10*result))%10
    d=(math.floor(100*result))%10
    digr.show("%d%d.%d%d"%(a,b,c,d,))
    buz.beepAction(0.3,0.3,3)
    # 流水灯庆祝
    for round in range(20):
        for index in range(8):
            ledr.on_for_index(index)
            ledr.off_for_index((index-1)%8)
            time.sleep(0.02)
    digr.off()
```

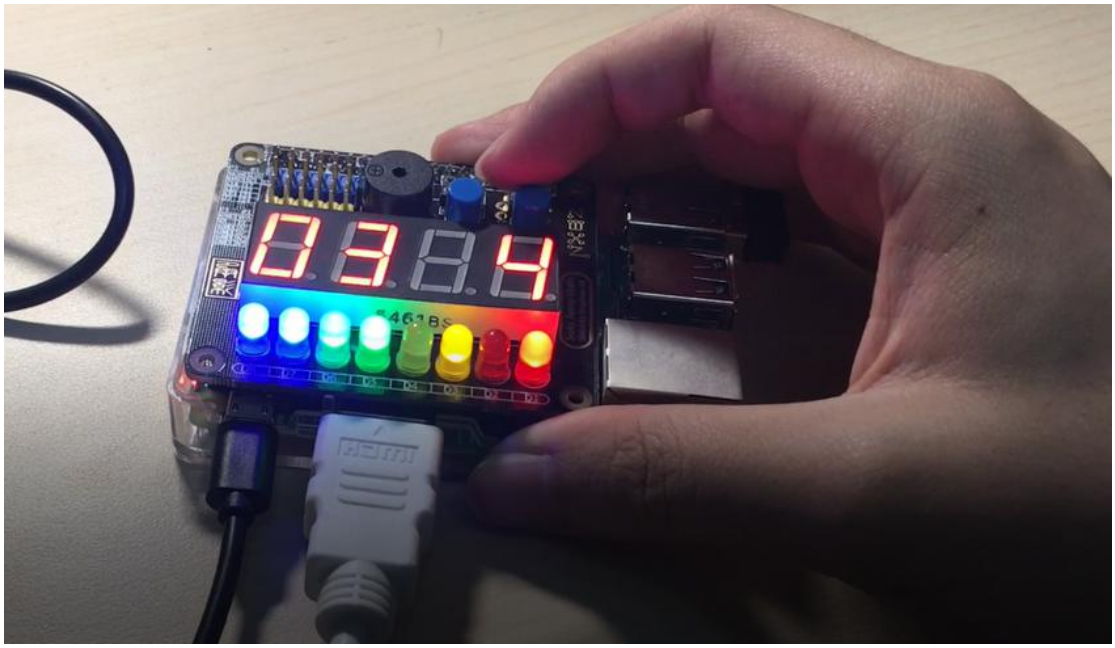
```
        ledr.off()
        input("Press 'ENTER' to exit")
        exit()

# 触控开关事件处理
def tact(pin,status):
    global curdig,actstep
    if status==False:
        return
    # 左开关移动所控制的位置
    if pin==PINS.TACT_LEFT:
        curdig=(curdig+1)%8
        digr.show("%d%d#%d"%(actstep//10,actstep%10,curdig,))
    # 右开关实现规定操作一次
    elif pin==PINS.TACT_RIGHT:
        rowstat[curdig]=switch(rowstat[curdig])
        if curdig!=0:
            rowstat[curdig-1]=switch(rowstat[curdig-1])
        if curdig!=7:
            rowstat[curdig+1]=switch(rowstat[curdig+1])
        ledr.set_row(rowstat)
        actstep+=1
        if actstep>99:
            end(0)
            return
        digr.show("%d%d#%d"%(actstep//10,actstep%10,curdig,))
        if lightcounter()==8:
            end(1)
            return

SAKS=SAKSHAT()
ledr=SAKS.ledrow
tactr=SAKS.tactrow
digr=SAKS.digital_display
buz=SAKS.buzzer
# 开始提示
buz.beepAction(0.2,0.2,5)
ledr.on()
digr.show("0000")
time.sleep(3)
ledr.off()

# 生成 puzzle
rowstat=[True]*8
```

```
steps=random.randrange(15,30)
for step in range(steps):
    dig=random.randrange(8)
    rowstat[dig]=switch(rowstat[dig])
    if dig!=0:
        rowstat[dig-1]=switch(rowstat[dig-1])
    if dig!=7:
        rowstat[dig+1]=switch(rowstat[dig+1])
best=Leaststeps[rowstat_to_int(rowstat)]
ledr.set_row(rowstat)
SAKS.tact_event_handler=tact
begintime=time.time()
digr.show("00#0")
curdig=0
actstep=0
```

5.实现结果



新生成的 Puzzle



操作过程

最终结果

五、后续工作展望

本次活动中我们编写的反应时间测试游戏与 LED 解密游戏已初步取得良好效果，但同时它们还有很大的发展空间.

对于反应时间测试游戏，我们可以在单一的听觉反应测试的基础上添加视觉反应测试，具体来说，可以在蜂鸣器鸣响之外添加 LED 灯闪亮这一提示方式，要求玩家在听到蜂鸣器鸣响后按下左边开关，在看到 LED 灯闪亮后按下右侧开关，这样游戏就能锻炼玩家全方位注意力把握的能力，可以用于多任务同时处理能力的训练.

对于 LED 解密游戏，我们可以在数学上将游戏由 8 个 LED 灯推广到 n 个 LED 灯的情况，

探究其内在性质；在计算最优解的步数时，可以采用动态规划算法来代替原先的广度优先搜索算法，从而简化计算。

六、分工合作

本次活动中，组长郭浩主要负责树莓派开发包代码的研究、主要代码的编写以及实验报告的书写。组员罗哲楷主要负责树莓派的开机与使用的探索、代码与报告的完善、PPT 和演示视频的制作。

七、总结与致谢

本次活动是我们对单片机、硬件编程的初次接触. 这一过程中我们自己学习树莓派和 SAKSSDK 开发包的使用，并把最初的设想变成了现实. 对我们而言，这是非常宝贵的经历. 当然，我们深知我们的作品还处在初等的阶段，而且有的细节也没有达到最理想的效果，我们需要学习多线程编程和更深入的硬件开发知识才能做得更好.

感谢 SESSDSA 团队主办此次活动，并为我们提供了器材；感谢李宇轩同学，他在 Led 灯问题的数学描述中做出了关键性的贡献；感谢郭晴宇、王玉琦、董子安同学，他们参与了游戏的测试.

2017 年 4 月 15 日初稿
2017 年 4 月 19 日改定

树莓派 Web 遥控小车实习报告

张峻伟 信息科学技术学院
陈梦珂 信息管理系

【摘要】本设计基于树莓派 3 代 B 型开发板，搭配 L298N 系统小车，构成主体硬件架构。利用客户端/服务器（C/S）模型，通过移动终端远程控制树莓派 GPIO 接口输出高低电平，来控制小车的运动。LED 显示屏可显示小车的行进状态，并和蜂鸣报警器一同具有紧急报警功能。超声波测距探测前方障碍物的距离，实现紧急制动。

【关键词】树莓派 C/S 结构 Web 遥控小车 超声波测距

一、选题及创意介绍

选题来自电影《太空旅客》中男女主角第一次约会时所用的遥控小车，我们基于树莓派想要制作一个在 WiFi 环境下，可以通过前端网页控制，后台服务器响应，实现小车的基本行进、实时测距、检测障碍物、碰撞警报与紧急制动等功能。

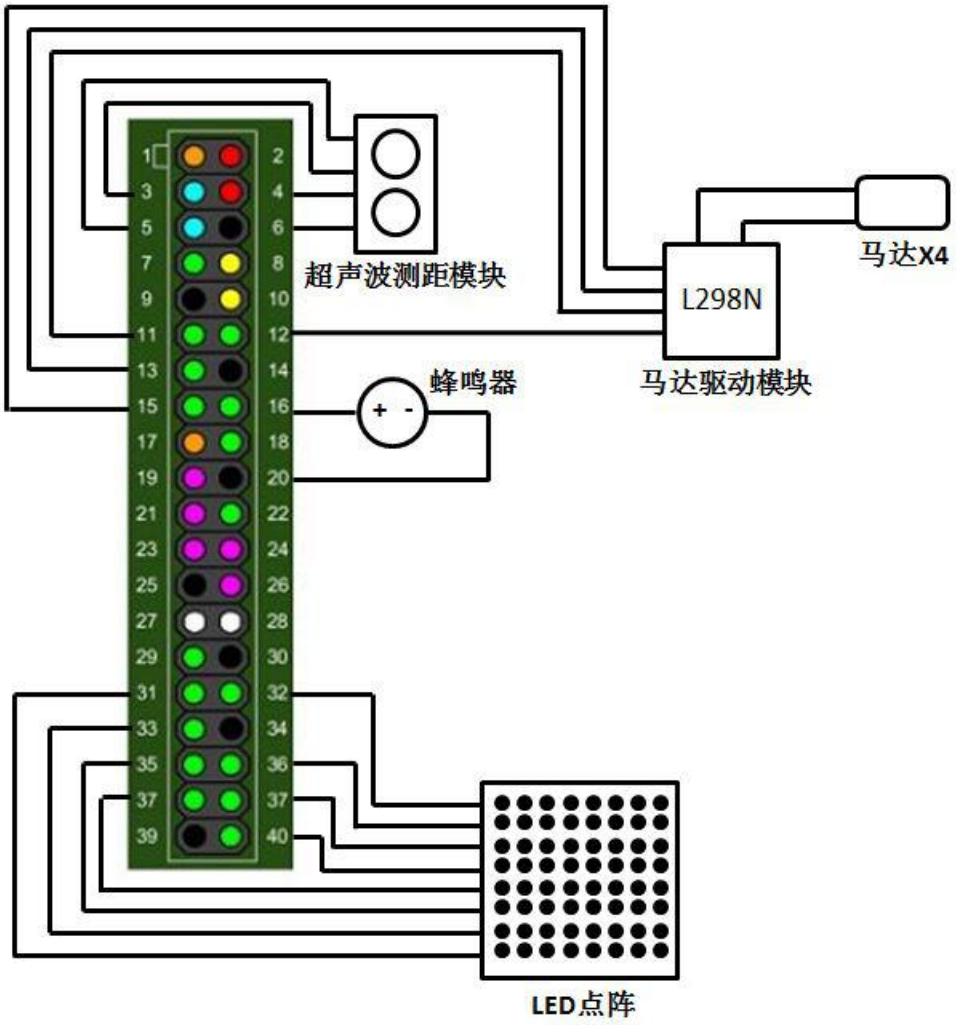
基于树莓派的 Web 小车不同于传统依靠红外线发射接收信号来控制的遥控小车，仅需要树莓派与用户位于在同一无线局域网（WLAN）下，用户便能通过前端 Web 界面自主控制小车。插入无线网卡，设备发出信号，如果有摄像头，小车便能在无人环境中边行进边监测周围环境。在排爆排险、古物探测（《功夫瑜伽》中的海底环境即通过一个可以游泳的小车监测）、刑事侦查等方面具有重要意义。

二、设计方案

1、项目组件与原理图

项目组件包括：树莓派主板、小车底盘、马达、杜邦线、移动电源、L298N 电机驱动模块、超声波测距模块、LED 点阵显示屏、八块 5 号电池、蜂鸣报警器等。

项目原理图如下：



树莓派的 GPIO 接口通过 L298N 驱动，控制电机运转，实现小车的前进、后退和变向。马达驱动模块需要外接电源 12V。LED 点阵显示屏和蜂鸣报警器与树莓派的 GPIO 相应接口连接。

2、服务器程序

本次实验借助 Python 程序 cherrypy 模块来在树莓派上搭建网络服务器。小车的基本行进功能通过编写 Python 程序控制树莓派的 GPIO 接口输出高低电平来实现。

超声波雷达测距通过超声波遇到前方障碍物返回的时间差来计算，所得距离返回到服务器，并通过前端界面实时更新响应，返回给用户。

服务器端 Python 程序还包括控制 LED 点阵、蜂鸣器报警等指令。

3、前端 web 页面

前端主要通过开关、前后左右共五个操作按钮来控制小车的开关状态和运行状态。使用了 HTML、JavaScript、CSS 等语言，保证在用户发出请求时能正确调用服务器端的相应函数。

三、实现方案及代码分析

整个项目的代码主要分为两部分：服务器端 Python 程序和前端网页代码。前端网页代码又分为 HTML 和 JavaScript 两部分。下面分别对各部分的关键代码进行分析。

1、服务器端 Python 程序：

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def change_status(self):
  if self.power == 'off':
    self.power = 'on'
    self.normal_sign()
  else:
    self.power = 'off'
    GPIO.cleanup()
  return self.power
```

Change_status 函数的作用是在前端 JavaScript 调用时改变小车的开关状态。小车的开关状态用 CAR 类中的 power 变量表示。如果当下小车的状态是‘off’，就将状态改成‘on’，并且调用 CAR 类中的 normal_sign()函数，让 LED 显示屏亮起来。反之，将状态改成‘off’。Change_status 函数将 power 变量作为返回值传给前端 JavaScript，保证前端能够随时更新小车的开关状态。

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def init_engine(self):
  GPIO.setmode(GPIO.BOARD)
  GPIO.setup(self.IN1, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.IN2, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.IN3, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.IN4, GPIO.OUT, initial = GPIO.LOW)

@cherrypy.expose()
@cherrypy.tools.json_out()
def init_radar(self):
  GPIO.setmode(GPIO.BOARD)
  GPIO.setup(self.Trig, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.Echo, GPIO.IN)

@cherrypy.expose()
@cherrypy.tools.json_out()
def init_buzzer(self):
  GPIO.setmode(GPIO.BOARD)
  GPIO.setup(self.BUZZ_NO, GPIO.OUT, initial = GPIO.LOW)

@cherrypy.expose()
@cherrypy.tools.json_out()
def init_led(self):
  GPIO.setmode(GPIO.BOARD)
  GPIO.setup(self.PIN1, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN2, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN3, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN4, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN5, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN6, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN7, GPIO.OUT, initial = GPIO.LOW)
  GPIO.setup(self.PIN8, GPIO.OUT, initial = GPIO.LOW)
```

这一部分是树莓派小车的初始化函数部分。初始化函数集包含四类，分别代表初始化马达，初始化超声波测距模块、初始化蜂鸣器、初始化 LED 点阵。初始化函数不仅在小车刚开始运作时调用，运作过程中为了将某些功能还原到初始状态也会随时调用该函数。比如，用户在网页上松开前进按钮时就会调用 init_engine()函数，让小车停下来。

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def forward(self):
    self.init_engine()
    GPIO.output(self.IN1, GPIO.HIGH)
    GPIO.output(self.IN2, GPIO.LOW)
    GPIO.output(self.IN3, GPIO.HIGH)
    GPIO.output(self.IN4, GPIO.LOW)

@cherrypy.expose()
@cherrypy.tools.json_out()
def left(self):
    self.init_engine()
    GPIO.output(self.IN3, GPIO.HIGH)
    GPIO.output(self.IN4, GPIO.LOW)

@cherrypy.expose()
@cherrypy.tools.json_out()
def right(self):
    self.init_engine()
    GPIO.output(self.IN1, GPIO.HIGH)
    GPIO.output(self.IN2, GPIO.LOW)

@cherrypy.expose()
@cherrypy.tools.json_out()
def backward(self):
    self.init_engine()
    GPIO.output(self.IN1, GPIO.LOW)
    GPIO.output(self.IN2, GPIO.HIGH)
    GPIO.output(self.IN3, GPIO.LOW)
    GPIO.output(self.IN4, GPIO.HIGH)
```

这部分代码的功能是控制小车运动。四个函数分别控制小车前进、左转、右转和后退。为了安全起见，每个函数在执行时，都先调用了马达初始化函数。管脚电平的高低与小车运动状态的关系是查阅马达驱动模块 L298N 的标准后得到的。

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def buzzer(self):
  self.init_buzzer()
  GPIO.output(self.BUZZ_NO, GPIO.HIGH)

@cherrypy.expose()
@cherrypy.tools.json_out()
def debuzzer(self):
  self.init_buzzer()
  GPIO.output(self.BUZZ_NO, GPIO.LOW)
```

这部分代码控制蜂鸣器的鸣叫。Buzzer()函数让蜂鸣器鸣叫，debuzzer()函数让蜂鸣器停止鸣叫。

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def check_distance(self):
  self.init_radar()
  GPIO.output(self.Trig, GPIO.HIGH)
  time.sleep(0.000015)
  GPIO.output(self.Trig, GPIO.LOW)
  while not GPIO.input(self.Echo):
          pass
  t1 = time.time()
  while GPIO.input(self.Echo):
          pass
  t2 = time.time()
  result = (t2-t1)*340/2
  print(result)
  return {'distance':'%.3f' % result, 'unit':' m'}
```

这部分代码的功能是通过控制超声波测距模块工作来计算小车与前方障碍物的距离，并将距离值返回到前端网页上实时显示。该模块的工作原理为，先向 Trig 管脚输入 15us 的触发信号。一旦检测到有回波信号则 Echo 管脚输出高电平回响信号。回响信号的脉冲宽度与所测的距离成正比。由此通过发射信号到收到的回响信号时间间隔可以计算得到距离。公式如下：距离=高电平时间*声速/2。

```
@cherrypy.expose()
@cherrypy.tools.json_out()
def normal_sign(self):
        self.init_led()
        GPIO.output(self.PIN1, GPIO.HIGH)
        GPIO.output(self.PIN6, GPIO.HIGH)
        GPIO.output(self.PIN8, GPIO.HIGH)

@cherrypy.expose()
@cherrypy.tools.json_out()
def normal_sign2(self):
        self.init_led()
        GPIO.output(self.PIN4, GPIO.HIGH)
        GPIO.output(self.PIN7, GPIO.HIGH)

@cherrypy.expose()
@cherrypy.tools.json_out()
def normal_sign3(self):
        self.init_led()
        GPIO.output(self.PIN2, GPIO.HIGH)
        GPIO.output(self.PIN5, GPIO.HIGH)
        GPIO.output(self.PIN3, GPIO.HIGH)

@cherrypy.expose()
@cherrypy.tools.json_out()
def all_sign(self):
        self.init_led()
        GPIO.output(self.PIN1, GPIO.HIGH)
        GPIO.output(self.PIN2, GPIO.HIGH)
        GPIO.output(self.PIN3, GPIO.HIGH)
        GPIO.output(self.PIN4, GPIO.HIGH)
        GPIO.output(self.PIN5, GPIO.HIGH)
        GPIO.output(self.PIN6, GPIO.HIGH)
        GPIO.output(self.PIN7, GPIO.HIGH)
        GPIO.output(self.PIN8, GPIO.HIGH)
```

这部分代码的功能是控制小车 LED 显示屏的工作。Normal_sign()函数让 LED 显示屏以正常状态显示（记为状态一），normal_sign2()函数与 normal_sign3()函数代表显示屏状态二、三。All_sign()函数让 LED 显示屏全亮。以一定的时序轮流调用前三个状态函数就能够实现小车显示屏上波浪型的动态效果，用于表示小车的前进、后退。交替调用 all_sign()函数与 init_led()函数可以实现闪烁的报警动态效果。

2、前端 JavaScript 程序

```
function timeCount_forward(){
  count = count + 1;
  var xhr = new XMLHttpRequest();
  if(count%3 == 0) xhr.open("POST", "/normal_sign", true);
  else if(count%3 == 1) xhr.open("POST", "/normal_sign2", true);
  else xhr.open("POST", "/normal_sign3", true);
  xhr.send();
  t = setTimeout("timeCount_forward()",150);
}

function timeCount_backward(){
  count = count + 1;
  var xhr = new XMLHttpRequest();
  if(count%3 == 0) xhr.open("POST", "/normal_sign", true);
  else if(count%3 == 1) xhr.open("POST", "/normal_sign3", true);
  else xhr.open("POST", "/normal_sign2", true);
  xhr.send();
  t = setTimeout("timeCount_backward()",150);
}

function flash(){
  count_flash = count_flash + 1;
  var xhr = new XMLHttpRequest();
  if (count_flash % 2 == 0) xhr.open("POST", "/all_sign", true);
  else xhr.open("POST", "/init_led", true);
  xhr.send();
  t_flash = setTimeout("flash()",150);
}

function stopCount(){
  clearTimeout(t);
  clearTimeout(t_flash);
  count = 0;
  count_flash = 0;
  xhr = new XMLHttpRequest();
  xhr.open("POST", "/normal_sign", true);
  xhr.send();
}
```

这四个函数的功能是实现 LED 显示屏不同的动态显示效果。以第一个函数 timeCount_forward()为例。小车前进时调用此函数。函数引用了全局变量 count，t=setTimeout("timeCount_forward()",150)的作用是每隔 150ms 重复调用该函数。每重复调用一次，count 加 1。函数通过计算 count 模三的余数来选择性调用 normal_sign()，normal_sign2()，normal_sign3()中的一个。这样，在不断重复调用的过程中，三个状态函

数被轮流执行，显示屏也就出现了动态效果。后退时原理一样，唯独调用的顺序有变化。

当小车停止前进（不论是因用户松开按钮的主动停止还是检测到碰撞的强制停止），stopCount()函数被调用。执行 ClearTimeout()清除循环调用，并初始化 count 变量，执行 normal_sign()让显示屏以状态一静态显示。

```
function buzz(){
  xhr = new XMLHttpRequest();
  xhr.open("POST", "/buzzer", true);
  xhr.send();
}

function debuzz(){
  xhr = new XMLHttpRequest();
  xhr.open("POST", "/debuzzer", true);
  xhr.send();
}
```

执行 python 中的 buzzer()、debuzzer()函数，控制蜂鸣器的鸣叫。

```
var forward_status = 0;

function forward()
{
  if (status == 'on' && alarm == 'off'){
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/forward", true);
    xhr.send();
    forward_status = 1;
    timeCount_forward();
    return;
  }
  else if(alarm == 'on') {
    buzz();
    flash();
  }
}

function left()
{
  if (status == 'on'){
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/left", true);
    xhr.send();
    return;
  }
}

function right()
{
  if (status == 'on'){
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/right", true);
    xhr.send();
    return;
  }
}
```

```
function backward()
{
  if (status == 'on'){
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/backward", true);
    xhr.send();
    timeCount_backward();
    return;
  }
}


function stop()
{
  if (status == 'on'){
    stopCount();
    debuzz();
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/init_engine", true);
    xhr.send();
    forward_status = 0;
    return;
  }
}
```

控制小车的运动。这部分的所有函数的执行条件都是 status==‘on’。当小车状态尚未开启时，网页上控制小车前进、后退、左右转的按钮都是无效的。函数内部只需要请求执行 python 程序中对应的函数即可。前进、后退函数还需要调用相应的 timeCount 函数控制 LED 显示屏。对于前进 forward()函数，还需要判断 alarm 是否为‘on’。如是，表明前方存在障碍物，小车不能前进，并调用 buzz()和 flash()来发出警报与闪灯提示。

当用户松开前进（或后退、左右转）按钮时，调用 stop()函数。Stop()函数通过执行 python 端 init_engine 函数使小车停止运动。

```
function force_stop(){
  stopCount();
  flash();
  buzz();
  var xhr = new XMLHttpRequest();
  xhr.open("POST", "/init_engine", true);
  xhr.send();
  forward_status = 0;
  return;
}
```

强制停止函数。当小车正在前进时检测到前方障碍物距离过近，调用此函数。此函数相比 stop()函数多了对 flash()的调用。

```
function radar()
{
    if (status == 'on'){
    var xmlhttp;
    xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST","/check_distance",true);
    xmlhttp.onreadystatechange=function(){
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
          resp = JSON.parse(xmlhttp.responseText);
          mystr = resp.distance + resp.unit;
          document.getElementById("myDiv").innerHTML=mystr;
          if(resp.distance < 1.2) {
            alarm = 'on';
            if(forward_status == 1) force_stop();
          }
          else if(alarm == 'on') {
            alarm = 'off';
            debuzz();
          }
        }
    }
    xmlhttp.send();
    setTimeout("radar()",500);
    }
}
```

超声波测距函数。同样在 status==‘on’条件下执行。执行过程中，请求调用 python 端 check_distance 函数，并记录下该函数返回给前端的距离值信息。document.getElementById("myDiv").innerHTML=mystr 是将距离值实时显示在网页界面上。如果测得的距离小于 1.2m，将 alarm 置为‘on’，此时如果小车正在前进，就调用强制停止函数 force_stop()。

超声波测距功能要求足够的实时性。setTimeout("radar()",500)指令就是让程序每 500ms 调用一次 radar()函数，完成一次测距，并更新网页上显示的实时距离值。

```
function change_status(){
    var xhr;
    var obj = document.getElementById("switch");
    obj.className = (obj.className == "class1"?"class2":"class1");
    xhr = new XMLHttpRequest();
    xhr.open("POST","/change_status",true);
    xhr.onreadystatechange = function(){
        if (xhr.readyState==4 && xhr.status==200)
        {
          status = JSON.parse(xhr.responseText);
          if (status == 'on') {
            buzz();
            setTimeout("debuzz()", 300);
            radar();
          }
          if (status == 'off')
            document.getElementById("myDiv").innerHTML='';
        }
    }
    xhr.send();
}
```

改变小车的开关状态。当用户点击网页中间的 ON/OFF 按钮时调用此函数。函数请求执行 python 端对应的 change_status()函数，并将其返回值（‘on’或‘off’）记录在 status 变量中。如果小车从‘off’状态变为‘on’状态，还需要通过 buzz()发出蜂鸣声表示小车启动，并调用 radar()开始实时测距。

3、前端 HTML 部分与其他

```
<div>

    <div align="center" style="color:black">
    <h2 style="color:#B0C4DE">Distance from the obstacle</h2></div>
    <br>
    <font id="myDiv" style="color:white" size="10"></font>

<br>
<br>
<br>
<br>

    <button id='up' type="button" ontouchstart="forward()" ontouchend="stop()" style="width:120px;height:120px;border-radius:400px;
    background-color:#A9A9A9;" class="btn btn-lg btn-primary glyphicon glyphicon-circle-arrow-up"><font size="7">&#8593</font></button>

</div>
<div>

    <button id='left' type="button" ontouchstart="left()" ontouchend="stop()" style="width:120px;height:120px;border-radius:400px;
    background-color:#A9A9A9;" class="btn btn-lg btn-primary glyphicon glyphicon-circle-arrow-left"><font size="7">&#8592</font></button>

    <button class="class1" id="switch" onclick = "change_status()" style="width:120px;height:120px;border-radius:400px";>
    <font size="5">ON/OFF</font></button>

    <button id='right' type="button" ontouchstart="right()" ontouchend="stop()"
    style="width:120px;height:120px;border-radius:400px;background-color:#A9A9A9;"
    class="btn btn-lg btn-primary glyphicon glyphicon-circle-arrow-right"><font size="7">&#8594</font></button>

</div>

<div>
   <button id='down' type="button" ontouchstart="backward()" ontouchend="stop()"
   style="width:120px;height:120px;border-radius:400px;background-color:#A9A9A9;"
   class="btn btn-lg btn-primary glyphicon glyphicon-circle-arrow-down"><font size="7">&#8595</font></button>
</div>
```

这是网页主操作界面的 HTML 代码部分，包括四个运动按钮、一个开关按钮与障碍物距离显示。需要说明的是，在手机移动端实现按住按钮、松开按钮调用不同函数的命令是 ontouchstart 和 ontouchend，而在电脑端实现此功能需要使用命令 onmousedown 和 onmouseup。而 onclick 命令普遍适用，表示完成一个点击操作（包括按下与抬起）后执行的指令。

为了克服移动端在用户长时间按住按钮时激活的其它功能（如放大镜、注释等等），我们借助如下 CSS 语句：

```
<style type = "text/css">
  #up
  {
    margin-left:1px;
    margin-bottom:3px;
    -webkit-user-select: none;
    -moz-user-select: none;
  }
  #down
  {
    -webkit-user-select: none;
    -moz-user-select: none;
  }
  #switch
  {
    -webkit-user-select: none;
    -moz-user-select: none;
  }
  #right
  {
    -webkit-user-select: none;
    -moz-user-select: none;
  }
  #left
  {
    -webkit-user-select: none;
    -moz-user-select: none;
  }
</style>
```

这样，用户在长时间点击手机屏幕上的按钮时就不会激活其他功能了。

四、后续工作展望

自动规避障碍物：通过红外避障模块和超声波测距模块，来实现小车对障碍物的自行规避。

小车的无人驾驶功能：基于小车的自动规避功能，可以编写类似于一段宏指令的代码让小车完成一场表演，也可以基于树莓派接口让小车同时闪灯和播放音乐；或者让小车像海龟一样试着走迷宫。通过对路面的检测，由单片机来判断控制其小车的反应情况，使其变得智能化，实现自动的前进，转弯，停止功能此系统还不断的完善后可以应用到道路检测，安全巡逻中，能满足社会的需要。

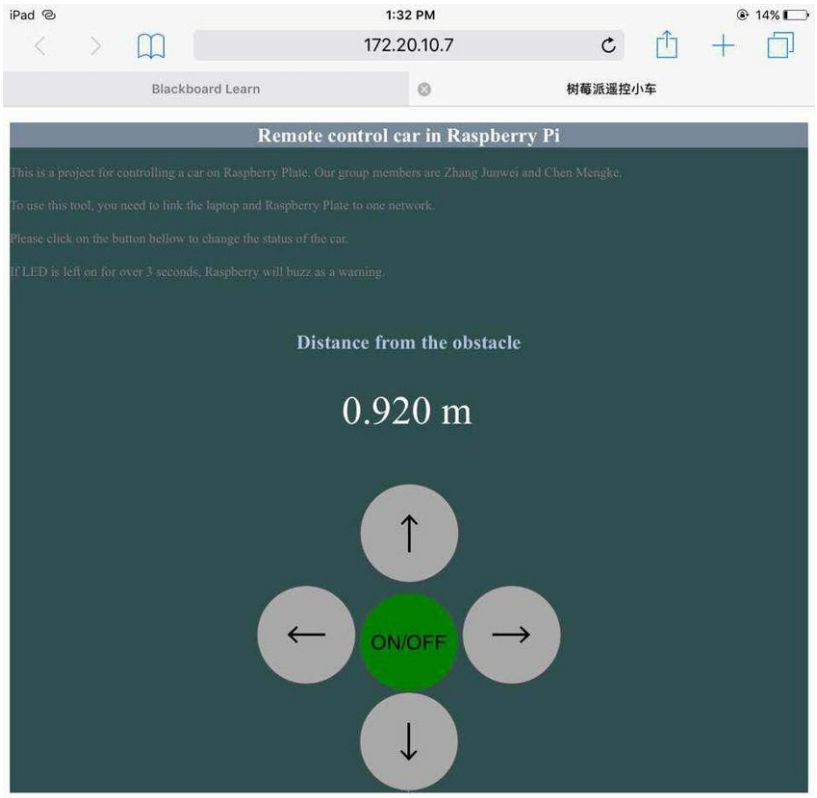
产品设计优化: 使小车原型更像实体小车，譬如可根据不同档位调节小车速度，让小车上下坡等等。用户界面上也可以继续优化，使之更试用于移动端的操作，掌控上更类似于方向盘的操作。

基于树莓派的车载无线传输视频系统：先在树莓派上安装树莓派官方摄像头,通过树莓派创建无线 AP。在 C/S 模式下使用 TCP Socket 编程技术实现树莓派与 PC 上位机的无线数据通信,PC 上位机可时实显示视频信息，返回小车周围的环境情况，并且通过超声波测距在图像上标出小车距离各个障碍物的距离，产品可用于危险地勘测和刑事侦察。

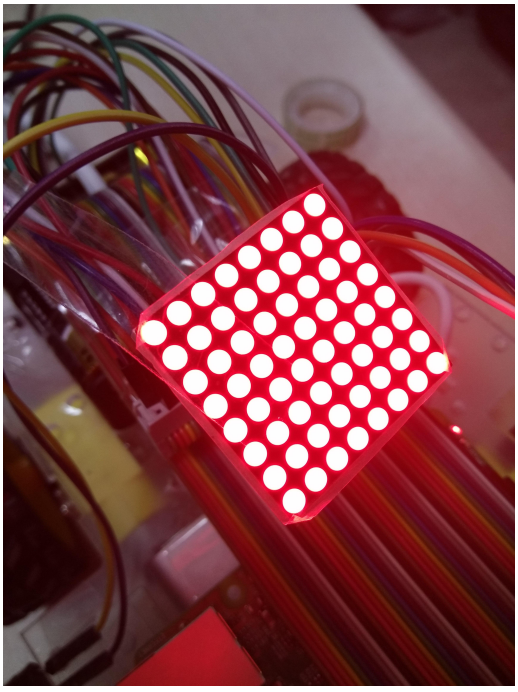
五、小组分工合作

张峻伟：项目设计、服务器搭建、代码实现与调试
陈梦珂：网页前端的开发、背景调研、物资采购

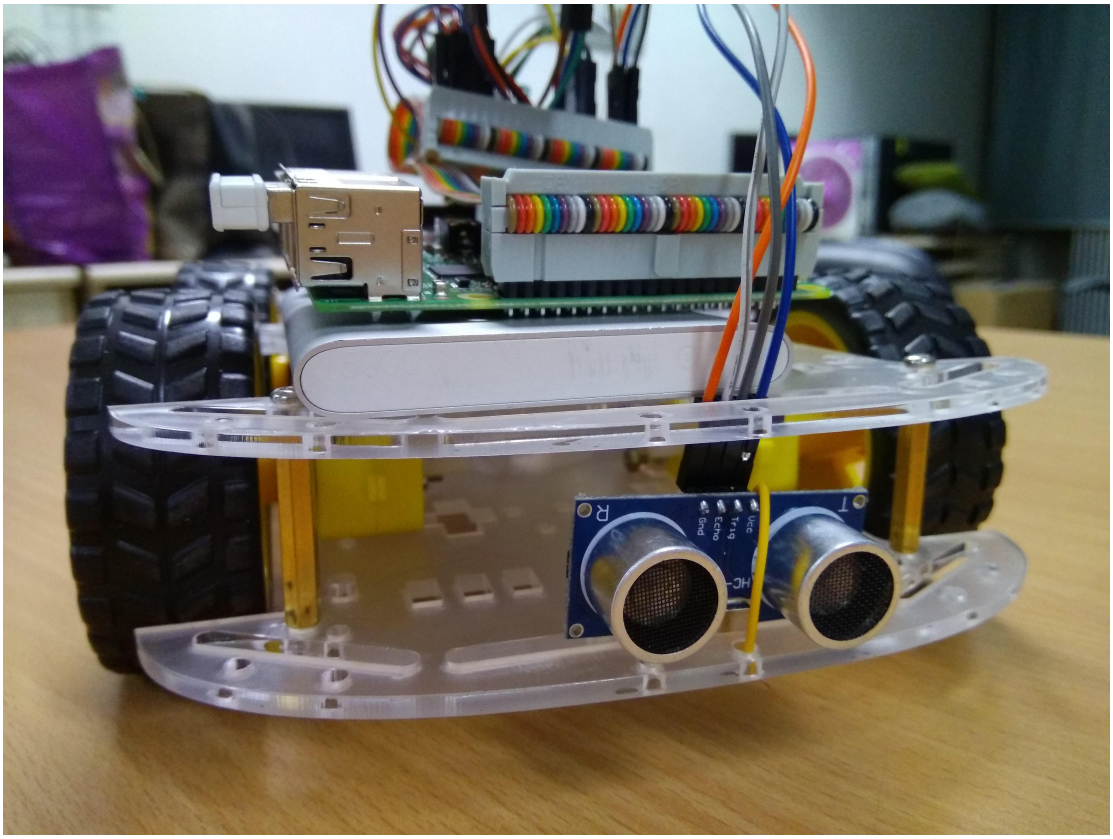
附相关图片：

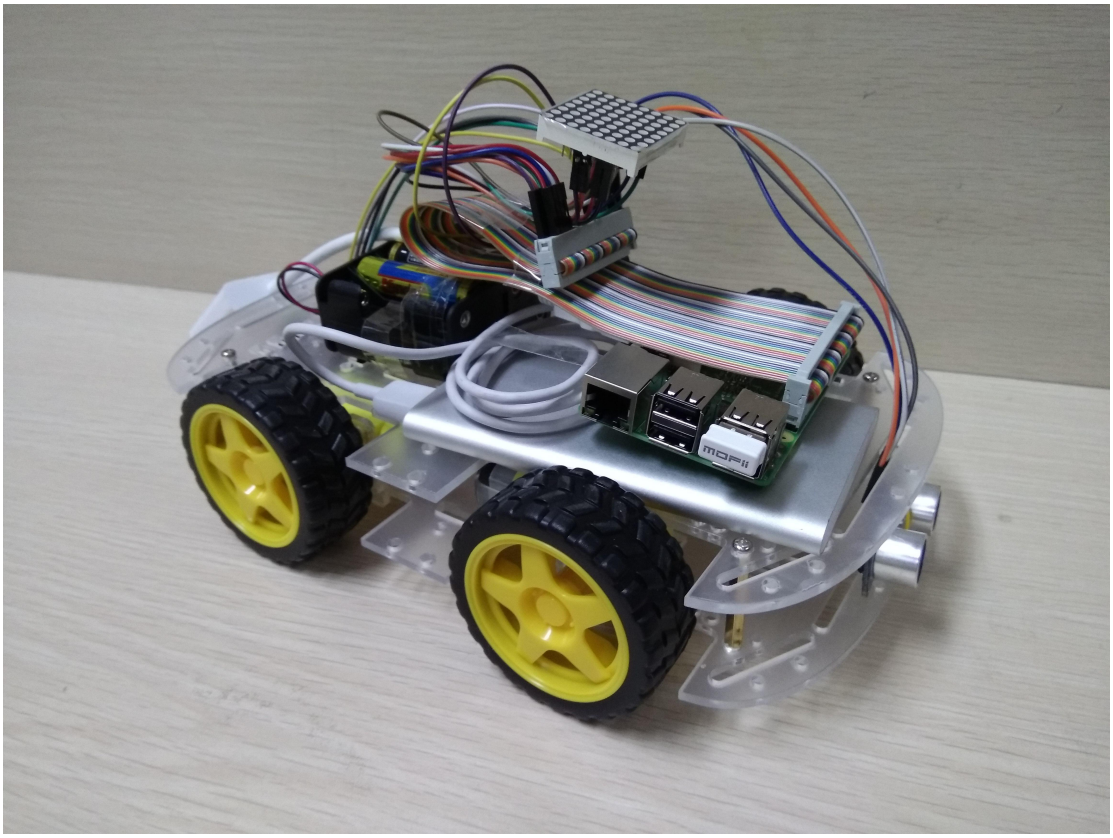


网页控制界面



LED 屏





遥控小车

树莓派实习报告

--用树莓派实现八分音符酱游戏

刘小辉 张颢丹 李佳益

【摘要】 树莓派是由注册于英国的慈善组织“Raspberry Pi 基金会”开发的一种微型电脑，加上瑞士军刀扩展板以及其他扩展设备如传感器等之后可以实现多种操作，功能多样。本文主要叙述我们小组使用树莓派来实现八分音符酱游戏的思路、过程等。

【关键词】 树莓派、八分音符酱游戏、瑞士军刀扩展板

一、选题及创意介绍

1.选题：用树莓派实现八分音符酱游戏

2.创意介绍：

A. 创意来源：由近期在网络上十分火爆的各种八分音符游戏的魔性视频产生的灵感，有趣之余更重要的是在我们现有水平上具有实现的可能性。

B. 创意想法:主体上是一个声控游戏，通过声音大小控制游戏主人公的走与跳，并在瑞士军刀扩展板上通过亮灯个数表示音量、数字表示分数来实现这个游戏。

二、设计方案

1.声音信息的输入与处理调用

通过外接 Microphone 获取声音信息，在代码中将其大小量化，做判断用。

2.游戏信息的输出

通过声音大小级别来控制亮灯的个数，通过对游戏状态的获取来显示分数，在扩展板上加以显示。

3.游戏主体的设计

游戏主人公设为学院吉祥物“地小空”，背景与地图就先暂时以最简单的方式显示，待后期加强。开始界面显示‘Please Speak Loudly to Start’，然后在界面的左上方显示得分与最高分，当本轮游戏失败后，显示‘Please Speak Loudly to Restart’提示玩家继续。

三、实现方案与代码分析

1、角色和方块实现

pygame 模块中的 sprite 类

```
class DXK(pygame.sprite.Sprite)
class Block(pygame.sprite.Sprite)
```

2、声音输入实现

pyaudio 模块获取声音，struct 模块对声音进行处理

```
pa = pyaudio.PyAudio()
SAMPLING_RATE = int(pa.get_device_info_by_index(0)['defaultSampleRate'])
NUM_SAMPLES = 4096
stream = pa.open(format=pyaudio.paInt16,
                 channels=1,
                 rate=SAMPLING_RATE,
                 input=True,
                 frames_per_buffer=NUM_SAMPLES)
string_audio_data = stream.read(NUM_SAMPLES)
k = max(struct.unpack('%dh' % (NUM_SAMPLES, ), string_audio_data))
```

这里面的 k 的大小与声音大小有关，可以用来表示声音的大小。

3、瑞士军刀扩展板的输出显示曲折实现

```
def number(n):
    SAKS.digital_display.show("%04d" % n)
def light(n):
    init()
    lst = [0x00, 0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF]
    writeByte(lst[n])
```

原本直接在主程序内调用以上函数，但是数字显示会引起声音输入报错，于是决定在主程序中循环向 score.txt 文件中写入分数，另外再开一个程序读入分数并显示。

（惭愧，没有学多线程和多进程）

4、时间记录

```
clock = pygame.time.Clock()
time_passed = clock.tick()
```

time_passed 即为与上次 tick()之间的时间间隔。

5、跳跃和着陆实现

```
def jump(self, speed):
    if self.can_jump:
    self.can_jump = False
    self.speed_y = speed
def land(self):
    self.can_jump = True
    self.speed_y = 0
if not dxk.can_jump and not fall:
    h -= dxk.speed_y * t
    dxk.speed_y -= dxk.g * t
    if h >= h0:
    dxk.land()
    xk.can_jump = True
    h = h0
```

h0 为地面高度，跳跃遵循物理规律，即 v=v0+gt，h=h0+vt。

6、行走和地图实现

```
map_queue_1 = list()
map_queue_2 = list()
```

第一个列表用来存放当前可见的方块，因为屏幕宽为 800，所以同时可能出现 9 个方块。第二个方块用来存放等待加入第一个列表的方块。用 1 表示方块，0 表示空方块。

```
block_s += dxk.speed_x * t
if block_s <= -100.:
    map_queue_1.pop(0)
    map_queue_1.append(map_queue_2.pop(0))
    block_s += 100.
for i in range(9):
    if map_queue_1[i][0]!=0:
        screen.blit(block.image, (i * 100. + block_s,
                                  screen_size[1]-block.height))
```

根据相对性原理，用地图的移动来表示地小空的行走。

用 block_s 记录方块的位移，每当有一个方块完全移动到界外，就移除它，在后面加一个新的方块，并把 block_s 归零。

7、计分实现

```
if map_queue_1[3][0] == 0 and map_queue_1[3][1] != 0:
    score += 1
    if score>score_max:
        score_max = score
    map_queue_1[3][1] = 0
```

积分规则为，每通过一个空方块，加一分。

在每个地图方块上加一个标记参数，当有空方块出现在地小空左边时，分数加一，并改变标记，防止重复计分。

8、死亡判定

```
if h >= h0:
    if map_queue_1[4][0]==0:
        if block_s < -22 and block_s > -78:
            fall = True
        elif map_queue_1[5][0]==0 and block_s <= -78:
            fall = True
        elif map_queue_1[3][0]==0 and block_s > -22:
            fall = True
if fall:
    if map_queue_1[5][0]==1 and block_s <= -78:
        dxk.stop()
        stop = True
    dxk.speed_y = -2.5
    if h < 600-53:
        h -= dxk.speed_y * t
    else:
        dxk.stop()
        break
```

当地小空脚下没有方块时，不会着陆，会一直下降，当接触地图边缘，判定为死亡。

9、开始界面

```
font = pygame.font.SysFont("arial", 24)
screen.blit(font.render("Please speak loudly to start!", True,
                        (0,0,0)),(260,100))
```

开始提示需要大声说话才能开始，并设置声音达到最大（k=32767）时开始游戏。

10、死亡界面

```
screen.blit(font.render("Please speak loudly to restart!", True,
                        (0,0,0)),(250,100))
```

死亡时提示需要大声说话才能重新开始开始，并设置声音达到最大时重新开始。

11、代码搬运

直接使用 github 实现 PC 和树莓派之间的代码互传。

四、后续工作展望

在后续工作中，我们希望在以下几个方面进行加强或改动：

1. 游戏背景与美观性

导入一些更好的背景作为游戏环境（如未名湖，并将地图块设计为荷叶）供玩家选择，或者允许用户自行导入背景。

2. 游戏地图去单一化与难度设计

在现有的程序中，地图只是单调的“坑”与“块”的组合，后续可能加上地图的高低起伏，或者在游戏人物上方添加障碍，成为声控的‘flappy bird’也不一定。

3. 充实游戏内容，增强可玩性

加入一些辅助道具（如增益药丸之类的），或者增加不同的游戏模式（例如：暴走模式，角色一直向前走，只能通过声音控制跳跃），另外可以利用扩展板上触控开关（如其中一个开关控制走，另一个控制跳等等）。

4. 代码更加面向对象化

原来的代码在写的时候大部分是用面向过程的思路写的，只有地小空和地图方块用了类来写。其中的碰撞检测、角色移动和地图生成都可以改用类来实现，可以提高程序的可读性，并且容易修改。

5. 优化窗口界面

原本使用的显示器分辨率较小，而且窗口固定大小为 800*600，代码中很多与位置有关的量都没有用变量名来表示，而是用 600、53 等这种不知所云的神秘数字。后来做展示时的分辨率较大，当时可以看出在不同分辨率的显示器中的效果有一定的差异。可以用 Python 动态检测屏幕的分辨率，考虑不同的输出环境下各个变量的改变，比如 1024*768 大小的窗口需要同时显示 11 个方块。

可以考虑完善全屏的显示方式。原本决定用 F 键一键全屏化，但是进入全屏后，键盘不可用，无法退出全屏，于是删除了全屏的设定。希望能够用更好的方式实现全屏。

6. 优化合作

利用 github 等工具，实现小组成员之间代码的直接交流。

五、小组分工合作

张颢丹	技术担当	负责最主要的游戏程序部分
李佳益	创意担当	提出创意，丰富创意，收集资料
刘小辉	组长	负责树莓派的基本操作（GPIO 的实现）

树莓派实习报告——音乐可视化

孙景南 杨德鼎

摘要：音乐可视化实现了音乐转化为图像的功能
关键词：音乐可视化 树莓派 电路

一、选题及创意介绍

我们的选题是“音乐可视化”，意思是用树莓派达到的效果是用音乐控制灯珠产生绚丽的闪光效果。对一首乐曲，以手动录入的方式将它的乐谱输入至树莓派通过程序对乐谱进行读取、转换、输出操作，并根据其中的音符，按照既定的音符配色方案控制灯阵列的亮灭。预期效果是灯阵随着音乐的音高、音乐节奏的快慢来变幻闪烁出合适的式样，达到音乐可视化的效果。

我们的创意来自音游，音游画面中的圆圈图案会随着音乐的节奏不断变换速度冒出，即使不听音乐也能感觉到音乐的节奏。我们打算利用树莓派的 40 个线脚和面包板实现该效果。

二、设计方案及创意想法

设计分为两部分：功能部分和硬件部分。在功能实现上，我们设计的是，输入一段乐谱，由程序转化为电信号，控制灯泡的绚丽闪烁。在硬件部分上，我们设计的是灯泡阵列，受树莓派输出的电信号控制绚丽闪烁。

三、实现方案及代码分析

考虑到我们是在进行一些可视化的设计研究，所以后来我们将我们的设计简化成了针对某些乐曲进行面包板显示的配合。起先我们希望通过程序自动识别乐曲，将乐谱中的每一个音符对应于一种灯光闪烁方式，进行输出，然而考虑到通过程序自动识别音高、行进速度过于困难，因此我们对于这一思路也进行了调整。

我们实现机理的简介

（一）、软件模块：

1.读取与录入模块

在程序的开始进行乐谱的读取、特定接线柱的初始化，这些是前期的一些准备工作。

2、对应亮灯函数的编写。

我们为乐谱中的每一个音高，以及不同的频率设置了不同的函数，控制 led 灯产生变化的效果。

亮灯核心代码为：

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
GPIO.output(PIN_NO,GPIO.HIGH)
GPIO.output(PIN_NO,GPIO.LOW)
```

3、主程序的执行

根据乐谱读取音符和速度，用控制函数控制对面包板电路的信号传入。

（二）、硬件模块：

1.设计电路

2.试线脚

3.无尽的电工活

最关键的一点，在整个操作之前有两件及其重要的事情。一个是灯的电路的设计，为了实现美观的效果，我们需要对整个电路进行一些规划；另外一个是电工活，唉，需要我们进行无穷的接线与测试工作，十分复杂。

四、后续工作展望

考虑到我们目前的设计仍有所可以提高的地方，因此我们对未来的开发研究抱有强烈的期待。而我们小组具体打算从以下几个方面进一步拓展我们创意：

1.首先，我们希望使整个设计更加美观。在我们的开发以及展示过程中，我们注意到，整体线路的连接对于整个面包板的外观实为不利，因此我们期望能够通过一些外设将整个面包板进行包装，同时使我们的接线隐藏在面包板的后面，这样，只有灯光的闪烁会使整体显得更有感染力。

2.我们还希望能够使我们对于音乐的展示更加多样化，或许可以不止限于LED灯的闪烁，还可以通过多元化的外设使我们能更全面、立体的展现音乐的魅力。而且我们也可以不仅限于一块面包板，而是通过面包板的空间摆放使整体更有感染力。

3.我们目前的程序只能做到对于一首歌曲进行乐谱性的读取，进而随着音乐的节奏进行灯光的闪烁，这也导致了手动录入歌词的工作的繁琐以及我们的设计不太可推广。因此，我们计划在未来进行基于音乐进行歌曲展现的研究，即通过读取乐曲本身，让程序自动获得音乐节奏的信息，这样我们的整个开发活动便大有推广的余地。另外一方面，在实现了上一点之后，我们或许可以实行音乐的剪切与拼接，让音乐在穿插演奏中展现出更大的魅力，这样势必会非常吸引用户。

4.最后，也是最长远的目标，我们也希望，在一定程度上，实现与用户的交互，甚至可以录入使用者本身的声音，伴随着他的声音进行灯光的闪烁和整个树莓派的展示活动。总之，这一点需要前几点的基础，并充分利用树莓派的各种扩展外设，以及复杂的程序进行控制。

综合以上几点，我们坚信，通过更加复杂的代码的控制，以及更加全面的外设辅助，我们可以实现对于音乐更加全面美丽的展现，也一定可以在交互中实现用户心灵与音乐的共鸣。

五、小组分工合作

孙景南：烧录系统、采购外设、初期摸索（大量BUG）、设计和接线路（电工活儿）

王泽鑫：寻找音频素材 、提供思路、多次找到问题的突破口。

杨德鼎：图像点阵化处理、乐谱录入、编写主要代码、设计灯泡样式。

有感：

在完成这个项目中，我们三人合作，共同学到了很多，享受到了学习的乐趣，有过为一个难题纠结数小时的苦闷，也有终于搞好的拍手叫好。在这个过程中，我们各有各的收获。本次树莓派实习项目真的很棒。

附效果图：

（此为课堂展示时的照片）

项目源码：

```
import random
import time
import RPi.GPIO as GPIO

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

GPIO.setup(11,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
GPIO.setup(22,GPIO.OUT)
GPIO.setup(36,GPIO.OUT)
GPIO.setup(40,GPIO.OUT)
GPIO.setup(37,GPIO.OUT)

adict = {1:22,2:36,3:37,4:11,5:12,6:40,7:1,8:1}

def beeep(seconds,PIN_NO):
    GPIO.output(PIN_NO,GPIO.HIGH)

def beep(seconds,PIN_NO):
```

```
    GPIO.output(PIN_NO,GPIO.HIGH)
    time.sleep(seconds)
    GPIO.output(PIN_NO,GPIO.LOW)
    time.sleep(seconds)

def beepall(seconds):
    for i in range(1,7):
        GPIO.output(adict[i],GPIO.HIGH)
    time.sleep(seconds)
    for i in range(1,7):
        GPIO.output(adict[i],GPIO.LOW)
    time.sleep(0.1)

def beepround(seconds):
    beeep(seconds,22)
    time.sleep(seconds)
    beeep(seconds,36)
    beeep(seconds,37)
    time.sleep(seconds)
    beeep(seconds,11)
    time.sleep(seconds)
    beeep(seconds,12)
    beeep(seconds,40)
    time.sleep(seconds)
def beepblingbling(seconds,num):
    beep(seconds,num)
    beep(seconds,num)
    beep(seconds,num)
def test():
    GPIO.output(PIN_NO,GPIO.LOW)
def hothothot(seconds):
    for i in range(1,7):
        GPIO.output(adict[i],GPIO.HIGH)
    time.sleep(seconds)
    for i in range(1,7):
        GPIO.output(adict[i],GPIO.LOW)
starttime=time.time()
while True:
    endtime=time.time()
    if endtime-starttime<6:
        beep(2,adict[random.randrange(1,7)])
        beepblingbling(0.2,22)
        hothothot(1)
    elif endtime-starttime>=6 and endtime-starttime<30:
```

```
    beep(1,adict[random.randrange(1,7)])
    beepblingbling(0.1,36)
    beepblingbling(0.2,40)
elif endtime-starttime>=30 and endtime-starttime<40:
    beep(0.2,adict[random.randrange(1,7)])
    hothothot(2)
else:
    beep(0.1,adict[random.randrange(1,7)])
    beepall(0.1)
    beep(0.1,adict[random.randrange(1,7)])
    beep(0.1,adict[random.randrange(1,7)])
    beep(0.1,adict[random.randrange(1,7)])
    beep(0.1,adict[random.randrange(1,7)])
    beep(0.1,adict[random.randrange(1,7)])
    for i in range(30):
        beep(0.1,adict[random.randrange(1,7)])
    beepblingbling(0.5,40)
    beep(0.1,adict[random.randrange(1,7)])
    hothothot(1.5)
    beepblingbling(0.2,36)
    for i in range(20):
        beep(0.1,adict[random.randrange(1,7)])
```

数据结构与算法课程

树莓派实验报告

(作品名称：摩尔斯电码)

(组长：戴琪 组员：黄岭贝)

摘要：由于摩尔斯电码在结构上较为简单，且有着极强的实用性和电影中的酷炫性，我们组决定用树莓派实现与摩尔斯电码有关三个功能，效果良好也有些环节可以后续开发。

关键字：树莓派，摩尔斯电码

一、选题及创意过程：

选题：摩尔斯电码

创意过程：周末在看电影《无间道》的时候，我看到间谍用摩斯密码传讯就在想这个摩尔斯电码既简单又实用，突然就想把摩尔斯电码运用在树莓派上面。而且树莓派的结构也很适于摩尔斯电码的应用。于是，这个创意就诞生了。如下为摩尔斯电码：

A	·—	N	—·	1	·— — — —
B	—···	O	— — —	2	··— — —
C	—·—·	P	·— —·	3	···— —
D	—··	Q	— —·—	4	····—
E	·	R	·—·	5	·····
F	··—·	S	···	6	—····
G	— —·	T	—	7	— —···
H	····	U	··—	8	— — —··
I	··	V	···—	9	— — — —·
J	·— — —	W	·— —	0	— — — — —
K	—·—	X	—··—	?	··— —··
L	·—··	Y	—·— —	/	—··—·
M	— —	Z	— —··	.	·—·—·—
()	—·— —·—	—	—···—	@	·— —·—·

二、设计方案：

通过树莓派实现有关于摩尔斯电码的三种功能：

1）英文转摩尔斯电码

2）声音输入摩尔斯电码，输出英文

3）键盘输入摩尔斯电码，输出英文

实现设备：

树莓派展板

树莓派瑞士军刀扩展板

无线键盘和鼠标

显示屏

声音输入设备

各种乱七八糟的电线就不一一赘述了

三、实现方案及代码分析：

摩尔斯电码由点、划、空格实现。不同字母，数字，符号都有相应点划组合。对于键盘输入字符串而言，我们就可以将输入的信号拆开，把全部信息变成点划的排列，再设计树莓派对于点和划的不同反应实现区分，从而输出我们的最终目标——摩尔斯电码。对于键盘输入摩尔斯电码，我们则通过空格将每个字母或者数字分开存入列表，通过一个对应关系实现输出。对于声音输入，我们通过记录输入时长和声音大小实现信号转化。

准备工作：

建立声音输入设备对象和树莓派瑞士军刀扩展板对象。

通过建立两个字典实现对应关系。

b 是蜂鸣器

设置遇到点是树莓派叫的秒数，划的时长为点的三倍。

然后选择 if 语句进入三种功能的其中一个

功能 1)：

通过建立字典实现映射。然后在读入后将字符串分成单个单词，将单个单词又变成多个字母的组合，每个字母通过字典的映射找到对应摩尔斯电码，再将每个电码拆成点划。对于点划，树莓派有不同反映即可。这里用了三重循环。

```
ty = input()
#英文输摩尔斯电码
if ty == 'word':
    print('Please input your word, my prince/princess')
    onone1 = [False,]*4+[True,]*4
    onone2= [True,]*4+[False,]*4
    s = input()
    everypart = s.split()
    #对每一个最小单元依次扫描
    for i in everypart:
        for j in i:
            jreal = dic1[j]
```

```
            for k in jreal:
                #如果遇到点
                if k=='.':
                    b.beep(point)
                    SAKS.digital_display.show('6.66.6')
                    SAKS.ledrow.set_row(onone2)
                #如果遇到划
                elif k=='-':
                    b.beep(line)
                    SAKS.digital_display.show('2.333')
                    SAKS.ledrow.set_row(onone1)
                time.sleep(point)
            SAKS.ledrow.off()
            time.sleep(point*2)
        #单词之间输出 8888
        SAKS.digital_display.show('8888')
        time.sleep(point*4)
```

遇到点的话，亮 4 右边盏灯，蜂鸣器叫预先设定的秒数，数字显示屏显示 2333。遇到划的话，亮 4 左边盏灯，蜂鸣器叫预先设定的秒数，数字显示屏显示 8888.单词之间显示 8888，休息 4 倍点数对应的秒数。

功能 2)：

通过声音输入设备记录声音大小 k，并且每隔很短时间就记录一次（声音输入设备自带功能）。当 k 大于 10000 值的时候就累计一次，用 i 记录，初始值为 0，从而反映输入的点和划的时长。不同 i 值对应点还是划，或者中断，感应到点就在数字显示屏上面输出 6.66.6，感应到划就在数字显示屏上面输出 2.333。如下代码：

```
#输入声音输出英文
elif ty=='voice':
    voi = []
    i = 0
    s = ''
    while True:
        stream = pa.open(format = pyaudio.paInt16,channels = 1,rate = SAMPLING_RATE,input =
True,frames_per_buffer = NUM_SAMPLES)
        string_audio_data= stream.read(NUM_SAMPLES)
        #声音大小
        k= max(struct.unpack('%dh'%(NUM_SAMPLES,),string_audio_data)    )
        stream.close()
        #大到一定程度记 1，连续记录
        if k>10000:
            i+=1
        else:
            #点
            if i>1 and i<=3:
```

```
            s+='.'
            SAKS.digital_display.show('6.66.6')
            i = 0
        #划
        elif i>=6 and i<=10:
            s+='-'
            SAKS.digital_display.show('2.333')
            i = 0
        #极短地叫一下极为空格
        elif i==1:
            voi.append(s)
            s=''
            i=0
            SAKS.digital_display.show('8888')
        elif i>13 and i<=25:
            break
        #声音持续时间过长或过短均极为没有输入
        elif i>25:
            i = 0
            s=''
        else:
            i = 0
    print(i)
    print(s)
    print(voi)
for i in voi:
    print(dic2[i]),
```

功能 3)：

再建一个字典 dict2，实现字母和摩尔斯电码对应关系，不过摩尔斯电码作为 key 值，输入后找对应关系再输出即可。

```
#摩尔斯电码输出英文
elif ty=='code':
    print('Please input your code, my prince/princess')
    s = input()
    everypart = s.split()
    for i in everypart:
        print(dic2[i]),
```

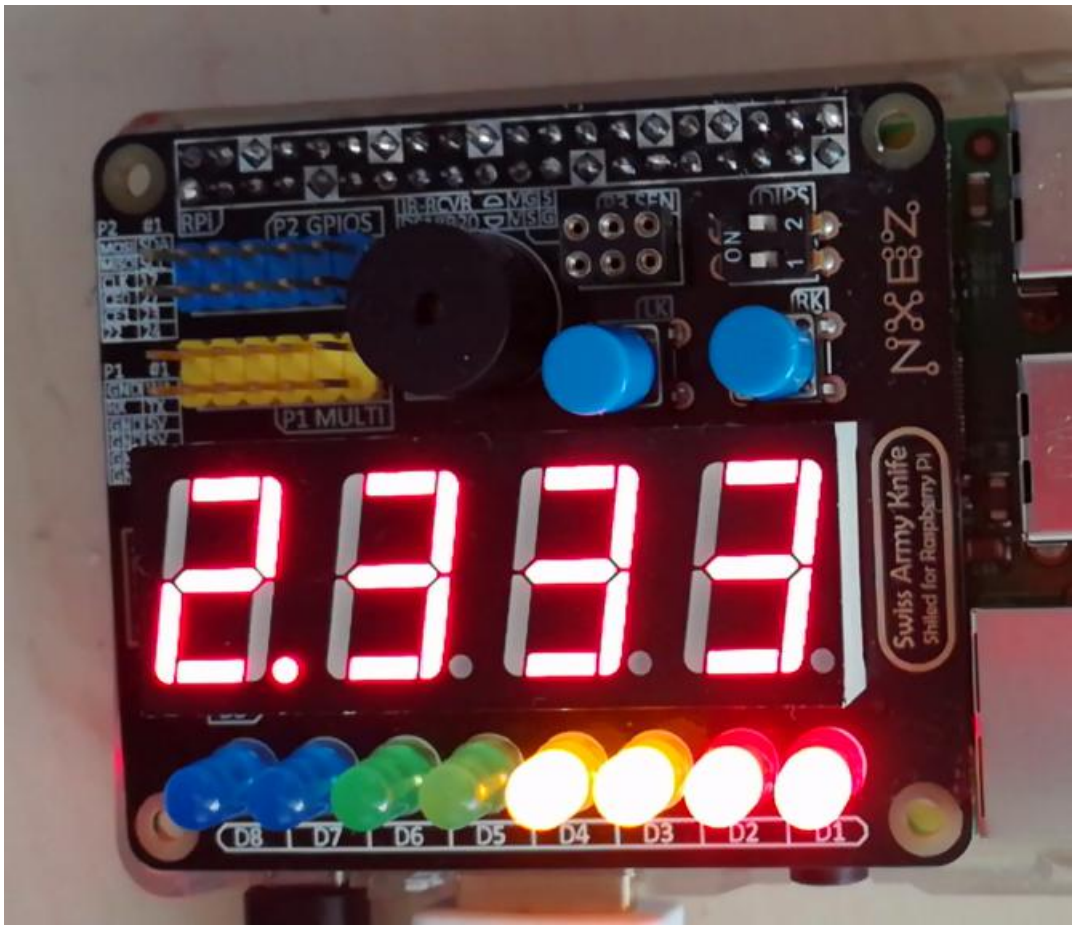
自我评价：

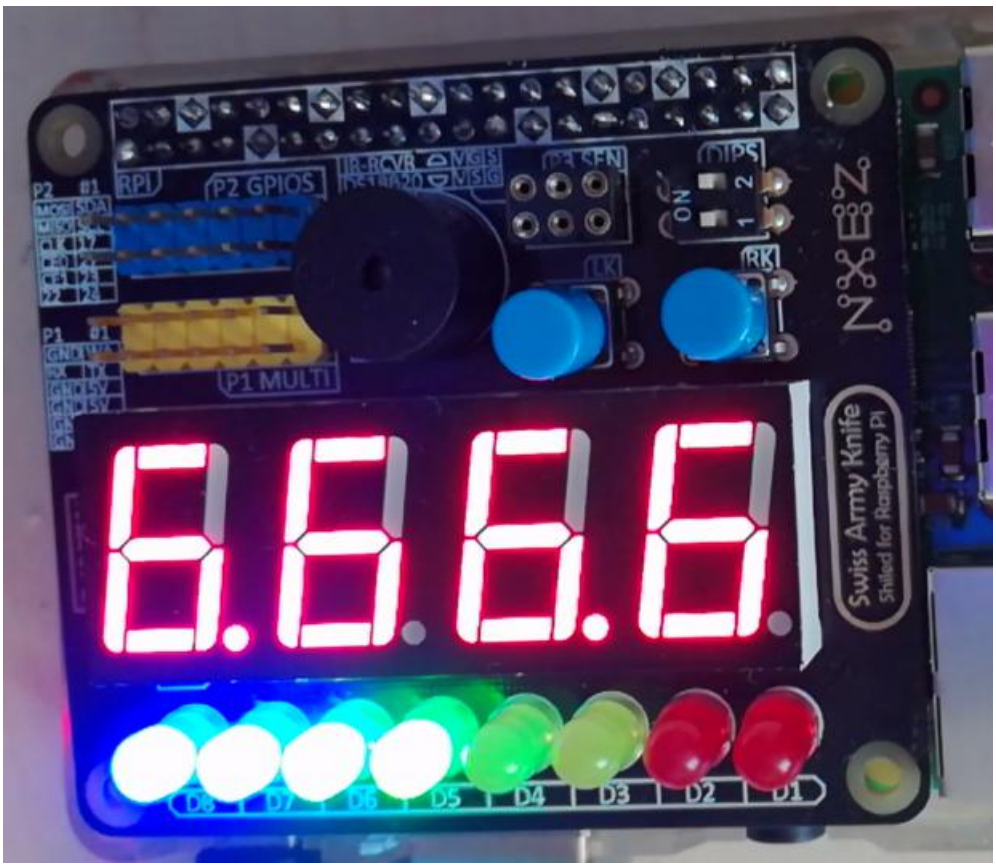
我们先写了功能 1）3）的代码，虽然功能上比这个稍优（不同情况更具体，能体现有几个点连着，几个划连着），但是因为要穷举导致代码极为冗长，所以我们简化这两个功能，大幅度减少了代码长度，大概是原来的 1/15。后来觉得如果加上声音输入更为贴切，也更符合摩尔斯电码的使用，于是增加了功能 2）。

结果与分析：

对于功能 1）3）实验进行得很完美，因为整个结构比较简单，所以在结果上找不出毛病。而功能 2）则是在声音输入端出现了问题，因为我们毕竟不是专业电报员，难免会出现一点偏差，只好增加容错率，导致输入时时间可能会很长。这也是没办法，当然，专业人员的话就不存在这个问题了。下面附上一些实验结果：

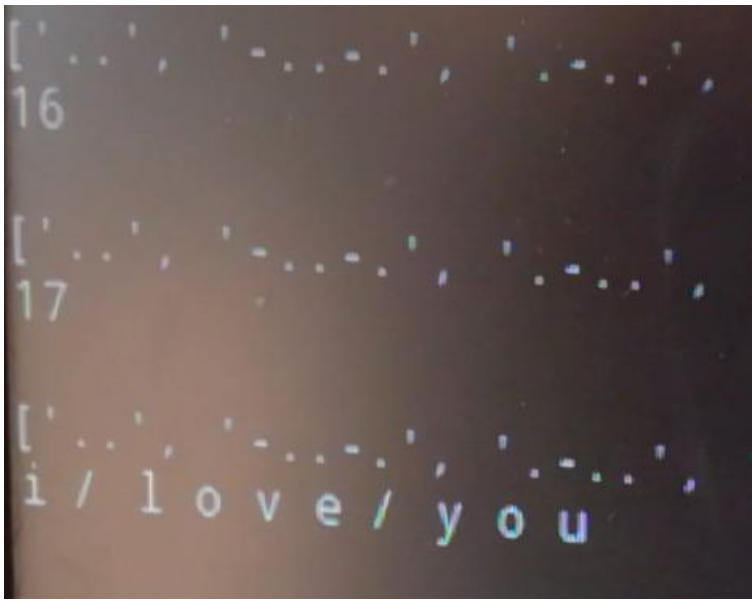
点：





划：

声音输入结果：“i/love/you”空格似乎没有对应摩尔斯电码，所以我们采用“/”分开不同单词。



四、后续工作展望

1）用树莓派的两个按键实现点，划，空格的输入（这也是这次的遗憾之处，由于时间原因没有完成）

2）提高声音输入的精准度（通过编程让感知声音的时长更准确，同时再增大一些容错率，让第二种功能更精准）

3）（借鉴其它小组思想）实现外界面包板，显示输出内容（面包板上面接 LED 灯，实现字母和数字的可视化）

五、小组分工合作

戴琪：原始代码（只有两种模式且较为冗长），PPT， 实验报告

黄岭贝：大幅度简化组长原始代码…增加声音输入功能

树莓派扩展板创意编程项目（可视化钢琴小组）实习报告

组长：杨帆 组员：张昊、孙唯一

【摘要】 本组的可视化钢琴将音乐同树莓派的 LED 灯结合，打造出良好的展示效果。使用 pygame 为基础的多线程编程实现。在与 CPU 不足、延迟的抗争中，我们提出了一些可以进一步完善的方案。本组的代码规范文件对多人合作项目有积极意义。

一、选题及创意介绍

在我们小组成立之初，我们想到的是要做一个 DJ 与舞池灯光的互动；而在我们小组三个人一致希望减少外设的想法下，我们最终选择制作该互动的基础部分，即实现声音与显示的互动，也即一个可以实现声音多元化并且可以以灯光的形式进行反馈的装置，于是我们的选题就这样诞生了——可视化电子琴。我们力求以简单的外设使多种乐曲的便捷演奏成为可能，为乐曲添加视觉效果，并为产品的进一步拓展提供可能。

二、设计方案

工作的一开始，我们就明确了我们要做什么——一个电子琴。它自然应具有电子琴最基本的功能：录制与播放，多种音色，延音踏板，伴奏等等。相比较于传统电子琴，树莓派有它天然的优势，可编程的特性带来了出色的拓展性，我们可以随时添加不同的乐器，甚至添加更多的功能！

然而，在后续的工作中，我们也发现了许多问题。首先，树莓派令人失望的 CPU 和仅仅 1G 的内存带来了许多麻烦，尤其是对需要良好节奏感的电子琴来说，延迟是最大的敌人！为了应对这个问题，我们将对音源文件的读取提前到程序开始，减少了每次按键读取文件带来的延时。但令人遗憾的是，我们不得不取消了打击乐伴奏功能，因为在存在延迟的情况下强行打节奏会造成更大的混乱。此外，我们取消了叠加音轨，这其中既有延迟的原因，但时间紧张也是必须面对的因素。

我们本计划用左右 Shift 键来实现音域的升降，但经过测试，我们发现树莓派并不能区分出左右 Shift！我们不得不取消了这个功能。当然，我们很高兴能用到键盘这个不同于钢琴键盘的输入工具。虽然在小小的键盘上放置如此多的键位显得十分拥挤，但我们也可以利用键盘上的各个键位。比如，我们用 Shift 代替延音踏板，Alt 代替弱音踏板，用 Tab 来切换乐器，用 Space 来控制音符升降。我们还利用拓展版上的 LED 来实现灯光效果。当然，由于时间原因，有些功能还未能实现或不够令人满意，有机会我们会再加

以完善。

三、实现方案及代码分析

根据以上设计方案，我们需要做到以下内容：捕获用户按下键盘的事件，并把键盘事件的内容映射为对应的单音信息（比如音高、是否弱音或延音）；对单音信息进行处理，包括发出声音、控制树莓派灯光闪烁、数字显示变化等。其中，比较重要的键盘、声音的处理可以用通用游戏设计框架 pygame 来实现，树莓派灯光闪烁与数字显示变化可以通过调用 SAKS-SDK 实现。由于调用树莓派、声卡的过程是比较耗时的，而键盘事件的调度耗时不长，因此采用 Python 多线程组织整个程序。同时由于树莓派的相关功能与音乐相关功能相对独立，因此我们将树莓派的部分和其他部分分开，单独写成了一个模块。因此，pypiano_main 模块和 Piano_SAKS 模块构成了我们工程的主体，共计 500 行。

这里有必要简单介绍一下有核心地位的 pygame 部分。pygame 有完整的事件处理机制，所有捕获到的键盘、鼠标事件等等都被封装成了一个 event 对象，我们只需要根据 event 对象的各种属性，编写 on_key 部分的代码即可。pygame 的音效处理模块分为两步：首先调用 pygame.mixer.Sound 构造方法，编译 ogg 文件，得到一个 pygame.mixer.channel 对象；其次调用 channel 对象的 play, stop, fadeout, set_volume 等方法，即可向声卡写入相应的数据，发出声音。

具体而言，主体部分除了常量池外，主要由三个类构成：音符类 note，主要负责对音符信息的封装，包括音高，乐器，有没有弱音，有没有延音效果等等，作为一个单纯的信息类，没有其他方法。

频道类 music_channel，这个类本来打算作为一个音轨的专用输入类，内置了该音轨对应乐器的所有 pygame.mixer.channel 对象，包含键盘音轨和文件音轨。键盘音轨实现了 key_input 方法，捕获键盘事件，并根据键位表映射为音符对象，实时地放入处理队列。文件音轨实现 file_input 方法，从文件中读取内容，生成音符对象，并按照对应的时间放入处理队列。我们希望通过键盘音轨实现键盘实时演奏，而通过文件音轨实现对录制文件的播放还原，可以通过一个键盘音轨和多个文件音轨实现一首曲目的多音轨叠加。由于延迟问题和时间紧张，所有 file_input 相关内容都没有实现，因此没有实现录放音、叠加音轨的功能。

混声器类 music_mixer，包含对音符对象处理的部分。混声器对象包含一个处理队列，所有 music_channel 对象生成的音符对象都通过处理队列被混声器实时处理。处理

队列原则上可以注册多个 music_channel 对象，但是由于没有 file_input 方法的实现，只有一个键盘音轨被注册到这里。对每个音符对象处理的过程大致分为两步，第一步是调用音符对象中 music_channel 的对应声音文件，进行发声；第二步是启动一个新的线程处理树莓派操作。

树莓派相关操作比较简单，主要是根据音符的音高调用相应的灯，并给数字显示屏的全局变量加 1。考虑到处理树莓派的操作是多线程的，且 SAKS-SDK 的操作不是线程安全的，我们用了一个队列来异步处理多个亮、灭请求。

编写代码之初我们犯了一个严重的错误，就是把 pygame 音效处理中编译的阶段也放在了键盘的捕获阶段，而这是一个非常需要实时性的阶段，这么做会产生大量的延迟。好的解决办法是把这部分放在开始捕获键盘之前的初始化阶段。此外，这个项目中间曾经试验用一次键盘事件对应一次音效处理替代多线程，但是并没有性能提升，因此虽然这个程序 CPU 占用率极高，仍然是 I/O 密集型的程序（比如针对键盘、SAKS 板、声卡的 I/O），多线程的使用是合理的。

四、后续工作展望

根据上文的代码分析，我们缺失的功能也可以补充在这一部分：

1. 录放音、叠加音轨。根据上文分析，这两个实现的关键在于实现 file_input 函数。实际上这是一个相对简单的问题。需要解决的问题是如何存储录音文件并方便地读取。这一点我们需要参考一些常见的音乐格式以及音乐播放器的写法。录音阶段可以放在音频处理的平行阶段，先将相关信息缓存在字符串中，一段时间后（比如切换乐器，或者使用其他功能键），将信息按某种格式写入文件。此外，延迟也是需要考虑的因素。如果认为 PC 端延迟可以接受，那么这个功能可以仅对 PC 开放。这个功能是实现本组初始方案——DJ 与舞池互动的关键部分，也是以下多个拓展方案的基础。

2. 节奏伴奏。这个实际上包含两个子问题，一是关于打击乐的输入。打击乐是一种特殊的乐器，它弱化了音高属性，但是强化了音色属性。我们可以采取打谱软件 Overture 的处理方式，用音高来代指音色。二是关于打击乐的输出，这个实际上和普通乐器输出是没有区别的，关键仍然在于 file_input 方法。

3. GUI。小组成立之初我们曾打算使用树莓派数字显示屏作为主要 GUI，但是发现数字显示屏的刷新速率远远达不到我们的要求，此外 pygame 强制要求打开一个窗口，因此我们计划将 GUI 显示在窗口中。内容可能会包括音频波形、音高、乐器选择界面、录放音按钮等。如果叠加音轨能够实现的话，我们甚至可以做一个乐谱 GUI，把电子琴

包装成一个只用键盘操作的打谱软件。

4. 延迟。这个问题在树莓派框架下是无法解决的。我曾经做过一个精简版的电子琴，删除了不必要的功能，发现 CPU 占用仍然是长时间 100%，偶尔 99%。然而，在 PC 端，我们可能采取一些措施。比如常见 PC 都是多核，而 Python 多线程由于 GIL 无法实现真正多核，但是可以通过一些方式绕过 GIL 实现多核，从而提高性能，降低延迟。

5. 关于复杂和弦处理。举例：G7 和弦 G-B-D-F#，凡是和弦中出现了黑键、白键同时的情况，全部无法演奏，因为这个程序为了操作的简洁，使用了空格键提升半音。实际上这个问题只需要修改一下键位、取消空格黑键机制即可，然而考虑到树莓派延迟的问题，基本不太可能弹奏过于复杂的曲目，所以保留了按键的解决。如果延迟问题可以解决，这个问题应当提在日程当中。当然，如何安排一个合理的按键分布，也是一个问题。

6. 常见格式导入导出。这个问题在整个开发过程是回避的。但是作为一个完整的功能软件，应当具有这个功能，比如导出成 midi, wav 等常见格式，或者从 midi 中读取。

7. 蜂鸣器使用。我们开始考虑到使用蜂鸣器补充乐器库，但是由于 SAKS 板的蜂鸣器是有源蜂鸣器，不可以调节频率，所以放弃。实际上可以通过 RPi.GPIO 的 PWM 模块对无源蜂鸣器的频率进行调制，来达到演奏效果。

五、小组分工合作

小组成立之初，便确立了明确分工：杨帆负责树莓派编程环境配置，主体代码的编写。张昊负责测试 SAKS 板的性能，树莓派部分代码的编写。孙唯一负责寻找播放的单音音源，转换格式，整理命名归类。大家的工作虽然难度不同，但是都是整个项目的重要组成部分。

此外本组的一大管理特色就是，在小组成立之初，起草了一份代码管理规范（见附件）。虽然这份规范没有得到很好的落实，但是这份规范对团队协作具有十分积极的意义。另外在合作开发期间，为统一 API，也有一些其他的规范，作为项目的必要组成部分，也被打包在附件中。

【编者注】：本项目已在 Github 上开源，网址为
https://github.com/RibomBalt/RPi3-MusicPiano
读者可由此下载本项目代码。
报告中提到的代码开发规范，读者可参阅漂移乒乓大作业报告集 Quebec 组部分。

老虎机在树莓派上的实现

陶天阳

摘要：充分利用树莓派瑞士军刀扩展版上的开关、数码管、LED 灯等组件，在只外接充电宝的情况下独立完成老虎机功能

一、选题及创意介绍

老虎机是广大市民朋友都喜欢玩的一种赌博机器，通过玩家下注和机器随机产生结果来决定金额与输赢。而树莓派瑞士军刀扩展版上的组件恰好足以实现老虎机的这些功能，而且树莓派小巧轻便，可以做成一个成随身携带的小游戏机。

二、设计方案

1.组件介绍

数码管：显示剩余金额和下注金额

LED 灯：演示开奖动画和显示下注的灯

轻触开关(左)：选择下注金额和下注的灯，输入秘籍

轻触开关(右)：确定下注金额和下注的灯，输入秘籍，进行下一轮游戏

拨码开关 1：重新开始游戏

拨码开关 2：调节静音和进入秘籍模式

2.流程介绍

游戏开始时，先显示初始金额 100，此时按右键(右轻触开关)进入下注模式，此时按左键(左轻触开关)调节个位数字大小，按右键确认后再按左键调节十位数字大小，再按右键进入选灯模式，此时按左

键选灯再按右键确认。之后会播放开奖动画，若最后停留在下注的灯则奖励十倍下注金额，若只是同一颜色则奖励五倍下注金额，之后会显示剩余金额，等待玩家按右键进行下一轮游戏。

若金额减为 0，则游戏失败，无法继续下注。

若金额超过 10000，则游戏通关，播放通关动画。

3.游戏性功能

当连续十次未中时，作为惩罚金额减半。

把拨码开关 2 关闭，此时所有 LED 灯和数码管均熄灭，进入秘籍模式，用轻触开关输入固定秘籍后再将拨码开关打开，此时会发现金钱加了 9000，这样就很容易通关了。

4.人性化功能

为了在课堂等不方便发出声音的地方玩，进入秘籍模式后再回来时会进入静音状态，再进入一次后复原。

为了不外接其他设备，重新开始游戏只需拨动一下拨码开关 1 即可。

以上两条巧妙利用了两个拨码开关独立完成了重新开始和静音功能，但如果不小心断电，则需要外界键盘才能运行程序，若记住按键顺序则可不必外界显示器：

开机-20s-win-⬇ -➡ -⬇ -⬇ -⬇ -⬇ -⬇ -⬇ -回车-5s-ctrl+o-输入 b.py-回车-3s-ctrl+F5

三、实现方案及代码分析

程序总的流程是四个模式的切换，用变量 mode 来控制，mode 值

为 0、1、2、3 时分别代表开奖模式、下注模式、选灯模式、秘籍模式。

程序的主体是 dip_switch_status_changed_handler 和 tact_event_handle 两个函数，分别用来控制拨码开关和轻触开关被触发时的响应。

当拨码开关 1 被触发且为开时：初始化所有数据

当拨码开关 2 被触发时：若为关则进入秘籍模式；若为开则回到正常模式并改变是否静音。

当左轻触开关被触发且为按下时：若在下注模式则给正在设置的数位+1，若超过 9 或者超过可设置金额则变为 0；若在选灯模式则选择下一个灯，若超过最后一个灯则变为第一个灯；若为秘籍模式则记为输入一个 0，若秘籍输入正确则金额+9000

当右轻触开关被触发且为按下时：若为开奖模式则进入下注模式；若为下注模式检测是否可以继续设置下一数位，是则进入下一数位的设置，否则进入选灯模式；若为选灯模式则进入开奖模式，随机选出中奖的灯，若选中同一灯则加 10 倍赌注，若选中同一颜色灯则加 5 倍赌注，若金额超过 10000 则播放通关动画，若连输 10 次则金额减半

其他函数详见代码注释。

四、后续工作展望

1. 下注时可以多选几个灯，这样会增加游戏性，因为每轮赢的可能性会增大，可以不断激励人的兴致。但相应策略性会降低，因为这样就不需要有长远的眼光，更多的是凭每一局的运气。

2.可以外接投币机，同时降低概率，原先数学期望是+7/8，把数学期望调至 0 以下即可赚钱。

五、小组分工合作

刘启航：买读卡器，尝试开机，归还仪器

沈泽盛：把显示屏搬上楼，尝试开机，尝试固定树莓派，输入 ppt 上部分代码，试玩游戏，视频摄像

陶天阳：其他工作

（感谢孙景南同学帮忙烧录系统）

附项目源码：

```
from sakshat import SAKSHAT
from sakspins import SAKSPins as PINS
import time
import random

SAKS = SAKSHAT()#瑞士军刀扩展版组件
b = SAKS.buzzer#蜂鸣
c = 100#金额
d = 0#选的灯
l = 0#中奖的灯
x = 0#正在设置的数位
o = 1#是否静音
f = 0#连输次数
a = [0,0,0,0]#下注金额
mode = 0#模式
m = {0:1,1:0,2:3,3:2,4:5,5:4,6:7,7:6}#用以匹配同颜色的灯

def bzy(n):#把数字转成可调用的形式
    s = ''
    s += str(int(n/1000)) if n >= 1000 else '#'
    s += str(int(n/100%10)) if  n >= 100 else '#'
    s += str(int(n/10%10)) if  n >= 10 else '#'
    s += str(int(n%10))
    return s
```

```
def tty(a,x):#把四位数字的列表转为可调用的形式
    s = ''
    for i in range(3-x):
        s += '#'
    for i in range(x+1):
        s += str(a[x-i])
    return s

def num(a):#把四位数字的列表转为数字
    t = 0
    for i in range(4):
        t += a[i]*10**i
    return t

def win():#播放通关动画
    global o
    SAKS.digital_display.show('9999')
    if o:
        b.beep(1.5)
    t = 0
    while True:
        SAKS.ledrow.off()
        SAKS.ledrow.on_for_index(t)
        time.sleep(0.1)
        t += 1
        if t == 8:
            t = 0

def dip_switch_status_changed_handler(status):#触发拨码开关时运行
    global f,o,a,c,d,l,mode,x,bzytty
    if not status[0]:#当拨码开关 1 被触发且为开时：初始化所有数据
        c = 100
        d = 0
        l = 0
        x = 0
        o = 1
        f = 0
        a = [0,0,0,0]
        mode = 0
        SAKS.ledrow.off()
        SAKS.digital_display.show('#100')
    if not status[1]:#当拨码开关 2 被触发时：若为关则进入秘籍模式
        mode = 3
```

```
        bzytty = ''
        SAKS.ledrow.off()
        SAKS.digital_display.show('####')
    if status[1]:#当拨码开关 2 被触发时：若为开则回到正常模式并改变是否静音
        mode = 0
        if o:
            o = 0
        else:
            o = 1
        SAKS.digital_display.show(bzy(c))

def tact_event_handler(pin,status):#触发轻触开关时运行
    global f,o,a,c,d,l,mode,x,bzytty
    if pin == PINS.TACT_LEFT and status == True:#当左轻触开关被触发且为按下时：
        if mode == 1:
            #若在下注模式则给正在设置的数位+1
            a[x] += 1
            #若超过 9 或者超过可设置金额则变为 0
            if a[x] == 10:
                a[x] = 0
            if len(str(c)) == x + 1:
                if str(a[x]) > str(c)[0]:
                    a[x] = 0
                if str(a[x]) == str(c)[0]:
                    t = 0
                    for i in range(x):
                        t += a[i]*10**i
                    if c % 10**x < t:
                        a[x] = 0
            SAKS.digital_display.show(tty(a,x))
        if mode == 2:
            #若在选灯模式则选择下一个灯
            d += 1
            #若超过最后一个灯则变为第一个灯
            if d == 8:
                d = 0
            SAKS.ledrow.off()
            SAKS.ledrow.on_for_index(d)
            SAKS.digital_display.show(bzy(c))
        if mode == 3:
            #若为秘籍模式则记为输入一个 0
            bzytty += '0'
            #若秘籍输入正确则金额+9000
            if bzytty == '100110':
```

```
                c += 9000

    if pin == PINS.TACT_RIGHT and status == True:#当右轻触开关被触发且为按下时：
        if mode == 0:#若为开奖模式则进入下注模式
            mode = 1
            SAKS.ledrow.off()
            SAKS.digital_display.show(tty(a,x))
        elif mode == 1:#若为下注模式检测是否可以继续设置下一数位
            t = 0
            for i in range(x+1):
                t += a[i]*10**i
            if len(str(c)) == x + 1 or (len(str(c)) == x + 2 and c % 10**(x+1) <
t and str(c)[0] == '1'):#否则进入选灯模式
                mode = 2
                c -= num(a)
                SAKS.ledrow.off()
                SAKS.ledrow.on_for_index(d)
                SAKS.digital_display.show(bzy(c))
            else:#是则进入下一数位的设置
                x += 1
                SAKS.digital_display.show(tty(a,x))
        elif mode == 2:#若为选灯模式则进入开奖模式
            mode = 0
            l = random.randint(0,7)#随机选出中奖的灯
            for i in range(41+l):
                time.sleep(-1/(i-41-l))
                SAKS.ledrow.off()
                SAKS.ledrow.on_for_index(i%8)
                if o:
                    b.beep(0.01)
            if l == d:#若选中同一灯则加 10 倍赌注
                f = 0
                c += 10*num(a)
                if c >= 10000:#若金额超过 10000 则播放通关动画
                    win()
                SAKS.digital_display.show(bzy(c))
                if o:
                    b.beep(1)
            elif l == m[d]:#若选中同一颜色灯则加 5 倍赌注
                f = 0
                c += 5*num(a)
                if c >= 10000:#若金额超过 10000 则播放通关动画
                    win()
                SAKS.digital_display.show(bzy(c))
```

```
                if o:
                    b.beep(0.3)
            else:
                f += 1
                if f == 10:#若连输 10 次则金额减半
                    f = 0
                    c = int(c/2)
                    if o:
                        b.beep(0.1)
                        time.sleep(0.1)
                if o:
                    b.beep(0.1)
                SAKS.digital_display.show(bzy(c))
            a = [0,0,0,0]
            x = 0
        else:
            bzytty += '1'

if __name__ == '__main__':
    SAKS.tact_event_handler = tact_event_handler
    SAKS.dip_switch_status_changed_handler = dip_switch_status_changed_handler
    SAKS.ledrow.off()
    SAKS.digital_display.show('#100')
    input()
```

树莓派--防止狗扒垃圾桶报警装置

生命科学学院
谢冠旖（组长）1300012134　　端韵成 1300012136

一. 选题介绍

1. 家中有宠物的人经常会遇到宠物乱翻垃圾的烦恼。主人既不想过多的限制宠物的自由，又不想让宠物因为误食垃圾受到伤害。一个能够阻止宠物乱翻垃圾的装置可以很好的解决这一问题。此次发的扩展板上面有蜂鸣器，红外和温度传感器以及开关，可以制成报警装置，因此定下这一主题。
2. 如果只是设置一个普通的在宠物常出现的位置上报警装置，那么人也可能经过，倘若不能区分宠物与人，这个报警装置就没有意义了。所以我们希望制作一个放在垃圾桶附近的增加了功能的装置，就可以在不影响人类活动的基础之上，防止宠物扒垃圾桶，即只有宠物出现的时候才会报警的装置。

二. 设计方案

1. 怎样能使装置分辨出人和狗：人类和狗在身高上的差异很大，所以针对这个不同进行装置的设置。
2. 首先在靠墙的高低两个位置放置两个传感器，这两个传感器理论上可以为任何类型的传感器，比如温度，声音，红外。此次设计，考虑到灵敏度的问题，我们选择了两个红外传感器。
3. 两个红外传感器都同时有红外发射管和红外接收管。连上电源后，通过调整树莓派内置系统，设置红外输入接口，在一米的范围内，如果接收到信号，则输入为 1，否则为 0。
4. 当狗出现在垃圾桶附近时，只能触发低端的红外传感器，而人出现的时候，可以同时触发两个红外传感器，那么只有当低端传感器被触发时才让蜂鸣器报警就可以了。

三. 思路流程图

1. 狗或人接近时，低端的传感器有信号。
2. 此时去检测高端的传感器有无信号。
3. 如果有信号，则此时是人出现在附近，蜂鸣器不响；如果没有信号，则认为是狗在附近，此时蜂鸣器报警，为了演示效果，我们让 LED 灯也闪烁。
4. 此时如果主人及时赶到，可以通过按开关，使报警装置停下，LED 灯灭；否则，在铃响 20 次后，报警装置自动停下。
5. 如果人手动关闭报警开关，则所有的 LED 灯均熄灭；如果是报警装置自动停下的情况，最后一个红色 LED 灯会持续亮。

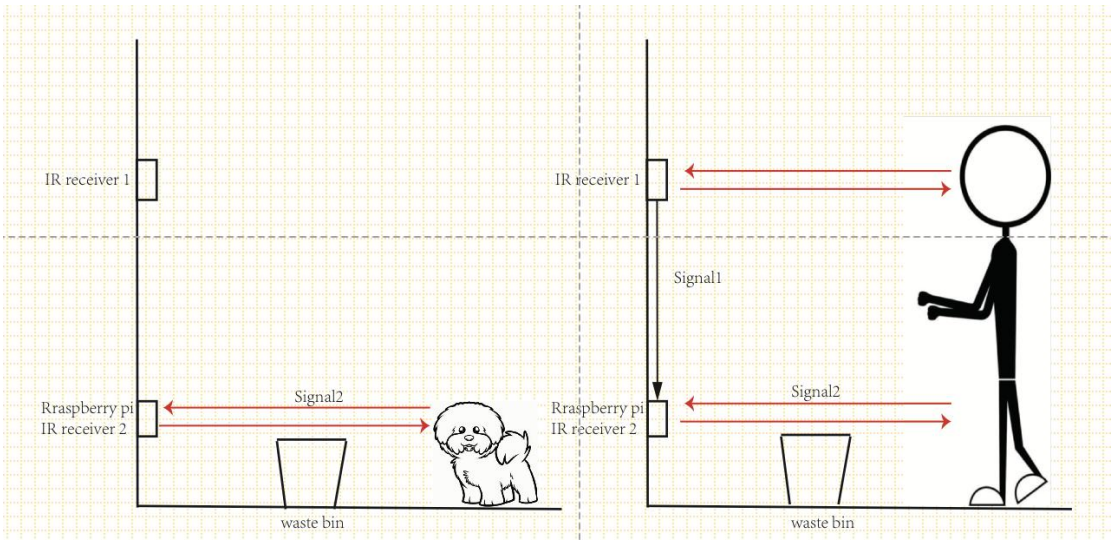


图 1 设计示意图

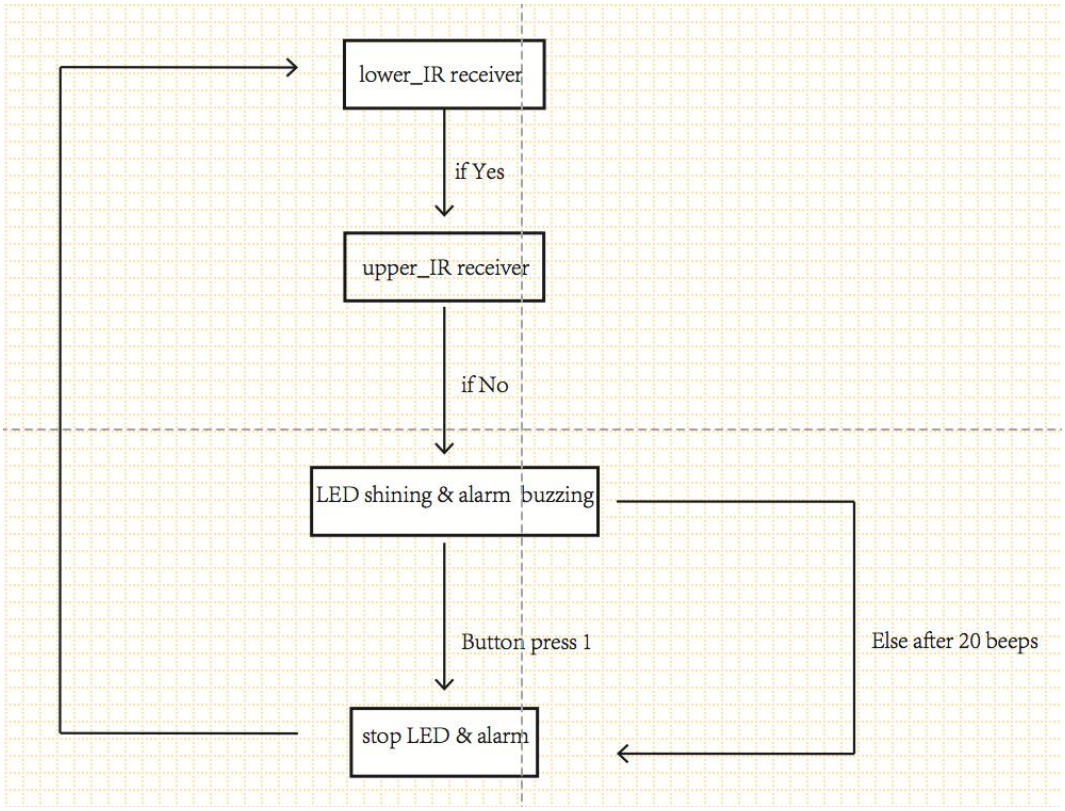


图 2 程序流程图

四.实现介绍

（一）主函数：

主要实现通过监测上下两个红外线是否受到遮挡，确定是否启动报警程序。
1.默认开关未被按下，计数值为 0，报警循环数为 0。
2.持续监测下部红外线，当检测到信号为 1 时，开始计数。
3.此时检测上方红外传感器，如果此时未接收到上方信号，即上方为 0，且计数达到 80 时，开始执行报警程序。
4.当检测到开关被按下或是报警循环数达到 5 次时，停止报警。
5.另外设置如果检测到上方信号为 1 时，则计数值为 0。

（二）报警系统：

1.当传感器被触发后，八盏 LED 灯轮流亮，蜂鸣器间隔报警。
2.LED 灯：代码中，Data>> i 就是把 data 里面的 1 向右挪 i 位，&0x01 就是比较挪过的数字和 0x01 是否相等，Writebit 之后是把那个二进制的数字串变成了分立的 0，1 码，按顺序存到寄存器中，表示 LED 灯的代码，通过调用拓展版内置的操作，实现对相应的 LED 灯的控制（详见代码）。
3.蜂鸣器：通过设置 GPIO 输出的 LOW 和 HIGH 来实现间断报警，LOW 的时候蜂鸣器响，HIGH 的时候为蜂鸣器关闭状态，期间根据需要设置间歇时常，形成间断报警声。

（三）如何停止报警：

1.主函数中设置按下按钮则启动报警系统终止的程序，这个函数首先设置对报警系统整体进行终止，使该轮报警后不再连续报警，类似清零的作用。
2.如果不进行操作，最后会亮一盏红灯。
3.如果在报警期间触发开关，则在 LED 和蜂鸣器报警一轮后进行关灯操作。
4.最后停止报警时，把 0x00 对应的 8 位存入寄存器，不代表任何灯，使 LED 灯熄灭，同时将蜂鸣器输出设置为 HIGH 状态，关闭蜂鸣器。

五.实验结果

1.实验结果符合预期，在宠物狗出现的时候，报警系统开启，蜂鸣器报警，LED 灯闪烁；在人类出现的时候，报警装置不开启。
2.当报警装置开启之后，主人手动可以关闭报警装置；而倘若没有手动关闭装置时，20s 后装置自动关闭，且最后一盏红色 LED 灯亮。

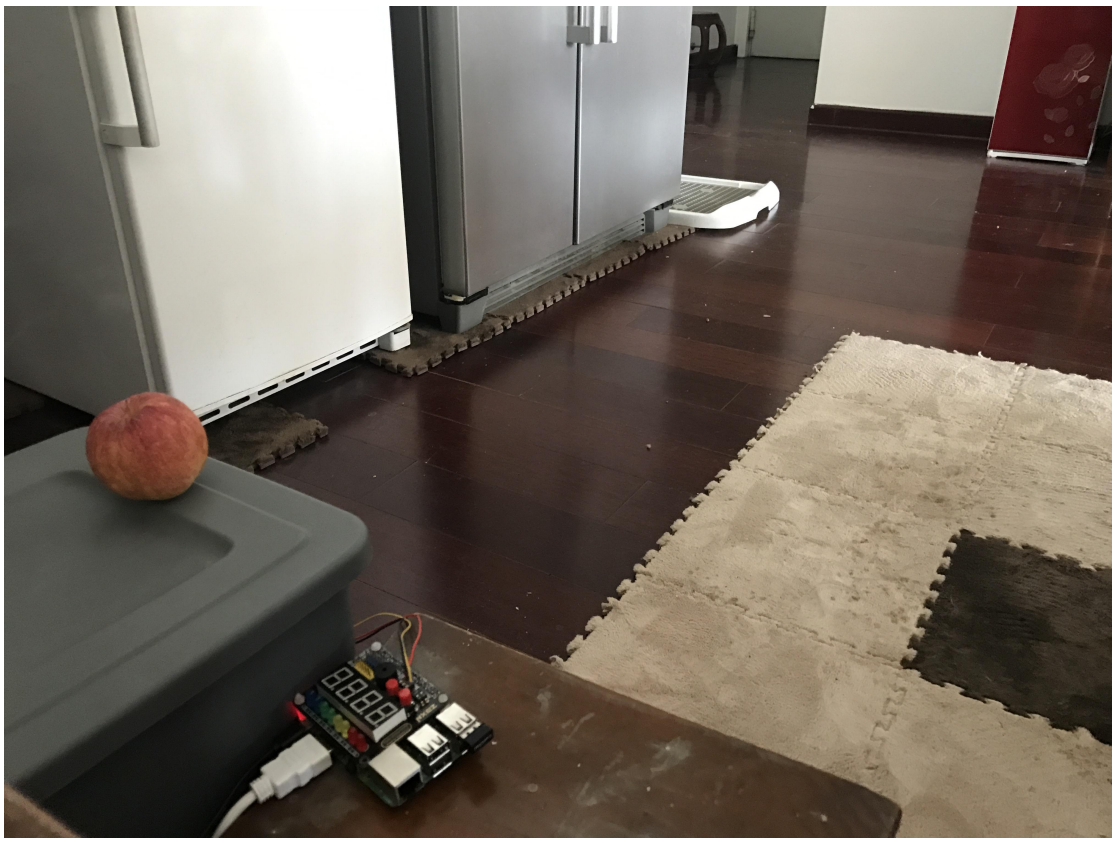
六. 后续工作展望

1. 首先，不同的狗的反应速度不同，可能有的狗扒拉的时候一下就搞定，有的需要时间长一点，这样的话我们最好不要每次都在程序里重新设置时间；扩展板上还有数码管，我们可以通过按按钮来调整反应时间并显示，比如每按一次钮，增加五秒的反应时间，这样可以因狗而异。
2. 同时，还可以优化硬件，使硬件更美观，我们现在的装置，六根杜邦线都露在外面，可以外面加一个硬壳固定好，完成装置一体化。
3. 最后，由于测试太多次，狗狗们容易不害怕报警器，树莓派 3 已经可以接蓝牙了，可以录制好主人的声音，当狗狗接近装置时，通过蓝牙音箱播放，起到警戒作用。

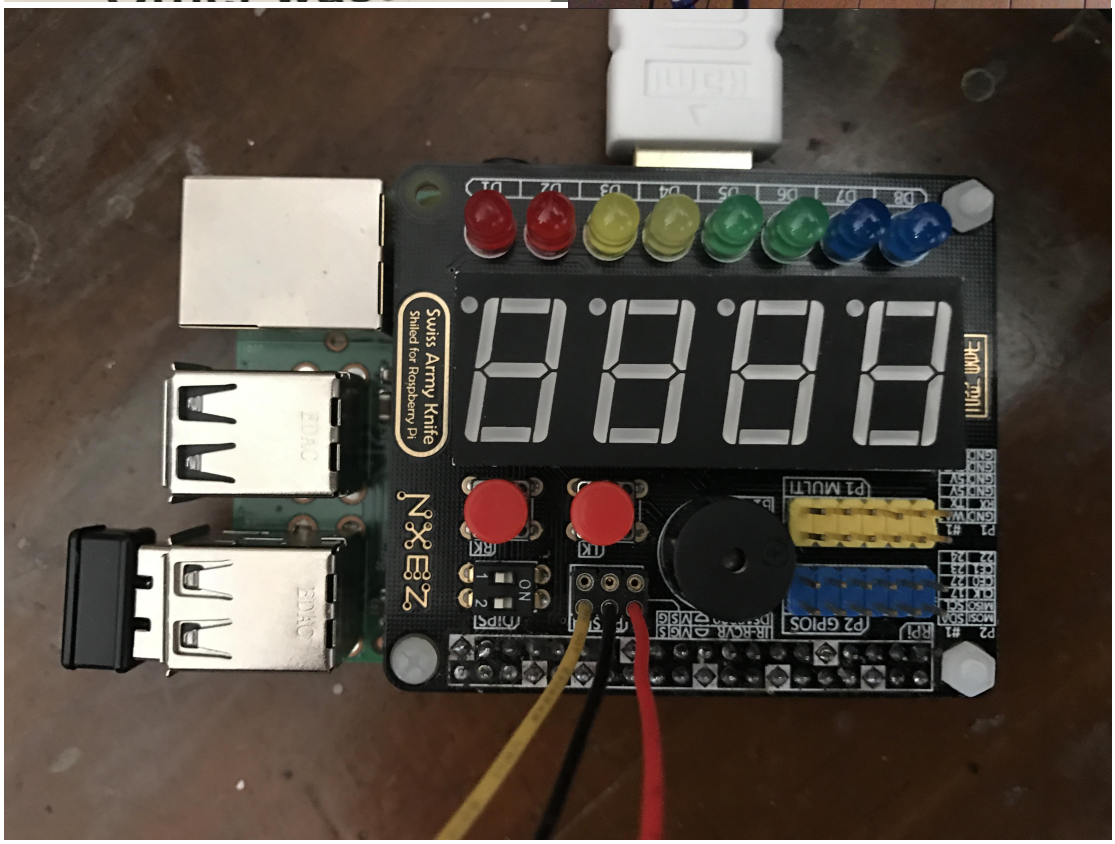
七. 小组分工

1. 两个传感器的创意来源：端韵成
2. 软件和硬件的实现：谢冠旖，端韵成
3. 视频制作：谢冠旖
4. ppt 制作：端韵成
5. 撰写实验报告：谢冠旖，端韵成

附相关图片：



实验场景



左上图：传感器位置　　右上图：高低传感器连接

下图：低传感器连接

附项目源码：

```
import RPi.GPIO as GPIO
#from sakshat import SAKSHAT
import time
import os,sys
import signal

#SAKS = SAKSHAT()
GPIO.setmode(GPIO.BCM)       # BOARD 编号方式，基于插座引脚编号
GPIO.setwarnings(False)
DS = 6
SHCP = 19
STCP = 13                    #LED 灯
LKEY = 16
RKEY = 20                    #左右开关
BEEP = 12
LIR_RCV = 18                  #地上红外线
HIR_RCV = 17                 #较高位置红外线

def init():
    #输出方式
    GPIO.setup(DS,GPIO.OUT)
    GPIO.setup(SHCP,GPIO.OUT)
    GPIO.setup(STCP,GPIO.OUT)        #led 灯
    GPIO.setup(BEEP, GPIO.OUT, initial = GPIO.HIGH)        #蜂鸣器
    #输入方式
    GPIO.setup(LIR_RCV,GPIO.IN)
    GPIO.setup(HIR_RCV,GPIO.IN)     #红外线
    GPIO.setup(LKEY,GPIO.IN, pull_up_down = GPIO.PUD_UP)
    GPIO.setup(RKEY,GPIO.IN)        #开关,输入方式

def writeBit(data):
    GPIO.output(DS, data)
    GPIO.output(SHCP, GPIO.LOW)
    GPIO.output(SHCP, GPIO.HIGH)

def beep(seconds):                    #蜂鸣器状态
    GPIO.output(BEEP, GPIO.LOW)
    time.sleep(seconds)
    GPIO.output(BEEP, GPIO.HIGH)

def led_buzzer(data):
    for i in range(0, 8):
```

```
        writeBit((data >> i) & 0x01) #写入 8 位 LED 的状态
    GPIO.output(STCP, GPIO.LOW)
    GPIO.output(STCP, GPIO.HIGH)
    beep(0.2)

def alarm():
    global done
    done=False
    for i in [0x01,0x04,0x10,0x40,0x02,0x08,0x20,0x80]:  # 一组 8 个编码由一组二
进制转换而成,分别对应 8 个 LED 点亮状态
        led_buzzer(i)
        time.sleep(0.2)
    done=True

def turn_off(channel):       #终止报警系统
    global key,done
    key = True       #对报警系统整体进行终止，该轮报警后不再连续报警
    #关闭 LED 最后的灯（不进行该操作最后的灯会一直亮）
    while done is False:         #在 LED 一轮结束后再进行关灯操作
        if done==True:
            break
    if done==True:
        for i in range(0, 8):
            writeBit(0x00 & 0x01)      #把 0x00 对应的 8 位存入寄存器，不代表任何灯
        GPIO.output(STCP, GPIO.LOW)
        GPIO.output(STCP, GPIO.HIGH)
    time.sleep(1)

def main():
    init()
    global  alarm_beep_times, key, count
    GPIO.add_event_detect(LKEY,GPIO.FALLING, callback=turn_off)
    while True:                                #持续监测下部红外线
        key = False
        count = 0
        alarm_beep_times = 0
        while GPIO.input(LIR_RCV) is 1:      #检测到下部的红外线受到遮挡
            count+=1
            if GPIO.input(HIR_RCV) is 0:     #未检测到上部红外线受遮挡--说明不是
人!! 开始计数
                count += 1
                print(count)
                print(GPIO.input(HIR_RCV))
```

```
                while alarm_beep_times < 4 and key is False and count > 80:      #
计数达到 80 则开始执行报警程序
                    alarm()
                    alarm_beep_times += 1
            else:   #检测到上部红外线受阻--说明是人在倒垃圾
                count=0
main()
```

树莓派——多功能篮球计分器

组长：石晓霏
组员：胡若冰、陈玄同

一、选题及创意介绍

如今，中国高校的篮球比赛在校园里越来越流行，北京大学作为中国最高学府之一，不仅学术上有所成就，而且课余活动也尤为丰富。PKUBA（北大篮协）主要有三个面向全校的比赛：新生杯、北大杯和超级杯。这让我们不禁想到最近如火如荼的北大杯男篮决赛。

对于比赛的最终比分，裁判占有主要权利。出于对球赛公平的考虑，避免比赛选手的纠纷，裁判主观因素的影响和减少裁判的压力四点考虑，我们拟定遵循“计分计时电子化，进球自动化”的原则，即进球就有提示功能，在篮框内置红外线传感器，进球即发出“滴~~”一声；另外为了方便裁判计分，我们将把纸质版的比分电子化，顺带加上计时功能。同时，我们还会为观众考虑，如果观众想来学校看比赛却因为有事来不了，就可以通过拍摄直播给不能到场的同学们提供服务。

二、设计方案

材料准备：树莓派摄像头（淘宝购买）、红外线传感器（同学送的）、网线……
目标：完成五个功能
要求：完成基本功能，在此基础上添加有趣的功能
①利用 pygame 在树莓派上写计时计分的界面
②连接红外线传感器和摄像头
③运行代码
④执行代码时：
1)拓展版基本功能，在关键时刻会给出提示以及手动计时计分
—两个开关：一个管 24s 倒计时，3s 灭一盏灯，全灭之后 buzzer 就发出声音；另一个控制暂停时间，2min 后全闪和发出提示声（显示板时间暂停）
—两个按钮分别控制双方得分
（即按一下加一分）
—支持对每节结束、中场休息开始和结束进行提示。

2)在 PC 端 IP 地址输入树莓派的端口号，摄像头指向我们录得视频，传输给 PC 端进行直播
—红外线传感器，感应到有遮挡物就灯光+声音提示
—摄像头 运用流媒体实现

三．实现方案及代码分析

（1）实现方案：

我们使用瑞士军刀拓展版上的按键和拨码开关来操作比赛的计分，控制 24 秒的计时和比赛中间的暂停；代码内有计时的操作，可以在每一节比赛的开始和结束是有声音和灯光的提示，并在比赛结束后显示获胜的队；为了实现感应进球的功能，我们外加了红外感应器，可以放置在篮筐内，用于提示比赛时有进球；我们还外接了摄像头，用于实现篮球比赛的直

播功能，在具体实践的过程中，我们选择使用流媒体的方式将视频发布出来，使所有在同一网络下的 pc 都可以通过简单的操作看到树莓派的直播。我们尝试了很多的方式来实现直播功能，最先尝试的 pygame 中的 movie 和 camera 模块，但可能是由于 pygame 版本的问题，至两个模块无法成功导入，所以我们又尝试了可以用于监控的 motion 操作，我们虽然成功实现了 motion 模块，但是 motion 的拍摄时 2s 一帧，无法满足比赛直播的需求，因此为我们最终选择用流媒体来实现我们的创意，通过设定一个网络协议将树莓派拍摄到的视频实时发布出去，让很多人都可以在自己的 pc 端上观看，这样的播放比较流畅，与我们的设想很符合。最后我们设计了一个简单的 pygame 界面用来显示比赛的比分和剩余时间，使计时更加直观。

（2）代码分析（为主体的操作部分）

```
#系统的计时和拓展版的操作
    while clock<=2400:
        #clock 是计时器，计时间隔为一秒
        time.sleep(1)  #用 time.sleep 来计时
        clock=clock+1
        print(clock)
        if clock%600==0:
        #以下为自动记录比赛时间，在每一节的开始和结束有提示
            SAKS.ledrow.on()
            ring()
            if clock!=1200:
                time.sleep(120)
            if clock==1200:
                time.sleep(600)
            SAKS.ledrow.on()
            ring()
       #以下是两个函数，分别用于控制两个按钮和两个拨码开关，来完成相应的功能
        SAKS.dip_switch_status_changed_handler=dip_switch_status_changed_handle
        r
        SAKS.tact_event_handler=tact_event_handler


 #以下为 pygame 的界面设置
        pygame.init()
        screen = pygame.display.set_mode((640, 480), 0, 32)
        background_image_filename = 'tim3.jpg'
        background = pygame.image.load(background_image_filename).convert()
        screen.blit(background, (0,0))
        pygame.display.update()
        if clock==2400:
            de='GAME OVER
'            my_font2=pygame.font.SysFont(None,120)
            text_screen2=my_font2.render(de, True, (255, 0, 0))
            screen.blit(text_screen2, (50,200))
```

```
            pygame.display.update()
            time.sleep(10)
        points = []
        my_font=pygame.font.SysFont(None,50)
        my_font1=pygame.font.SysFont(None,350)
        screen.blit(background, (0,0))
        textstr='time:'+str(clock)
        s=str(m)+' '+':'+' '+str(a)
        text_screen=my_font.render(textstr, True, (255, 0, 0))
        text_screen1=my_font1.render(s, True, (255, 0, 0))
        screen.blit(text_screen, (0,0))
        screen.blit(text_screen1, (100,120))
        pygame.display.update()
    print("GAME OVER")

#以下为比赛结束后执行的部分，同时展示出获胜的队伍
    if m<a:
        SAKS.digital_display.show("0011")
        SAKS.ledrow.off_for_index(0)
        SAKS.ledrow.off_for_index(1)
        SAKS.ledrow.off_for_index(2)
        SAKS.ledrow.off_for_index(3)
        time.sleep(4)
    else:
        SAKS.digital_display.show("1100")
        SAKS.ledrow.off_for_index(4)
        SAKS.ledrow.off_for_index(5)
        SAKS.ledrow.off_for_index(6)
        SAKS.ledrow.off_for_index(7)
        time.sleep(4)
    GPIO.cleanup()

#红外线感应部分
#主要是通过判断输出信号的状态来判断是否进球

#流媒体的实现部分
声明：为了实现这个功能，我们搜索了一些前人的工作用于参考。
代码中首先是几个类的定义（这里是其中的一部分），创建连接所需要的协议，代码的实现
需要一些模块的支持，在这里没有展示。
class StreamingHttpHandler(BaseHTTPRequestHandler):
    def do_HEAD(self):
        self.do_GET()

    def do_GET(self):
```

```
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
            return
        elif self.path == '/jsmpg.js':
            content_type = 'application/javascript'
            content = self.server.jsmpg_content
        elif self.path == '/index.html':
            content_type = 'text/html; charset=utf-8'
            tpl = Template(self.server.index_template)
            content = tpl.safe_substitute(dict(
                ADDRESS='%s:%d' % (self.request.getsockname()[0], WS_PORT),
                WIDTH=WIDTH, HEIGHT=HEIGHT, COLOR=COLOR, BGCOLOR=BGCOLOR))
        else:
            self.send_error(404, 'File not found')
            return
        content = content.encode('utf-8')
        self.send_response(200)
        self.send_header('Content-Type', content_type)
        self.send_header('Content-Length', len(content))
        self.send_header('Last-Modified', self.date_time_string(time()))
        self.end_headers()
        if self.command == 'GET':
            self.wfile.write(content)

class StreamingHttpServer(HTTPServer):
    def __init__(self):
        super(StreamingHttpServer, self).__init__(
                ('', HTTP_PORT), StreamingHttpHandler)
        with io.open('index.html', 'r') as f:
            self.index_template = f.read()
        with io.open('jsmpg.js', 'r') as f:
            self.jsmpg_content = f.read()

class StreamingWebSocket(WebSocket):
    def opened(self):
        self.send(JSMPEG_HEADER.pack(JSMPEG_MAGIC, WIDTH, HEIGHT), binary=True)

......
......
#以下为主体的实现部分
def main():
    print('Initializing camera')
```

```
    with picamera.PiCamera() as camera:
        camera.resolution = (WIDTH, HEIGHT)
        camera.framerate = FRAMERATE
        sleep(1) # camera warm-up time
        print('Initializing websockets server on port %d' % WS_PORT)
        websocket_server = make_server(
            '', WS_PORT,
            server_class=WSGIServer,
            handler_class=WebSocketWSGIRequestHandler,
            app=WebSocketWSGIApplication(handler_cls=StreamingWebSocket))
        websocket_server.initialize_websockets_manager()
        websocket_thread = Thread(target=websocket_server.serve_forever)
        print('Initializing HTTP server on port %d' % HTTP_PORT)
#这里显示的是连接的端口号，用户想在自己的 pc 看到直播，只需要在输入网址的那一栏中
输入树莓派的 ip 地址，加上一个冒号，在输入这里显示的端口号即可
        http_server = StreamingHttpServer()
        http_thread = Thread(target=http_server.serve_forever)
        print('Initializing broadcast thread')
        output = BroadcastOutput(camera)
        broadcast_thread = BroadcastThread(output.converter, websocket_server)
        print('Starting recording')
        camera.start_recording(output, 'yuv')
#一下是对用户的一些信息提示，和出错时的报错
        try:
            print('Starting websockets thread')
            websocket_thread.start()
            print('Starting HTTP server thread')
            http_thread.start()
            print('Starting broadcast thread')
            broadcast_thread.start()
            while True:
                camera.wait_recording(1)
        except KeyboardInterrupt:
            pass
        finally:
            print('Stopping recording')
            camera.stop_recording()
            print('Waiting for broadcast thread to finish')
            broadcast_thread.join()
            print('Shutting down HTTP server')
            http_server.shutdown()
            print('Shutting down websockets server')
            websocket_server.shutdown()
            print('Waiting for HTTP server thread to finish')
```

```
            http_thread.join()
            print('Waiting for websockets thread to finish')
            websocket_thread.join()

if __name__ == '__main__':
    main()
#运行函数就可以实现功能
```

四．小组分工

创意想法和编程实现均为小组成员合作实现

五．后续工作展望

我们的树莓派篮球计时器还有很多方面需要改进，我们也想出了一些相应的解决方案：

（4） 视频直播时无法同时在视频下方显示比分和时间

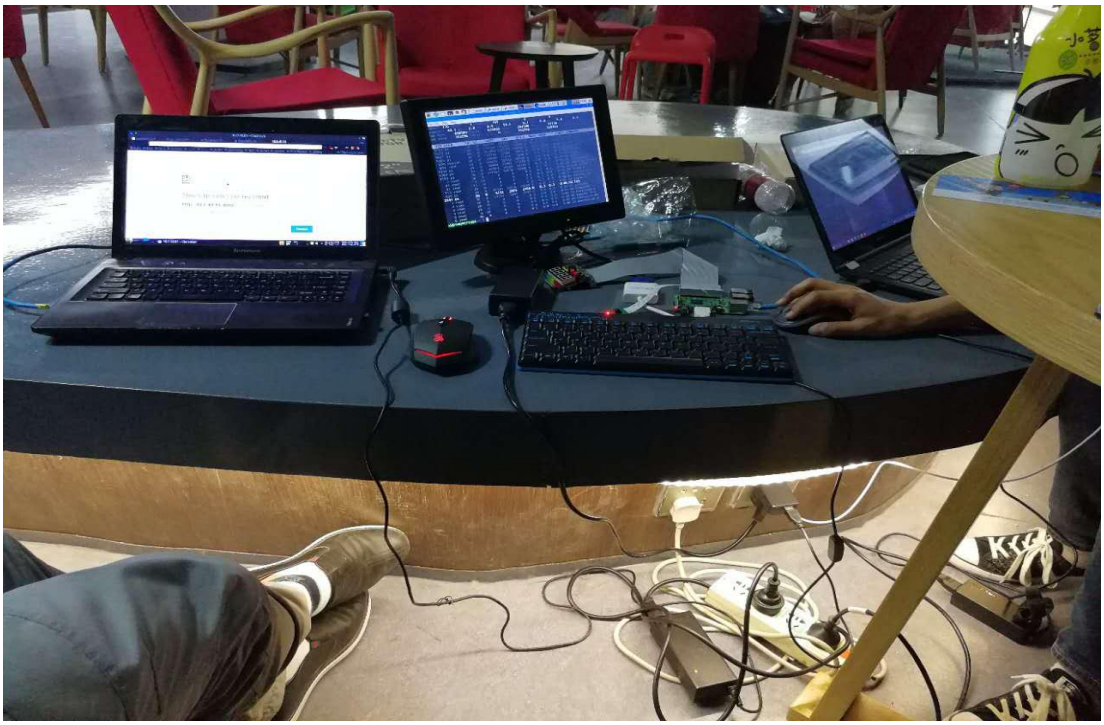
解决方案：用 pyqt 代替 pygame，将 pyqt 与流媒体结合起来。Pyqt 界面中可以有一个框显示比分，另一个框显示时间，第三个框演示视频，并将这个界面发布出去，实现计时计分和视频直播的结合。我们因为对 pyqt 的掌握不够，并且时间有限，没能完成这个想法。

（5） 视频直播时无法播放声音

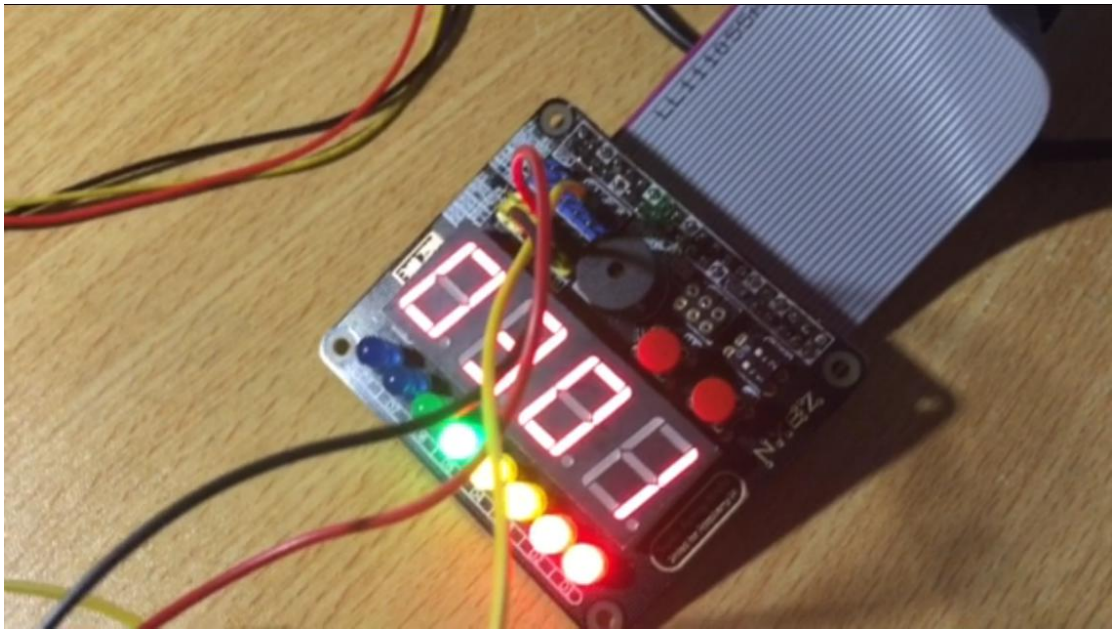
解决方案：这个问题主要是硬件方面的限制，可以添加相应的声音获取装置，将声音和视频同时发布出去

（6） 无法多角度观看比赛

解决方案：这个问题也是出于硬件方面的限制，可以连接多个摄像头，放置在不同的位置，在树莓派操作界面上选择需要为观众播放哪一个角度的视频。



主界面



树莓派显示

树莓派活动报告

经济学院 任庆杰 1400015500

经济学院 王梦瑶 1400015507

主题

树莓派面部签到系统

内容

1. 选题及创意介绍

签到是人们在集体活动中经常运用的一种确定人员出席的方式。常用的签到方式包括点名、签名，课堂上常使用的签到方式还包括布置随堂作业、通过固定座位，一些新奇的签到方式还包括固定座位、网络签到（如果微信抢红包签到等）。

本小组考虑到之前的签到方式的复杂性，与树莓派特有的方便与智能因素，设计了树莓派面部签到系统。通过面部识别和面部对比，实现一个简单的自动化签到系统。

2. 设计方案

实现工具：树莓派、瑞士军刀扩展版、摄像头、显示器、树莓派的其他必要零件（如电源、数据线、内存卡等）。

实现语言：Python3.4。

设计流程：

本系统的工作流程主要分为三个阶段：

1. 接入数据。实现签到系统，首先要获取应该签到的人员的名单。第一步，即为从数据库或已有的内存中加载数据。对于本系统而言，则需要从内存中加载已知名册的人员名单以及对应的面部图片（寸照），并处理，便于下一步进行比对。
2. 拍照分析签到情况。利用树莓派的摄像头进行拍照，获取当前进行签到（实际到达）人员的合照。利用合照，首先通过面部识别，得到每个出勤人的面部照片；然后所得到的照片与第一步加载的数据进行比对，确认每位应出席人是否实际抵达，并保存数据；最后，将结果进行汇总。
3. 返回结果。将上一步得到的结果，进一步通过可视化的界面展示出来。包括通过树莓派扩展版、通过显示器及其它的可展示方法。

3. 实现方案及代码分析

下面我们开始具体地分析代码的运行。

本程序主要调用了系统程序包、树莓派函数的程序包与 *python* 中图像处理和机器学习的程序包。

```
import os,time
import picamera, RPi, sakshat
import PIL, dlib, face_recognition as fr
```

具体过程主要分为五个部分：

1. **加载数据库。**前面我们已经讨论过，签到系统需要加载数据库中的姓名与照片名单，作为比较的对象。

```
# 处理已知的名单和已知的图片
path="" ## path of image save
known_image = []
for i in os.listdir(path+'leaderFaces'):

known_image.append(face_recognition.load_image_file('leaderFaces
/' + i))

# 2 个字典用来记录姓名与编号
leaders = {}
inv_leaders = {value:key for key, value in leaders.items()}
```

2. **拍照**。利用树莓派自带的相机，来实时抓取获取照片，并将照片储存。本实验中由于相机需要手动对焦，所以设置为延时拍照。

```
# 延时拍照函数
def take(camera=c, t=10):
    camera.start_preview()
    time.sleep(t)
    camera.capture("sample.jpg")
    camera.stop_preview()
```

3. **识别**。将上一步获得的照片进一步处理。首先从合照中逐一识别人脸，并将每一个人脸截图保存。然后将人脸与第一步在数据库中加载的名单进行比对。

```
# sample 为样本，是一个图片格式，通过拍照获得
image = face_recognition.load_image_file(sample)
face_locations = face_recognition.face_locations(image)

#记录人脸
face_image_list = []

# 逐一截取人脸
for face_location in face_locations:
```

```
        #先获得每一张面部的上下左右边界
        top, right, bottom, left = face_location
        #依据边界进行截图并放到一个列表中
        face_image = image[top:bottom, left:right]
        face_image_list.append(face_image)
      #将截图的人脸与数据库中数据比对
      unknown_encoding                                    =
face_recognition.face_encodings(face_image)[0]
```

4. **汇报结果**。此处只得到了字典型结果。并没有通过其他的数据形式展示。此外，结果还利用扩展版进行展示。如果全员到齐，则扩展版显示绿灯；否则显示红灯。同时，扩展版数字屏显示缺席人数。

```
# results 记录成功签到和未签到的人数
results = {inv_leaders[i]: False for i in inv_leaders}

# 将每个人员是否被签到（识别成功）的情况，放到一个字典中储存。
if    True    in    face_recognition.compare_faces(known_encoding,
unknown_encoding, 0.5):  # 0.5 是识别准确度
    # 识别情况
    recognition  =  face_recognition.compare_faces(known_encoding,
unknown_encoding, 0.5)
    # 确定对应的人
    leader = recognition.index(True)
    # 添加到结果中
    results[leaders[leader]] = True
```

4. 后续工作展望

本小组的工作只是完成了面部签到系统的理论设计，距离一个成熟的、可以投入使用的签到系统尚有一些距离。首先，在接下来的工作中，可以考虑将加载的数据库从内存中的静态数据库转向服务器或者云端的动态数据库。其次，汇报分析结果的方式也比较单一。可以考虑将字典型结果进一步处理，如上传到服务器中、和历史数据进行合并与比较等。当然，最困难的一项工作是识别准确度的提高。本程序中设置的值为 0.5，当选择更大的值时，会造成即使为同一人也难以识别的现象；而如果值设置过小，那么则会难以区分一些图片（之前的测试中，难以识别区分刘诗诗和唐嫣）。

5. 小组分工合作

小组分工：由于本小组只有 2 名成员，而实现本次活动的代码与其他劳动均为之前未接触的领域，所以主要内容为两人合作共同学习完成。

大致上，王梦瑶同学负责了主程序的书写，任庆杰负责了代码在树莓派上的调试与实现。活动展示内容与活动报告的撰写等其它工作由二人共同完成。

最后，特别感谢在本次活动中一直提供帮助的助教陈旭。因为显示器和其它的各种问题，多

次打扰到助教；而助教总是耐心地给予答复和鼓励。是我们得意顺利完成本次作业。同时感谢陈斌老师在本学期数据结构与算法课程上的付出，使我们度过了一个快乐而充实的学期的同时，从编程小白逐渐熟悉灵活强大的 *python* 语言与神奇的程序算法。

附图：

测试用图片

树莓派小游戏开发

组长：傅昊博
组员：祝奇文

一、选题及创意介绍

听完陈斌老师在课上对树莓派的介绍，我们对树莓派的最直接印象居然都是——这是一个小型游戏机。（捂脸.jpg）我们想法的出发点是树莓派上的 IO 设备，主要是两个触控按键、LED 灯以及数字显示屏。结合傅同学丰富的游戏经历，我们决定先实现一个名为“连携技”的小游戏。然后在此基础上利用两个触控按键实现更多类型、更多玩法的小游戏。但是在最后，由于时间和技术有限，我们只完成了“连携技”和“别踩白块儿”两个小游戏。

二、设计方案

我们的设计基于树莓派的瑞士军刀扩展板的功能。

“连携技”的想法非常简单，每次在数字屏上随机显示由“1”或“2”组成的 4 个数字，其中“1”对应左键，“2”对应右键。要求玩家在规定时间内按照完全正确的顺序按下对应的按键，每正确按下一个按键，该按键显示即变为“6”以表示鼓励和膜拜，同时也可以告诉玩家目前的按键进度，如果规定时间内四个按键全部完成那么就得一分，同时难度增加继续游戏。LED 流水灯实现计时显示功能，即八个流水灯从左至右依次亮起，当八个灯全亮时，计时结束。每当按下错误的按键或者八个灯全亮时，数字屏上会显示“2333”表示嘲讽。展示玩家的最终得分并将其清零，然后重新开始游戏。

随后我们按照类似的思路，在树莓派上实现我们宿舍的人气游戏“别踩白块儿”和“完美钢琴”。基本的想法是用数字 1 到 7 代替 7 个音高，在数字屏上从右往左滚动显示简谱数字；当某个数字移动到最左端时，需要在这时按下对应的按键（奇数对应左键，偶数对应右键）；正确按下按键时播放这个数字对应的音，并得到一分，漏按和错按不给分；最后显示按对数字的比例来表示玩家对这首音乐的完成度。

三、实现方案及代码分析

“连携技”部分

【按键操作】

使用 SAKS 模块中的 tact_event_handler 来实现按键事件的捕捉。设置对每两次按键事件间隔时间的监控，并且如果这个时间间隔小于或等于 0.08 秒，那么不认为这是两次按键操作，而将其只视为 1 次。这样就避免了扩展板按键将一次按下并抬起事件误认为是两次按下按键的问题。

tact_event_handler 可以实现对按键事件的随时捕捉，所以计时函数可以和 tact_event_handler 分开。每次捕捉到按键事件后，判断按下的是左键还是右键，再判断是否按下了正确的按键。并在此条件下做出相应的处理和判断。

【鼠标操作】

使用 pygame 模块实现鼠标事件的捕捉。由于 pygame 对鼠标的捕捉需要等待，也就是不能实现对鼠标行为的随时捕捉。我们的方案是监测时间间隔。如果时间间隔没有达到流水灯的计时单位，那么就在 while 循环里面等待鼠标输入并不断更新时间间隔。当时间间隔达

到流水灯的计时单位时，那么就让下一个 LED 小灯亮起。用 for 语句反复循环八次，如果还没有全部正确点击鼠标，那么循环结束，判定为游戏失败。同时，与按键操作一样，如果出现按错的情况，也会被判定为游戏失败。

“别踩白块儿”部分

【滚动数字简谱】

将乐曲简谱按每个数字代表半拍，数字 1 到 7 代表七个音高的规则编写成一个长字符串，同时按照简谱规则，用“0”代替空拍。将显示用的字符串从左到右依次读取，每次做切片取四个字符，每隔一段时间取切片的位置增加 1，实现简谱数字的滚动显示。我们仍然采用【按键操作】中的 tact_event_handler 来实现对按键事件的随时监控，并将监测到的按键与切片中第一个字符对比，判断是否按下了正确的按键。如果按下了正确的按键，那么就调用 pygame 模块的 music 部分函数播放对应音高的声音。为了防止在一个音符内同时多次按下正确按键导致重复得分与音符重复播放，我们还设置了一个 flag，在第一次播放后即被修改。

四、后续工作展望

【不稳定的速度】

为了保证音乐的节奏感，需要“别踩白块儿”程序在运行时稳定地读取简谱字符串。但是经过我们的测试，实际运行时会产生随机的速度不稳定，甚至在速度设置的较快时会发生跳过某些音符的情况。这就使我们不得不降低简谱滚动的速度，从而造成音乐节奏变慢。这可能和树莓派的性能有关，但我们相信可以通过简化代码的时间复杂度来使程序运行更稳定。这是我们继续努力的一个方向。

【不够良好的音乐效果】

我们设计的乐曲播放方式是单纯的采用一种乐器进行音符的线性组合，这种音乐效果常见于新手刚刚开始练习演奏某个乐曲的情况。但作为一个音游，我们设计的小游戏的音乐效果应该远好于此。我们认为一方面可以考虑采用多音轨，设置背景音乐，另一方面可以尝试剪取原音乐的片段来组成完整的歌曲。

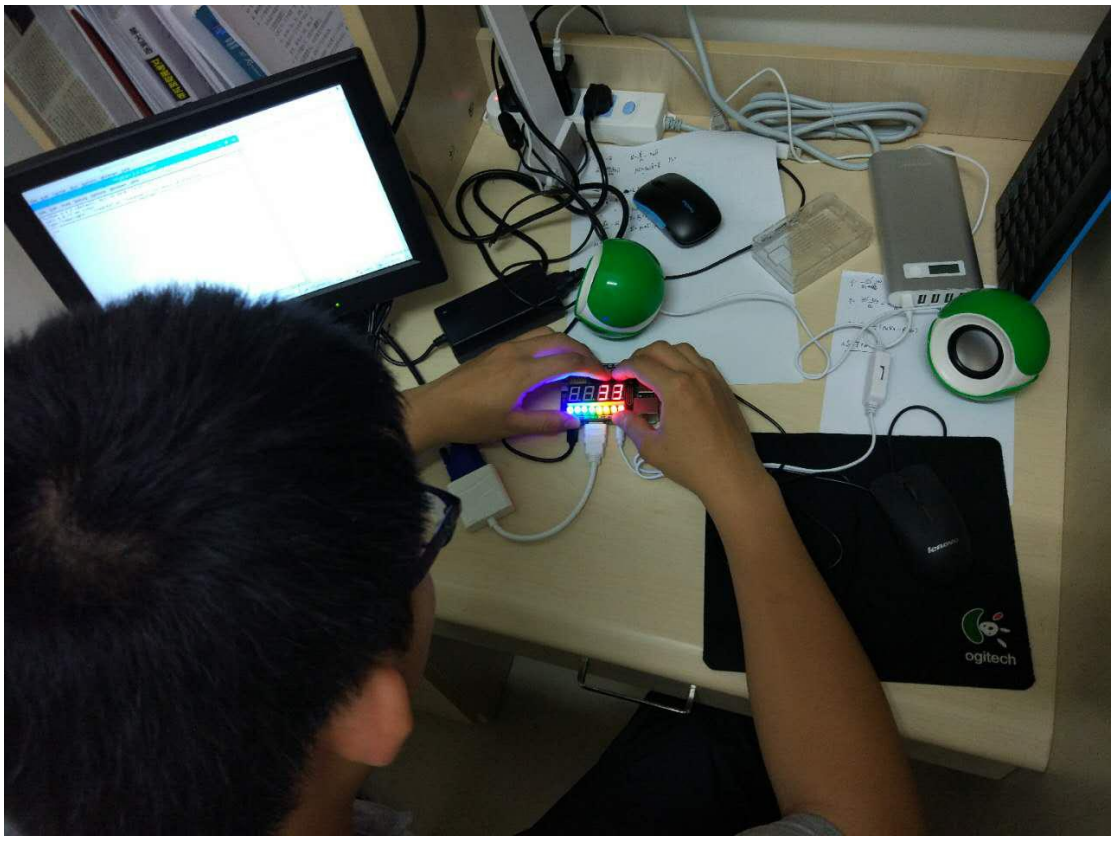
【更多的小游戏】

我们初步的预想是在树莓派上实现“2048”。利用数字屏上只能显示 4 个数字的限制，使用左键作为切换键，用小灯和液晶屏幕配合显示，当 1、2 两个 LED 小灯亮起时，表示当前显示为第一行的数字，同时表示在此时按下确认键表示向上滑动屏幕。2048 游戏中有 4 行显示，同时有四个滑动方向，用 8 个小灯刚好可以与其对应，而点击左键的效果就是改变当前显示的小灯位置，而右键的作用则是确认当前对应的操作，并按照游戏规则刷新各行的数字。其他的游戏规则和正常的 2048 一样。

五、小组分工介绍

祝同学主要负责实现树莓派 IO 功能部分的代码，傅同学主要负责逻辑部分的代码。

附图片：



附程序源码：

1.连携技（鼠标输入模式）

```
# 鼠标输入模式代码
from sakshat import SAKSHAT
from sakspins import SAKSPins as PINS
import time
import random
import RPi.GPIO as GPIO
import pygame
from pygame.locals import *

# GPIO 采用 BCM 编码
GPIO.setmode(GPIO.BCM)

# LED 灯对应的三个端口号
DS = 6
SHCP = 19
STCP = 13
```

```
# 初始化 pygame
pygame.init()

# 创建一个 pygame 窗口
screen = pygame.display.set_mode((1024, 768), 0, 32)

# LED 灯功能的实现：

# 定义 GPIO 的初始化函数
def init():
    GPIO.setup(DS, GPIO.OUT)
    GPIO.setup(SHCP, GPIO.OUT)
    GPIO.setup(STCP, GPIO.OUT)

    GPIO.output(DS, GPIO.LOW)
    GPIO.output(SHCP, GPIO.LOW)
    GPIO.output(STCP, GPIO.LOW)

# 通过 writebit 实现 LED 灯的状态控制
def writeBit(data):
    GPIO.output(DS, data)
    GPIO.output(SHCP, GPIO.LOW)
    GPIO.output(SHCP, GPIO.HIGH)

LEDlist = [0x00, 0x01, 0x03, 0x07, 0x0f, 0x1f, 0x3f, 0x7f, 0xff]

# 传入一个 16 进制的数最为 LED 灯的状态
def writeByte(data):
    for i in range(0, 8):
        # 转成 2 进制
        writeBit((data >> i) & 0x01)
    # 状态刷新信号
    GPIO.output(STCP, GPIO.LOW)
    GPIO.output(STCP, GPIO.HIGH)

# 调用 SAKS
SAKS = SAKSHAT()
presstimes = 0
ledposition = 0

def beforegame():  # 准备开始游戏，返回一个点击目标字符串
    global ok
    # 做一个 3 秒的倒计时
```

```
    for i in range(1, 5):
        time.sleep(1)
        t = 4 - i
        SAKS.digital_display.show('###' + str(t))
    # 随机产生一个由 1 或 2 组成的 4 位数字符串
    t = ''
    for i in range(4):
        t += str(random.randint(1, 2))
    SAKS.digital_display.show(t)
    if not ok:
        ok = True
    return t

init()
# 得分记作 score
score = 0
# 没有犯规记作 ok
ok = True
# 目标记作 target
target = beforegame()
# 时间未到记作 flag
flag = True

inittime = time.time()
# inittime 用作记录当前时间，用于计算两次点击时间差（仅在触控开关函数中使用）

def tact_event_handler(pin, status):  # 触控开关函数
    global __light_status, presstimes, score, target, ledposition, ok, inittime,
flag
    if pin == PINS.TACT_RIGHT and status == True:  # 如果按下的是右键：
        nexttime = time.time()  # 利用当前时间计算两次敲击时间差
        timecost = nexttime - inittime
        inittime = nexttime

        if timecost > 0.08 and flag:  # 如果时间没到而且时间差多于 0.08s
            if target[presstimes] == '2':  # 右键对应 2 说明按对了
                target = target.replace('2', '6', 1)  # 将按对的字符替换为 6
                SAKS.digital_display.show(target)
                presstimes += 1  # 正确点击次数+1

            else:  # 如果不是那就说明按错了
                SAKS.digital_display.show('2333')
                ok = False  # 按键错误则 ok 变为 False
                writeByte(0x00)  # 流水灯清空，等待 1.5 秒后显示分数
```

```
                time.sleep(1.5)
                SAKS.digital_display.show('%04d' % (score))
                time.sleep(2)
                score = 0  # 一切重置重新开始游戏
                presstimes = 0
                ledposition = 0
                target = beforegame()
                flag = True

    elif pin == PINS.TACT_LEFT and status == True:  # 按下左键的情况与按下右键的
情况刚好对称
        nexttime = time.time()
        timecost = nexttime - inittime
        inittime = nexttime
        if timecost > 0.08 and flag:
            if target[presstimes] == '1':
                target = target.replace('1', '6', 1)
                SAKS.digital_display.show(target)
                presstimes += 1
            else:
                SAKS.digital_display.show('2333')
                ok = False
                writeByte(0x00)
                time.sleep(1.5)
                SAKS.digital_display.show('%04d' % (score))
                time.sleep(2)
                score = 0
                presstimes = 0
                ledposition = 0
                target = beforegame()
                flag = True

    if presstimes == 4 and flag:  # 如果正确点击次数为4而且时间还没到
        score += 1  # 说明玩家正确完成一组游戏，得分+1
        presstimes = 0  # 部分重置，开始下一次游戏
        ledposition = 0
        ok = False
        time.sleep(0.5)
        SAKS.digital_display.show('####')
        writeByte(0x00)
        target = beforegame()
        flag = True
```

```
while True:  # 无限循环，主循环
    ledposition += 1
    writeByte(LEDlist[ledposition])
    SAKS.tact_event_handler = tact_event_handler

    if ledposition == 8:  # 如果这个时候发现时间到了
        writeByte(0xff)
        SAKS.digital_display.show('2333')  # 显示“2333”
        flag = False  # 推倒 flag
        time.sleep(1.5)

        SAKS.digital_display.show('%04d' % (score))  # 显示最终的得分
        time.sleep(1.5)

        score = 0  # 重置参数重新开始游戏
        presstimes = 0
        ledposition = 0
        ok = False
        writeByte(0x00)

        target = beforegame()
        for sdbg in pygame.event.get():  # 清空 event 列表
            shenmedoubuzuo = 0

        flag = True

SAKS.digital_display.show('####')  # 游戏结束，初始化树莓派扩展板
GPIO.cleanup()
```

2.别踩白块儿

```
import pygame
from pygame import *
import time
from sakshat import SAKSHAT
from sakspins import SAKSPins as PINS
import random
import RPi.GPIO as GPIO

#screen = pygame.display.set_mode((640, 480), 0, 32)

GPIO.setmode(GPIO.BCM)  #采用 BCM 编码模式
SAKS = SAKSHAT()        #调用 SAKSHAT 模块
DS = 6                  # 控制 LED 灯的三个端口的 BCM 编码
SHCP = 19
```

```
STCP = 13
speed = 0.36            # set speed

def init():             # 初始化端口状态
    GPIO.setup(DS, GPIO.OUT)
    GPIO.setup(SHCP, GPIO.OUT)
    GPIO.setup(STCP, GPIO.OUT)
    GPIO.output(DS, GPIO.LOW)
    GPIO.output(SHCP, GPIO.LOW)
    GPIO.output(STCP, GPIO.LOW)

# 实现 1 到 8 对应流水灯的状态，i 对应从左到右前 i 个 LED 灯亮
LEDlist = [0x00,0x01,0x03,0x07,0x0f,0x1f,0x3f,0x7f,0xff]
def writeBit(data):
    GPIO.output(DS, data)
    GPIO.output(SHCP, GPIO.LOW)
    GPIO.output(SHCP, GPIO.HIGH)

#写入 8 位 LED 的状态
def writeByte(data):
    for i in range (0, 8):
        writeBit((data >> i) & 0x01)
    #状态刷新信号
    GPIO.output(STCP, GPIO.LOW)
    GPIO.output(STCP, GPIO.HIGH)

# 燕园情简谱字符串，print 用来显示在数字屏上
yyq_print =
'''000300235000300235003560056100065612123200030023500030023500356005610006561212310001160040016500300322032013000116004001650030032201203200055300100553003003440560540006630020061300200322012031000000011600016500011600046500055305206120350001160001650001160004650005530520612003000100 0'''
# string 用来播放对应的音频文件
yyq_string =
'''000600561000600568006890089400021245456500060056100060056800689008940002124545640004490070049800600655065046000449007004980060065504506500011600400186006006770890870002260050094600500655045064000000044900049800044900079800011608502450680004490004980004490007980001160850245006000400 0'''

flag = []   # 列表，记录简谱字符串中的每个数字有没有被访问过，1 表示没被访问过，0 表示访问过
score = 0   # 得分，初始为 0

music_print = yyq_print     # 封装
```

```
music_string = yyq_string
music_list = list(music_string)

for i in range(len(music_print)):
    flag += [1]

def tact_event_handler(pin, status):        # 用来捕捉按键事件的函数

    global __light_status,cst,t0,flag,score,LEDlist
    # 如果按右键
    if pin == PINS.TACT_RIGHT :
        # 判断是否按对
        if int(music_print[cst-1])%2 == 0 and flag[cst-1] == 1 and
music_print[cst-1] != '0':
            score += 1 # 按对，得分加一
            flag[cst-1] = 0
            writeByte(LEDlist[score%9])

            pygame.mixer.music.load(sound[int(music_list[cst-1])])# 播放对应音
高的音乐文件
            pygame.mixer.music.play()

    elif pin == PINS.TACT_LEFT :    #如果按下右键，和按下左键时情况对称
        if int(music_print[cst-1])%2 == 1 and flag[cst-1] == 1:
            score += 1
            flag[cst-1] = 0
            writeByte(LEDlist[score%9])
            pygame.mixer.music.load(sound[int(music_list[cst-1])])
            pygame.mixer.music.play()

# 制作翻译简谱字符串的字典
sound = {}
for i in range(1,4):
    sound[i] = '3%s.ogg' % ('GAB'[i-1])
for i in range(4,10):
    sound[i] = '4%s.ogg' % ('CDEFGA'[i-4])
sound[0] = 0

for cst in range(len(music_string)):        # 依次读取简谱字符串中的内容
    if sound[int(music_list[cst])] == 0:        # 如果读到0，那么对应位置不显示
        SAKS.digital_display.show(music_print[cst-1:cst+3].replace('0','#')) #
切片显示，并将‘0’替换成‘#’
        time.sleep(speed)
```

```
    elif cst <= len(music_string) - 4:
        SAKS.digital_display.show(music_print[cst-1:cst+3].replace('0','#'))
        SAKS.tact_event_handler = tact_event_handler
        time.sleep(speed)
    else:
        SAKS.digital_display.show('1###')
        SAKS.tact_event_handler = tact_event_handler
        time.sleep(speed)

score = score  / (len(music_string) - music_string.count('0')) # 计算并展示最后
得分
score = int(score * 100)
score = str(score)
while len(score)<4:
    score = '#'+score
SAKS.digital_display.show(score)
time.sleep(3)

writeByte(0x00) # 结束游戏
```

“最强大佬”游戏开发实习报告

滕沅建 陶韵竹 王瑞敏

【摘要】 “最强大佬”作为一款益智小游戏，兼有放松娱乐，锻炼思维的益处。本实习报告将从创意来源、实现方案、相关代码三方面介绍这款益智小游戏的开发过程，以及介绍极具趣味性的游戏语音，在报告的最后简单阐述我们的后期设想和小组分工。

一、选题及创意介绍

在如今随时随地“膜大佬”大环境下，我们应景推出了这款名叫“最强大佬”的益智小游戏。我们希望利用树莓派瑞士军刀以及 pygame 实现游戏界面的搭建。大家经过了一天繁重的工作学习（特别是目前深受“级数”困扰的我们），在闲暇之余，玩玩我们开发的这款游戏，听着“魔性”的提示音，放松身心，又锻炼右脑，何乐而不为？

二、设计方案及实现方案

我们设想的游戏具有一个比较完整的框架，包括游戏模式、游戏难度、游戏中的“命”（失误次数）、计分算法等等。我们利用树莓派瑞士军刀扩展板以及 pygame 搭建游戏界面，推出了两款游戏，分别是“数字闪现”小游戏和“颜色辨别”小游戏。

下面介绍我们设计的这两个游戏的具体规则：

“数字闪现”是通过代码实现在数码管上根据难度级别显示 2-4 位的数字，不同的难度数字闪现时间也不尽相同，数字闪现结束后，玩家需要按照提示音要求（“请输入最大数”“请输入最小数”“请输入中位数”），尽可能快的输入正确数字，系统会根据玩家的正确率以及反应输入时间给出一个分数，每一轮游戏次数为 5，但要注意最多只能答错 2 次，游戏结束后将计算出总得分。

在“颜色辨别”游戏，我们利用 pygame 作出了图形界面，并实现了倒计时的功能，在这个游戏模式下，可以直接用键盘操作，实现人机交互。游戏开始后，屏幕会出现相应的颜色和干扰汉字（比如“紫”这个汉字它却是黄色的，而玩家应该快速排除“紫”字的干扰，输入黄色。），在规定的时间内，根据玩家答对的题目数目给出相应的得分。

特别要提到的是我们“绞尽脑汁”想出的能引起玩家兴趣的游戏提示音，以此来提高我们游戏的趣味性。下面列出部分：

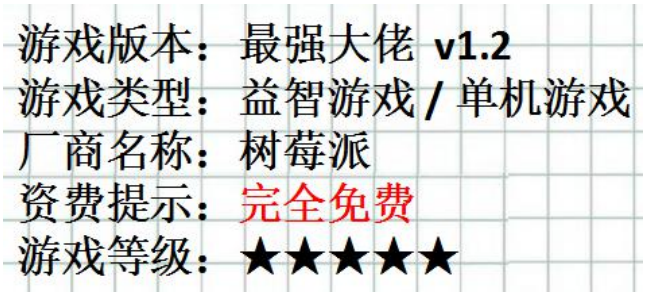
嘲讽：“我看你那脑子啊，就要退化成苔藓了。”

无奈：“我还能说什么呢，当然是选择原谅你啦！”

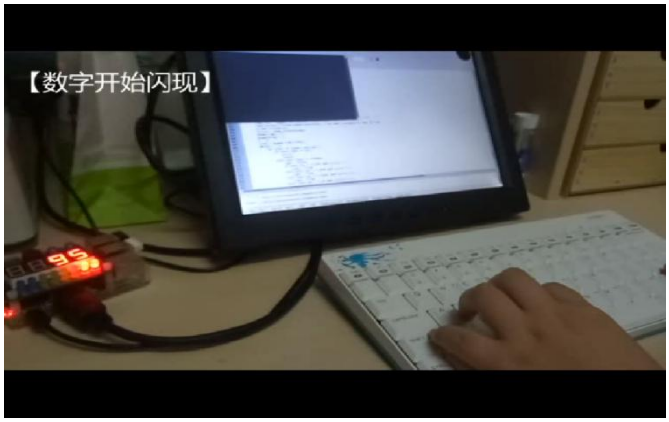
赞美：“你太棒了，不愧是我儿子。”

游戏最高荣誉：“大佬在上，请受本渣一拜！”……

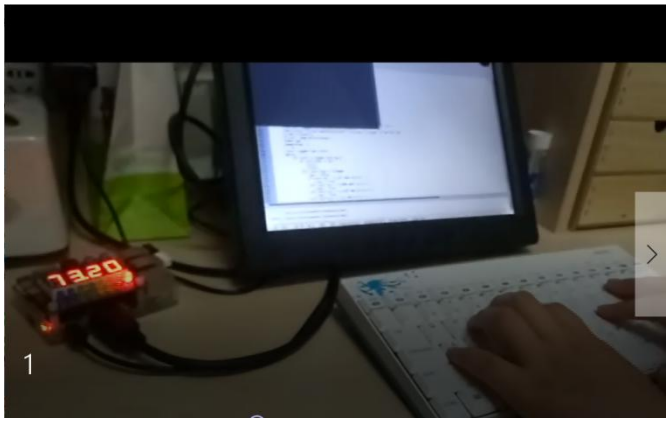
下面用图片展示的是游戏的相关信息以及在运行过程中的具体情境：



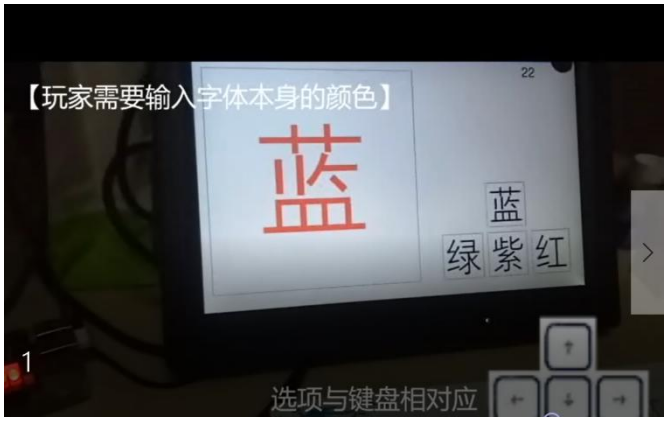
（图 1：游戏基本信息）



（图 2：数字闪现）



（图三：“数字闪现”得分显示）



（图四：“颜色辨别”游戏界面）

三、代码分析

鉴于代码有详细的注释，下面直接粘贴部分重要代码：

1、关于亮灯的代码

```
SAKS = SAKSHAT()
ds = 6
shcp = 19
stcp = 13
di = 25
clk = 5
# about lights
def GPIO_init():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(ds, GPIO.OUT)
    GPIO.setup(shcp, GPIO.OUT)
    GPIO.setup(stcp, GPIO.OUT)
    GPIO.setup(clk, GPIO.OUT)
    GPIO.setup(di, GPIO.OUT)
    GPIO.output(ds, GPIO.LOW)
    GPIO.output(shcp, GPIO.LOW)
    GPIO.output(stcp, GPIO.LOW)
    GPIO.output(clk, GPIO.LOW)
    GPIO.output(di, GPIO.LOW)
def writeBit(data):
    GPIO.output(ds, data)
    GPIO.output(shcp, GPIO.LOW)
    GPIO.output(shcp, GPIO.HIGH)
def writeByte(data):
    for i in range(0, 8):
        writeBit((data >> i) & 0x01)
    GPIO.output(stcp, GPIO.LOW)
    GPIO.output(stcp, GPIO.HIGH)
def light():
    GPIO_init()
    for j in range(2):
        for i in [0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80]:
            writeByte(i)
            time.sleep(0.1)
    GPIO.cleanup()
    return 0
```

2、第一个游戏显示数字和显示成绩的代码

```
def showgrade_numbers(correct, tryanswer, lasttime, difficulty):    # get the grade of number game
    if int(tryanswer) == correct:    # if    correct
        ## play: congratulations
        if lasttime < (2 + difficulty / 3):
            grade = 100
        elif lasttime < (4 + difficulty / 3):
            grade = 100 - lasttime * 6
        elif lasttime < (6 + difficulty / 3):
            grade = 100 - lasttime * 6
        elif lasttime > (8 + difficulty / 3):
            grade = 10
        else:
            grade = 100 - lasttime * 10
    else:    # if wrong
        grade = 0
    if grade == 0:
        s = "00.00"
    else:
        s = "%2.2f" % grade
    GPIO_init()
    SAKS.digital_display.show(s)    # show the grade
    return grade
def show_numbers(numlist, listsize, sleeptime, difficulty):
    for i in range(listsize):    # show the numbers
        if difficulty < 4:
            num = random.randint(10, 99)
        elif difficulty < 7:
            num = random.randint(100, 999)
        else:
            num = random.randint(1000, 9999)    # the numbers are from 10 to 9999
        numlist.append(num)
        if num / 10 < 1:
            s = "###" + str(num)
        elif num / 100 < 1:    # the number is double-digit
            s = "##" + str(num)
        elif num / 1000 < 1:    # the num is 3-digit
            s = "#" + str(num)
        else:    # the num is 4-digit
            s = str(num)
        GPIO_init()
```

```
            SAKS.digital_display.show(s)
            time.sleep(sleeptime)
            SAKS.digital_display.show("####")
            time.sleep(0.08)
        SAKS.digital_display.show("####")    # end of the show time
        return numlist
```

3、第二个游戏显示颜色和显示成绩的代码

```
pygame.init()
white = (255, 255, 255)
black = (0, 0, 0)
blue = (0, 0, 255)
red = (255, 0, 0)
green = (0, 250, 0)
yellow = (250, 250, 0)
purple = (200, 0, 200)    # the colors
# color game
def show_colors(screen):
        pygame.draw.rect(screen, black, rect1, 1)
        pygame.draw.rect(screen, black, rect2, 1)
        pygame.draw.rect(screen, black, rect3, 1)
        pygame.draw.rect(screen, black, rect4, 1)
        pygame.draw.rect(screen, black, rect5, 1)    # draw the rectangles
        colordict = {'黑': black, '蓝': blue, '红': red, '绿': green, '黄': yellow, '紫': purple}
        colorlist = list(colordict.keys())
        word = random.choice(colorlist)    # choose a random color
        word1 = word
        while word1 == word:    # choose a different color as the word1
                word1 = random.choice(colorlist)
        words = [word] * 4
        correct = random.randint(0, 3)    # choose a random position as the correct one
        for i in range(4):
                if i != correct:    # change the word which is not in the right position
                        while words[i] == word or words[i] in words[:i]
                                words[i] = random.choice(colorlist)
        screen.blit(font1.render(word1, True, colordict[word]), (190, 180))    # write the
big word
        screen.blit(font2.render(words[0], True, black), (870, 620))
        screen.blit(font2.render(words[1], True, black), (1050, 620))
        screen.blit(font2.render(words[2], True, black), (1230, 620))
        screen.blit(font2.render(words[3], True, black), (1050, 440))    # write the small
words
        return correct
```

```
def showgrade_color(right, wrong, screen):
    grade = 100
    if right + wrong == 0:
        grade = 0
    else:
        grade -= 25 - (right + wrong) * 2
        grade = grade * right / (right + wrong)
    if grade > 100:
        grade = 100
    screen.fill(white)
    screen.blit(font1.render('%.2f' % grade, True, black), (240, 220))    # show the grade
    pygame.display.update()
    if grade > 90:    # if the grade is great
        play_grade1()
    elif grade > 75:
        play_grade2()
    elif grade > 60:
        play_grade3()
    elif grade > 40:
        play_grade4()
    else:
        play_grade5()
    time.sleep(2)
    return grade
```

四、后续工作展望

在展示中，我们发现游戏独具特色的提示音极具吸引力，所以首先我们会引入更多的游戏语音，使得游戏更有趣味性。其次，对于“颜色辨别”游戏，我们可以在框内加入颜色背景提高难度，并利用背景使整个界面更丰富多彩。

除了这两款游戏，还有很多有趣的创意益智小游戏可以开发，比如立体积木的识别，“石头剪刀布高阶版”等，

五、小组分工合作

组长：滕沅建（编写代码）

组员：陶韵竹（创意来源、游戏语音来源、制作视频）

王瑞敏（录制视频、制作展示 ppt、撰写实习报告）

后记：我们这个小组成员是一个寝室的室友，在宿舍讨论开发这个游戏时，从宿舍里会时常传出我们“魔性”的笑声。其实在此之前，我们三个对树莓派甚至编程都不是特别精通，但是在组长滕沅建的带领下，不断探索，不断学习，终于实现了我们开发小游戏的设想，当看到数码管和屏幕出现我们预想中的游戏界面是，三个人都很有成就感。如此欢乐的合作气氛，以后想来也是会骄傲地扬起嘴角的事。

树莓派实习报告——树莓派定制美图秀秀

实习者：叶勃 曹寒冰 金恬

【摘要】实现对图片中人脸的识别保存，并通过图形界面（GUI）对识别出来的人脸进行元素饰品添加。
【关键词】树莓派 人脸识别 图形界面

一、选题及创意介绍

一开始，我们的设想是，用树莓派做一个门禁系统，即通过安装在门上的摄像头获取图片，通过树莓派进行人脸识别来控制门禁。由于采购的摄像头有精度不高等问题，这个方案最后被弃用。

最终我们组决定通过树莓派,实现美图秀秀中的一部分功能。也即对图片进行铅笔画操作(滤波)，以及进行人脸识别，再在识别出来的人脸上加胡须、耳朵等饰品。基于树莓派的美图秀秀构想就这样被确定下来。

二、设计方案

1.完成人脸识别的程序，识别出一张照片中的所有人脸，加框并截图输出。
2.选择合适的素材，对人脸进行装饰，参照美图秀秀的“饰品”功能。
3.另写一些函数来实现对图片的其它处理方法，比如通过滤波实现铅笔画风格。
4.完成图形用户界面（GUI），提高程序的用户友好程度。
5.通过瑞士军刀扩展板增加可视化效果，比如识别出 5 张人脸之后，扩展板上就会显示 5 并且鸣叫 5 下，亮 5 盏灯。（如果识别人脸大于 8，那也只是叫 8 下亮 8 盏灯）

三、实现方案及代码分析

以下是识别人脸的代码以及注释：

```
def detect_object(image):# 调用库数据判别五官来查找人脸
    grayscale = cv.CreateImage((image.width, image.height), 8, 1)
    cv.CvtColor(image, grayscale, cv.CV_BGR2GRAY)
    cascade                                                    =
cv.Load("/usr/share/opencv/haarcascades/haarcascade_frontalface_alt_tree.xml")
    rect = cv.HaarDetectObjects(grayscale, cascade, cv.CreateMemStorage(), 1.1, 2,
        cv.CV_HAAR_DO_CANNY_PRUNING, (20,20))
    result = []
    for r in rect: # 向 result 里添加找到的人脸的左上右下点坐标
        result.append((r[0][0], r[0][1], r[0][0]+r[0][2], r[0][1]+r[0][3]))
    return result

def process(infile): # 调用以上函数保存查找到的人脸
    image = cv.LoadImage(infile);
```

```
    if image:
        faces = detect_object(image) # 调用探测脸的函数
    im = Image.open(infile)
    path = os.path.abspath(infile)
    save_path = os.path.splitext(path)[0]+"_face"
    try:
        os.mkdir(save_path) #储存路径
    except:
        pass
    count = 0
    if faces:
        draw = ImageDraw.Draw(im)
        for f in faces: # 循环将探测到并截图的人脸顺序保存
            count += 1
            draw.rectangle(f, outline=(255, 0, 0))
            a = im.crop(f)
            file_name = os.path.join(save_path,str(count)+".jpg")
            a.save(file_name)

        drow_save_path = os.path.join(save_path,"out.jpg")
        im.save(drow_save_path, "JPEG", quality=80)
        im.show()
    else:
        print("Error: cannot detect faces on %s" % infile)
    print(count) # 输出人脸个数
```

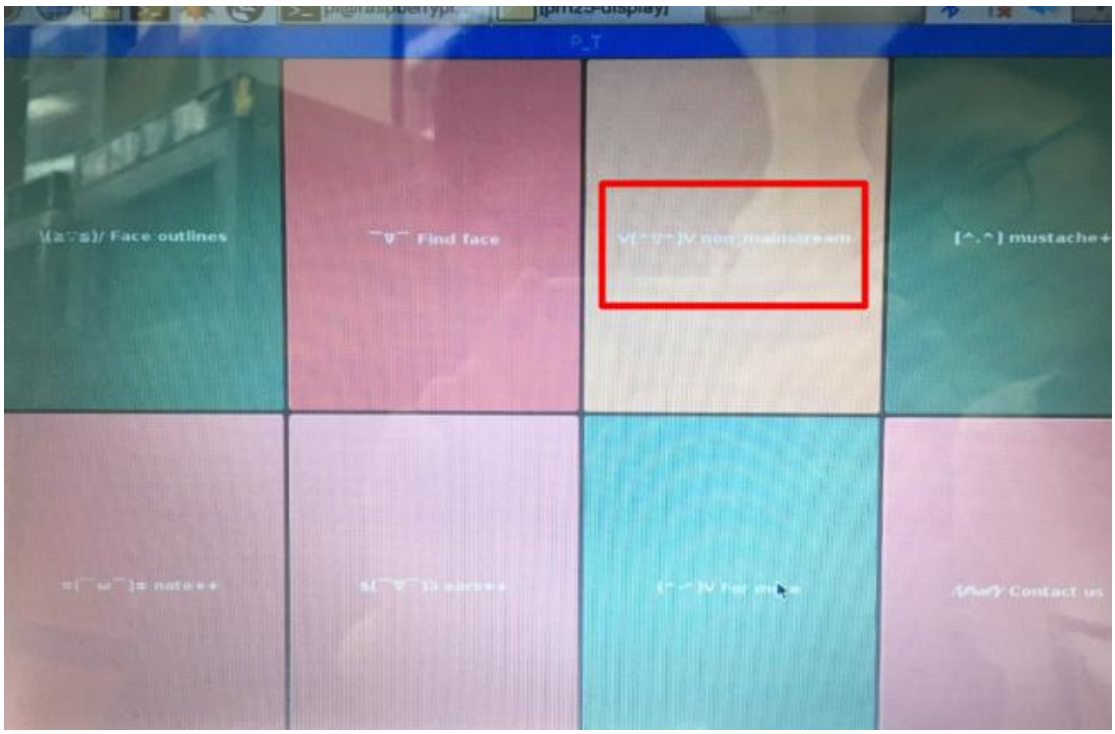
以下是加胡须的代码以及详细注释：

```
def mustache(evt):
    # 加载底图
    base_img = Image.open('/home/pi/Desktop/face/2.1.jpeg')
    # 转化为 RGBA
    base_img = base_img.convert('RGBA')
    # 可以查看图片的 size 和 mode，常见 mode 有 RGB 和 RGBA，RGBA 比 RGB 多了 Alpha 透
明度
    # print base_img.size,base_img.mode
    # 加载需要 P 上去的图片
    tmp_img = Image.open('/home/pi/Desktop/face/shipin/mtsc11572.png')
    # 转化为 RGBA
    tmp_img = tmp_img.convert('RGBA')
    # 可以查看图片的 size 和 mode，常见 mode 有 RGB 和 RGBA，RGBA 比 RGB 多了 Alpha 透
明度
    # print tmp_img.size,tmp_img.mode
    # 这里可以选择一块区域或者整张图片
    # region = tmp_img.crop((0,0,304,546)) #选择一块区域
```

```
    # 或者使用整张图片
    region = tmp_img
    # 使用 paste(region, box) 方法将图片粘贴到另一种图片上去.
    # 注意，region 的大小必须和 box 的大小完全匹配。但是两张图片的 mode 可以不同，合并的时候回自动转化。如果需要保留透明度，则使用 RGMA mode
    # 提前将图片进行缩放，以适应 box 区域大小
    # region = region.resize((box[2] - box[0], box[3] - box[1]),Image.ANTIALIAS)
    # region.show()
    region = region.resize((int(base_img.size[0] / 3), int(base_img.size[1] / 5)))
    box = ((int(base_img.size[0]/2)-int(tmp_img.size[0]/3.4)),(int(base_img.size[1]*0.93)-int(tmp_img.size[1]))) # 底图上需要 P 掉的区域的左上角坐标
    # 分离透明通道
    r, g, b, a = region.split()
    base_img.paste(region, box, mask = a)
    base_img.show() # 查看合成的图片
    base_img.save('/home/pi/Desktop/face/4.1.jpeg','jpeg',quality = 95) # 保存图片
```

以下是 GUI 实现中一个按键的代码及注释以及实际效果：

```
bt_non_mainstream = wx.Button(panel,-1,label='V(^ ▽ ^)V non_mainstream',size=(45,30))
bt_non_mainstream.SetForegroundColour((255,255,255))
bt_non_mainstream.SetBackgroundColour((249,205,173)) #设置背景色
bt_non_mainstream.SetFont(wx.Font(10, wx.SWISS, wx.NORMAL, wx.BOLD, False)) # 设置文字属性
bt_non_mainstream.Bind(wx.EVT_BUTTON,non_mainstream) #添加 CUTE++按钮事件绑定
```



以下是军刀拓展版的注释：

‘’’

识别出 num 张人脸之后，扩展板上就会显示 num 并且鸣叫 num 下，亮 num 盏灯。持续显示 10 秒后，鸣叫 2 秒、显示器清零、LED 灯熄灭。如果识别人脸大于 8，叫 8 下亮 8 盏灯，显示 num。

‘’’

四、后续工作展望

人脸识别，或者说面部识别在当前的政府、军事、经济、生活、社会保障、电子支付、娱乐、安全防务等等领域都发挥着重大的作用，而更高精度、更加快速、更加廉价的面部识别技术也是我们现如今科技发展的一个方向。

对于我们小组，我们可以增加人脸识别功能的应用性。比如说建立一个数据库，那么识别出的人脸就可以和数据库中的人脸进行比较，获得更有意义的识别信息，像是做成一个门禁系统这样。

另外，目前我们的程序中是基于框架的人脸识别，就是通过识别眼睛、耳朵等部位来确定人脸的大致位置。

那么在识别侧脸，以及面部不清晰的时候，识别精度就会大大下降了。可以通过细化程序，做到更精准的确定脸元素——眼睛、耳朵、鼻子、嘴巴的位置，实现更精确的效果添加。

人脸识别算法有很多种，在我们的知识水平更进一步之后就有可能通过机器学习等等方法来实现人脸识别，增加人脸识别的精度。此外，我们还可以加入更多的变化元素，为使用者提供更多的饰品选择；还可以实现更友好的图形界面，并使用端口，实现更方便友好的图片调用方式，提高使用者的使用体验。

五、小组成员及分工

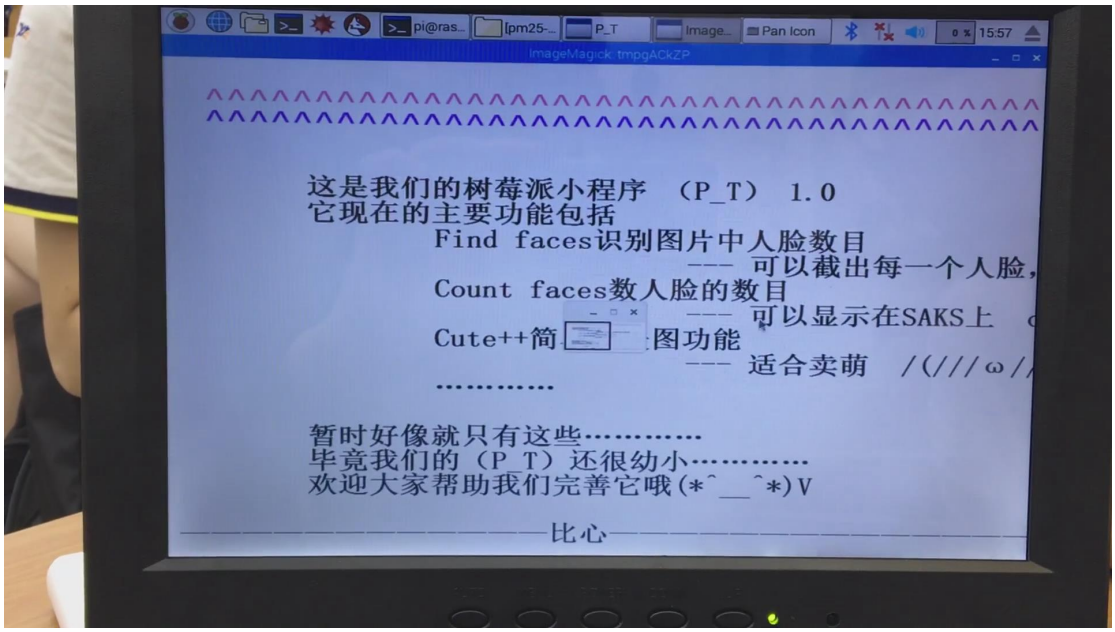
组长：叶勃 组员：曹寒冰、金恬
叶勃：组织小组讨论、进行分工，完成“饰品”功能，进行展示视频的后期处理。
金恬：完成在瑞士军刀扩展板上的功能实现，制作 PPT。
曹寒冰：完成人脸识别功能实现，完成 GUI，代码调用整合。

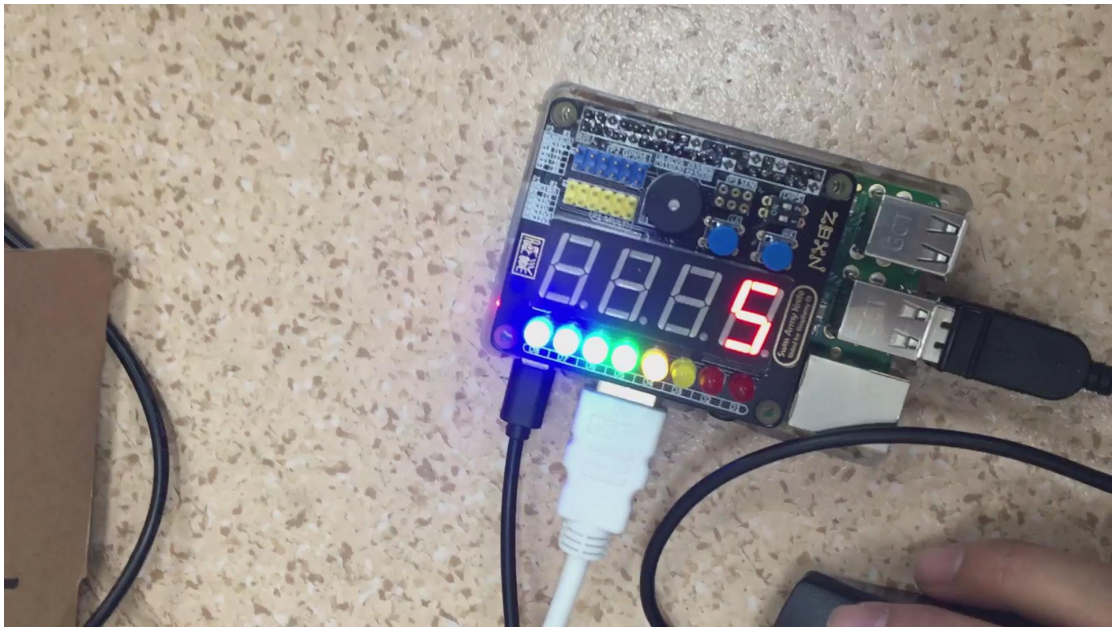
六、感受：

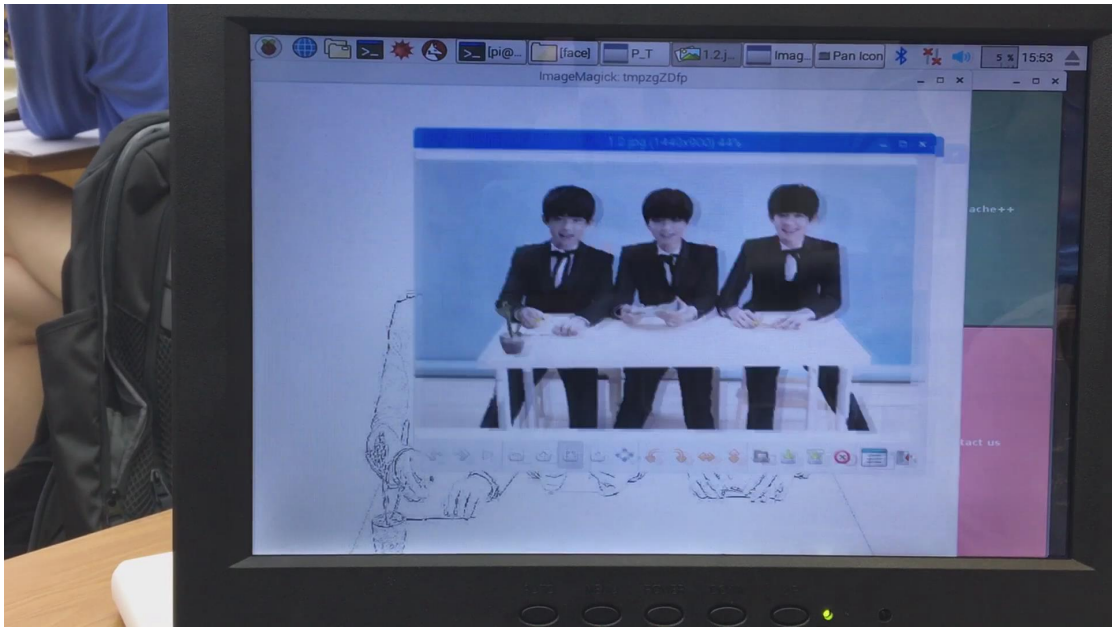
我们小组三人齐心合作，从零开始学习树莓派，在经历各种克服困难的同时感受到学习的乐趣。在这个过程中，我们成功实现了很多曾经觉得特别高大上的功能，从实战中感受到编程的力量，大大激发了进一步深入学习的热情和信心。

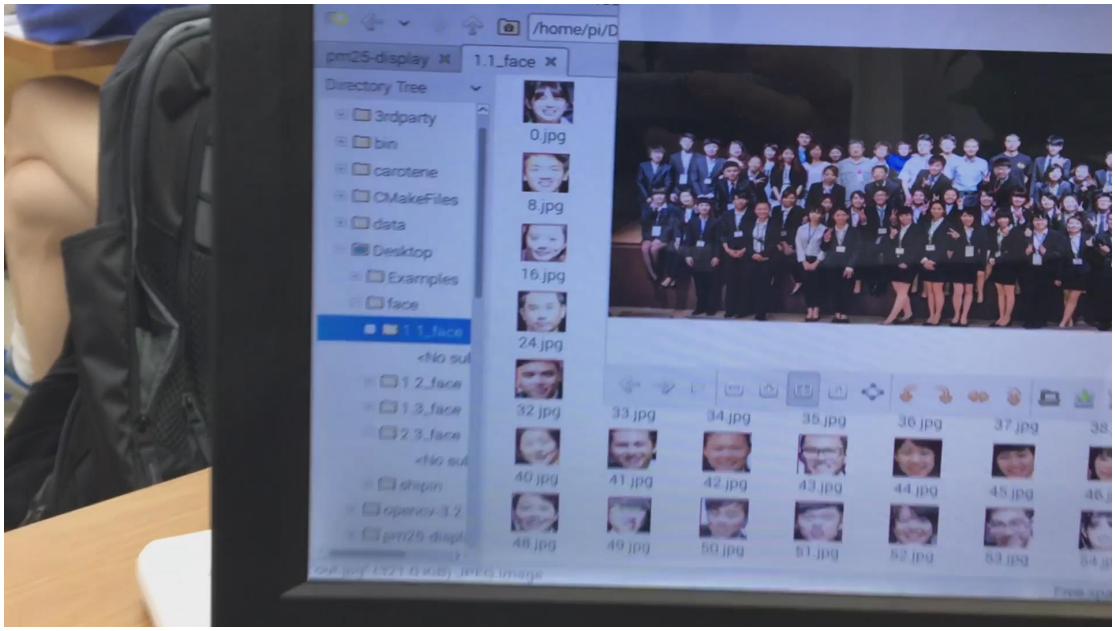
总而言之，树莓派实习项目使我们获益匪浅！感谢老师提供这样的实习机会，让我们运用所学知识，实现现实功能，充分感受到编程的乐趣。

附其它图片：


这是我们的树莓派小程序 （P_T） 1.0
它现在的主要功能包括
Find faces识别图片中人脸数目
--- 可以截出每一个人脸，
Count faces数人脸的数目
--- 可以显示在SAKS上
Cute++简 图功能
--- 适合卖萌 /(///ω//
.....
暂时好像就只有这些.....
毕竟我们的（P_T）还很幼小.....
欢迎大家帮助我们完善它哦(*^__^*)V
比心




ImageMagick: tmpzgZDfp


/home/pi/D
pm25-display
1.1_face
Directory Tree
3rdparty
bin
carotene
CMakeFiles
data
Desktop
Examples
face
0.jpg
8.jpg
16.jpg
24.jpg
32.jpg
33.jpg
34.jpg
35.jpg
36.jpg
37.jpg
40.jpg
41.jpg
42.jpg
43.jpg
44.jpg
45.jpg
48.jpg
49.jpg
50.jpg
51.jpg
52.jpg
53.jpg


PEKING UNIVERSITY
1898