

浙江工业大学



微机接口技术 课程设计报告

实验名称 车宝——一款智能车载语音控制助手

姓 名

贡 献 度

学 号

教 师

日 期 2021 年 1 月

一、设计内容与要求

1.1 设计内容

本次课程设计基于树莓派 4B 实现了车宝——一款智能车载语音控制助手，以满足行车过程中司机对于一些查询功能的需求，并且通过全语音的方式进行交互，进一步提升了便捷性和安全性。目前市面上多数方便的车载控制设备大多为位于正副驾驶位中间的触摸屏，对于一个车载设备来说，虽然触摸的方式已经极大程度上提升了驾驶员操作的便捷性，但仍然需要驾驶员腾出一只手才能进行操作。在这样的情况下，驾驶员要么为了安全而仅能在停车时进行操作，要么不安全地边单手开车边单手操作，造成了追求便利而反而舍弃了安全性的舍本逐末的操作，得不偿失。因此，我们小组成员希望通过树莓派 4B 这款拥有强力性能的芯片的深入研究，设计并实现一款通过语音方式进行人机交互的车载控制设备，从而在兼顾安全性和便捷性的情况下，为车载设备控制提供一种新思路。

在本次课程设计中诞生的车宝所具有的功能大体如下：系统上电，各设备连通但无法运行，需要通过指纹匹配后方可使用其他功能（指纹解锁模拟车主身份识别后开门）；通过语音唤醒词“snowboy”唤醒语音控制系统，唤醒之后可以继续通过语音的方式进行诸如车内环境温度检测、当前时间查询、中译英、指定城市的天气查询、模拟天色按下自动开启前照灯、指纹录入、指纹更新以及闲聊的功能，这部分功能无论在开车前还是开车过程中均可使用；通过语音交互告诉车宝“准备开车”的信息之后，车宝便会通过监测指定 GPIO 电平，模拟安全带系好的功能，若未系好则会一致通过语音进行提示并无法开车；最后，为了进一步提升驾驶的安全性，通过官方摄像头基于 opencv 和 dlib 库实现了对驾驶员的疲劳检测功能，并在满足一定疲劳条件后不断提醒驾驶员“注意自己的疲劳状态”，必要时升级提示的内容为“请立即停车休息”。整体功能较多且完整并充分；功能间的转换与衔接行云流水，十分流畅；各功能实现采用了多线程编程技术，使得一些警报提示信息和语音功能选择和并行地执行，也增强了对现实场景使用的考虑，故具有较强的实际意义！

1.2 设计要求

- (1) 设计的丰富性，即需要多个模块与芯片间的联动；
- (2) 设计的实用性，即需要设计内容具有较强的实际意义；
- (3) 设计的可用性，即所设计的目标功能是可以被实现的，不能纸上谈兵；
- (4) 设计的复杂性，即模块较丰富，功能较完善，且实现有一定的技术性与难度；
- (5) 提交正式实验报告、视频和源代码。

本次课程设计使用了树莓派 4B 芯片，合理运用了其 GPIO 引脚资源连接 DHT11 温湿度传感器、光敏电簇、LED 灯、模拟安全带、指纹识别 ATK-AS608 模块、摄像头视觉模块、麦克风及音箱等模块，预期能够成功实现一个完整的车载语音控制助手的所有功能，同时具有较强的现实意义！

1.3 设计目的

我们小组成员希望通过树莓派 4B 这款拥有强力性能的芯片的深入研究，外接合理且适量的芯片模块，设计并实现一款通过语音方式进行人机交互的车载控制设备，从而在兼顾安全性和便捷性的情况下，一改现有车载设备清一色触屏控制的现状，为车载设备控制系统的设计与实现提供一种新思路。

二、设计原理与硬件连线

2.1 设计原理

2.1.1 树莓派 4B 简介

树莓派由注册于英国的慈善组织“Raspberry Pi 基金会”开发，Eben · Upton/埃·厄普顿为项目带头人。2012 年 3 月，英国剑桥大学埃本·阿普顿（Eben Epton）正式发售世界上最小的台式机，又称卡片式电脑，外形只有信用卡大小，却具有电脑的所有基本功能，这就是 Raspberry Pi 电脑板，中文译名“树莓派”。

自问世以来，受众多计算机发烧友和创客的追捧，曾经一“派”难求。别看其外表“娇小”，内“心”却很强大，视频、音频等功能通通皆有，可谓是“麻雀虽小，五脏俱全”。自从树莓派问世以来，经历了 A 型、A+型、B 型、B+型、2B 型、3B 型、3B+型、4B 型等型号的演进。

树莓派 4B 于 2019 年推出，本次课程设计开发中由于涉及部分需要 AI 支持的图像处理技术（实时、高精度的人脸和物体识别），因此选择使用最新的树莓派 4B。树莓派 4B 着重对性能进行了优化以支持更多涉及到 AI 的开发，4B 采用了四核 64 位的 ARM(RISC)，较前代提升 3 倍性能。同时在内存方面可以选择 1G, 2G, 4G 版本（该版本可以运行 Win10）。接口方面有较大改动的地方如下：

- USB3.0×2, USB2.0×2
- 同时支持 2 台 4K 输出（mini-HDMI×2）
- 使用 Type-c 接口进行供电

接下来以课程设计中购买的树莓派为例，图中框出了树莓派的各个功能部件：

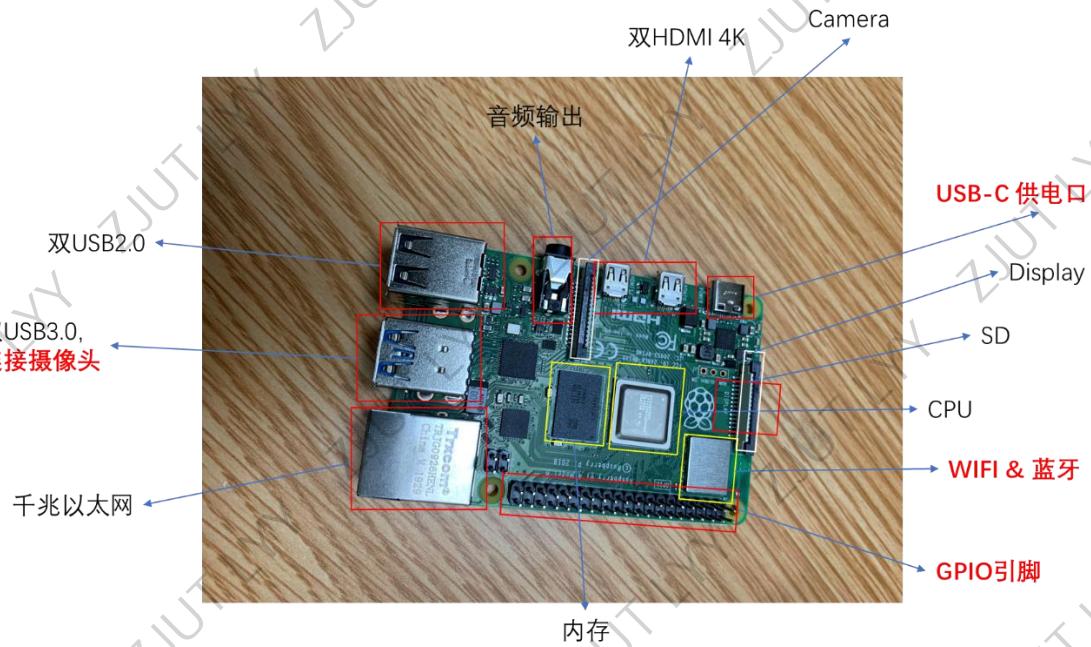


图 1 树莓派 4B 各功能模块

图中红色框为连接外设的各个端口，包括千兆以太网，双 **USB3.0**，双 **USB2.0**，音频输出，双 **mini-HDMI**，Type-C 供电，**GPIO 引脚**以及 **SD 卡插槽**（加粗的为本次实验所使用的）。图中黄色框为分别为 CPU，内存以及 WIFI，蓝牙模块。此外还需要指出的是本次实验中所使用的高分辨率摄像头通过 USB 端口与树莓派进行连接，而非树莓派上的 Camera 端口。

树莓派接口逻辑图如下图 2 所示：



图 2 树莓派接口逻辑图

2.1.2 指纹模块原理

指纹识别，即对人手指表面纹路进行识别的过程，在这个过程中，通常并不对比整个指纹，

而是提取上述细节特征点的类型及位置等进行比对，如下图 3 所示：



图 3 指纹识别特征点信息

指纹模块的种类众多，各有特点。半导体指纹模块：无论是电容式或是电感式，其原理类似，在一块集成有成千上万半导体器件的“平板”上，手指贴在其上与其构成了电容(电感)的另一面，由于手指平面凸凹不平，凸点处和凹点处接触平板的实际距离大小就不一样，形成的电容/电感数值也就不一样，设备根据这个原理将采集到的不同的数值汇总，也就完成了指纹的采集。射频指纹模块：利用生物射频指纹识别技术，通过传感器本身发射出微量射频信号，穿透手指的表皮层去控测里层的纹路，来获得佳的指纹图像。防伪指纹能力强，射频识别原理只对人的真皮皮肤有反应，从根本上杜绝了人造指纹的问题。

本次课程设计中所使用的 AS608 属于光学指纹模块，相较于其他类型指纹模块，其具有以下几点突出的优点：

- 1) 抗静电能力强、系统稳定性较好、使用寿命长；
- 2) 灵敏度特别的高；
- 3) 能提供高分辨率的指纹图像（可以达到 500 dpi）。

其工作原理为：利用光的折摄和反射原理，将手指放在光学镜片上，手指在内置光源照射下，光从底部射向三棱镜，并经棱镜射出，射出的光线在手指表面指纹凹凸不平的线纹上折射的角度及反射回去的光线明暗就会不一样。用棱镜将其投射在电荷耦合器件上 CMOS 或者 CCD 上，进而形成脊线(指纹图像中具有一定宽度和走向的纹线)呈黑色、谷线(纹线之间的凹陷部分)呈白色的数字化的、可被指纹设备算法处理的多灰度指纹图像。如下图 4 中 a) 和 b) 所示：

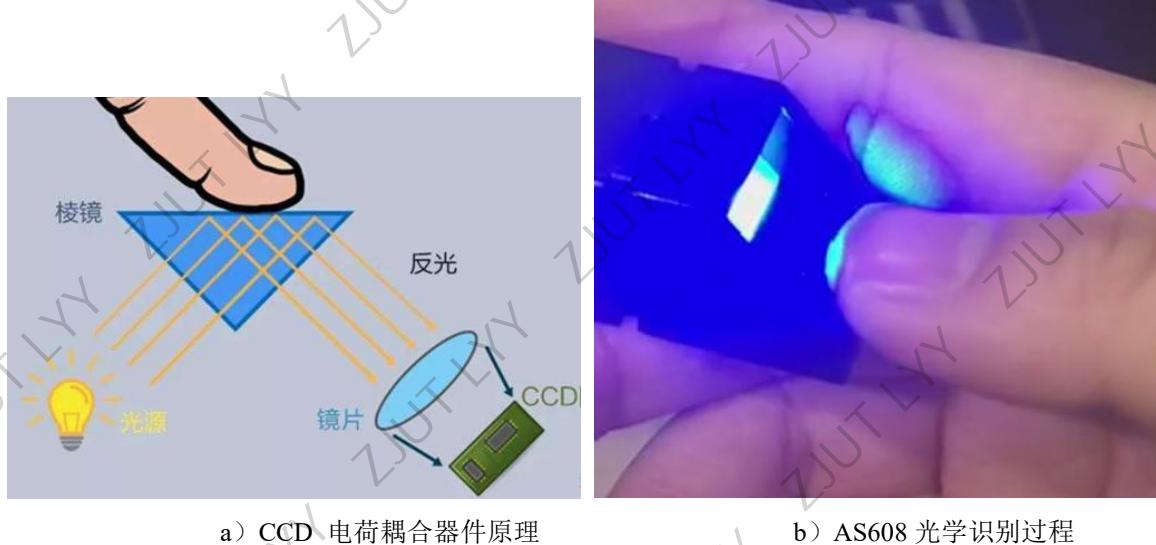


图 4 AS608 指纹识别原理

其优点主要表现为经历长期实用检验、系统稳定性较好、成本亦较低、能提供分辨力为 500 dpi(dot per inch)的图像。特别是能实现较大区域的指纹图像采集，有效克服大面积半导体指纹传感器价格昂贵缺点。该传感器局限性主要体现于潜在指印方面，不但会降低指纹图像的质量，严重时，还可能导致两个指印重叠，显然难以满足实际应用需要。此外，台板涂层及 CCD 阵列会随时间推移产生损耗，可能导致采集的指纹图像质量下降。

2. 1. 3 疲劳检测原理

2.1.3.1 人脸关键点定位技术

在研究前人探索的经验后，检测人脸位置信息的方法是先将输入的图片转化为统一格式 $20*20$ ，再根据先前标定的人脸位置，提取出上万个 haar 特征，如下图 5 所示：

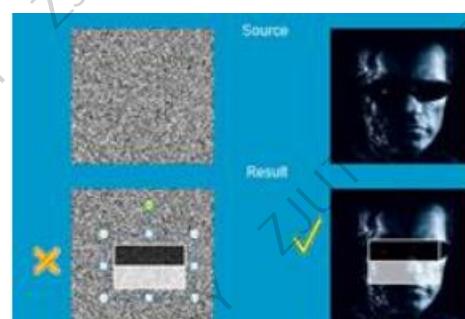


图 5 提取 haar 特征

该特征在人脸与背景上归一化后显示是不同的，通过训练了上千上万张照片后，可以得到

使用中的模型。实质上也是一个分类器，不过特征数量十分的多。将输入的图片灰度处理后，缩放成 20*20 的大小供给检测器使用，就可以框出图中所有的人脸的位置。过程如下图 6 所示：

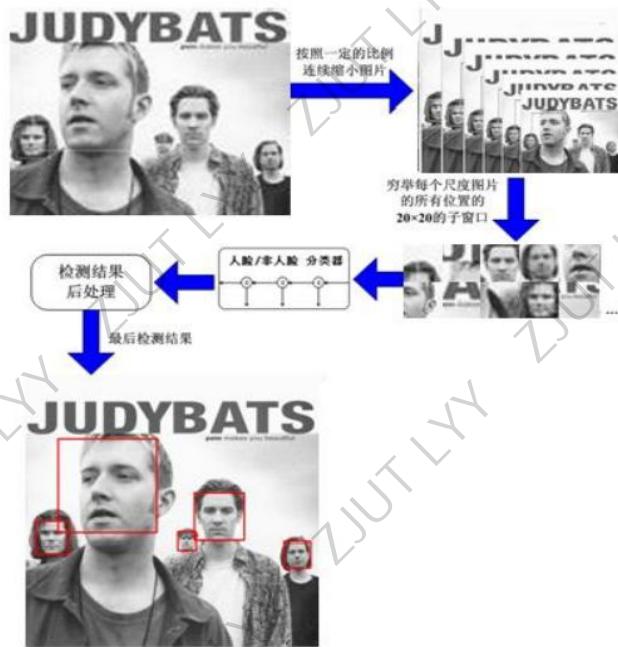


图 6 人脸位置识别过程

获得了相应的人脸位置后，可以进行接下来的人脸配准（人脸关键点定位）操作，在模型的建立过程中主要采用了随机森林和总体线性回归两个方法，能够达到每秒检测 3000 张图片。方法是在图片中随机 68 个点，基于第一次随机生成的点，通过随机点周围的两点的像素差值（LBF），标记特征。计算当前的点和人工标记点的差值。训练出一个增量函数，当使用这个模型的时候就可以先通过随机出的点，加上训练出的这个增量函数的值来得到关键点的位置。当然一次增加是不够的，通过多次重复上述步骤。才可以得到较为准确的关键点位置。计算这个增量函数是这个环节最关键的点。过程如下图 7 所示：

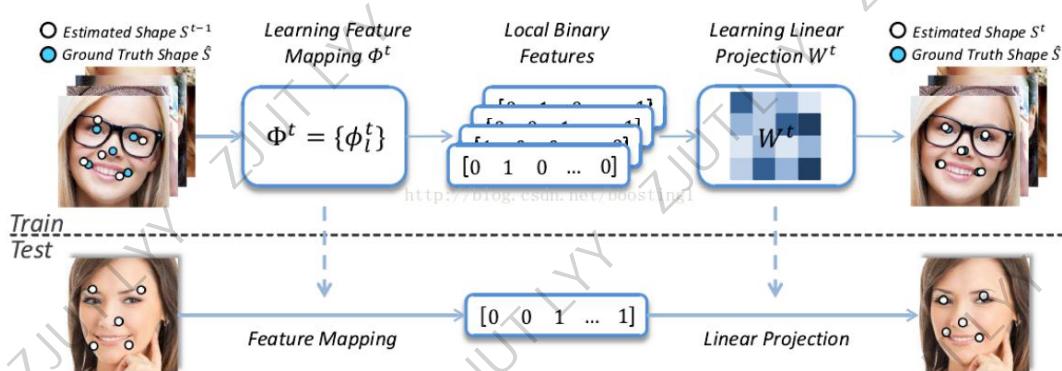


图 7 人脸配准操作流程

2.1.3.2 较疲劳眨眼检测原理

我们对较疲劳状态的眨眼动作检测是通过计算眼睛长宽比（Eye Aspect Ratio, EAR）值来完成的，在 68 点中，左眼关键点为 43~48，如下图 8 所示：

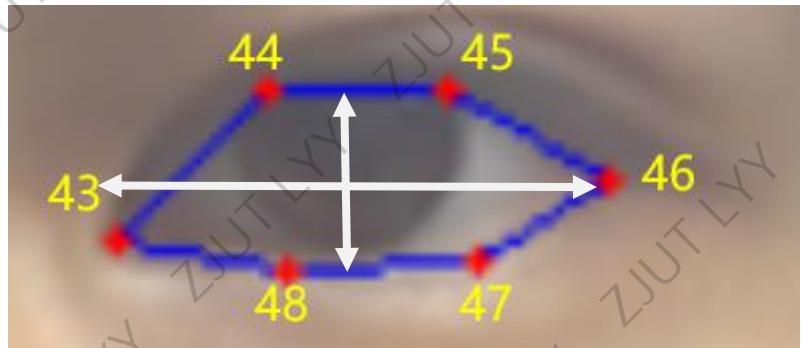


图 8 左眼关键点图

当人眼睁开时，因为在视频中人眼睛位置会有微小变化，所以 EAR 在某个值上下波动，但当人眼睁开时，是不会低于某个阈值的；当人眼闭合时，EAR 值会迅速下降，并在理论上会接近于零。但由于我们所用的 68 点人脸关键点检测器还没有这么精确，所以我们认为当 EAR 低于某个阈值时，眼睛就处于近似闭合状态；但平时不疲惫时，眨眼的速度较快，而疲惫时，眨眼的速度会变慢。查阅文献，我们设置若连续 3 帧两双眼睛长宽比平均值小于 0.2，则认为是疲劳眨眼，根据左右眼特征点的 EAR 值计算公式如下：

$$EAR_{left} = \left\| \frac{P44_y + P45_y - P48_y - P49_y}{P46_x - P43_x} \right\|$$
$$EAR_{right} = \left\| \frac{P38_y + P39_y - P41_y - P42_y}{P40_x - P37_x} \right\|$$
$$EAR_{avg} = \frac{EAR_{left} + EAR_{right}}{2}$$

总结一下即用 EAR 值衡量眼睛的开闭，小于 0.2 则近似认为进入闭眼状态，计算最大进入闭眼状态的帧数，若多于 3 帧，则认定为疲劳眨眼，相应次数加一。

2.1.3.3 打哈欠检测原理

打哈欠检测与较疲劳眨眼检测方式以及选取的计算公式类似。

我们对打哈欠的动作检测也是通过计算嘴巴长宽比（Mouth Aspect Ratio, MAR）值来完成的，在 68 点中，嘴巴相关关键点为 49~68，如下图 9 所示：

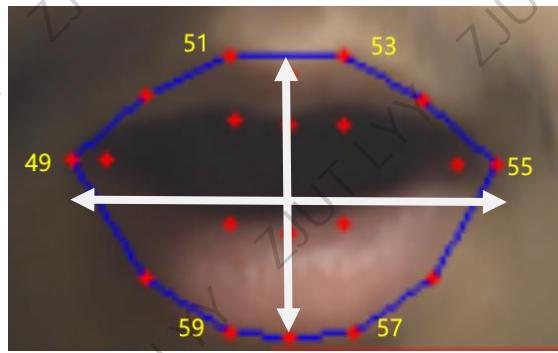


图 9 嘴巴关键点图

当嘴巴闭合时，因为在视频中人嘴巴位置会有微小变化，所以 MAR 在某个值上下波动，但当人嘴闭合时，是不会高于某个阈值的；当嘴巴张开时，MAR 迅速上升。所以我们认为当 MAR 高于某个阈值时，嘴巴处于近似张开状态。查阅文献，我们设置若连续 6 帧嘴巴长宽比大于 0.5，则认为是打哈欠状态，根据嘴巴的 MAR 值计算公式如下：

$$MAR = \left\| \frac{P51_y + P53_y - P57_y - P59_y}{P55_x - P49_x} \right\|$$

总结一下即用 MAR 值衡量嘴巴的开闭程度，大于 0.5 则近似认为处于打哈欠状态，计算最大进入打哈欠状态的帧数，若多于 6 帧，则认定为打了一次哈欠，相应次数加一。

2.1.3.4 瞌睡点头检测原理

对视频中驾驶员瞌睡点头动作的检测，可以通过一种比较经典的 Head Pose Estimate 算法来完成，它一般经过四个步骤：①2D 人脸关键点检测；②3D 人脸模型匹配；③求解 3D 点和对应 2D 点的转换关系；④根据旋转矩阵求解欧拉角。而所解出的欧拉角，即可反映驾驶员人脸正面方向与垂直于屏幕向外这个轴之间的夹角大小，通过这个角度的大小可以反映人点头的幅度，当连续一段时间人点头幅度大于某一阈值，即可认为人处于瞌睡点头状态，相应计数器加一。

第一步的 2D 人脸关键点检测和第二步的 3D 人脸模型匹配已经在之前调用的人脸关键点识别器中给出，调用即可。一个物体相对于相机的姿态可以使用旋转矩阵和平移矩阵来表示，其中平移矩阵为物体相对于相机的空间位置关系矩阵，用 T 表示；旋转矩阵为物体相对于相机的空间姿态关系矩阵，用 R 表示。

若为理想不考虑径向和切向畸变的情况，那么从世界坐标系中 3D 点投影到一个理想相机中 2D 像点的关系如下所示：

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

其中 r_{ij} 是旋转矩阵中元素, t_i 是平移矩阵中元素, (u, v) 是 2D 像素点坐标, f_x, f_y 是相机在 x, y 轴的比例尺, c_x, c_y 是相机中心点坐标, s 是未知的尺度系数 (因为对任意图片物方深度是不可知的, 点可能在射线的任何位置), (X, Y, Z) 是物体在 3D 世界坐标系中的坐标。

世界坐标系、像素坐标系和相机坐标系三者关系如下图 10 所示:

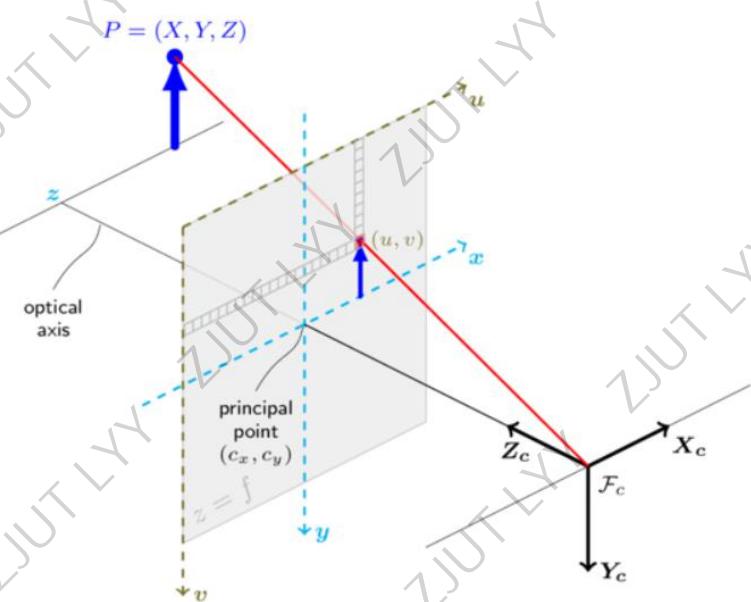


图 10 世界坐标系、像素坐标系和相机坐标系关系图

但是, 实际相机会带有径向和切向的畸变, 那么三者间的转换关系就不像上式理想坐标系中那样简单, 对其修正后的投影关系如下:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$

$$y' = y/z$$

$$r^2 = x'^2 + y'^2$$

$$x'' = x' \frac{1+k_1 r^2 + k_2 r^4 + k_3 r^6}{1+k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2)$$

$$y'' = y' \frac{1+k_1 r^2 + k_2 r^4 + k_3 r^6}{1+k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_2 x' y' + p_1 (r^2 + 2y'^2)$$

$$u = f_x x'' + c_x$$

$$v = f_y y'' + c_y$$

虽然一共有 9 个变量，但这些变量的自由度是 6，所以是完全可求解的。

接下来就是根据这些 2D,3D 点进行求解旋转矩阵。求解方法可使用 opencv 提供的是优化的 DLT 算法，普通的 DLT 算法是选取 6 个点并最小化重投影误差，优化的算法在原本的解之上随机扰动求更优解。优化的 DLT 算法大致如下：

$$\begin{aligned} sAx &= y \\ (Ax) \times y &= \vec{0} \end{aligned}$$

最后可以归结为求解形如 $AX = \vec{0}$ 的方程的解。显然方程有平凡解，可以通过奇异值分解 (SVD) 得到 $|X|=1$ 的一个解。选择特征值的绝对值最小的右特征向量就可以达到最小二乘的效果，因为残差为特征值的绝对值。

旋转矩阵（吉文斯旋转矩阵）是如下的矩阵：

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

对于对任意轴的旋转可以分解为 3 个轴的旋转矩阵的乘积，这三个角度被称为：Yaw (偏航)，指欧拉角向量的 y 轴；Pitch (俯仰)，指欧拉角向量的 x 轴；Roll (翻滚)，指 欧拉角向量的 z 轴。要注意的是由于学界没有一个特定的旋转顺序规定，而按不同的顺序旋转的结果是不同的，所以此处的旋转顺序由我们自己定义。

$$R = \begin{bmatrix} c_y c_z & c_z s_x s_y - c_x s_z & s_x s_z + c_x c_z s_y \\ c_y s_z & c_x c_x + s_x s_y s_z & -c_z s_z \\ -s_y & c_y s_x & c_x c_y \end{bmatrix}$$

求出欧拉角，在我们的实验算法中使用连续 3 帧 Pitch>20°（弧度制约为 0.34rad）作为瞌睡点头的量化标准。

2.1.4 python 多线程编程简介

2.1.4.1 线程、多线程简介

进程就是一个应用程序在处理机上的一次执行过程，它是一个动态的概念，而线程是进程中的一部分，进程包含多个线程在运行。 多线程可以共享全局变量，多进程不能。多线程中，所有子线程的进程号相同；多进程中，不同的子进程进程号不同。多线程（multithreading），是指从软件或者硬件上实现多个线程并发执行的技术。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。具有这种能力的系统包括对称多处理机、多核心处理器以及芯片级多处理（Chip-level multithreading）或同时多线程（Simultaneous multithreading）处理器。在一个程序中，这些独立运行的程序片段叫作“线程”（Thread），利用它编程的概念就叫作“多线程处理（Multithreading）”。具有多线程能力的计算机因有硬件支持而能够在同一时间执行多于一个线程，进而提升整体处理性能。

2.1.4.2 python 线程模块

python 主要是通过 `thread` 和 `threading` 这两个模块来实现多线程支持。python 的 `thread` 模块是比较底层的模块，而 python 的 `threading` 模块是对 `thread` 做了一些封装，可以更加方便地被使用。但是 python（cpython）由于 GIL 的存在无法使用 `threading` 充分利用 CPU 资源，如果想充分发挥多核 CPU 的计算能力需要使用 `multiprocessing` 模块(Windows 下使用会有诸多问题)。

2.1.4.3 如何创建一个线程

python3.x 中已经摒弃了 Python2.x 中采用函数式 `thread` 模块中的 `start_new_thread()` 函数来产生新线程方式。现在 python3.x 中通过 `threading` 模块创建新的线程有两种方法：一种是通过 `threading.Thread(Target=executable Method)`，即传递给 `Thread` 对象一个可执行方法（或对象）；第二种是继承 `threading.Thread` 定义子类并重写 `run()` 方法。第二种方法中，唯一必须重写的方法是 `run()`。

- 1) 通过 `threading.Thread` 进行创建多线程如下所示：

```

1. import threading
2. import time
3. def target():
4.     print("the current threading %s is runing"
5.           %(threading.current_thread().name))
6.     time.sleep(1)
7.     print("the current threading %s is ended"% (threading.current_thread()
8.           .name))
9.     print("the current threading %s is runing"% (threading.current_thread(
9.           .name)))
10.    ## 属于线程 t 的部分
11.    t = threading.Thread(target=target)
12.    t.start()
13.    ## 属于线程 t 的部分
14.    t.join() # join 是阻塞当前线程(此处的当前线程时主线程) 主线程直到 Thread-1
15.      结束之后才结束
16.      print("the current threading %s is ended"% (threading.current_thread()
17.           .name))

```

2) 通过继承 threading.Thread 定义子类创建多线程

使用 Threading 模块创建线程，直接从 threading.Thread 继承，然后重写 init 方法和 run 方法，如下所示：

```

1. class myThread(threading.Thread): # 继承父类 threading.Thread
2.     def __init__(self, threadID, name, counter):
3.         __init__(self)
4.         self.threadID = threadID
5.         self.name = name
6.         self.counter = counter
7.     def run(self): # 把要执行的代码写到 run 函数里面 线程在创建后会直接运
行 run 函数
8.         pass
9. # 创建新线程
10. thread1 = myThread(1, "Thread-1", 1)
11. thread2 = myThread(2, "Thread-2", 2)
12. # 开启线程
13. thread1.start()
14. thread2.start()
15. # 等待线程结束
16. thread1.join()
17. thread2.join()

```

2.1.4.4 如何杀死一个线程

在 python 的多线程编程库中，可用 ctypes 库中的函数强行地杀死一个线程，代码如下：

```
1. def _async_raise(tid, exctype):
2.     """raises the exception, performs cleanup if needed"""
3.     try:
4.         tid = ctypes.c_long(tid)
5.         if not inspect.isclass(exctype):
6.             exctype = type(exctype)
7.         res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, ctypes.
py_object(exctype))
8.         if res == 0:
9.             raise ValueError("invalid thread id")
10.        elif res != 1:
11. #     """if it returns a number greater than one, you're in trouble
12. #     and you should call it again with exc=NULL to revert the effect"""
13.             ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
14.             raise SystemError("PyThreadState_SetAsyncExc failed")
15.     except Exception as err:
16.         print(err)
17. def stop_thread(thread_ident):
18.     """终止线程"""
19.     _async_raise(thread_ident, SystemExit)
```

可以看出，只需要知道待杀死线程的线程标识号 tid（对应 Thread 类型的 ident 属性），即可调用 ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,none) 函数将线程干净利落地杀死。

至此，我们了解了如何开启一个进程，获取其进程号，并在合适的想要的时刻杀死一个进程，这样已经足够支持我们进行后续的实验了。

2.2 硬件连线

需要指出的是针对相关视觉任务的支持，我们使用的是官方的树莓派摄像头，通过树莓派 4B 自带的 CSI 接口与摄像头间互联，而本次开发中语音模块涉及到的硬件连线。包括以下两处：

- 利用 USB 免驱麦克风实现音频的输入，麦克风的淘宝购买地址为 <https://item.taobao.com/item.htm?spm=a1z10.3-c.w4002-678297270.12.130d1ac8NjZ4iu&id=44917780915>。该麦克风的使用仅需将 USB 插入树莓派任一 USB3.0 端口即可，且无需安装额外驱动。
但需要指出的是该麦克风仅支持 44100HZ 的采样频率，并且录制的音频质量不高，需要靠近麦克风给出语音识别的指令。
- 由于在语音模块中设计了主动聆听（持续进行语音交互）和静默睡眠（等待被唤醒）两种工作状态，从静默睡眠到主动聆听模式的切换可以通过监听唤醒词或检测按键实现，本实验采用前者，将在具体语音交互模块的实现部分详细阐述。

至于实物的 GPIO 连线，首先通过下图 11 了解树莓派的 GPIO 资源情况：

树莓派 40Pin 引脚对照表						
wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD 编码	功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V	
8	2	SDA.1	3	4	5V	
9	3	SCL.1	5	6	GND	
7	4	GPIO.7	7	8	TXD	14
		GND	9	10	RXD	15
0	17	GPIO.0	11	12	GPIO.1	18
2	27	GPIO.2	13	14	GND	1
3	22	GPIO.3	15	16	GPIO.4	23
		3.3V	17	18	GPIO.5	4
12	10	MOSI	19	20	GND	5
13	9	MISO	21	22	GPIO.6	25
14	11	SCLK	23	24	CE0	8
		GND	25	26	CE1	10
30	0	SDA.0	27	28	SCL.0	1
21	5	GPIO.21	29	30	GND	31
22	6	GPIO.22	31	32	GPIO.26	12
23	13	GPIO.23	33	34	GND	26
24	19	GPIO.24	35	36	GPIO.27	16
25	26	GPIO.25	37	38	GPIO.28	20
		GND	39	40	GPIO.29	28

表格由树莓派实验室绘制 <http://shumeipai.nxez.com>

图 11 树莓派 GPIO 引脚 (BCM 与 wiringPi 编码)

之后根据各硬件模块所需接口资源，可给出硬件的逻辑连线图如下图 12 所示：

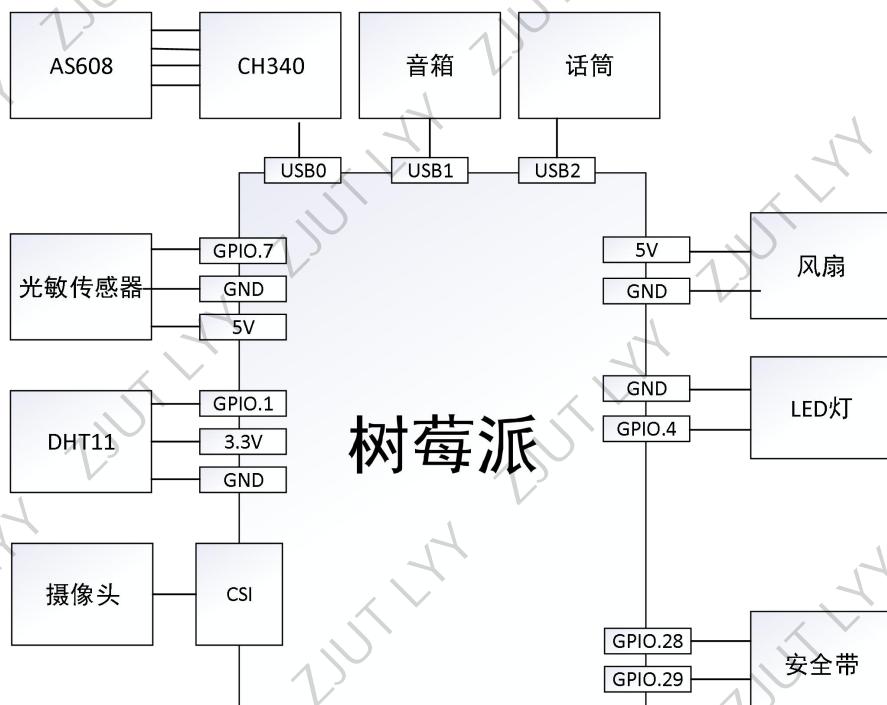


图 12 硬件逻辑连线图

之后根据各硬件模块所需接口资源的逻辑连线，可给出硬件的实际连线图如下图 13 所示：

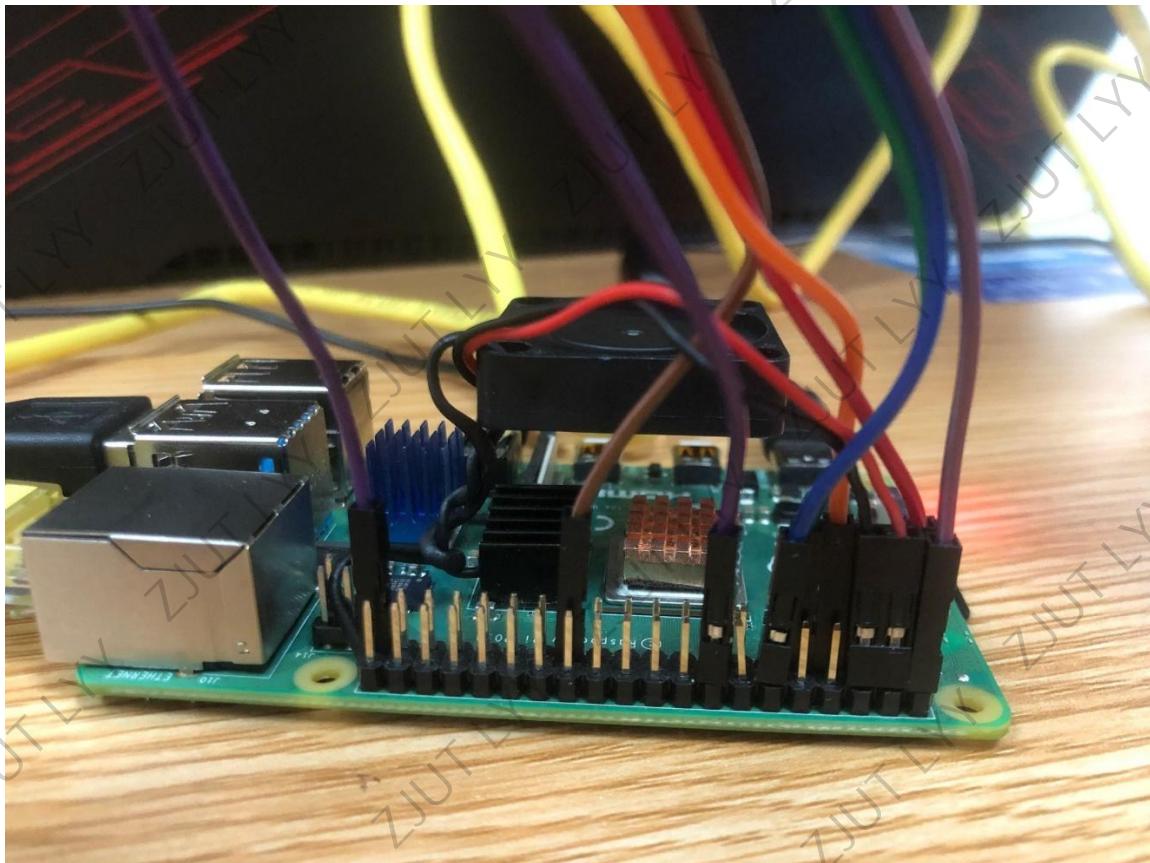


图 13 硬件实物连线图

三、设计思路、步骤和程序流程图

3.1 设计思路

本课程设计的初衷在于设计一款能通过语音方式与驾驶员交互以控制车内设备及功能的助手，从而弥补现有车载设备需要通过触屏交互的不安全性和不便利性。通过指纹识别模块，首先可以识别驾驶员的身份以防止不法分子进入车内行窃；通过温湿度传感器模块，实时监控车内温度，并在温度过高或过低时进行语音提示；通过光敏电阻和 LED 灯模块，实现智能地根据外界光线强弱而自动开启前照灯；通过摄像头模块，实现对驾驶员驾驶过程中的疲劳监测，并在确认疲劳时给出相应的语音提示；加入模拟安全带模块，在开车前检测安全带是否系好，并在没系好安全带时不断给出语音提示；通过语音交互模块，将所有硬件模块与功能相融合，并加入天气查询、时间查询、中译英、闲聊等个性化交互查询功能；最后通过多线程编程的方式实现各模块间的并行运行，从而使产品在现实意义上更上一层楼。综上，本课程设计在功能上总体可分为指纹模块、语音模块和视觉模块和传感器模块，并下设若干子模块，整体功能模块

的设计思路如下图 14 所示：

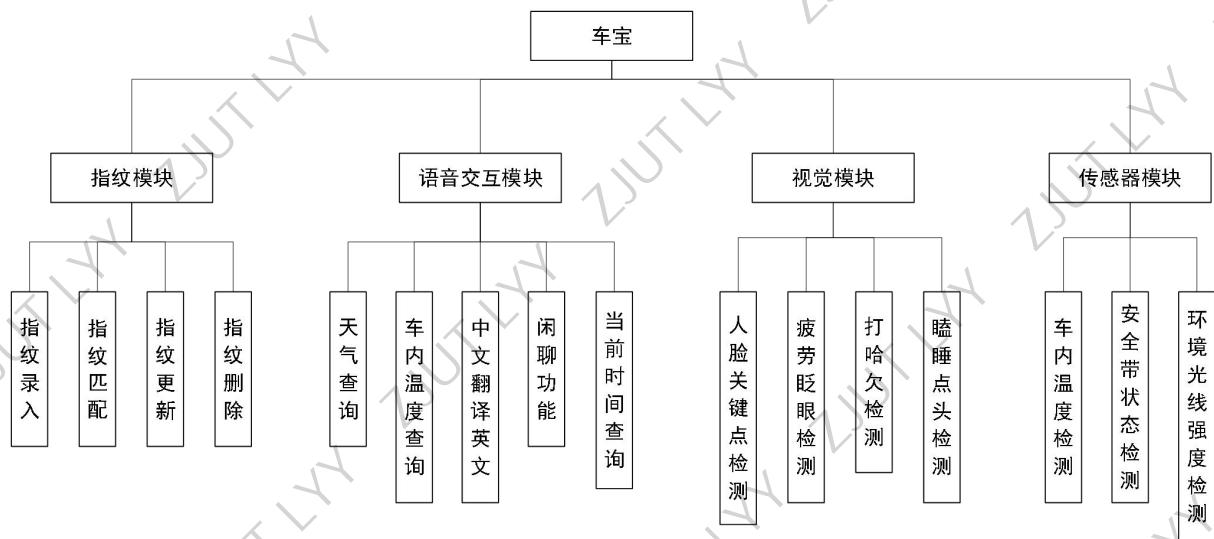


图 14 车宝总体功能模块设计思路

注意上图是对所有功能在逻辑上高阶的划分，而并非与实际的编程实现一一对应。其中，语音交互模块、摄像头模块、传感器模块三者间通过多线程编程技术可实现并行工作；语音交互模块中还有对传感器模块检测到异常后的语音报警，对摄像头模块中检测到驾驶员疲劳状态后的报警功能，以及对录入、删除、更新、匹配指纹成功与失败时的语音提示，但为了总体设计图的清晰与简介，在上图的模块划分中均没有画出。

具体的应用功能列表如下表所示（具体的功能介绍见 3.3 节中各个模块的程序流程图）：

表 1 实际功能列表

功能名称	功能简介	所属模块
指纹录入	对车主所认同的人的指纹进行录入	指纹识别模块
指纹匹配	通过指纹进行开车门前的身份验证	指纹识别模块
指纹更新	对已录入指纹进行特征文件的更新	指纹识别模块
指纹删除	对以录入指纹进行删除	指纹识别模块
天气查询	对指定城市的当天与下一天天气进行查询	语音交互模块
车内温度查询	调用温度传感器进行当前车内温度检测	语音交互模块
中译英	讲中文，翻译成英文	语音交互模块
当前时间查询	当前时间查询	语音交互模块
闲聊	无聊时与车宝聊天消磨时光	语音交互模块
其他模块信息提示	如指纹录入成功的结果通过语音方式提示	语音交互模块
其他模块发出警报	如检测到疲劳驾驶时进行语音报警	语音交互模块

人脸关键点检测	摄像头获取驾驶员人脸照片后标识出 68 关键点	视觉模块
疲劳眨眼检测	第一种疲劳状态，闭眼时间较长	视觉模块
打哈欠检测	第二种疲劳状态，打哈欠	视觉模块
瞌睡点头检测	第三种疲劳状态，瞌睡点头	视觉模块
车内温度检测	DHT11 温湿度传感器模块进行车内温度检测	传感器模块
安全带状态检测	模拟安全带系好才能开车的情景	传感器模块
环境光强检测	车外光线强度检测，并在暗时自动打开 LED 灯	传感器模块

上表中依据底纹颜色区分为 4 个不同的类别，每一类别对应一种模块。

3.2 实验步骤

实验将按照以下几个步骤依次展开：

- (1) 配置树莓派相关开发的环境；
- (2) 各模块软件编程工作同步开展，并各自封装为函数；
- (3) 以摄像头模块、指纹模块、语音模块和传感器模块顺序进行硬件调试；
- (4) 设计完整程序通过语音模块接口串接各功能模块；
- (5) 对系统及各个模块进行测试；
- (6) 对系统整体功能串连进行调试

在本节中首先对步骤（1）进行说明，可认为其是实验开展的准备工作。而后续步骤将在后续的章节中逐一进行展开。

树莓派系统烧录，首次登陆等相关配置在网上有很多参考资料，可是大多参考资料良莠不齐，对于自身的情况很难进行清晰定位，难以有万能的解决方案。本次树莓派配置也花费了我们小组成员大量的时间，但也经过此次经历，对树莓派内部设置，整体把握也有了更加清晰的认识，故在此记录，供一起学习！

●系统烧录

系统烧录的方法比较单一，即通过准备一个 16G（8G 不够装之后实验所需的 opencv 库与 dlib 库等人脸识别相关的 python 开发工具）以上的 SD 卡，然后下载系统，树莓派系统下载网址：<https://www.raspberrypi.org/downloads/>网址加载可能有点慢，有必要的话，可以试试科学上网或在 csdn 上下载（不推荐，还是官方的下载靠谱）。进入网址后选择要下载的系统 RASPBIAN。如下图 15 所示：

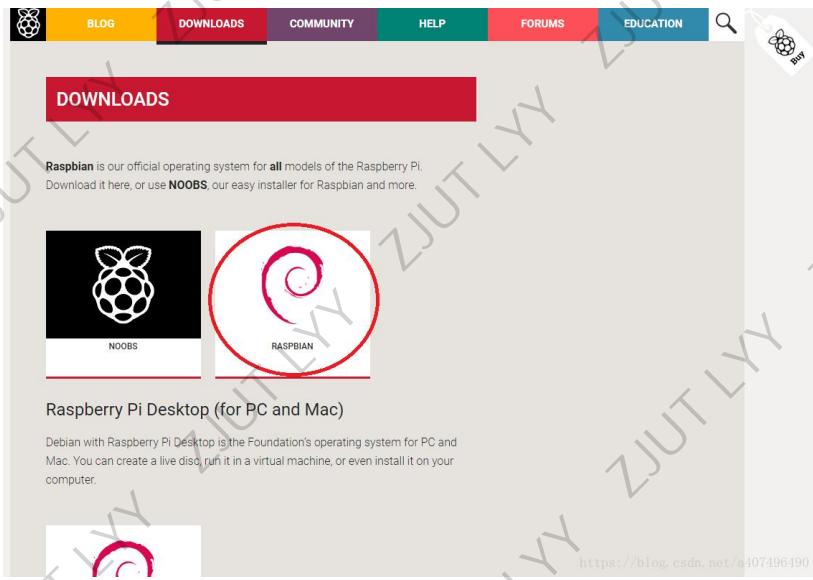


图 15 选择镜像为 RASPBIAN

然后选择 RASPBIAN STRETCH WITH DESKTOP，选择 Download ZIP 或者 Download Torrent 进行下载都可以。如下图 16 所示：

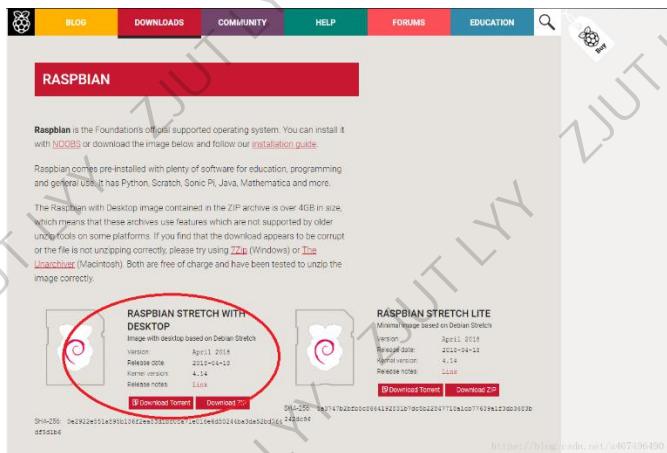


图 16 选择第一个

下载完成后，解压，再下载一个 Win32 Disk Imager，用来写.img 文件到 SD 卡中。Win32 Disk Imager 下载网址：百度一下，你就知道。将 SD 卡插入电脑，打开 Win32 Disk Imager，选择 SD 的盘符。之后点击 Write，写入成功后如下图 17 所示：

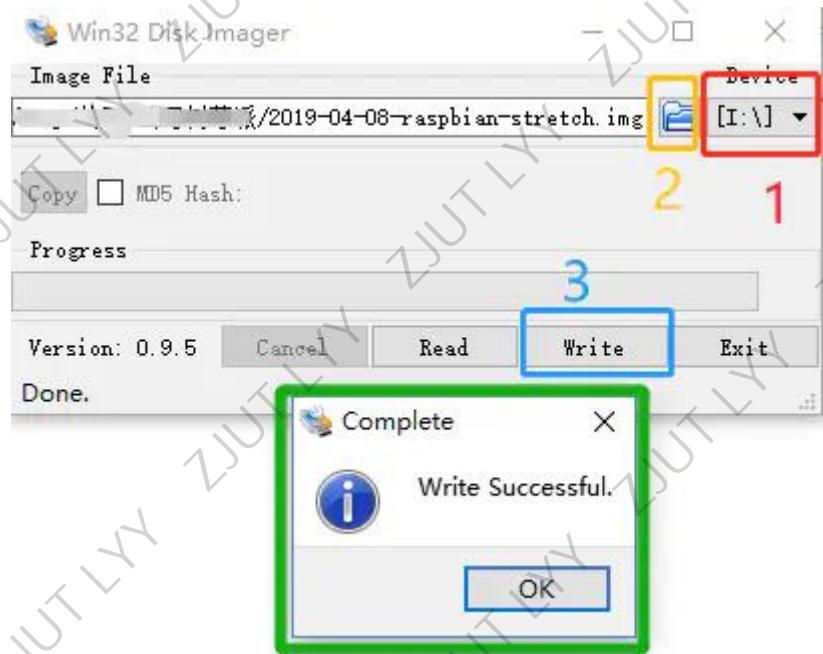


图 17 烧录镜像成功

然后会发现 SD 卡改名为 boot，并且容量变小了，成功后**千万不要**按照系统提示进行格式化 SD 卡的盘，以后插入 SD 卡也不要按照提示格式化 SD 卡的盘。如果要重新烧录镜像，再用 SDFormatter 进行格式化。

●树莓派 IP 查找

本小节内容为查找树莓派 IP 的具体方法，由于大多数使用树莓派的同学往往是没有显示器直接外接树莓派的。因此如何想通过远程登录方式启动树莓派成为一个主要难点。这也是大多数人树莓派配置**难以进行的最关键原因**。下面我将分几种方式分别阐述寻找树莓派 IP 的方法。

◆有显示树莓派找 IP

在树莓派的终端命令行中输入 ifconfig，按“enter”键运行，其中 wlan0 是无线的 ip；eth0 是网线连接的 ip；这种方式简单粗暴，也是我们小组最后采用的方式。由于树莓派 4B 配置的接口为 micro-HDMI 接口，因此我们起初无法找到这种 HDMI 的转换头的时候一门心想通过无显示器配置树莓派，也正是如此屡屡失败，颇有心得体会。

◆无显示找 IP

没有显示器找树莓派的 ip 有几种方法，按优劣排序，（1）最优是路由器网线直接连接树莓派找 ip，之后是用无线的方法。（2）无线的方式有三种，最方便的是用手机开热点给树莓派连接，然后是笔记本电脑开热点给树莓派连接或树莓派连接路由器的 wifi。（3）**最坑的方法**是树莓派和电脑用网线连接找 ip，由于树莓派没有屏幕，如果找不到 ip 完全不知道是什么原因，可能是树莓派没有启动成功（供电不足、没有读取到系统、没有进入到系统卡在开机界面、物理因素等），可能是网络没有连接上（开机时间较长还没进入系统分配 ip、没有自动分配到 ip、分配的 ip 和连接的网络不是同个网段、物理因素等）。

上述为四种我查询到的无显示器找 IP 的方法，我本人一直追求采用的方式是无线网络找 IP（因为在学校没有能直接上网的路由器供我连接），具体如下：

首先要确保您的电脑能看到文件的后缀，可以按照 windows 电脑查看文件后缀 进行设置，设置完就可以查看文件的后缀了。将插有 SD 卡的读卡器插入电脑，在电脑的 boot 盘新建一个 txt 后缀的文本文件。如下图 18 所示：

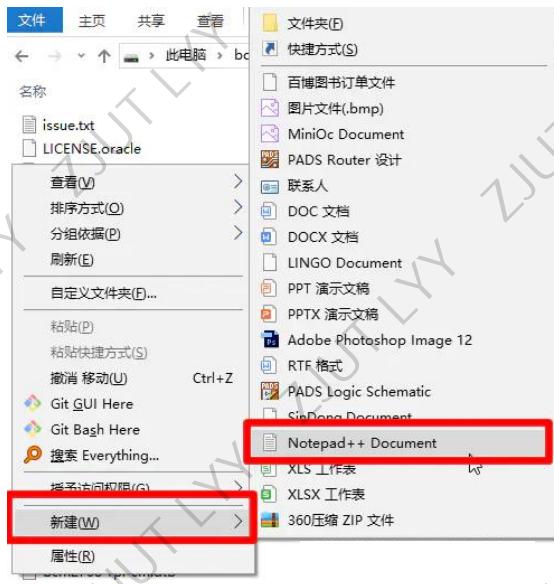


图 18 新建 txt 后缀的文本文件

打开文本文件，注意填写 wifi 名称和密码，两者尽量设置简单点的纯英文和数字，并且不能是中文，在其中输入下面这段文字，符号必须是英文的符号，不能是中文的符号：

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=CN
network={
    ssid="填写 wifi 名称"
    psk="填写 wifi 密码"
}
```

保存后将该文件进行重命名为 wpa_supplicant.conf。弹出警告按“是”确定更改。

电脑弹出 SD 卡的 boot 盘后，将 SD 卡插入树莓派的卡槽，可以在 wifi 源查看到树莓派有没有连接上。通过这个方法即可将树莓派的 IP 地址搜索得到，并且可以进行后续的远程登录模块。

原理：上述步骤是为了预存放一个 wifi 源连接配置文件，即插上树莓派后，树莓派会依据其内部信息（wifi 名， wifi 密码）进行网络连接。其旨在于帮助我们在无显示器的情况下将树莓派连接到我们个人的手机热点等无线网络中。

注意: 如果发现树莓派没有连接上 wifi 源的话, 将 SD 卡再插回电脑看看 wpa_supplicant.conf 这个文件还在不在, 如果不在, 应该是 wifi 的名字或密码输入错误了(要注意不能是中文的名字, 写符号也要注意是什么符号, 字母注意大小写); 如果文件还在, 并且名字和我文件的名字一样的话, 可能是系统没有启动成功, 重新烧录系统试试。

●远程登录

由于树莓派远程登录模块网上资料较为统一, 也不是大家会碰到的疑点难点, 因此我不在此进行赘述, 但我可稍微提一下我远们程登录的方式。本小组共使用了两种方式连接树莓派。

- 1) 第一种, 为方便连接树莓派, 在开启树莓派后并且连接到我的手机热点的情况下(即树莓派与我的 PC 端处于统一局域网中)。利用 windows10 自带的远程登录方式进行登录。

在树莓派命令行下输入 sudo apt-get install xrdp 命令安装 xrdp。

接着在 windows 下的运行界面输入 mstsc 来打开远程桌面。如下图 20 所示:



图 20 利用 mstsc 进行远程登录

Raspberry pi 系统中的 Raspbian 默认用户是 pi 密码为 raspberry

至此我们就可以进入树莓派界面, 进行后续的功能实现与操作了。如下图 21 所示:

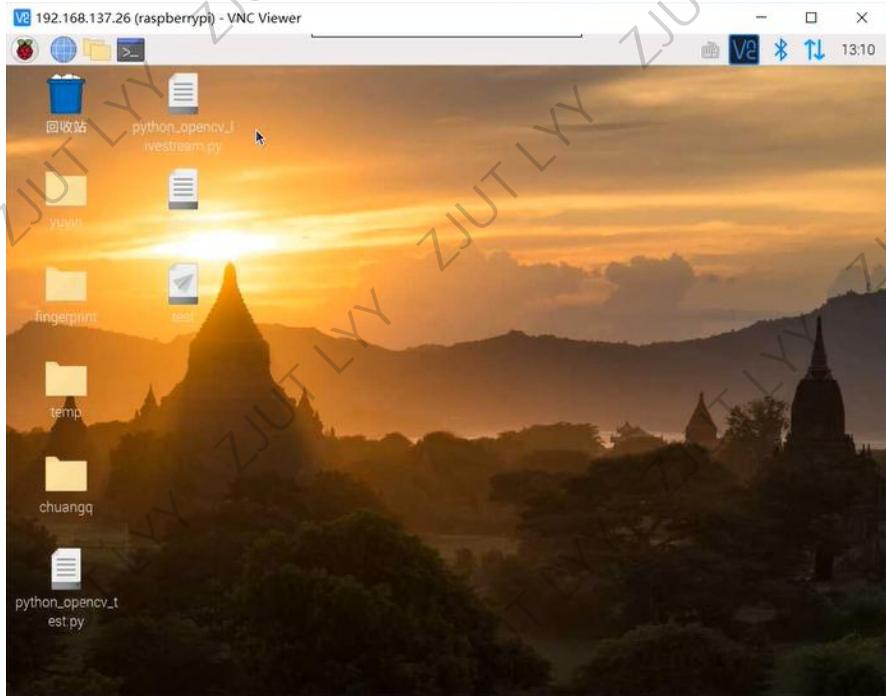


图 21 远程登录成功界面

第二种方式通过网线代替热点，为树莓派提供网络，并通过 VNC Viewer 软件远程登录树莓派。用网线连接树莓派和笔记本，打开笔记本的网络连接部分，可通过更改适配器选项打开，之后选择当前正在使用的网口，我们组使用的是 wlan 网络，选中后右键打开打开命令行，点击“属性”，选择“共享”，打上第一个框的勾，并且在网络连接部分选择以太网，最后点击确定。输入 arp -a 命令并回车。显示界面如下图所示：

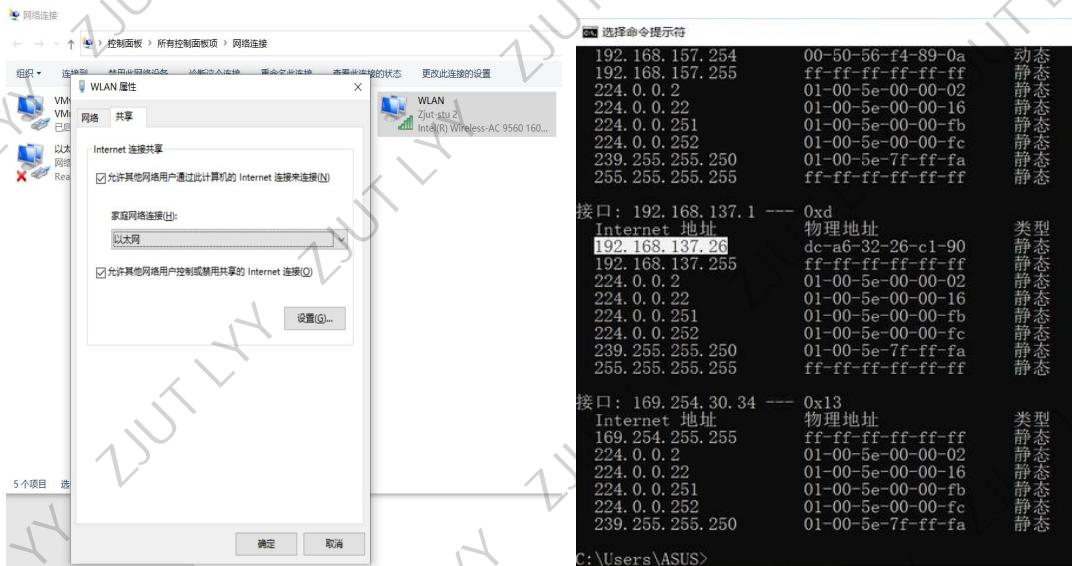


图 22 查询 ip 地址

arp -a 命令用于记录查看出现的 IP 地址与物理地址的列表信息，当树莓派和笔记本通过网线连接以后，通过此命令可以查看到树莓派的 MAC 地址和对应的 ip 地址，通过之前的提及的热点连接可以得知树莓派的 MAC 地址，因此也就得知了此时的 ip

地址。打开 VNCViewer 软件，在界面上的搜索栏中填入 ip 地址，并键入回车，如图 23 所示。此时就能够成功连接树莓派，此时还需要输入相应的账号和密码才能进行登录，成功后就可以进入树莓派主界面。如下图 23 所示：

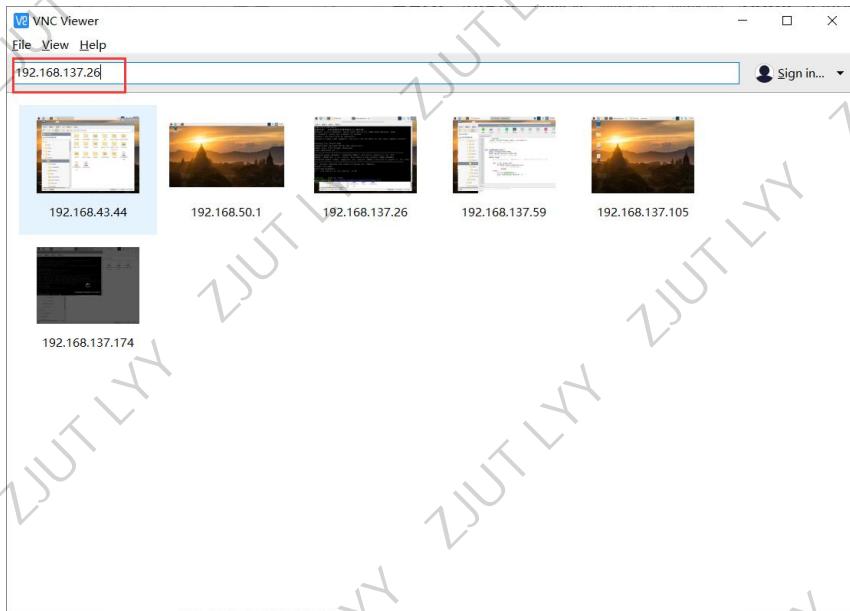


图 23 VNCViewer 连接树莓派

3.3 各功能模块的实现

基础功能模块均以函数的形式进行封装，方便应用功能进行调用。此次基于 python 3.7 环境进行开发，主要是基于从 2020 年 1 月 1 日起 python 2.x 将不再更新和维护，各模块实现过程中的参考资料将随文给出，需要指出的是参考资料中大多基于 python 2.x 环境构建，但在本次实验过程中全部做了更新和移植。

3.3.1 指纹模块的实现

3.3.1.1 所用模块介绍

指纹识别模块主要运用了 AS608 模块，该模块可以链接上位机进行指纹识别，上位机可以是 windows 系统也可以是树莓派，在这里我们的运行思路是通过上位机将指纹录入，再通过树莓派检测指纹是否匹配。

首先需要介绍 AS608 的链接组件 CH340，这是一个 usb 转 ttl 的接口，因为对于笔记本来说，并没有丰富的串口线用于进行数据交互，所以在笔记本上必须通过 CH340 模块来进行 ttl 转 usb 再进行操作，而对于树莓派来说，ttl 转 usb 也有着得天独厚的优势：

- 我们小组的树莓派并未额外购置面包板，在数据通信上，gpio 串口通信只能进行一次，而该指纹识别模块需要实时插入树莓派，所以使用树莓派上的 4 个 usb 端口，可以很好地解决该问题

- Usb 通信接口对比 ttl 来说，有着较为稳定的特性，该特性一开始也是从 csdn 上听说，但在后续实践中，多次排查问题，发现是传回数据包的问题，具体是不是因为 ttl 串口通信存在问题我也不清楚，但是就是不能通，虽然不可否认 ttl 串口也有一些优点，比如串口损坏后易于更换等。
- 树莓派的串口通信如果要配置，需要把 ttyAMA0 和 ttys0 进行更换，不然会不稳定。配置这些影响树莓派的其他使用，可能对其他功能造成影响

首先讲解这两个部件的链接，我们购买了 AS608 带有的 pin 口线转杜邦线，如果购买的是 pin 口线直接接出的话，我就不知道怎么链接了，根据最开始买错的 pin 口线接出结构，引脚定义也不相同，在这里使用杜邦线接出的 as608. 使用杜邦线链接 ch340 的杜邦线父口，ch340 有五个接口，如下图 24 所示：



图 24 CH340 实物图

AS608 的 8 个接口及功能如下表 2 所示：

表 2 接口表

序号	名称	说明
1	Vi	模块电源正输入端。
2	Tx	串行数据输出。 TTL 逻辑电平
3	Rx	串行数据输入。 TTL 逻辑电平
4	GND	信号地。内部与电源地连接
5	WAK	感应信号输出， 默认高电平有效 6
6	Vt	触摸感应电源输入端， .3v 供电
7	U+	USB D+
8	U-	USB D-

接着，用 AS608 的 Rx 接 CH340 的 Tx； CH340 的 Tx 接 AS608 的 Rx。给 AS608 接上 3.3v 正电源引脚，GND 接 GND 即可，如下图 25 所示：

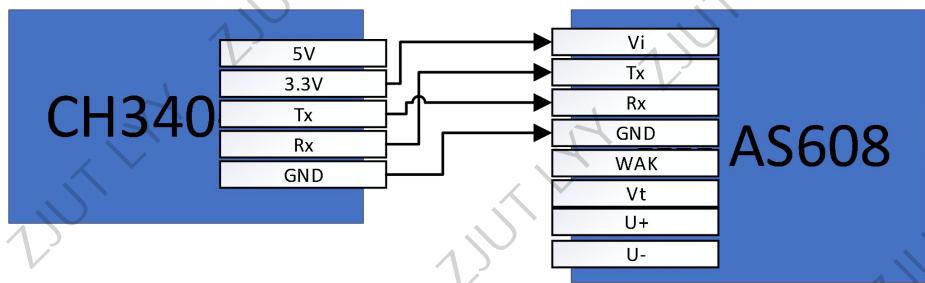


图 25 引脚逻辑连线图

3.3.1.2 模块连接与 usb 通信实现

下面分别讲解与上位机和树莓派的链接。

与上位机的连接较为简单，商家材料中自带一个图形化界面，使用 ch340 与 as608 接完后，直接通过 usb 接口链接上位机，在上位机上安装 ch340 的驱动程序后，可以在电脑上搜索到相应设备，如果不安装驱动，那找到的设备不会显示 ch340。之后找到对应的端口，再启动 xcom。即可在上位机对 as608 进行所有模块的操作。包括指纹录入识别删除，并且该图形化界面十分的好用。在本次实验中上位机主要使用到的是它的录入指纹操作和指纹匹配更新功能。模拟的是可以使用用户的录入过程。图形化界面展示如下图 26 所示：



图 26 Xcom 界面示意图

对于指纹识别模块与树莓派的链接，同样也是对于端口选取，首先插入树莓派任意一个 usb

端口，并输入 lsusb 指令以查看是否成功连接，如果显示出 HL-340 则已经成功连接了，如下图 27 所示：

```
Bus 001 Device 003: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
```

图 27 连线成功示意图

接下来就需要看连接上哪个端口，输入 ls -l /dev/tty* 会显示所有端口，包括串口和 usb 口，找到开头为 usb 的端口，如果只连接了一个 usb 端口那么就是这个端口进行通信，识别出进行通信的端口后，就需要使用 minicom 的通信工具，使用 apt-get 指令进行下载，并使用管理员权限运行，在设置界面中把串口通信的端口和波特率信息修改后即可，minicon 的作用类似于嵌入式设计中的串口通信助手，如下图 28 所示：

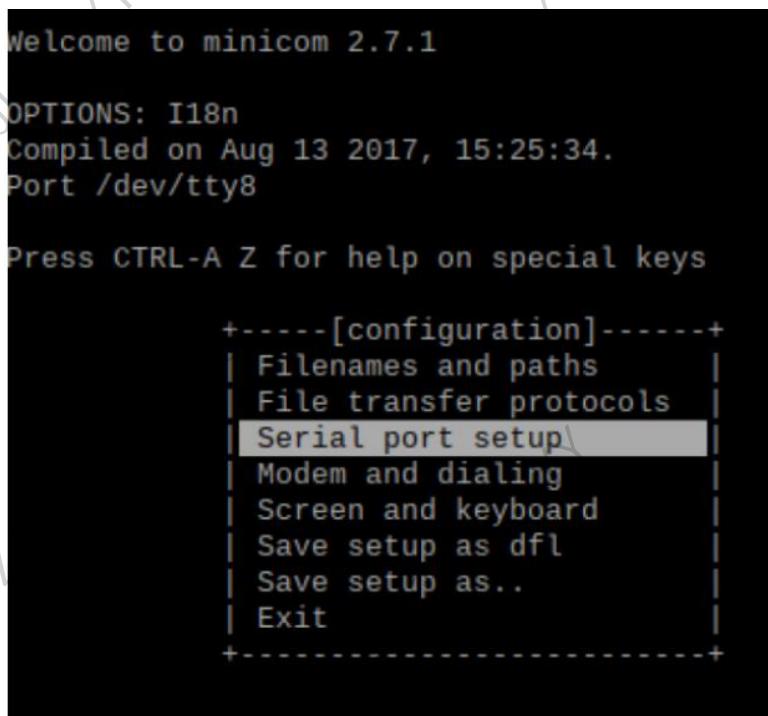


图 28 Minicom 示意图

其中的波特率建议选择 57600 其他波特率都验证失败了，过大 AS608 会损坏（没试过，三四十一个还挺贵）。端口就是 ttyusb0 或者是 ttyusb1，具体要看链接上的端口之前显示是多少。链接成功后，就可以在树莓派上正常使用 AS608 了。在目标环境下，我们只主要进行指纹识别，指纹录入和更新指纹功能。具体配置结果如下图 29 所示：

```

A - Serial Device      : /dev/tty0
B - Lockfile Location : /var/lock
C - Callin Program   :
D - Callout Program  :
E - Bps/Par/Bits     : 57600 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

Change which setting? □

```

图 29 USB 通信配置图

完成如上配置后，就可以使用 usb 口与指纹模块进行通信了。

3.3.1.3 实验过程与结果展示

- 数据传输分析。

在通信过程中，数据传输主要通过数据包进行通信，此处主要用到两类数据包，命令包与应答包，格式分别如下表 3 和表 4 所示：

表 3 命令包格式

字节数	2 bytes	4 bytes	1 byte	2 bytes	1 byte	N bytes			2 bytes
名称	包头	芯片地址	包标识	包长度	指令	参数 1	...	参数 n	校验和
内容	0xEF01	XXXX	01	N=					

表 4 应答包格式

字节数	2 bytes	4 bytes	1 byte	2 bytes	1 byte	N bytes		2 bytes
名称	包头	芯片地址	包标识	包长度	确认码	返回参数		校验和
内容	0xEF01	XXXX	07	N=				

AS608 与树莓派间的通信主要通过树莓派发出命令包，AS608 解析包，完成相应函数并传回应答包。树莓派读取应答包，并从中取出确认码和返回参数完成后续编程，具体流程图如下图 30 所示：

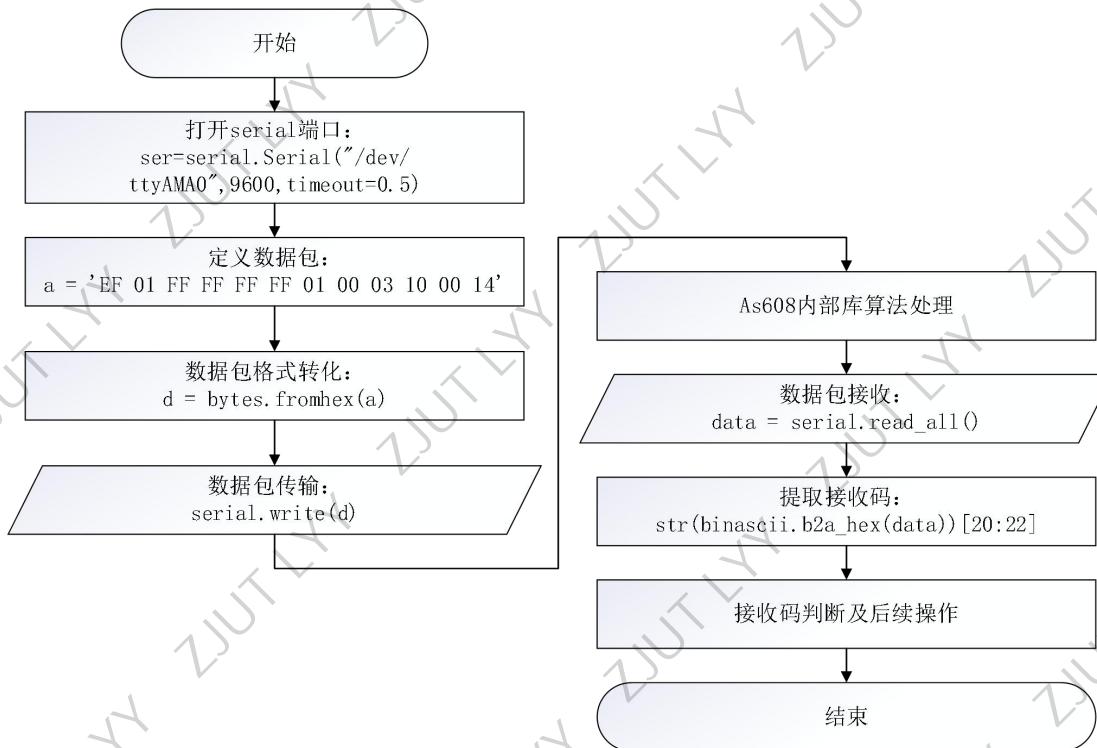


图 30 AS608 通信流程图

两类包的区分通过包标识进行，地址如果不进行额外配置，则默认为 `ffffffff` 即可，在命令包中最重要的是“指令”，该内容代表了 as608 需要完成的功能，本次使用的“指令”共如下表 5 所示有以下几种：

表 5 指令表

指令码	指令含义
01H	读入图像，将图像存储在缓冲区
02H	生成特征，将缓冲区的图像生成特征值
04H	使用特征值搜索指纹库
10H	单次生成指纹并存入指纹库
11H	录入指纹并在库中搜索相同指纹并更新

调用何种函数，而应答包中最重要的是确认码，确认码代表了 as608 处理结果，具体确认码很多，本次主要用到了由下表 6 所示的以下四种确认码：

表 6 确认码表

确认码	确认码内涵
00H	指令成功
02H	传感器未检测到手指

03H	录入指纹图像失败
08H	指纹不匹配
10H	快速注册指纹失败

了解了完成功能所需的数据包后，开始对实际操作实践作整体逻辑设计，设计指纹识别为所有功能的入口，所以需要对指纹识别的权限进行所有定义，设计计算机为管理员，树莓派为用户，对于用户，可以进行登陆，如果自己的指纹识别并不清晰，可以对自己的指纹进行更新，可以录入简单指纹进行使用，同时，为了防止无指纹存在在库中导致的无法登陆录入第一个指纹，用户指纹被用户意外删除导致的一系列 bug，将删除指纹功能，删除所有指纹等功能包装至上位机使用。额外的，设计模板操作，批量录入指纹，指纹库初始化等管理员功能给予上位机。普通用户和管理员的权限功能具体如下表 7 所示：

表 7 功能权限表

	用户	管理员
录入指纹	单次录入指纹	单个、批量录入指纹
更新指纹	对指纹库中已存在的指纹进行更新	
指纹验证	对指纹库中的指纹模板进行验证	
删除指纹	无该功能	单个、批量清除，指纹

下面先对用户所有功能进行详细逻辑分解和部分流程展示。

1) 录入指纹

录入指纹主要使用的是指令表的第 11 号指令。该指令可以读取一次按在 as608 上的指纹并按照顺序存储在模板库中，返回存储位置。存储位置 4byte 存储在确认码后，实际使用过程中，为了与语音模块连接，将其包装在函数内，函数返回值即为录入成功与否和录入位置。发包示例为：EF 01 FF FF FF FF 01 00 03 10 00 14

As608 逻辑流程图如下图 31 所示：

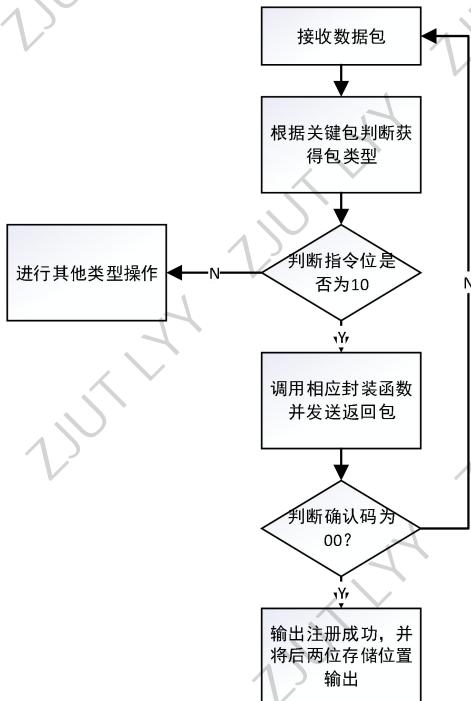


图 31 录入指纹流程图

此时，树莓派结果如下图 32 所示：



图 32 注册成功图

2) 更新指纹

更新指纹使用的是指令表中的第 11H 号指令，该指令读取一次按在 AS608 上的指纹并在模板库中搜索指纹存储位置，如果找到了对应位置，则合并特征值，返回特征值得分和存储位置。存储位置以 4byte 存储在确认码后，再其后 4byte 是特征分值，在实际过程中，将三个信息一同返回。发包示例为：EF 01 FF FF FF FF 01 00 03 11 00 15

该数据包中，表示指令位的是“11”，前四位是固定包头，往后八位是地址。默认 0xffffffff 不变。AS608 逻辑流程图如下与录入指纹大致相同。树莓派结果展示如下图 33 所示：

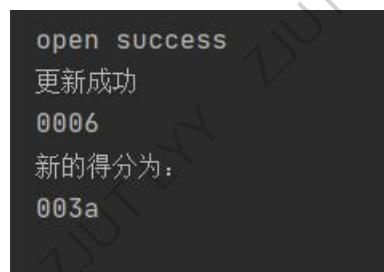


图 33 更新成功结果图

3) 指纹验证

指纹验证较为复杂，需要通过三次收发数据包完成该功能，第一次获得指纹图像，第二次将获得图像进行转存，第三次在库中进行匹配。但返回值较为简单，只会返回成功或是失败的确认码，并没有额外数据返回。在实际使用中，该验证为树莓派功能启动函数，通过该模块作为主函数的入口。AS608 逻辑流程图如下图 34 所示：

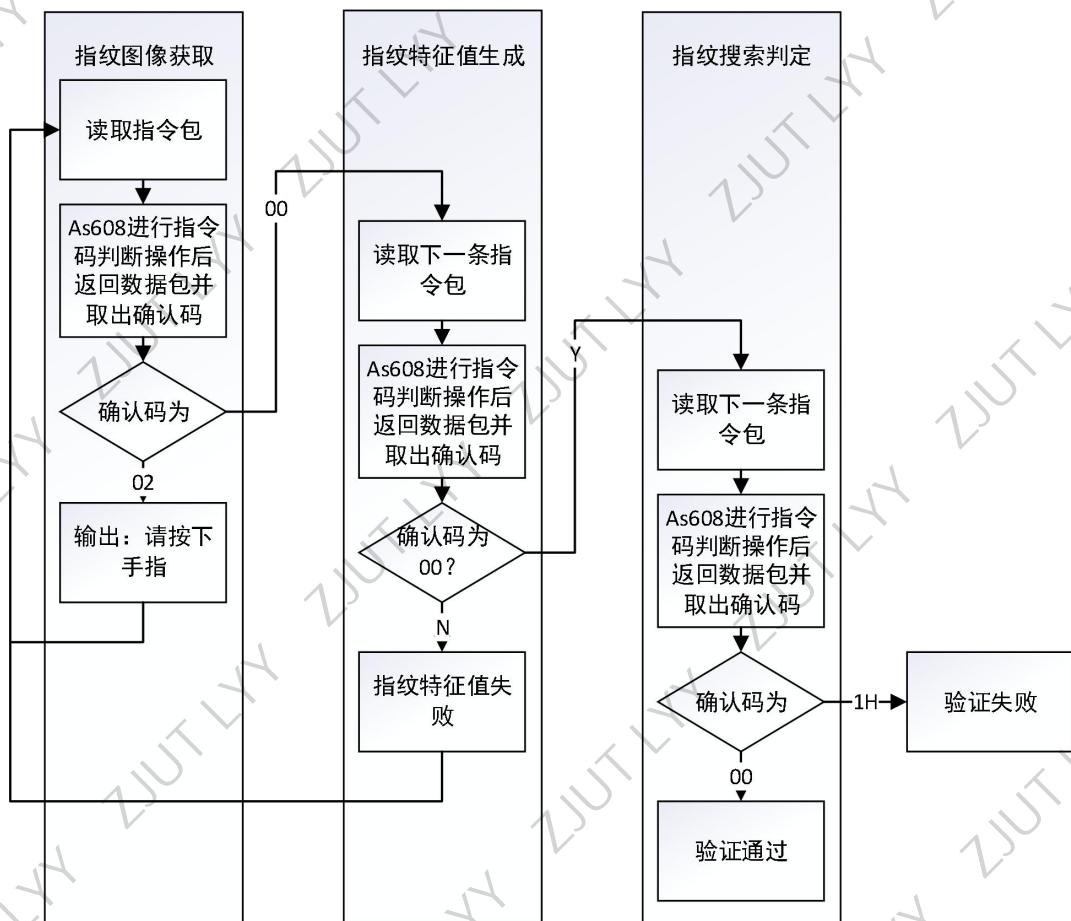


图 34 指纹验证程序流程图

树莓派结果展示如下图 35 所示：

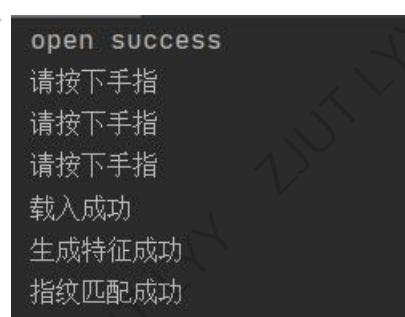


图 35 匹配成功结果图

- 对于笔记本电脑作为上位机，xcom 的使用示例如下所示。

首先展示录入指纹模块，下图为详细录入上位机实践，在输入指纹存入位置后，放上手指：



图 36 指纹录入 1

第一次识别成功后会展示指纹图片，并提示再次录入指纹图片：



图 37 指纹录入 2

两次录入成功后，如果两指纹特征值匹配成功，则会将该指纹存入设置位置：



图 38 指纹录入 3

Xcom 还有一个可以用直观展示目前被使用位置的区块，该区块内提供对选中指纹进行比对删除等功能：



图 39 指纹库区块

按下搜索后并在 AS608 上输入指纹，可以在指纹库中进行比对，展示匹配指纹：



图 40 搜索指纹

- 调试说明：一开始采用串口通信进行了很长一段时间，但是串口通信始终接收不到收回包，可以之后再进行尝试，解决方案就是使用 ch340 改变思路通过 usb 而绕过了串口。在对端口配置的时候，如果因为不熟悉而输错了端口导致端口甚至不是一个可以启用的端口，那么 minicom 会因此不能正常启用，这时候就需要通过 minicom -s 直接绕过启动进行配置。
- 特点：采用上位机系统和树莓派两个系统联合操作，不同系统完成不同功能，很好地模拟了管理员和用户两个层面的设置。
- 心得体会，串口通信是真的不如 usb，usb 替代串口是有原因的。

3.3.2 视觉模块的实现

由于在 2.1 设计原理部分已经详细解释过视觉模块中核心部分——疲劳检测的三种疲劳状

态检测原理，所以这一部分不再赘述，取而代之的将是实际开发的准备过程、效果以及一些问题。所以，视觉模块的实现，将从官方摄像头的连接与调试，疲劳检测程序流程图以及实际效果这三方面展开。

3.3.2.1 官方摄像头的连接与调试

本次课程设计中所使用的摄像头模块为官方提供的原装摄像头，它支持 720P 和 30 帧的摄像效率，基本可以满足我们实验的需求，实物如下图 41 所示：



图 41 官方摄像头实物图

然后是树莓派 4B 与摄像头的连接，如下图 42 所示，在树莓派板子上，白色字体显示 Camera 字样的插槽为 CSI 接口，它便是板子上自带的和官方摄像头进行通信的接口。我们在连接时，首先需要将插槽的黑长条往上轻轻拔起，注意不是拔出来，防止装不回去！

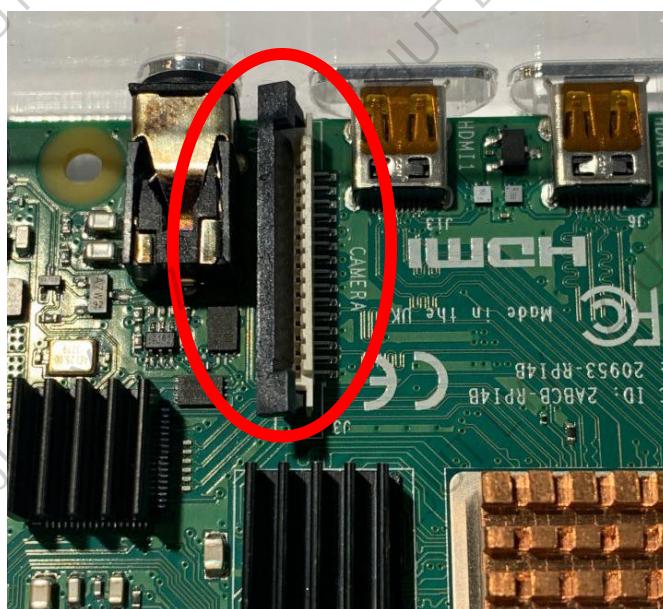


图 42 树莓派板子上的 CSI 接口图

然后把树莓派摄像头白色线路插入，然后把拔起来的黑色长条再压入原位，起固定作用，如下图 43 所示：



图 43 连接完成效果图

接下来开机启动树莓派并登陆，做一下摄像头使用前的准备工作。

执行下文介绍的命令行进行下载并安装最新的内核，GPU 固件及应用程序。当然，我们需要连接到互联网才可以实现以下操作：

1. `$sudo apt-get update`
2. `$sudo apt-get upgrade`

接下来，我们首先需要在 Raspberry Pi 的 raspi-config 程序中启用摄像头的支持。

- 1) 连接摄像头与树莓派
- 2) 修改树莓派配置，开启摄像头模块。

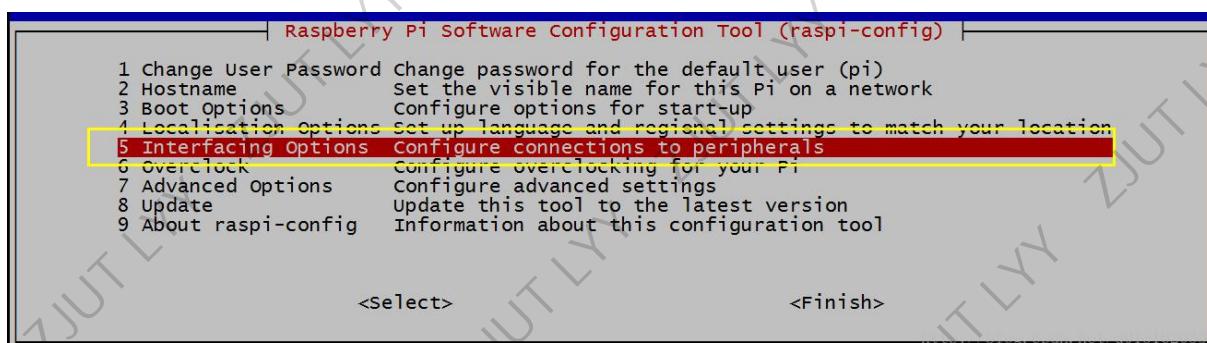


图 44 进入配置界面选择 Interfacing Options

将光标移动到摄像头选项 (Camera option) 处，并选择启用 (Enable)。在退出 raspi-config 时会要求您重新启动。启用选项是为了确保重启后 GPU 固件能够正确运行（包括摄像头驱动和调节电路），并且 GPU 从主内存划分到了足够的内存使摄像头能够正确运行。

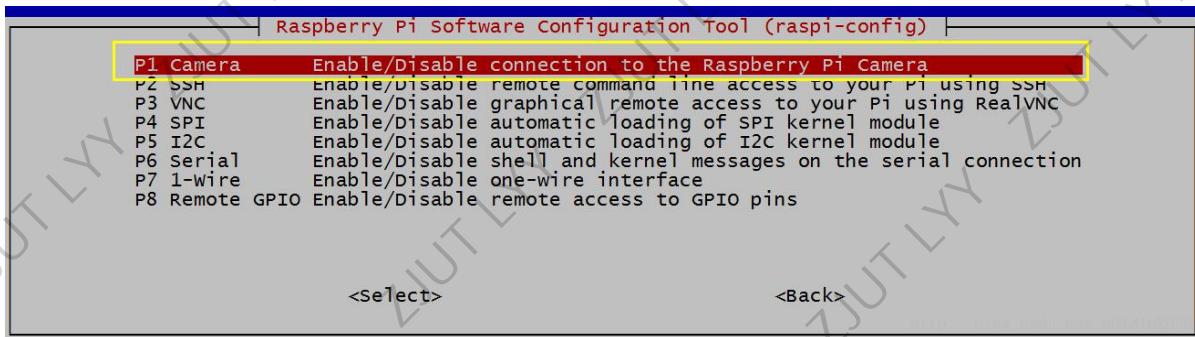


图 45 使能摄像头

在使能了摄像头之后，我们可以测试系统已经完成安装并正常工作，使用以下命令：

1. `$raspistill -v -o test.jpg`

这个命令将显示来自摄像头 5 秒钟的预览图像，并且拍摄一张照片，然后保存为文件 test.jpg，保存在/home/pi 的目录下面，同时显示出需要相关信息。如此操作，若得到了类似下图（我们实际开发中得到的图片）的名为 test.jpg 的图片，就说明摄像头配置成功，可以进行之后的操作了。



图 46 所得 test.jpg

3.3.2.2 开发准备

这一部分将介绍本次课程设计中所使用的图像获取、人脸关键点定位与标定部分重要的两个工具 opencv 以及 dlib 库，和它们在树莓派 4B 上的安装方式。

- OpenCV 的介绍

本次实验的视觉部分主要由利用 OpenCV 这一工具进行展开描述，OpenCV 的全称是 Open Source Computer Vision Library，是一个跨平台的计算机视觉库。OpenCV 是由英特尔公司发起并参与开发，以 BSD 许可证授权发行，可以在商业和研究领域中免费使用。OpenCV 可用于开发实时的图像处理、计算机视觉以及模式识别程序。该程序库也可以使用英特尔公司的 IPP 进行加速处理。

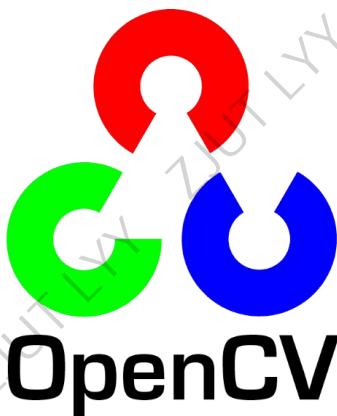


图 47 OPENCV 符号

- OPENCV 的安装经验

由于大部分同学在树莓派上安装 OpenCV 时会遇到各种各样的麻烦，并且本小组成员在此过程中也踩坑不少。因此在这一小节对 OpenCV 的安装进行部分经验总结。

本次实验采用的版本号为：python3.7.3 + OpenCV3.4。安装步骤大致如下：

- 1) 首先，在树莓派设置中把根目录扩大到整个 SD 卡

为在 python3 上安装 OpenCV 我们首先需要通过命令行 `sudo raspi-config` 进入树莓派配置界面，在树莓派设置中将根目录扩大到整个 SD 卡。因为默认情况下的树莓派根目录大小仅仅分配了几百兆，很容易造成部分 python 包的下载失败（我们小组就在此吃亏多次，苦苦查找问题毫无头绪，一经扩大 SD 的内存，即轻松完成安装。）

- 2) 命令行界面输入命令，进入树莓派配置界面。用上下键和左右键切换光标位置。

- 3) 第七行：Advanced Options，选择 Expand Filesystem，将根目录扩展到这个 SD 卡，充分利用 SD 卡的存储空间。如果不进行这一步，后续命令会出现卡死。

- 4) 退出设置界面，重启树莓派。

- 5) 安装 OpenCV 所需的库

由于 OpenCV 在安装时，需要很多依赖项，而这些依赖项都是必不可少的（注意倒数第三条命令中要安装四个 dev 包）本人在初次安装时，安装网上教程的说法总是缺失一些依赖项导致安装失败，抱着软件包多没坏处的思想，我将其全部进行安装，从而使得错误消失。命令如下图 48 所示：

```
sudo apt-get install build-essential git cmake pkg-config -y  
sudo apt-get install libjpeg8-dev -y  
sudo apt-get install libtiff5-dev -y  
sudo apt-get install libjasper-dev -y  
sudo apt-get install libpng12-dev -y  
  
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y  
  
sudo apt-get install libgtk2.0-dev -y  
sudo apt-get install libatlas-base-dev gfortran -y
```

图 48 安装依赖库相关命令

6) 下载 OpenCV

下载 OpenCV 的命令其实很简单，只需要两条指令即可。如下图 49 所示：

```
wget https://github.com/Itseez/opencv/archive/3.4.0.zip  
  
wget https://github.com/Itseez/opencv_contrib/archive/3.4.0.zip
```

图 49 安装 OpenCV 的命令

我们需要注意的是，第二个包的安装速度很慢，可能很多同学都会跟我一样，一旦开始安装下载，但命令行界面久久不得更新，一直停滞在某一界面，误以为安装失败便草草关门命令行框，取消安装。其实 OpenCV 由于网络的原因会导致下载的速度很慢很慢（比如每秒几个 KB）。

为加速 OpenCV 包的安装，我总结经验以供自己参考学习。我采用的方法为：可以在电脑浏览器中输入 wget 后面的链接下载压缩包，再用 Filezilla 或者 U 盘等方法把文件传输到树莓派的 /home/pi/Downloads 目录下（一定不能错）。通过电脑下载包再间接拷贝到树莓派的方法确实在速度上有明显的提升。

7) 编译

对于下载成功的 OpenCV 安装包，在执行编译操作前，我们首先需要进行编译参数的设置。

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D  
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D  
OPENCV_EXTRA_MODULES_PATH=/home/pi/Downloads/opencv_contrib-3.4.0/modules -D  
BUILD_EXAMPLES=ON -D WITH_LIBV4L=ON PYTHON3_EXECUTABLE=/usr/bin/python3.5
```

```

PYTHON_INCLUDE_DIR=/usr/include/python3.5
PYTHON_LIBRARY=/usr/lib/arm-linux-gnueabihf/libpython3.5m.so
PYTHON3_NUMPY_INCLUDE_DIRS=/home/pi/.local/lib/python3.5/site-packages/numpy/core/include ..

```

并根据下图 50 来判断你的 CMAKE 配置的成功与否：

```

-- OpenCV: YES (no extra features)
-- Include path: /home/pi/opencv-3.4.0/3rdparty/include/opencv3.4.0
-- Link libraries: Dynamic load
-- Python 2:
--   Interpreter: /usr/bin/python2.7 (ver 2.7.13)
--   Libraries: /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.13)
--   numpy: /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.12.1)
--   packages path: lib/python2.7/dist-packages
-- Python 3:
--   Interpreter: /usr/bin/python3 (ver 3.5.3)
--   Libraries: /usr/lib/arm-linux-gnueabihf/libpython3.5.so (ver 3.5.3)
--   numpy: /usr/lib/python3/dist-packages/numpy/core/include (ver 1.12.1)
--   packages path: lib/python3.5/dist-packages
-- Python (for build):
-- Java:
--   ant: NO
--   JNI: NO
--   Java wrappers: NO
--   Java tests: NO
-- Matlab: NO
-- Install to: /usr/local

-- Configuring incomplete, errors occurred!
See also "/home/pi/opencv-3.4.0/build/CMakeFiles/CMakeOutput.log".
See also "/home/pi/opencv-3.4.0/build/CMakeFiles/CMakeError.log".
pi@raspberrypi:~/opencv-3.4.0/build$ 
```



```

-- OpenCV: YES (no extra features)
-- Include path: /home/pi/Downloads/opencv-3.4.0/3rdparty/include/opencv3.4.0
-- Link libraries: Dynamic load
-- Python 2:
--   Interpreter: /usr/bin/python2.7 (ver 2.7.13)
--   Libraries: /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.13)
--   numpy: /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.12.1)
--   packages path: lib/python2.7/dist-packages
-- Python 3:
--   Interpreter: /usr/bin/python3 (ver 3.5.3)
--   Libraries: /usr/lib/arm-linux-gnueabihf/libpython3.5.so (ver 3.5.3)
--   numpy: /usr/lib/python3/dist-packages/numpy/core/include (ver 1.12.1)
--   packages path: lib/python3.5/dist-packages
-- Python (for build):
-- Java:
--   ant: NO
--   JNI: NO
--   Java wrappers: NO
--   Java tests: NO
-- Matlab: NO
-- Install to: /usr/local

-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/Downloads/opencv-3.4.0/build
pi@raspberrypi:~/Downloads/opencv-3.4.0/build$ 
```

图 50 左图为 CMAKE 配置失败，右图为配置成功

最后一步，也是最重要的一步：编译保证树莓派有至少 5G 的存储空间，建议本命令用树莓派桌面上的命令行工具运行，而不要使用远程 ssh 连接。因为执行命令时间太长，中途如果 ssh 断线的话无法得知是否已经安装完毕。

```

cd /home/pi/Downloads/opencv-3.4.0/build
make
make clean
make
sudo make install

```

最后一步，也是最重要的一步：编译保证树莓派有至少 5G 的存储空间，建议本命令用树莓派桌面上的命令行工具运行，而不要使用远程 ssh 连接。因为执行命令时间太长，中途如果 ssh 断线的话无法得知是否已经安装完毕。编译大概需要 4~5 个小时，所以，在此期间，树莓派要供电充足，不要运行其它任务，以免因为内存不够之类的问题而报错。

8) 测试 OpenCV 是否安装成功

安装好之后，可以在自带的命令行中通过 import 命令进行测试，若出现如下图 51 所示的提示则说明 OpenCV 安装成功：

```
[... Installing: /usr/local/share/opencv/samples/python/watershed.py
pi@raspberrypi:~/Downloads/opencv-3.4.0/build $ cd
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> ]
```

图 51 命令行中测试安装 OpenCV 成功

- DLIB 库的安装经验

Dlib 库主要是用于载入一个开源的人脸 68 关键点检测器模型，并通过其中封装好的函数将这些点的坐标值取出，以进行后续的处理。

对于 Dlib 的安装，相当痛苦，安装的时候基本上到 95% 树莓派就死掉了，这往往是因为在编译 Dlib 库的时候需要的内存过大，会导致系统杀死进程。虽然树莓派 4B 有 2GB 的 RAM。但其中有相当一部分用于所有的系统操作，显示 GUI / 桌面以及处理我们的编译。因此，需要使用几步来解决这个问题。（原文可以参考：<https://www.pyimagesearch.com/2017/05/01/install-dlib-raspberry-pi/>）原文中是在 python 虚拟环境中安装 dlib 和相关依赖，我的实际环境不需要。

安装前还有一些前置操作：

- 1) 扩大虚拟内存

通过 \$ sudo nano /etc/dphys-swapfile 命令修改虚拟内存配置文件：

将 CONF_SWAPSIZE=100 改为 CONF_SWAPSIZE=1024，即把虚拟内存从 100M 改为 1G，然后重启虚拟内存相关服务：

```
$ sudo /etc/init.d/dphys-swapfile stop
$ sudo /etc/init.d/dphys-swapfile start
```

- 2) 设置启动后进入命令终端而不是图形界面

通过 \$ sudo raspi-config 进入配置界面，再按以下路径选择选项并修改：

Boot Options => Desktop / CLI => Console Autologin

- 3) 设置 GPU 占用内存的大小

通过 \$ sudo raspi-config 进入配置界面，再按以下路径选择选项并修改：

Advanced Options => Memory Split

将 GPU 可用内存改为 16。作用就是把更多的内存给 CPU 用。

- 4) 安装依赖库

Dlib 需要①Boost、②Boost.Python、③CMake、④X11 这四个库的依赖，由于在安装 OpenCV 时已经安装了 CMake，所以此处可以不单独安装，但方便且保险起见，命令中仍

可以体现，系统会自动跳过已安装的 CMake，使用如下的命令自动安装：

```
$ sudo apt-get update  
$ sudo apt-get install build-essential cmake libgtk-3-dev libboost-all-dev -y
```

5) 用 pip3 安装其他 Dlib 库运行所依赖的库

```
$ pip3 install numpy  
$ pip3 install scipy  
$ pip3 install scikit-image
```

6) 下载并解压官方的 Dlib 库

先到 <http://dlib.net/> 下载自己所需的版本的 dlib，自动安装到 /home/pi/Downloads 文件目录下，再在将其解压在同目录下，最后从命令行进入这个目录，使用如下命令安装：

```
$ sudo python3 setup.py install
```

这一步耗时较长，需要耐心等待。

7) 验证安装成功

连同前一步安装的 OpenCV，一起在树莓派 4B 自带的 Python IDE 中通过 import 的方式进行测试，若出现如下图 52 所示的反馈，则说明没有保存，成功 import，也就说明成功安装了！



```
Shell  
Python 3.7.3 (/usr/bin/python3)  
>>> import dlib  
>>> import cv2  
>>> |
```

图 52 命令行中测试安装 OpenCV 成功

8) 最后一步，别忘了把虚拟内存和 GPU 使用内存改回初始值，详见 2) 和 3)。

3.3.2.3 疲劳检测程序流程

通过摄像头获取的视频流数据进行疲劳检测的程序相当复杂，这里略去各种指标计算的方式（设计原理部分已经详细介绍），大致可分为七步：①初始化 DLIB 的人脸检测器（HOG），然后创建面部标志物预测；②图像帧预处理；③对于每个所检测到的人脸，获取左眼、右眼和嘴巴的关键点信息；④计算 EAR 和 MAR 值；⑤根据结果判断是否超过或低于阈值从而判断是否发生疲劳眨眼和打哈欠的行为；⑥计算头部姿态并根据 pitch 值在一段时间内的波动平均值来判断是否进行了一次瞌睡点头；⑦根据疲劳行为进行警报，并判断是否疲劳驾驶。更加细致、

具体的程序流程图如下图 53 所示：

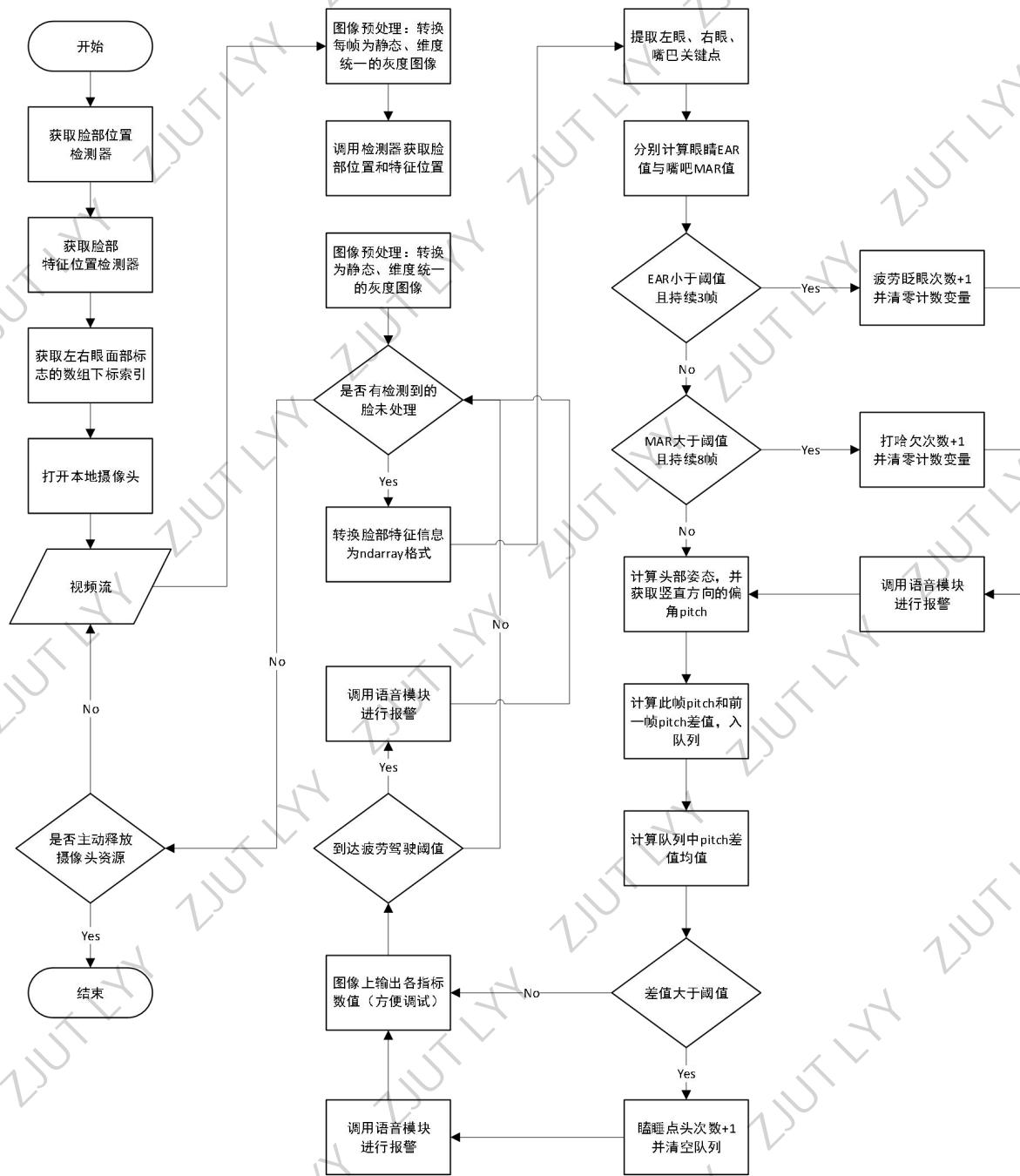


图 53 疲劳检测程序流程图

3.3.2.4 树莓派上疲劳检测实际效果

在这里必须说明的让我们较为失望的一点是：虽然摄像头经过测试确实是支持 30 帧的检测，但是由于疲劳检测程序的处理量较大，使得处理器无法完全利用摄像头的资源，在实际实验过程中发现帧率远远下降，仅剩 6~7 帧。相对庆幸的是这对我们疲劳检测的程序有影响但不致命，可能是由于处理器资源不够将图像实时地显示出来，但对于数据的处理速率稍高于 6~7 帧，勉强支持后续的疲劳检测操作。

首先展示的是通过摄像头获取视频信息后，经人脸定位和关键点定位之后，重新反馈在原图像上标定出 68 关键点的结果如下图 54 所示，

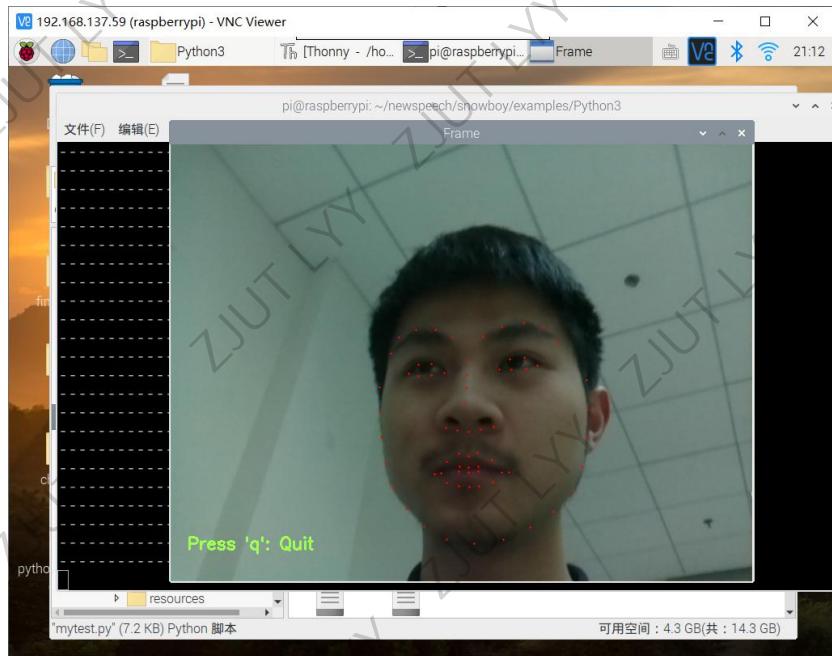


图 54 标定 68 关键点图

从上图不难看出，整体的特征点标定地非常符合被试的脸型，可以认为这个模型效果极佳，可以支持我们后续的检测。

下图 55 展示的是检测到疲劳眨眼之后的信息输出 “Tired Wink!”：

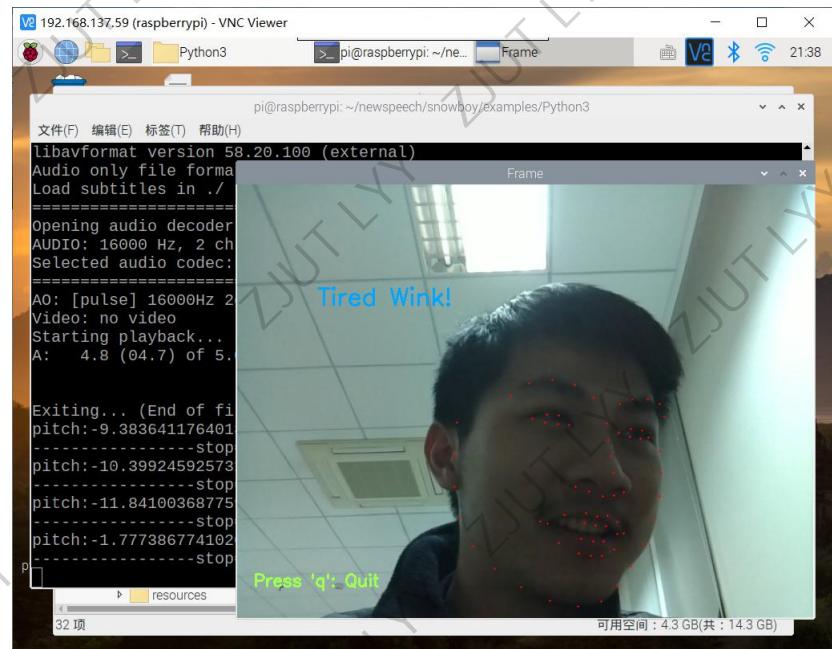


图 55 检测到疲劳眨眼的提示信息

这里需要说明的是提示信息文本框的显示为了让我们能看清楚是做了延时处理的，所以虽然图上看上去人眼睁着，但是之前是确实发生了一次疲劳眨眼的；加上疲劳眨眼确实是要检测

到眼睛睁开才能说明这是一次“眨眼”的，所以现象与描述并不矛盾。

下图 56 则是检测到打哈欠之后的信息提示 (Yawning!):

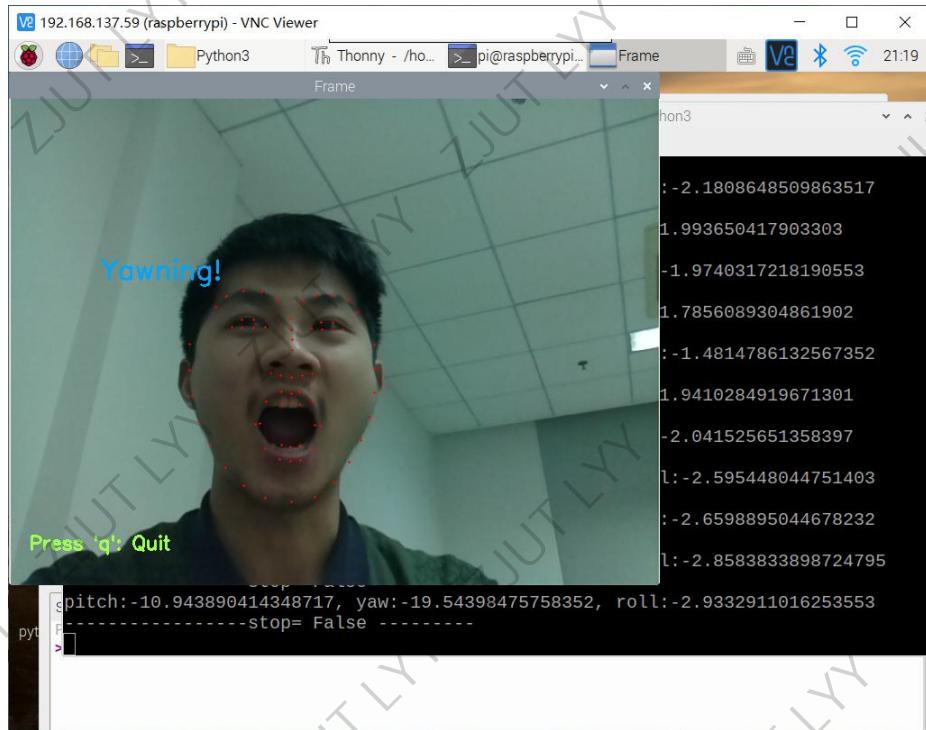


图 56 检测到打哈欠的提示信息

下图 57 则是检测到瞌睡点头之后的信息提示 (Nodding!):

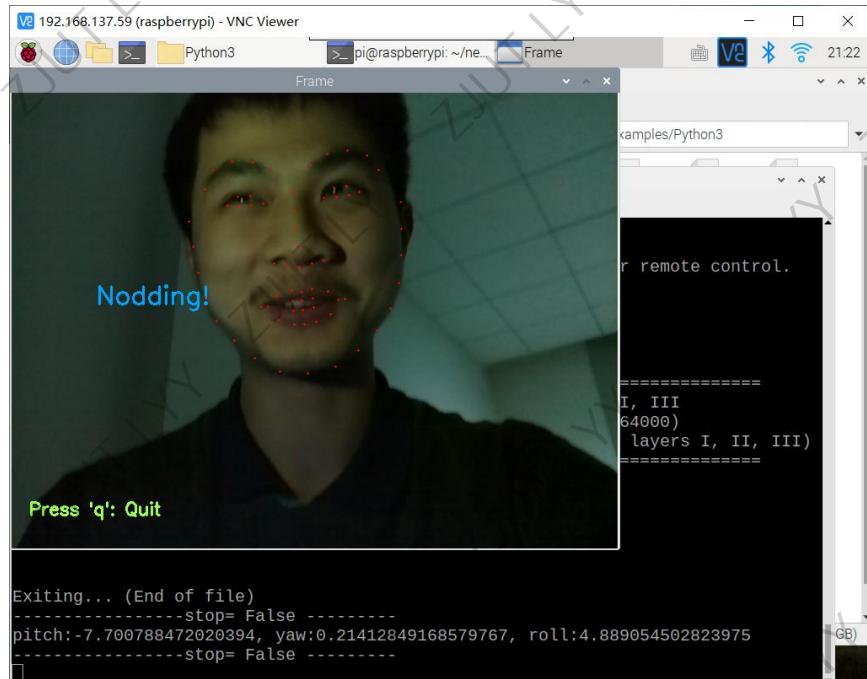


图 57 检测到瞌睡点头的提示信息

在对疲劳驾驶状态的推定时，用的是 20 次疲劳眨眼或 5 次打哈欠或 3 次瞌睡点头，这三个

条件满足其中一个之后就会出现下图 58 所示的提示信息：

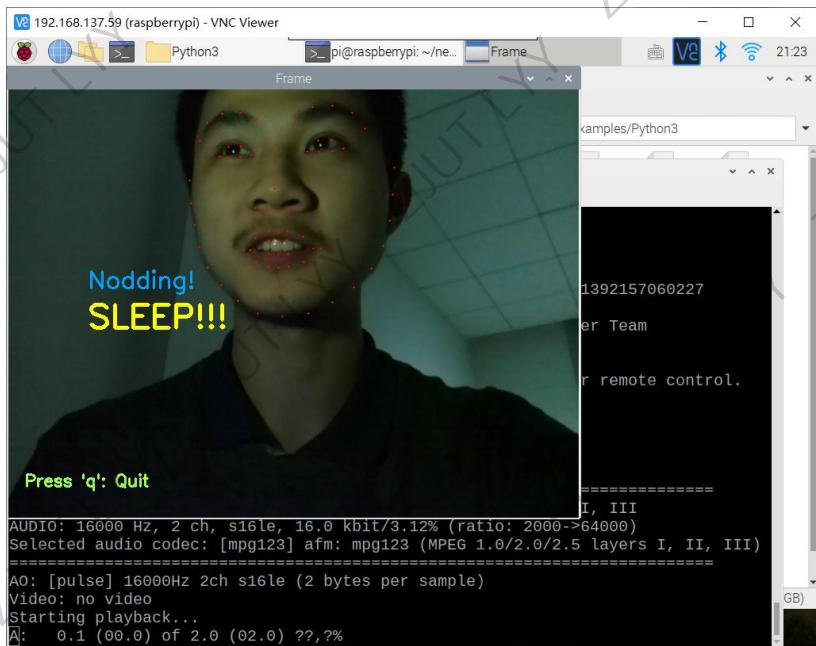


图 58 确认疲劳驾驶后的提示信息

3.3.3 语音模块的实现

语音模块的实现，我们将从语音唤醒、开发 SDK 的下载使用、语音识别/语音合成、录音功能、时间与天气查询功能、翻译功能和闲聊功能这七方面展开叙述，由浅入深，循循善诱，并给出实际实现过程中的实例与结果来增强这部分文档的可理解性。

3.3.3.1 语音唤醒

语音唤醒算是语音识别领域里最基础的应用，具体的场景如 Android 手机里的“OK, Google”或者苹果设备里的“Hey, Siri”。

简单来说就是在后台静默地运行着一个占用较少系统资源的服务（语音识别组件），该组件一直处于监视麦克风输入的状态，如果有检测到特定的语音输入（即唤醒词），则激活与之绑定的某个程序“开关”。语音唤醒一般只对某一个特定的词汇进行响应，识别后也只完成某一件指定的任务。

我们小组使用的语音唤醒引擎是“**Snowboy**”，它是一款轻量级的语音唤醒引擎，也就是说在后台静默运行时的占用内存量是非常少的，十分适合我们基于树莓派的开发；它的另一个最大的有点就是完全开源，是我们的开发成本、难度大大下降，它还有以下的一些特性：

- 高度可定制性。可自由创建和训练属于自己的唤醒词；
- 始终倾听。可离线使用，无需联网，保护隐私。精确度高，低延迟；
- 轻量可嵌入。耗费资源非常低（单核 700MHz 树莓派只占用 10% CPU）；

- 开源跨平台。开放源代码，支持多种操作系统和硬件平台，可绑定多种编程语言。

下面介绍在树莓派中配置 snowboy 语音唤醒功能的办法。

1) 硬件连接：

硬件连接相对简单，将麦克风和音箱连接到树莓派上，并测试麦克风和音箱，麦克风采用 USB 方式连接，音箱通过 USB 插口供电，3.5mm 插头进行数据传输。有线的连接方式非常简单，不需要多余的配置即可启用硬件模块。

2) 语音环境配置：

首先使用如下指令更新 pip3:

```
1. pip3 install --upgrade pip
```

接着按如下的命令安装依赖环境:

```
1. sudo apt-get install python3-pyaudio
2. sudo apt-get install python3-pyaudio sox
3. sudo apt-get install swig
4. sudo apt-get install libatlas-base-dev
```

通过如下命令测试音频的录制:

```
1. arecord temp.wav
```

若可以得到下图 60 所示的结果，则说明录音模块已经可以正常使用:

```
pi@raspberrypi:~ $ arecord temp.wav
Recording WAVE 'temp.wav' : Unsigned 8 bit, Rate 8000 Hz, Mono
```

图 60 录音模块正常结果图

因为没有指定参数，因此冒号后出现的是默认的参数。可以通过 arecord -h 命令来查看相关参数的含义。如下图 61 所示:

```
pi@raspberrypi:~ $ arecord -h
Usage: arecord [OPTION]... [FILE]...

-h, --help           help
--version          print current version
-l, --list-devices    list all soundcards and digital audio devices
-L, --list-pcms      list device names
-D, --device=NAME    select PCM by name
-q, --quiet         quiet mode
-t, --file-type TYPE file type (voc, wav, raw or au)
-c, --channels=#    channels
-f, --format=FORMAT sample format (case insensitive)
-r, --rate=#        sample rate
-d, --duration=#   interrupt after # seconds
-s, --samples=#    interrupt after # samples per channel
-M, --mmap          mmap stream
```

图 61 arecord 录音命令的参数一览

参数很多，不止图中这些，读者可以自行查看。其中比较常用的为-d 设置时间，

-r 设置采样率， -c 设置通道数， -t 设置文件类型。

接下来是根据所选 API 进行参数调整的过程。因为本小组打算使用百度语音来实现语音识别，因此需要符合百度 api 的规则：

原始 PCM 的录音参数必须符合 16k、8k 采样率、16bit 位深、单声道，支持的格式有：pcm（不压缩）、wav（不压缩，pcm 编码）、amr（压缩格式）因此，本小组设置参数的命令如下：

```
1. arecord -d 5 -r 16000 -c 1 -t wav -f S16_LE mysound.wav
```

如果没有设置-d 参数，可以输入 `ctrl+c` 停止录音。

通过 `aplay` 命令可以播放刚刚录制的音频。

```
1. aplay test.wav
```

如果播放成功代表配置完成。

接着由于本小组实验的语音合成功能基于百度语音，合成的音频格式为.mp3，因此还需要实现对 MP3 格式的音频文件的播放。`mplay` 命令可以播放 MP3 文件。下面两条命令分别是安装支持 mp3 格式音频文件播放的 `mplayer` 库与测试播放：

```
1. apt-get install mplayer
```

```
2. mplayer ***.mp3
```

3) 唤醒词 `snowboy`

使用以下命令下载 git，并且获取 `snowboy` 源代码：

```
1. sudo apt git
```

```
2. git clone https://github.com/Kitt-AI/snowboy.git
```

进入目录 `snowboy-master/resources/models/`，可以看到几个扩展名为.umd1 的文件，这些就是唤醒词模型，调用不同的模型，只要喊出对应的唤醒词就可以激活程序做出设计好的动作。

进入目录 `snowboy/examples/Python3/`，输入：

```
1. python3 demo.py resources/models/snowboy.umd1
```

这时候，会报错，因为官方代码有一个问题。打开 `snowboydecoder.py` 文件，将头文件中的 `from * import snowboydetect`，改为 `import snowboydetect`，再次输入上述命令。如果出现下图 62 所示界面：

```
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
```

图 62 语音唤醒就绪状态下提示信息

并且这时候，对着麦克风说出“snowboy”，能听到“Ding”的声音，那么表示 `snowboy` 配置成功。

关于唤醒词的定制，比较可惜的一点在于，由于 KITT.AI 在 2020 年 12 月 31 日（我们做语音功能的前一天，哭了）关闭了 snowboy 服务，因此唤醒词无法定制为我们的产品名——车宝。自己定制唤醒词的还有一点问题在于，由于自己定制的唤醒词的只提供三个样本，因此制作出来的唤醒词经常出现识别率低的问题；而官方提供的唤醒词的采集了成千上万个人的样本，因此识别率很高。综上这些因素，本小组打算直接采用官方提供的 snowboy 唤醒词作为本次课设产品的唤醒词。下图 63 为官方关闭唤醒词定制的发文：

Dear KITT.AI users,

We are writing this update to let you know that we plan to shut down all KITT.AI products (Snowboy, NLU and Chatflow) by Dec. 31st, 2020.

we launched our first product Snowboy in 2016, and then NLU and Chatflow later that year. Since then, we have served more than 85,000 developers, worldwide, across all our products. It has been 4 extraordinary years in our life, and we appreciate the opportunity to be able to serve the community.

The field of artificial intelligence is moving rapidly. As much as we like our products, we still see that they are getting outdated and are becoming difficult to maintain. All official websites/APIs for our products will be taken down by Dec. 31st, 2020. Our github repositories will remain open, but only community support will be available from this point beyond.

Thank you all, and goodbye!

The KITT.AI Team
Mar. 18th, 2020

图 63 关闭唤醒词定制的通知

3.3.3.2 开发 SDK 的下载使用

在进行相关的功能开发的时候使用的接口涉及到了百度和腾讯两家公司，需要各自下载相对应的 SDK，同时需要选择语言版本，我们下载的是 Python-SDK。以腾讯云的 Python SDK 为例，简要介绍以下 SDK 的依赖环境和安装。

1) 依赖环境

- 腾讯云的 Python-SDK 需要 Python 的版本为 2.7, 3.6 至 3.9 版本。
- 获取安全凭证。安全凭证包含 SecretId 及 SecretKey 两部分。SecretId 用于标识 API 调用者的身份，SecretKey 用于加密签名字串和服务器端验证签名字串的密钥。需要前往 API 密钥管理页面获取。获取许可结果如下图 64 所示

新建密钥						
APPID	密钥	创建时间	上次访问时间	上次访问服务	状态	操作
1304767554	SecretId: AKID02d1CHbsLh8MTKwEQSlvJPFqDUbTd1fF SecretKey: *****显示	2021-01-14 14:53...	-	-	已启用	禁用

图 64 成功获取安全凭证

- 获取调用地址。调用地址一般形式为*.tencentcloudapi.com，不同产品的调用地址有一定区别，支持就近地域接入，例如，机器翻译的就近地域接入域名为

tmt.tencentcloudapi.com，也支持指定地域域名访问，例如广州地域的域名为 tmt.ap-guangzhou.tencentcloudapi.com。具体调用地址可参考对应产品的 API 文档。

2) 安装 SDK

可通过 pip 安装的方式来安装腾讯云的 Python SDK，如下命令即可：

```
1. pip3 install tencentcloud-sdk-python
```

需要注意的是，由于树莓派 4B 官方镜像同时具备 python2 及 python3 环境，所以需要使用 pip3 命令进行安装。

中国大陆地区的用户可以使用国内镜像源提高下载速度。如果仅使用国内源，则可使用 -i 参数指定。示例如下：

```
1. # 以腾讯源为例  
2. # 源地址后为所要安装的包: tencentcloud-sdk-python  
3. pip3 install -i https://mirrors.tencent.com/pypi/simple/ --upgrade tencentcloud-sdk-python
```

3.3.3.3 语音识别/语音合成

语音合成是采用了调用百度 API 的方式，首先需要在百度智能云注册账号，在“人工智能”这一栏中选择“语音技术”这一项，如下图 65 所示：



图 65 注册百度智能云账号

创建两个语音应用，分别用作语音识别和语音合成，并查看自己的获得的 AppID, API Key, Secret Key。我们小组的对应信息如下图 66 所示：

应用列表						
	应用名称	AppID	API Key	Secret Key	创建时间	操作
1	语音合成	23476319	6czQuWV4tavXgX0Gw2EHH4v5	***** 显示	2021-01-01 22:30:33	报表 管理 删除
2	语音识别	23476240	XXH3nkO18oDqreZTuvlhvc3	***** 显示	2021-01-01 21:55:00	报表 管理 删除

图 66 AppID、API Key 和 Secret Key 信息

创建完应用后，需要下载 SDK，本小组采用的 python 编程，因此下载了 python-SDK。之后就可以开始相应的开发了。

语音识别可以将 60 秒以下的音频识别为文字。适用于语音对话、语音控制、语音输入等场景。

- 接口类型：通过 REST API 的方式提供的通用的 HTTP 接口。特点是适用于任意操作系统，任意编程语言
- 接口限制：需要上传完整的录音文件，录音文件时长不超过 60 秒。浏览器由于无法跨域请求百度语音服务器的域名，因此无法直接调用 API 接口。
- 支持音频格式：pcm、wav（我们采用的格式）、amr、m4a
- 音频编码要求：采样率 16000、8000，16 bit 位深，单声道

语音合成基于 HTTP 请求的 REST API 接口，将文本转换为可以播放的音频文件。

- 合成的文件格式：mp3, pcm (8k 及 16k), wav (16k)
- 多音字可以通过标注自行定义发音。格式如：重(chong2)报集团。
- 目前只有中英文混合这一种语言，优先中文发音。

下面以语音合成为例，介绍一下调用流程：

- 1) 创建账号及应用：在控制台中，创建应用，勾选开通“语音技术”-短语音识别、短语音识别极速版“能力。获取 AppID、API Key、Secret Key，并通过请求鉴权接口换取 token。
- 2) 创建识别请求：POST 方式，音频可通过 JSON 和 RAW 两种方式提交。JSON 方式音频数据由于 base64 编码，数据会增大 1/3。
- 3) 短语音识别请求地址：http://vop.baidu.com/server_api
- 4) 返回识别结果：识别结果会即刻返回，采用 JSON 格式封装，如果识别成功，识别结果放在 JSON 的“result”字段中，统一采用 utf-8 方式编码。

语音识别的主要流程伪代码如下：

```
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)
result = client.asr(语音文件, 语音格式, 采样频率, 语种)
if 返回值 result 中的字段为 success:
    将数据转换为 utf-8 格式
    if 转换后的数据不为空
        将数据写入本地文件
    else:
        报错文件不存在
else:
```

语音合成的主要流程伪代码：

```
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)
result = client.synthesis(待合成内容, 语种)
if not isinstance(result, dict):
    保存数据
else:
    报错
```

其中的 **AipSpeech** 是语音识别和语音合成的 Python SDK 客户端，提供了一系列的交互方法。

常量 APP_ID 在百度云控制台中创建，常量 API_KEY 与 SECRET_KEY 是在创建完毕应用后，系统分配给用户的，均为字符串，用于标识用户，为访问做签名验证，可在 AI 服务控制台中的应用列表中查看。通过账号信息创建 **AipSpeech** 对象，再通过 **AipSpeech** 对象调用接口获得数据，最后对获得的数据进行基本判断。

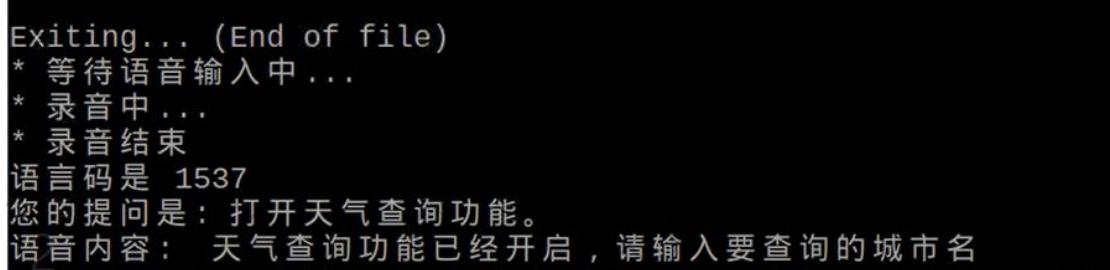
下面给出一些具体的结果界面的截图。

将语音合成和语音唤醒功能结合在一起可以实现树莓派在被唤醒后的语音提醒，树莓派在被语音唤醒后会生成一条唤醒记录，唤醒后可以将设置好的文本应答信息转换成语音数据进行输出。如下图 67 所示：

```
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
INFO:snowboy:Keyword 1 detected at time: 2021-01-16 13:14:34
-----
语音内容： 您好，请问有什么可以帮到您？
MPlayer 1.3.0 (Debian), built with gcc-8 (c) 2000-2016 MPlayer Team
```

图 67 语音合成后输出结果

同时，通过语音识别和语音合成可以实现和树莓派的语音交互功能。首先树莓派接收操作者发出的语音命令，存到本地文件中，通过语音识别功能将语音文件中的内容提取成文本数据，在对文本数据进行检测后，可以调用不同的功能，并且按照功能的不同回复不同的文本信息，最后将生成的应答文本信息通过语音合成功能转换成 MP3 文件，将 MP3 文件播放出来就可以实现完整的人机语音交互。如下图 68 所示：



```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：打开天气查询功能。
语音内容：天气查询功能已经开启，请输入要查询的城市名
```

图 68 语音交互结果

功能的具体代码见附件 speech_recognition.py(语音识别)和 speech_synthesis.py(语音合成)。

3.3.3.4 录音功能

录音功能可以按照指定的时间进行录音，这是最简单的录制要求，但是在正常人机交流中，操作者说出的话通常持续时间长短不同，因此固定的录音时间并不能满足我们的要求。我们需要在操作者说话的时候，智能地判断是否说完，以此确定结束录音的时间。

最简单的方法就是设置一个音量阈值和时间阈值，检测一段固定时间内（如 1 秒内），如果未超过音量阈值（如 20 分贝）的持续时间百分比超过了时间阈值（如 60%），就判定当前操作者已经结束说话。

此外，考虑到操作者在开始说话这一时间点的不确定性，在未开始说话时，未超过音量阈值的持续时间百分比也会超过时间阈值，因此在监听过程中还需要有一个开始说话的判断。在打开录音功能时，操作者的录入状态往往是未开始说话->说话->结束说话，因此在录音时需要对这三个状态的转换的中间两个转换点进行判断。

相应的伪代码如下：

设置录音参数

```
bool_开始说话 = False  
num_在说话 = 0  
num_未说话 = 0  
  
采集 1 秒的声音数据:  
if 音量 > 声音阈值:  
    num_在说话 += 1  
else:  
    num_未说话 += 1  
if num_未说话 / (num_在说话 + num_未说话) > 时间阈值:  
    if bool_开始说话 == True:  
        说话结束, 结束录音  
    else:  
        if bool_开始说话 == False:  
            bool_开始说话 = True
```

下面是实验过程中的具体界面截图（开始时未检测到人声，处于等待语音输入的状态，检测到人声后进入录音状态，最后检测到人声消失，判断录音已结束，退出录音），如下图 69 所示：

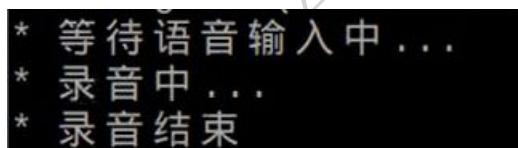


图 69 录音成功的结果

具体代码见附录 luyin.py

3.3.3.5 时间、天气查询功能

当语音识别后的文本信息中出现关键词“时间”，“几点”，车宝就会汇报当前的时间，时间的查询可以直接读取系统的时间，再进行格式化输出。格式化输出语句如下所示：

```
1. time.strftime("%Y 年%m 月%d 日%H 点%M 分", time.localtime())
```

语音功能还可以开启天气查询功能，在天气查询功能开启后，再通过读入城市名可以查询出目标城市当天和第二天两天的总体天气，最低气温，最高气温，风向和风力。

通过调用中国天气网的天气查询接口，只需要改变 url 最后的城市名称就可以返回目标城市

未来几天的天气，本小组取了查询当天以及第二天的部分数据。

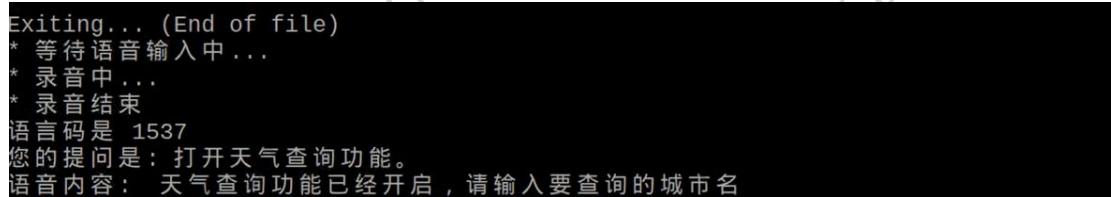
1. http://wthrcdn.etouch.cn/weather_mini?city=城市名

将返回的数据转换为 json 类型就可以非常方便地取出想要的数据。

下面是一些具体的界面截图结果：

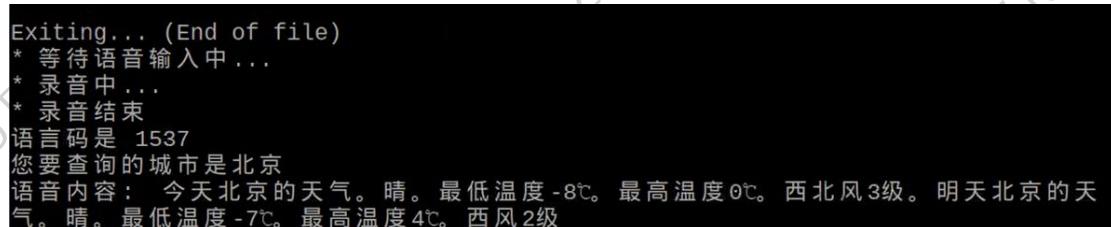
首先需要打开天气查询功能，之后再读入要查询的城市名，将返回的查询结果进行格式转化成字符串信息，调用语音合成功能就可以语音播报要查询城市的当天天气和第二天的天气。

分别如下图 70 和图 71 所示：



```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：打开天气查询功能。
语音内容：天气查询功能已经开启，请输入要查询的城市名
```

图 70 成功打开天气查询功能



```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您要查询的城市是北京
语音内容：今天北京的天气。晴。最低温度 -8℃。最高温度 0℃。西北风 3 级。明天北京的天气。晴。最低温度 -7℃。最高温度 4℃。西风 2 级
```

图 71 某城市天气查询成功的结果

3.3.3.6 翻译功能

我们的车宝系统还提供了中英翻译功能，其实不限于中英翻译，但是由于语音识别和语音合成无法较好的完成除了中英文之外的语种的识别和合成，因此使得语音翻译功能受限。本小组使用的翻译功能是通过调取腾讯的机器翻译中的文本翻译 api 实现的，需要提前在腾讯智能云的控制台进行申请。

腾讯的机器翻译支持 5 种翻译类型，其中有直接的语音翻译功能，但是为了更好的测试使用另外的功能，我们采用了文本翻译这种使用更为方便的翻译功能。如下图 72 所示：

API 接口	
接口名称	接口功能
ImageTranslate	图片翻译
LanguageDetect	语种识别
SpeechTranslate	语音翻译
TextTranslate	文本翻译
TextTranslateBatch	批量文本翻译

图 72 腾讯机器翻译 API 及功能

因为翻译接口的调用需要确定源语言和目标语言。因此在每次开启翻译功能时，主动指定翻译的源语言以及目标语言，当然，也可以设置默认的翻译方式。在开启翻译功能后，再读入想要翻译的内容，通过翻译接口的调用，即可获得对应的语言。

在确定源语言和目标语言时，通过查找通过语音识别后的文本信息里是否符合以下两种格式：

1. **part1**到**part2**翻译**part3**
2. **part1**翻译成**part2**

在符合当前格式后，在 part1 中查找源语言，在 part2 中查找目标语言。如果能查找到，就可以成功开启翻译功能。接下来再接收待翻译的一句源语言的语音。

如果想要退出翻译功能，只需要说出包含“退出”、“翻译”这两个关键词的语音消息即可。因为在执行翻译功能前会优先查询这两个关键词。

一些实验中具体的功能界面截图如下图 73 和图 74 所示：

```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：打开中文到英文的翻译功能。
源语言是中文
目标语言是英文
语音内容： 中文翻译英文的功能已开启，请输入要翻译的中文对话
```

图 73 成功打开中文翻译英文的功能

```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您要翻译的句子是今天的太阳很大，天气很好。
语音内容： 这句话的英文翻译是It's sunny today and the weather is fine.
MPlayer 1.3.0 (Debian), built with gcc-8 (c) 2000-2016 MPlayer Team
```

图 74 成功对所输入中文语句进行翻译为英文

将返回的翻译结果前加上“您要翻译的句子是”，在通过语音合成功能转化成语音文件进行播放就可以听到翻译结果，语音合成功能支持中英文混合的文本信息。

详细代码见附录 translation.py

3.3.3.7 闲聊功能

在日常驾车过程中，驾驶员通常会感到无聊，因此我们的车宝系统还提供了闲聊功能，通过语音控制打开闲聊功能后，驾驶员即可和车宝系统开始一问一答式的聊天功能。

闲聊功能主要通过调用腾讯的腾讯智能对话平台 TBP 来实现，其使用流程主要包括以下五个步骤：

1) 申请使用:

访问 腾讯云官网:【产品】>【人工智能】>【AI 平台服务】>【腾讯智能对话平台】，单击【立即申请】，登录腾讯云账号并填写申请表。

2) 创建和配置 Bot:

初次使用 TBP 首先需要新建一个 Bot，并对其进行配置。Bot 的配置主要包括意图和词典的配置，分别对应【意图管理】和【词典管理】模块。

3) 配置意图和词典:

进入 Bot 配置页面，需要新建和配置意图，意图的配置包括用户说法、槽位、服务实现以及机器人自动回复。在配置意图的槽位时需要对词典进行相关配置，包括新建自定义词典或引用内置词典，并在意图配置完成后为所需的自定义词典添加词条。

4) 测试和发布 Bot:

Bot 配置完成后必须先通过编译，才可以对 Bot 进行测试。若测试无误，便可在【发布管理】模块对调试版本进行发布上线，发布后的线上版本可用于实际使用。

5) 接入 Bot:

若 Bot 已存在线上版本，可在【应用接入】模块按照配置说明将 Bot 快速接入到多种应用，目前支持微信公众号和腾讯小微的快速接入。此外也可以通过 API 文档提供的接口进行接入。如下图 75 所示:

线上版本		
版本 ID	上线时间	操作
0001	2021-01-14 14:59:37	下线

图 75 线上版本的 Bot

我们采用了基础的配置，发布后采用 API 接入。

此外，通过腾讯的自然语言处理模块也可以实现闲聊功能，并且通过自然语言处理实现的闲聊功能还可以实现对聊天模式的选择，可以选择通用模式和儿童模式，儿童模式会更偏向儿童语言风格及说话方式，从而让聊天变得更睿智、简单和有趣。

- 儿童闲聊实例

输入实例如下：

```
https://nlp.tencentcloudapi.com/?Action=ChatBot  
&Flag=1  
&OpenId=no23391003  
&Query=给我讲个故事可以  
&<公共请求参数>
```

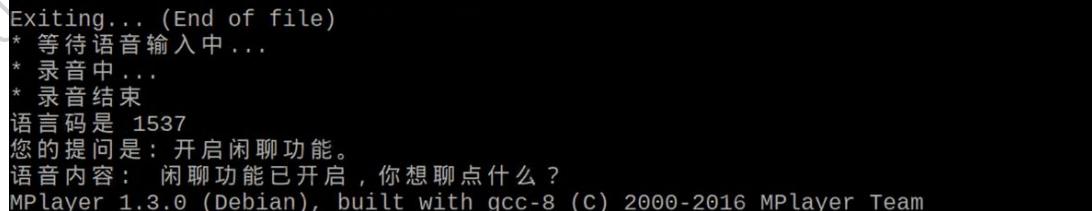
一个输出实例如下：

```
{  
    "Response": {  
        "Reply": "宝贝，最近好多小朋友都在听下面这些故事呢",  
        "Confidence": 1,  
        "RequestId": "eac6b301-a322-493a-8e36-83b295459397"  
    }  
}
```

需要注意的是，不管是腾讯智能对话平台 TBP 还是自然语言处理的闲聊功能都无法获得精确化的信息，比如无法获得当前时间，当前天气等精确回答，只能进行一些常识性的普通对话。

下面给出一些具体运行中的结果。

通过语音功能开启闲聊功能，检测关键词为“开启”或者“打开”，“闲聊”。同时也可退出闲聊功能，检测关键词为“退出”或者“关闭”，“闲聊”。一些结果如下图 76~79 所示：



```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：开启闲聊功能。
语音内容： 闲聊功能已开启，你想聊点什么？
MPlayer 1.3.0 (Debian), built with gcc-8 (C) 2000-2016 MPlayer Team
```

图 76 成功打开闲聊功能



```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
myword: 退出语音闲聊功能。
语音内容： 闲聊功能已退出
```

图 77 退出闲聊功能

```
* 录音中...
* 录音结束
语言码是 1537
myword: 运动时应该注意点什么？
1.应在半饱状态下进行各种运动。因为过饱会造成胃肠负担;空腹又易导致低血糖,出现意外。
2.要养成运动开始前的热身活动和运动结束后的缓冲活动,使心脏和关节、肌肉都能有个准备、适应过程。3.要选择柔软、合脚的鞋袜,避免因鞋子过大过小而造成足掌和腿部肌肉紧张,加剧活动疲劳度。4.在运动过程中,一旦因种种原因而造成不适,应立即中止运动,查找原因;千万不可硬撑着坚持,以免意外发生。5.运动过程中如果出了汗,一定要及时补充适量淡盐水;即使未出汗,也应及时补充些温开水,以帮助血液循环,同时有利于排除体内毒素。6.运动时一定要记得邀一两个伙伴同去,以防意外;也不可去太过偏僻的地方...
语音内容：1.应在半饱状态下进行各种运动。因为过饱会造成胃肠负担;空腹又易导致低血糖,出现意外。
2.要养成运动开始前的热身活动和运动结束后的缓冲活动,使心脏和关节、肌肉都能有个准备、适应过程。3.要选择柔软、合脚的鞋袜,避免因鞋子过大过小而造成足掌和腿部肌肉紧张,加剧活动疲劳度。4.在运动过程中,一旦因种种原因而造成不适,应立即中止运动,查找原因;千万不可硬撑着坚持,以免意外发生。5.运动过程中如果出了汗,一定要及时补充适量淡盐水;即使未出汗,也应及时补充些温开水,以帮助血液循环,同时有利于排除体内毒素。6.运动时一定要记得邀一两个伙伴同去,以防意外;也不可去太过偏僻的地方...
```

图 78 提问与回复结果 1

```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
myword: 四大名著是哪些？
四大名著公认为:吴承恩《西游记》 施耐庵《水浒传》 罗贯中《三国演义》 曹雪芹《
红楼梦》
语音内容：四大名著公认为:吴承恩《西游记》 施耐庵《水浒传》 罗贯中《三国演义》
曹雪芹《红楼梦》
MP3player 1.2.0 (Debian) built with gcc 4.6 (Debian) 20120620 MP3player Team
```

图 79 提问与回复结果 2

详细代码见附录 `get_response.py` (腾讯智能对话平台 TBP), `chat.py` (自然语言处理)。

3.3.4 传感器模块的实现

在传感器模块中, 主要分为以下三个功能: 车内环境温度检测、检测安全带系的清空和检测车外环境光线强度, 使用的模块分别是: DHT11、树莓派两个 GPIO 口、光敏电阻和 LED 灯。接下来将分别从温度检测功能的实现、智能灯控功能的实现和安全带未系警报功能的实现三部分进行传感器模块实现部分的叙述。

3.3.4.1 温度检测功能的实现

1) 所用模块介绍

温度检测模块主要运用了 DHT11 温湿度检测模块, DHT11 数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器, 它应用专用的数字模块采集技术和温湿度传感技术, 确保产品具有极高的可靠性和卓越的长期稳定性。传感器包括一个电阻式感湿元件和一个 NTC 测温元件, 并与一个高性能 8 位单片机相连接。其精度湿度 $+5\%$ RH, 温度 $+2^{\circ}\text{C}$, 量程湿度 20-90%RH, 温度 0~50°C, 其测温范围完全适用于系统开发环境, 因此我

们选择了 DHT11 温湿度传感器进行周边温度检测。实物如下图 80 所示：

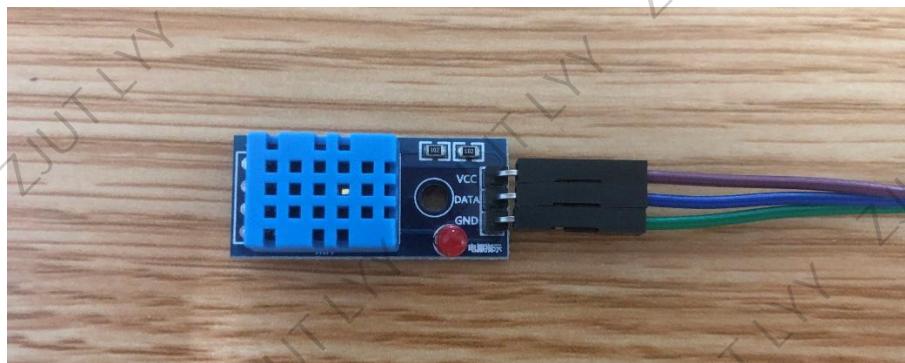


图 80 DHT11 实物图

如上图所示，温湿度传感器 DHT11 共有三个引脚，分别是“VCC”、“DATA”、“GND”，其中 VCC 引脚连接+3.3V~+5V 的电源，DATA 引脚连接树莓派上的 GPIO 接口，GND 引脚接地。

2) DHT11 测温功能实现

DHT11 的 DATA 引脚为数据输出的引脚。

首先，树莓派的 GPIO 引脚发送一次开始信号（即低电平）后，DHT11 从低功耗模式（即此时不通过 DOUT 把温湿度数据传出）转换到高速模式（即此时开始传输温湿度数据），等待主机开始信号结束后，DHT11 发送响应信号，送出 40bit 的数据，并触发一次信号采集，用户可选择读取部分数据。从模式下，DHT11 接收到开始信号触发一次温湿度采集，如果没有接收到主机发送开始信号，DHT11 不会主动进行温湿度采集。采集数据后转换到低速模式。

DHT11 开始进入高功耗模式，首先会发送一个高电平，以表示准备输出。接下来开始输出数据，每一 bit 数据都以 50us 低电平时隙开始，高电平的长短定了数据位是 0 还是 1。

数据采集完毕后，我们将其转换成十进制数据，DHT11 中需要采集温度和湿度两个数据，除了这两个数据外，DHT11 还会得到这两个数据的校验和，当且仅当两个校验和的数值相等时，模块才会判定此时得到的数据时正确的，而后输出正确的温度和湿度；如果校验和不相等，则会提示此次检测错误，并输出当次结果下的校验和值。

本组设计仅需采集当前周边环境温度，因此实际设计里删去了湿度的数据。考虑到数据的得出需要两个校验和的一致，因此我们采用的方法是读出正确的之后再提取其中的温度值。

在完成利用 DHT11 芯片读取温度值的代码后，将其封装成函数，与语音模块进行连接，利用语音进行功能唤醒，最后读出的数据也可利用语音读出。最终成功启动车内环境温度检测功能的交互结果图如下图 81 所示：

```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：检测当前周围环境温度。
正在检测中 -----
语音内容： 正在检测中
```

图 81 成功启动温度检测功能结果图

输入语音检测当前周围环境温度，进行语音检测后，成功调用 DHT11 温度查询模块，DHT11 接收到低电平信号立刻进行温度检测，得到正确值后将信息反馈回语音模块。可得下图 82 所示的检测结果：

```
Exiting... (End of file)
0
temp:22.4,hum:30.0
语音内容： 当前周围的环境温度为 22.4 摄氏度。
MPlayer 1.3.0 (Debian), built with gcc-8 (c) 2000-2016 MPlayer Team
do_connect: could not connect to socket
connect: No such file or directory
Failed to open LIRC support. You will not be able to use your remote control.
```

图 82 成功检测出环境温度

如上图所示，检测出当前的周围环境温度为 22.4 摄氏度（当时环境为开空调保温 20℃ 的专教），可见结果相对较为准确，同时将结果语音输出。

3) 调试说明：

在对 DHT11 芯片检测温度的模块实现中遇到了不少的问题。

问题一：首先是接线的问题，由于树莓派引脚有物理引脚 BOARD 编码、BCM 编码以及 wiringPi 编码三种编码方式，一开始对于这三种编码方式的了解过少，所以在选择引脚及其对应编码时走了一些弯路，之后选择了更加简洁方便的 BCM 编码方式，不过需要注意的是，在代码中一定要指明编码方式，否则会出现连接不上芯片得不到数据的问题。以 BCM 编码格式进行编码的代码如下所示：

```
GPIO.setmode(GPIO.BCM)
```

问题二：DHT11 芯片虽然能够满足检测环境的要求，但是其检测效率不高，平均需要检测 10 次左右才能输出正确结果。对于这一问题，最简单的方法是换一个更加精确稳定的温度传感器，但由于时间问题，我们不得不对代码进行优化，使 DHT11 的这一缺点带来的时间延迟尽可能地小。我们在代码中从原本输入一次命令检测一次温度改进为输入一次命令后一直检测直至检测出正确数值输出后该次命令结束，极大地减少了这一问题带来的影响。

3.3.4.2 智能灯控模块的实现

1) 所用模块介绍

智能灯控功能所需实现的功能是自动检测车外光照强度，如果检测到光照强度低于某一阈值时，自动打开灯光；若检测到光照强度重新高于该阈值时，自动关闭灯光。

因此在该功能中，我们所需使用的模块是光敏电阻传感器以及 LED 灯。光敏电阻传感器实物图如下图 83 所示：

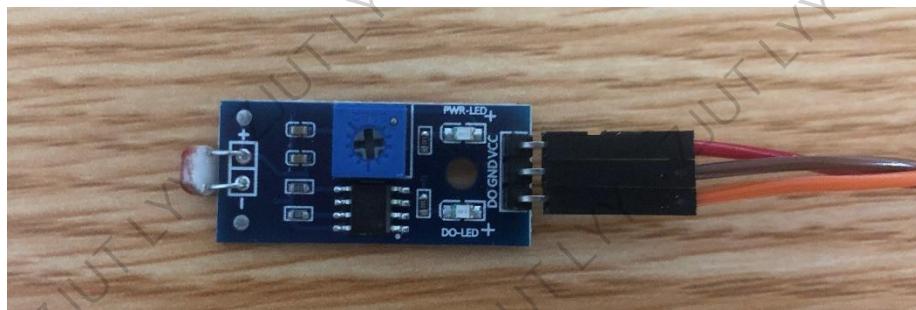


图 83 光敏电阻实物图

光敏电阻传感器共有三个引脚，分别是“VCC”、“DO”、“GND”，其中 VCC 引脚连接 +3.3V~+5V 的电源，DO 引脚连接树莓派上的 GPIO 接口，GND 引脚接地。光敏电阻最左侧为光照强度监测点，蓝色方块处可通过旋转十字口改变光敏电阻传感器的灵敏度，也就是改变决定灯光开启关闭的阈值。

LED 灯如下图 84 所示：



图 84 LED 灯实物图

LED 灯共有两个引脚，一个引脚为正极，一个引脚为负极可以将其用杜邦线与树莓派的引脚进行相连。

2) 智能灯控功能实现

光敏电阻传感器是通过把光强度的变化转换成电信号的变化来实现控制的，它的基本结构包括光源、光学通路和光电元件三部分，它首先把被测量的变化转换成光信号的变化，然后借助光电元件进一步将光信号转换成电信号。光敏电阻传感器的数据从 DO 引脚输出，其数据输出为 0 或 1。光敏电阻模块对环境光强最敏感，一般用来检测周围环境的亮度和光

强。模块在无光条件或者光强达不到设定阈值时，DO 口输出高电平，当外界环境光强超过设定阈值时，模块 DO 输出低电平。DO 输出可以直接驱动 LED 模块，由此组成一个光电开关。光敏电阻传感器原理图如下图 85 所示：

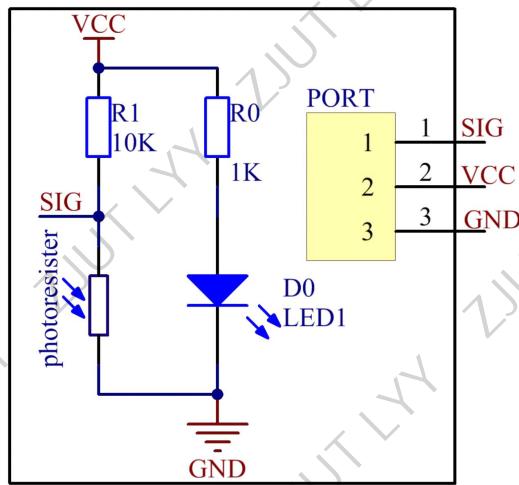


图 85 LED 灯实物图

随着光强度的增加，光敏电阻的电阻将降低，因此输出电压降低，超过一定阈值输出低电平；相反，当光强度减弱时，光敏电阻的电阻将升高，因此输出点么升高，超过一定阈值输出高电平。

LED 模块一端接 GPIO 口，另一端接地。当周围光强不够时，DO 口的数值输出至 LED 所连的 GPIO 口，输出高电平，使 LED 灯发光，而当周围光照强度足够时，DO 口输出低电平，此时并不能使 LED 灯点亮，因此此时 LED 灯不亮。

实际应用效果如下图 86 所示：

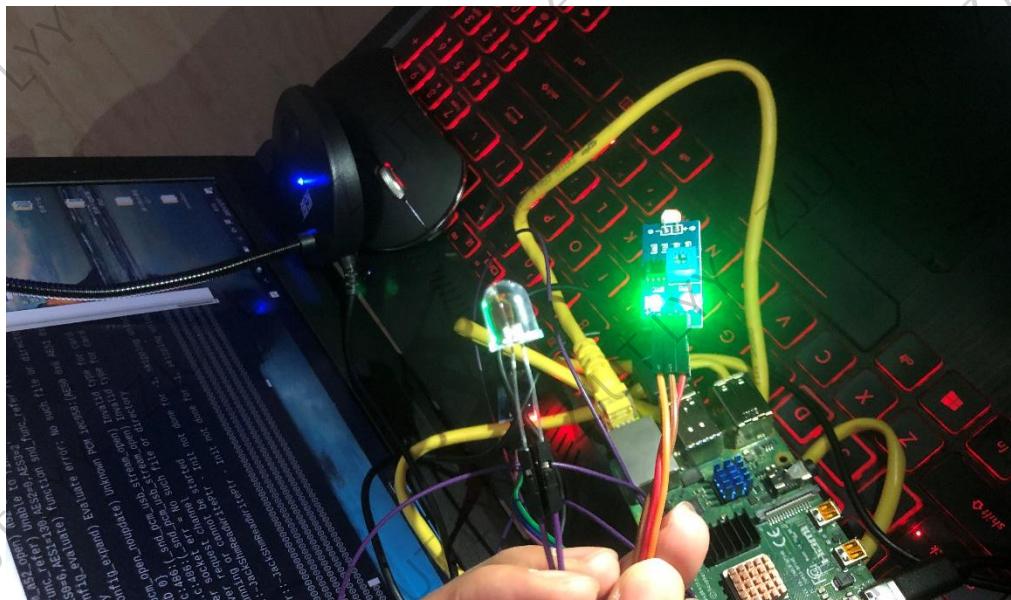


图 86 光照较强时 LED 结果展示（不开灯）

光敏电阻传感器有两个 LED 指示灯。PWR-LED 是电源指示灯，DO-LED 是数字输出指示灯，此时光照强度较高，两盏 LED 指示灯均亮起，此时 DO 口输出低电平，LED 灯不亮。光照强度较弱时 LED 灯结果如下图 87 所示：

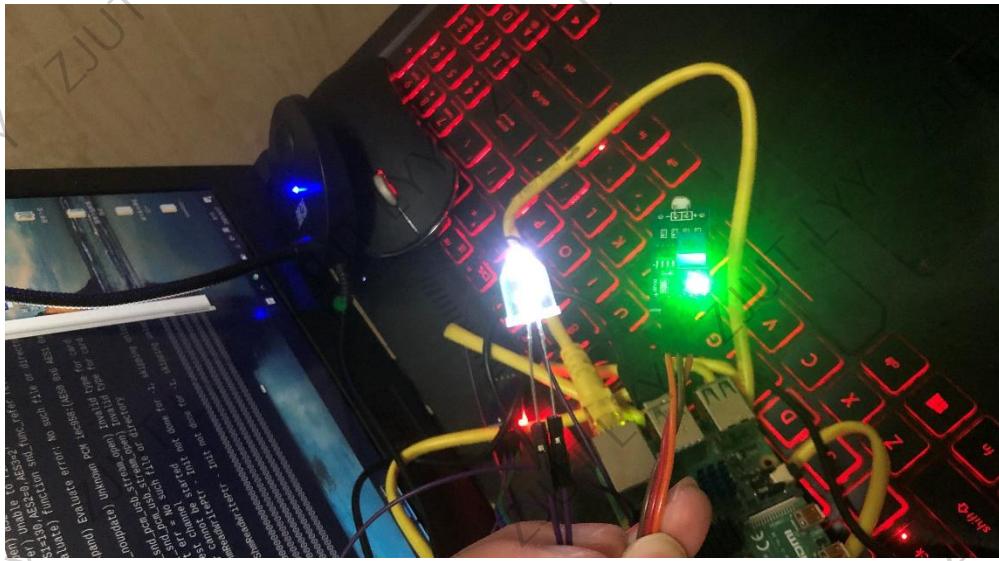


图 87 光照较弱时 LED 结果展示（开灯）

由上图可以看出，此时环境光照强度较弱，电源指示灯 PWR-LED 亮，数字输出指示灯 DO-LED 灭，此时 DO 输出高电平，LED 亮起。

3) 调试说明：

在有了 DHT11 温湿度传感器的经验之后，光敏电阻传感器的测试就顺利了很多，唯一需要多次尝试的便是光照强度的阈值，我们需要调到最适合的灵敏度以达到最好的效果。还有一点需要注意的说，此模块用到了两个 GPIO 口，因此在输入输出时应该要更加小心对照，以免出错。

3.3.4.3 安全带未系警报功能的实现

1) 所用模块介绍

安全带未系警报模块是目前车辆的基本功能，我们团队做出了一些创新，将其与语音模块相连，有语音进行安全带未系提示。因此在该模块中，硬件方面仅需一条杜邦线来演示系上安全带这一动作即可。

2) 功能实现

我们选择使用两个 GPIO 口来对安全带状态进行检测，其中一个 GPIO 口为监测口，另一个 GPIO 口模拟安全带状态口。杜邦线的一头固定在监测口，若杜邦线的另一端未插入模拟状态口，则认为此时安全带未系，相反，若杜邦线的另一端也接入模拟状态口，此时安全带处于系好的状态。

为了实现监测功能，我们设置监测口处电流最初为高电平，模拟状态口处固定为低电

平。在安全带未系的情况下，监测口始终为高电平不变，此时将会一直提示车主系上安全带；如若安全带系上，模拟状态口的低电平输出，将会改变监测口的状态，此时监测口为低电平，不再提示车主系好安全带。

具体实现结果如下图 88 所示：

```
Exiting... (End of file)
* 等待语音输入中...
* 录音中...
* 录音结束
语言码是 1537
您的提问是：开始开车。
安全带未系好
语音内容：安全带未系好，禁止开车！
MPlayer 1.3.0 (Debian), built with gcc-8 (c) 2000-2016 MPlayer Team
do_connect: could not connect to socket
connect: No such file or directory
Failed to open LIRC support. You will not be able to use your remote control.

Playing tts_result.mp3.
libavformat version 58.20.100 (external)
Audio only file format detected.
Load subtitles in ./
=====
```

图 88 安全带模块实现（未系安全带时警报）结果图

由上图可以看出，在发出“开始开车”命令后，系统将进行安全带检测，如果未系上安全带，系统将一直用语音发出“安全带未系好，禁止开车！”的提示，直至安全带系好以后，提示停止。

3) 调试说明：

该模块硬件内容较为简单，难点在线程的分配，由于此内容为总体实现，因此不在此过多赘述。基于此模块，我们采取了更加人性化的设计，考虑到存在使用车的功能但并不开车的情况，因此我们并非在做任何操作时都会进行安全带监测从而进行提示，而是在发出“开始开车”命令后才进行监测。

四、程序清单与执行结果

由于整套系统功能模块较多，因此将以功能模块为单位列出程序清单和执行结果。仍然需要强调，小派的开发各功能模块之间有高度的独立性，均被封装成了相应的类和函数。各模块源代码如下所示：

4.1 主函数

Main.py
import speech_recognition

```
import speech_synthesis  
import os  
import get_response  
import luyin  
import translation  
import chat  
import weather  
import time  
import DHT11  
import fp_add  
import fp_update  
import detect_sleepy  
import safebound  
  
import pyautogui as pk  
import pyperclip  
from playsound import playsound  
import RPi.GPIO as GPIO  
  
light = 4  
led = 23  
  
def light_test():  
    GPIO.setmode(GPIO.BCM)  
    GPIO.setup(light,GPIO.IN)  
    GPIO.setup(led,GPIO.OUT)  
    GPIO.output(led,GPIO.LOW)  
    if GPIO.input(light)==1:  
        GPIO.output(led,GPIO.HIGH)  
    else:  
        GPIO.output(led,GPIO.LOW)
```

```
def say(content, isstr=True):
    if not content:
        print("内容为空")
        return
    path = speech_synthesis.tts(content, isstr)
    if path:
        os.system('mpplayer ' + path)
    else:
        print('语音输出出现错误')

def trans(content):
    lansdict = {"中文": "zh", "英文": "en", "英语": "en"}
    "pos = content.find("翻译")
    if pos == -1:
        pos = content.find("到")
        if pos == -1:
            return
        pos = content[pos:].find("翻译")
        if pos == -1:
            return ""
    source = None
    target = None
    sourcecode = None
    targetcode = None

    posdict = {}
    for lan in lansdict:
        posdict[lan] = content.find(lan)
    minp = len(content)
    minv = None
    for lan in posdict:
        if posdict[lan] != -1 and posdict[lan] < minp:
```

```

minp = posdict[lan]
minv = lan

if not minv:
    say("翻译的源语言或目标语言不明确")
    return

source = minv
sourcecode = lansdict[source]
print("源语言是"+source)
for lan in lansdict:
    if lan in content[minp+1:]:
        target = lan
        targetcode = lansdict[target]
        break

if not target:
    say("翻译的源语言或目标语言不明确")
    return

print("目标语言是"+target)
if source==target:
    say("源语言和目标语言相同，不用翻译")
    return

say(source+'翻译'+target+'的功能已开启，请输入要翻译的'+source+'对话')
while True:
    luyin.recording('mysound.wav')
    trans_word = speech_recognition.stt('mysound.wav', lan=source)
    if trans_word:
        if '退出' in trans_word and "翻译" in trans_word:
            say("翻译功能已退出")
            return
        print('您要翻译的句子是'+trans_word)
        trans_res = translation.translation(trans_word, sourcecode, targetcode)
        return '这句话的'+target+'翻译是'+trans_res

```

```
else:  
    print('没有监听到正常的提问对话')  
    say('没有监听到正常的提问对话。请输入要翻译的'+source+'对话')  
  
def func_choose(old_sleepy_thread):  
    light_test()  
    say('您好，请问有什么可以帮到您？', True)  
    luyin.recording('mysound.wav')  
    stt_res = speech_recognition.stt('mysound.wav')  
  
    sleepy_thread = None  
    result = None  
    if stt_res:  
        print('您的提问是:', stt_res)  
        if '天气' in stt_res:  
            say("天气查询功能已经开启，请输入要查询的城市名")  
            while True:  
                luyin.recording('mysound.wav')  
                city = speech_recognition.stt('mysound.wav')  
                #city = '杭州'  
                if city:  
                    city = city.strip('。')  
                    if '退出' in city and "查询" in city:  
                        say("天气查询功能已退出")  
                        return  
                    print('您要查询的城市是'+city)  
                    result = weather.get_weather(city)  
                    if result:  
                        break  
                    say('没有监听到正确的城市。请重新输入要查询的城市名')  
                else:  
                    print('没有监听到正确的城市')
```

```
say('没有监听到正确的城市。请重新输入要查询的城市名')

#result = weather.get_weather('杭州')

# path = speech_synthesis.tts(wea, True)

elif '几点' in stt_res or '时间' in stt_res:

    curtime = time.strftime("%Y 年 %m 月 %d 日 %H 点 %M 分", time.localtime())

    result = '现在是' + curtime

    # path = speech_synthesis.tts('现在是' + curtime, True)

elif '翻译' in stt_res:

    result = trans(stt_res)

    if not result:

        return

elif "温度" in stt_res and "周围" in stt_res:

    print("正在检测中-----")

    say("正在检测中")

    tem, humi = DHT11.get_tem(18)

    result = "当前周围的环境温度为" + str(tem) + "摄氏度。"

elif "录入" in stt_res and "指纹" in stt_res:

    say("请按下指纹")

    result = fp_add.fp_add()

elif "更新" in stt_res and "指纹" in stt_res:

    say("请按下指纹")

    result = fp_update.fp_update()

elif "疲劳检测" in stt_res:

    if "打开" in stt_res or "开启" in stt_res:

        sleepy_thread = detect_sleepy.mythread_run()

        if not sleepy_thread:

            result = "疲劳检测功能开启失败"

        else:

            result = "成功开启疲劳检测功能"

    elif "关闭" in stt_res:

        if not old_sleepy_thread:

            result = "当前未开启疲劳检测功能"

        else:
```

```

        if detect_sleepy.mythread_stop(old_sleepy_thread):
            result = "成功关闭疲劳检测功能"
            sleepy_thread = 1
        else:
            result = "疲劳检测功能关闭失败"

    elif(("开启" in stt_res) or ("打开" in stt_res)) and "闲聊" in stt_res:
        say("闲聊功能已开启，你想聊点什么？")

    while True:
        luyin.recording('mysound.wav')
        myword = speech_recognition.stt('mysound.wav')
        print("myword:", myword)
        if not myword:
            say("我没有听懂你说的话")
            continue
        if(("退出" in myword) or ("关闭" in myword)) and \
           "闲聊" in myword:
            result = "闲聊功能已退出"
            break
        say(get_response.get_response(myword))

    elif("开始" in stt_res or "准备" in stt_res) and "开车" in stt_res:
        safebound.mythread_run()
        result = "end"

    elif "的女人" in stt_res:
        if "温柔" in stt_res:
            result = "雷艳静老师" + stt_res[1:len(stt_res)-1]
        elif "大方" in stt_res:
            result = "这还用说吗？当然还是雷艳静老师。"
        else:
            result = "你不用问了，只要是褒义词，都是形容雷艳静老师的"
    else:
        result = get_response.get_response(stt_res)
        # path = speech_synthesis.tts(response, True)

    if result != "end":

```

```

if not result:
    say("未接收到合适指令")
else:
    say(result)
else:
    say('没有监听到正常的提问对话')
    return
if sleepy_thread == 1:
    return None
elif not sleepy_thread:
    return old_sleepy_thread
else:
    return sleepy_thread

if __name__ == '__main__':
    #GPIO_init()
    func_choose()

```

4.2 视觉模块——疲劳检测功能

Detect_sleepy.py

```

# 导入所需库部分
from scipy.spatial import distance as dist
from imutils import face_utils
import numpy as np  # 数据处理的库 numpy
import imutils
import dlib
import cv2
import math
import threading
import ctypes
import inspect

```

```
import speech_synthesis
import os
import time

stop = False

def say(content, isstr=True):
    if not content:
        print("内容为空")
        return
    path = speech_synthesis.tts(content, isstr)
    if path:
        os.system('mplayer ' + path)
    else:
        print('语音输出出现错误')

def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    try:
        tid = ctypes.c_long(tid)
        if not inspect.isclass(exctype):
            exctype = type(exctype)
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, ctypes.py_object(exctype))
        if res == 0:
            # pass
            raise ValueError("invalid thread id")
        elif res != 1:
            # """if it returns a number greater than one, you're in trouble,
            # and you should call it again with exc=NULL to revert the effect"""
            ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
            raise SystemError("PyThreadState_SetAsyncExc failed")
    except Exception as err:
```

```

print(err)

def stop_thread(thread_ident):
    """终止线程"""
    _async_raise(thread_ident, SystemExit)
    # print("stop successful!")

def get_head_pose(shape): # 头部姿态估计
    # 一些初始化的参数
    # 世 界 坐 标 系 (UVW) : 填 写 3D 参 考 点 , 该 模 型 参 考
http://aifi.isr.uc.pt/Downloads/OpenGL/glAnthropometric3DModel.cpp

object_pts = np.float32([[6.825897, 6.760612, 4.402142], # 33 左眉左上角
                        [1.330353, 7.122144, 6.903745], # 29 左眉右角
                        [-1.330353, 7.122144, 6.903745], # 34 右眉左角
                        [-6.825897, 6.760612, 4.402142], # 38 右眉右上角
                        [5.311432, 5.485328, 3.987654], # 13 左眼左上角
                        [1.789930, 5.393625, 4.413414], # 17 左眼右上角
                        [-1.789930, 5.393625, 4.413414], # 25 右眼左上角
                        [-5.311432, 5.485328, 3.987654], # 21 右眼右上角
                        [2.005628, 1.409845, 6.165652], # 55 鼻子左上角
                        [-2.005628, 1.409845, 6.165652], # 49 鼻子右上角
                        [2.774015, -2.080775, 5.048531], # 43 嘴左上角
                        [-2.774015, -2.080775, 5.048531], # 39 嘴右上角
                        [0.000000, -3.116408, 6.097667], # 45 嘴中央下角
                        [0.000000, -7.415691, 4.070434]]) # 6 下巴角

    # 相机坐标系(XYZ): 添加相机内参
    K =[6.5308391993466671e+002, 0.0, 3.195000000000000e+002,
         0.0, 6.5308391993466671e+002, 2.395000000000000e+002,
         0.0, 0.0, 1.0] # 等价于矩阵[fx, 0, cx; 0, fy, cy; 0, 0, 1]
    # 图像中心坐标系(uv): 相机畸变参数[k1, k2, p1, p2, k3]

```

```

D = [7.0834633684407095e-002, 6.9140193737175351e-002, 0.0, 0.0, -1.3073460323689292e+000]

# 像素坐标系(xy): 填写凸轮的本征和畸变系数
cam_matrix = np.array(K).reshape(3, 3).astype(np.float32)
dist_coeffs = np.array(D).reshape(5, 1).astype(np.float32)

# 重新投影 3D 点的世界坐标轴以验证结果姿势
reprojects = np.float32([[10.0, 10.0, 10.0],
                         [10.0, 10.0, -10.0],
                         [10.0, -10.0, -10.0],
                         [10.0, -10.0, 10.0],
                         [-10.0, 10.0, 10.0],
                         [-10.0, 10.0, -10.0],
                         [-10.0, -10.0, -10.0],
                         [-10.0, -10.0, 10.0]])

# 绘制正方体 12 轴
line_pairs = [[0, 1], [1, 2], [2, 3], [3, 0],
               [4, 5], [5, 6], [6, 7], [7, 4],
               [0, 4], [1, 5], [2, 6], [3, 7]]

# (像素坐标集合) 填写 2D 参考点, 注释遵循 https://ibug.doc.ic.ac.uk/resources/300-W/
# 17 左眉左上角/21 左眉右角/22 右眉左上角/26 右眉右上角/36 左眼左上角/39 左眼右上角/42 右
# 眼左上角/
# 45 右眼右上角/31 鼻子左上角/35 鼻子右上角/48 左上角/54 嘴右上角/57 嘴中央下角/8 下巴角
image_pts = np.float32([shape[17], shape[21], shape[22], shape[26], shape[36],
                       shape[39], shape[42], shape[45], shape[31], shape[35],
                       shape[48], shape[54], shape[57], shape[8]])

# solvePnP 计算姿势——求解旋转和平移矩阵:
# rotation_vec 表示旋转矩阵, translation_vec 表示平移矩阵, cam_matrix 与 K 矩阵对应, dist_coeffs
# 与 D 矩阵对应。
_, rotation_vec, translation_vec = cv2.solvePnP(object_pts, image_pts, cam_matrix, dist_coeffs)

# projectPoints 重新投影误差: 原 2d 点和重投影 2d 点的距离 (输入 3d 点、相机内参、相机畸
# 变、r、t, 输出重投影 2d 点)

```

```

reprojectdst, _ = cv2.projectPoints(reprojsrc, rotation_vec, translation_vec, cam_matrix,
dist_coeffs)

reprojectdst = tuple(map(tuple, reprojectdst.reshape(8, 2))) # 以 8 行 2 列显示

# 计算欧拉角 calc euler angle
# 参考
https://docs.opencv.org/2.4/modules/calib3d/doc/camera\_calibration\_and\_3d\_reconstruction.html#decomposeprojectionmatrix

rotation_mat, _ = cv2.Rodrigues(rotation_vec) # 罗德里格斯公式（将旋转矩阵转换为旋转向量）
pose_mat = cv2.hconcat((rotation_mat, translation_vec)) # 水平拼接，vconcat 垂直拼接

# decomposeProjectionMatrix 将投影矩阵分解为旋转矩阵和相机矩阵
_, _, _, _, _, euler_angle = cv2.decomposeProjectionMatrix(pose_mat)

pitch, yaw, roll = [math.radians(_) for _ in euler_angle]

pitch = math.degrees(math.asin(math.sin(pitch)))
roll = -math.degrees(math.asin(math.sin(roll)))
yaw = math.degrees(math.asin(math.sin(yaw)))

print('pitch:{}, yaw:{}, roll:{}' .format(pitch, yaw, roll))

return reprojectdst, euler_angle # 投影误差, 欧拉角

def cal_EAR(eye): # 眨眼疲劳检测
    # 垂直眼标志 (X, Y) 坐标
    A = dist.euclidean(eye[1], eye[5]) # 计算两个集合之间的欧式距离
    B = dist.euclidean(eye[2], eye[4])
    # 计算水平之间的欧几里得距离
    # 水平眼标志 (X, Y) 坐标
    C = dist.euclidean(eye[0], eye[3])
    # 眼睛长宽比的计算
    ear = (A + B) / (2.0 * C)
    # 返回眼睛的长宽比

```

```

return ear

def cal_MAR(mouth): # 打哈欠疲劳检测
    A = np.linalg.norm(mouth[2] - mouth[9]) # 51, 59
    B = np.linalg.norm(mouth[4] - mouth[7]) # 53, 57
    C = np.linalg.norm(mouth[0] - mouth[6]) # 49, 55
    mar = (A + B) / (2.0 * C)
    return mar

pitch_list = []
pitch_list_max_num = 5

def update_pitchlist(now_pitch):
    if (len(pitch_list) < pitch_list_max_num):
        pitch_list.append(now_pitch)
    else:
        del pitch_list[0]
        pitch_list.append(now_pitch)
    return np.mean(pitch_list)

# sleepy_detect 入口参数默认值及含义
# EYE_AR_THRESH = 0.2 # 眼睛长宽比
# EYE_AR_CONSEC_FRAMES = 3 # 眨眼闪烁阈值
# MAR_THRESH = 0.5 # 打哈欠嘴巴长宽比
# MOUTH_AR_CONSEC_FRAMES = 6 # 打哈欠闪烁阈值
# HAR_THRESH = 0.34 # 瞌睡点头阈值, 约 20° 左右
# NOD_AR_CONSEC_FRAMES = 3
def sleepy_detect(EYE_AR_THRESH=0.2, EYE_AR_CONSEC_FRAMES=3, MAR_THRESH=0.65,
MOUTH_AR_CONSEC_FRAMES=5, HAR_THRESH=5.0,

```

```
NOD_AR_CONSEC_FRAMES=3):  
print("-----已经开始疲劳检测-----")  
# 初始化帧计数器和眨眼总数  
eCOUNTER = 0  
eTotal = 0  
# 初始化帧计数器和打哈欠总数  
mCOUNTER = 0  
mTotal = 0  
# 初始化帧计数器和点头总数  
hCOUNTER = 0  
hTotal = 0  
  
# 初始化 DLIB 的人脸检测器 (HOG), 然后创建面部标志物预测  
print("[INFO] loading facial landmark predictor...")  
# 第一步: 使用 dlib.get_frontal_face_detector() 获得脸部位置检测器  
detector = dlib.get_frontal_face_detector()  
# 第二步: 使用 dlib.shape_predictor 获得脸部特征位置检测器  
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')  
# 第三步: 分别获取左右眼面部标志的数组下标索引  
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]  
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]  
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]  
# 第四步: 打开 cv2 本地摄像头  
cap = cv2.VideoCapture(0)  
  
yawn_flag = False  
pre_pitch = 0  
global stop  
# 从视频流循环帧  
while True:  
    # 第五步: 进行循环, 读取图片, 并对图片做维度统一, 并进灰度化  
    ret, frame = cap.read()  
    frame = imutils.resize(frame, width=720)
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# 第六步：使用 detector(gray, 0) 进行脸部位置检测
rects = detector(gray, 0)

# 第七步：循环脸部位置信息，使用 predictor(gray, rect)获得脸部特征位置的信息
for rect in rects:

    shape = predictor(gray, rect)

    # 第八步：将脸部特征信息转换为 numpy 中数组 array 的格式
    shape = face_utils.shape_to_np(shape)

    # 第九步：提取左眼、右眼和嘴巴关键点
    leftEye = shape[lStart:lEnd] # 左眼
    rightEye = shape[rStart:rEnd] # 右眼
    mouth = shape[mStart:mEnd] # 嘴巴

    # 第十步：构造函数计算左右眼的 EAR 值，使用平均值作为最终的 EAR;计算打哈欠
    的 MAR 值
    leftEAR = cal_EAR(leftEye)
    rightEAR = cal_EAR(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    mar = cal_MAR(mouth)

    # 第十一步：使用 cv2.convexHull 获得凸包位置，使用 drawContours 画出轮廓位置进
    行画图操作
    # leftEyeHull = cv2.convexHull(leftEye)
    # rightEyeHull = cv2.convexHull(rightEye)
    # mouthHull = cv2.convexHull(mouth)
    # cv2.drawContours(frame, [leftEyeHull], -1, (240, 0, 0), 1)
    # cv2.drawContours(frame, [rightEyeHull], -1, (240, 0, 0), 1)
    # cv2.drawContours(frame, [mouthHull], -1, (240, 0, 0), 1)

    # 第十二步：进行画图操作，用矩形框标注人脸
```

```

# left = rect.left()
# top = rect.top()
# right = rect.right()
# bottom = rect.bottom()

# cv2.rectangle(frame, (left, top), (right, bottom), (0, 240 , 0), 1)

# 第十三步：循环，满足 EAR 值>0.2 条件的，眨眼次数+1

if ear< EYE_AR_THRESH: # 眼睛长宽比: 0.2

    eCOUNTER += 1

else:

    #如果连续 3 帧都小于阈值，则表示进行了一次较疲劳的眨眼活动

    if eCOUNTER >= EYE_AR_CONSEC_FRAMES: # 阈值: 3

        eTotal += 1

        say("检测到一次疲劳眨眼，请注意自己的疲劳状态！")

        cv2.putText(frame, "Tired Wink!", (100, 150),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 165, 0), 2)

        # 重置眼帧计数器

        eCOUNTER = 0

# 同理，判断是否打哈欠

if mar > MAR_THRESH: # 张嘴阈值 0.5

    mCOUNTER += 1

    cv2.putText(frame, "Yawning!", (100, 200), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 165, 0), 2)

elif mar < 0.45:

    # 如果连续几帧都大于等于阈值，则认为打了一次哈欠

    if mCOUNTER >= MOUTH_AR_CONSEC_FRAMES: # 阈值: 3

        mTotal += 1

        say("检测到您打了一次哈欠，请注意自己的疲劳状态！")

        # 重置哈欠帧计数器

        mCOUNTER = 0

# print("mar=",mar)

# 第十四步：进行画图操作，同时使用 cv2.putText 将关键数值量进行显示

```

```

#           cv2.putText(frame,      "Faces:      {}" .format(len(rects)),      (10,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

#           cv2.putText(frame,      "COUNTER:    {}" .format(COUNTER),      (150,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

#           cv2.putText(frame,      "EAR:        {:.2f}" .format(ear),      (300,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

#           cv2.putText(frame,      "Blinks:     {}" .format(TOTAL),      (450,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,0), 2)

#           cv2.putText(frame,      "COUNTER:    {}" .format(mCOUNTER),      (150,
60),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

#           cv2.putText(frame,      "MAR:        {:.2f}" .format(mar),      (300,
60),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

#           cv2.putText(frame,      "Yawning:    {}" .format(mTOTAL),      (450,
60),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,255,0), 2)

# 第十五步：获取头部姿态
reprojectdst, euler_angle = get_head_pose(shape)

har = euler_angle[0, 0] # 取 pitch 旋转角度
pitch_diff = abs(har - pre_pitch)
pre_pitch = har

if pitch_diff > HAR_THRESH: # 点头阈值 0.3
    hCOUNTER += 1
else:
    # 如果连续 3 帧都大于阈值，则表示瞌睡点头一次
    if hCOUNTER >= NOD_AR_CONSEC_FRAMES: # 阈值： 3
        hTotal += 1
        cv2.putText(frame, "Nodding!", (100, 250), cv2.FONT_HERSHEY_SIMPLEX, 1,
(255, 165, 0), 2)
        say("检测到瞌睡点头，请立即停车休息！")
    # 重置点头帧计数器
    hCOUNTER = 0

```

```

# 绘制正方体12轴

# for start, end in line_pairs:
    # cv2.line(frame, reprojectdst[start], reprojectdst[end], (0, 0, 255))

# 显示角度结果

# cv2.putText(frame, "X: " + "{:7.2f}".format(euler_angle[0, 0]), (10, 90),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), thickness=2) # GREEN

# cv2.putText(frame, "Y: " + "{:7.2f}".format(euler_angle[1, 0]), (150, 90),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 0, 0), thickness=2) # BLUE

# cv2.putText(frame, "Z: " + "{:7.2f}".format(euler_angle[2, 0]), (300, 90),
cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), thickness=2) # RED

# cv2.putText(frame, "Nod: {} ".format(hTOTAL), (450,
90), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 0), 2)

# 第十六步：进行画图操作，68个特征点标识

for (x, y) in shape:
    cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)

# print('嘴巴实时长宽比:{:.2f} '.format(mar)+"\t是否张嘴：" + str([False, True][mar > MAR_THRESH]))

# print('眼睛实时长宽比 : {:.2f} '.format(ear)+"\t是否眨眼：" + str([False, True][COUNTER>=1]))


# 确定疲劳提示:疲劳眨眼50次,打哈欠15次,瞌睡点头15次。满足条件之一就提示
if eTotal >= 20 or mTotal >= 5 or hTotal >= 3:
    cv2.putText(frame, "SLEEP!!!", (100, 300), cv2.FONT_HERSHEY_SIMPLEX, 1.4, (0,
255, 255), 3)

# 这下面加入对树莓派GPIO引脚的电平输出操作
say("危险！疲劳驾驶！")

# 按q退出
cv2.putText(frame, "Press 'q': Quit", (20, 500), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (84, 255,
159), 2)

# 窗口显示 show with opencv
cv2.imshow("Frame", frame)

```

```
# if the 'q' key was pressed, break from the loop
print("-----stop=", stop, "-----")
if stop or cv2.waitKey(1) & 0xFF == ord('q'):
    # 释放摄像头资源
    cap.release()
    cv2.destroyAllWindows()
    break

def mythread_run():
    global stop
    stop = False
    detect_thread = threading.Thread(target=sleepy_detect, args=())
    detect_thread.start()
    return detect_thread.ident

def mythread_stop(thread_ident):
    global stop
    stop = True
    time.sleep(10)
    stop_thread(thread_ident)
    return True

if __name__ == '__main__':
    # detect_thread = threading.Thread(target=sleepy_detect(), args=()) # args 可以给 run 传参
    detect_thread = threading.Thread(target=sleepy_detect, args=())
    detect_thread.start()
    # voice_thread = threading.Thread(target=voice_detect, args=(detect_thread.ident, 9999))
    # voice_thread.start()
    while True:
        # print(1)
        pass
```

4.3 传感器模块

DHT11.py 温湿度检测功能

```
import RPi.GPIO as GPIO    #引入 RPi.GPIO 模块，并实例化为 GPIO，简化后面的模块调用
import time                #引入 time 模块

def test(Pin=18):
    Bit=[]                  #定义列表，用以存储接收到的数据
    Time=[]                 #存储数字 0 和数字 1 的高电位持续时间
    K=0                     #定义循环初始值
    GPIO.setmode(GPIO.BCM)   #定义 GPIO 编码方式
    time.sleep(2)            #由于 dht11 的采样周期间隔为 2s，此处防止 dht 无方响应
    #Pin=16                 #定义引脚，使用哪个就定义哪个
    GPIO.setup(Pin,GPIO.OUT)  #将 GPIO 设置为输出模式
    GPIO.output(Pin,0)        #向 dht11 发送低电平(开始信号)(其中 0 可以用 Flase、GPIO.low
    替代，下文的 1 同理)
    time.sleep(0.02)          #低电平持续时间 0.02s 大于 dht11 要求的 0.018s
    GPIO.output(Pin,1)        #向 dht11 发送高电平（理论上可以取消，但是实际使用中不发
    送高概率出错）
    GPIO.setup(Pin,GPIO.IN)   #将 GPIO 转为输入模式，用以接收 dht11 的信号
    while GPIO.input(Pin)==0:  #采用轮训检测 dht 的响应信号，如果输入高电平将跳出此循环
    （不懂可以查阅 python 的 while continue 用法）
        continue
    while GPIO.input(Pin)==1:  #采用轮训检测 dht 的响应信号
        continue
    while K<40:              #循环 40 次用以接收 dht11 的 40bit 数据
        while GPIO.input(Pin)==0:#因为每 bit 数据以低电平开始，故此。
            continue
        begin=time.time()      #低电平循环结束故此时是高电平信号，因此开始计时
        while GPIO.input(Pin)==1:#轮训高电平
            continue
        end=time.time()        #获取高电平信号的结束时间
        #Time.append(end-begin)#此处为了确定 0 和 1 的高电平持续时间
```

```

if (end-begin)<0.00003: #检测高电平的持续时间判断输入是 0 还是 1 (0.00003 可以自测
一下, 取一个适合你的时间)
    Bit.append(0)          #高电平小于 0.00003s 证明是 0
else:
    Bit.append(1)          #高电平大于 0.00003s 证明是 1
K=K+1                         #记录循环次数
#print(Time)                   #观察时间特点以确定上文 0.00003 的取值
humidity1bit=Bit[0:8]          #根据 dht11 的信号原理获取所需的值 (下同理)
humidity2bit=Bit[8:16]
temperature1bit=Bit[16:24]
temperature2bit=Bit[24:32]
checkbit = Bit[32:40]
humidity1=0
humidity2=0
temperature1=0
temperature2=0
check=0
for i in range(0,8):           #循环 8 次, 将二进制数转换为十进制数
    humidity1+=humidity1bit[i]*(2***(7-i))
    humidity2+=humidity2bit[i]*(2***(7-i))
    temperature1+=temperature1bit[i]*(2***(7-i))
    temperature2+=temperature2bit[i]*(2***(7-i))
    check+=checkbit[i]*(2***(7-i))
temperature=temperature1+temperature2*0.1 #获取温度值 (注意 dht11 的使用说明中明文写到
整数位后是小数位故应乘 0.1)
humidity=humidity1+humidity2*0.1        # (网上大部分都没乘, 他们的运行结果是整数,
足可见网上的帖子都是复制粘着的水文)
checknum=temperature1+humidity1+temperature2+humidity2 #计算前 32 位数的值
if checknum==check:                  #检查前 32 位的值是否与校验位相等
    print("temp:%s,hum:%s"%(temperature,humidity))  #相等输出温度和湿度
    return temperature, humidity
else:
    print("dht11 check was wrong.  checknum:%s check:%s"%(checknum,check))#不等输出前

```

32 位值和校验位值

```
    return None
```

```
def get_tem(Pin=16):
```

```
    i=0
```

```
    while True:
```

```
        print(i)
```

```
        i+=1
```

```
        res = test(Pin)
```

```
        if res:
```

```
            return res
```

```
if __name__ =='__main__':
```

```
    i=0
```

```
    while True:
```

```
        print(i)
```

```
        i+=1
```

```
        if test():
```

```
            break
```

LED.py 智能光控功能

```
import RPi.GPIO as GPIO
```

```
import time
```

```
import threading
```

```
import ctypes
```

```
import inspect
```

```
light = 4
```

```
led = 23
```

```
def mythread_run():
```

```
    global stop
```

```

stop = False

detect_thread = threading.Thread(target=led_test, args=())
detect_thread.start()

return detect_thread.ident


def mythread_stop(thread_ident):
    global stop
    stop = True
    time.sleep(10)
    stop_thread(thread_ident)
    return True


def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    try:
        tid = ctypes.c_long(tid)
        if not inspect.isclass(exctype):
            exctype = type(exctype)
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, ctypes.py_object(exctype))
        if res == 0:
            # pass
            raise ValueError("invalid thread id")
        elif res != 1:
            # """if it returns a number greater than one, you're in trouble,
            # and you should call it again with exc=NULL to revert the effect"""
            ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
            raise SystemError("PyThreadState_SetAsyncExc failed")
    except Exception as err:
        print(err)

def stop_thread(thread_ident):

```

```
"""终止线程"""
_async_raise(thread_ident, SystemExit)
# print("stop successful!")

def led_test():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(light,GPIO.IN)
    GPIO.setup(led,GPIO.OUT)
    GPIO.output(led,GPIO.LOW)
    while True:
        if GPIO.input(light)==1:
            GPIO.output(led,GPIO.HIGH)
        else:
            GPIO.output(led,GPIO.LOW)

        # print("-----light----", GPIO.input(light))
```

safebound.py 安全带检测功能

```
import threading
import ctypes
import inspect

import RPi.GPIO as GPIO
import time

import speech_synthesis
import os

sender = 20
receiver = 21
```

```
def say(content, isstr=True):
    if not content:
        print("内容为空")
        return
    path = speech_synthesis.tts(content, isstr)
    if path:
        os.system('mpplayer ' + path)
    else:
        print('语音输出出现错误')

def mythread_run():
    global stop
    stop = False
    detect_thread = threading.Thread(target=safebound_test, args=())
    detect_thread.start()
    return detect_thread.ident

def mythread_stop(thread_ident):
    global stop
    stop = True
    time.sleep(10)
    stop_thread(thread_ident)
    return True

def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    try:
        tid = ctypes.c_long(tid)
        if not inspect.isclass(exctype):
            exctype = type(exctype)
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, ctypes.py_object(exctype))
        if res == 0:
```

```

# pass

raise ValueError("invalid thread id")

elif res != 1:
    # """if it returns a number greater than one, you're in trouble,
    # and you should call it again with exc=NULL to revert the effect"""
    ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
    raise SystemError("PyThreadState_SetAsyncExc failed")

except Exception as err:
    print(err)

def stop_thread(thread_ident):
    """终止线程"""
    _async_raise(thread_ident, SystemExit)
    # print("stop successful!")

def safebound_test():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(receiver,GPIO.IN)
    #GPIO.setup(sender, GPIO.IN)
    while True:
        #print(receiver, "端口电平为: ", GPIO.input(receiver), "#"
        GPIO.input(sender))

        for i in range(100):
            if GPIO.input(receiver)==0:
                # print("安全带已系好")
                break
            else:
                print("安全带未系好")
                say("安全带未系好,禁止开车! ")

```

4.4 指纹模块

fp_add.py 指纹录入功能

```
import binascii
import serial
import serial.tools.list_ports
import time

# volatile unsigned char FPM10A_RECEICE_BUFFER[32]; //定义接收缓存区
# code unsigned char FPM10A_Pack_Head[6] = {0xEF,0x01,0xFF,0xFF,0xFF,0xFF}; //协议包头
# code unsigned char FPM10A_Get_Img[6] = {0x01,0x00,0x03,0x01,0x00,0x05}; //获得指纹图像
# code unsigned char FPM10A_Img_To_Buffer1[7]={0x01,0x00,0x04,0x02,0x01,0x00,0x08}; //将图像放入到 BUFFER1
# code unsigned char FPM10A_Search[11]={0x01,0x00,0x08,0x04,0x01,0x00,0x00,0x00,0x64,0x00,0x72}; //搜索指纹搜索范围 0 - 999,使用 BUFFER1 中的特征码搜索

def recv(serial):
    while True:
        data = serial.read_all()
        if data == "":
            continue
        else:
            break
    return data

def fp_add():
    myserial = serial.Serial('/dev/ttyUSB0', 57600, timeout=0.5) #/dev/ttyUSB0
    if myserial.isOpen() :
```

```

        print("open success")
    else :
        print("open failed")
    while True:
        a = 'EF 01 FF FF FF FF 01 00 03 10 00 14'
        d = bytes.fromhex(a)
        myserial.write(d)
        time.sleep(1)
        data =recv(myserial)
        if data != b" ":
            data_con = str(binascii.b2a_hex(data))[20:22]
            if(data_con == '00'):
                print("注册成功")
                data_con = str(binascii.b2a_hex(data))[22:26]
                print(data_con)
                myserial.close()
                return data_con + "号用户指纹注册成功"
            elif data_con != '03':
                return "指纹注册失败"
        if __name__ == '__main__':
            print(fp_add())

```

fp_match.py 指纹匹配功能

```

import binascii
import serial
import serial.tools.list_ports
import time
import speech_synthesis
import os
# volatile unsigned char FPM10A_RECEICE_BUFFER[32];      //定义接收缓存区
# code unsigned char FPM10A_Pack_Head[6] = {0xEF,0x01,0xFF,0xFF,0xFF,0xFF}; //协议包头
# code unsigned char FPM10A_Get_Img[6] = {0x01,0x00,0x03,0x01,0x00,0x05}; //获得指纹图像

```

```

# code unsigned char FPM10A_Img_To_Buffer1[7]={0x01,0x00,0x04,0x02,0x01,0x00,0x08}; //将图像放入到 BUFFER1
#
#           code          unsigned         char
FPM10A_Search[11]={0x01,0x00,0x08,0x04,0x01,0x00,0x00,0x00,0x64,0x00,0x72}; //搜索指纹搜索范围 0 - 999,使用 BUFFER1 中的特征码搜索

def recv(serial):
    while True:
        data = serial.read_all()
        if data == "":
            continue
        else:
            break
    return data

def say(content, isstr=True):
    if not content:
        print("内容为空")
        return
    path = speech_synthesis.tts(content, isstr)
    if path:
        os.system('mplayer ' + path)
    else:
        print('语音输出出现错误')

def fp_match():
    myserial = serial.Serial('/dev/ttyUSB0', 57600, timeout=0.5) #/dev/ttyUSB0
    if myserial.isOpen() :
        print("open success")
    else :
        print("open failed")
    while True:
        a = 'EF 01 FF FF FF FF 01 00 03 01 00 05'

```

```
d = bytes.fromhex(a)
myserial.write(d)
time.sleep(1)
data =recv(myserial)
if data != b"":
    data_con = str(binascii.b2a_hex(data))[20:22]
    if(data_con == '02'):
        print("请按下手指")
    elif(data_con == '00'):
        print("载入成功")
        buff = 'EF 01 FF FF FF FF 01 00 04 02 01 00 08'
        buff = bytes.fromhex(buff)
        myserial.write(buff)
        time.sleep(1)
        buff_data = recv(myserial)
        buff_con = str(binascii.b2a_hex(buff_data))[20:22]
        if(buff_con == '00'):
            print("生成特征成功")
            serch = 'EF 01 FF FF FF FF 01 00 08 04 01 00 00 00 64 00 72'
            serch = bytes.fromhex(serch)
            myserial.write(serch)
            time.sleep(1)
            serch_data = recv(myserial)
            serch_con = str(binascii.b2a_hex(serch_data))[20:22]
            if (serch_con == '09'):
                print("指纹不匹配")
                say("指纹不匹配")
                continue
            elif(serch_con == '00'):
                print("指纹匹配成功")
                myserial.close()
                say("指纹匹配成功")
                break
```

```

#myserial.close()
#exit()
else:
    print("不成功")
os.system("python3 speech_detected.py snowboy.umdl")

if __name__ == '__main__':
    fp_match()

```

fp_update.py 指纹更新功能

```

import binascii
import serial
import serial.tools.list_ports
import time

# volatile unsigned char FPM10A_RECEICE_BUFFER[32]; //定义接收缓存区
# code unsigned char FPM10A_Pack_Head[6] = {0xEF,0x01,0xFF,0xFF,0xFF,0xFF}; //协议包头
# code unsigned char FPM10A_Get_Img[6] = {0x01,0x00,0x03,0x01,0x00,0x05}; //获得指纹图像
# code unsigned char FPM10A_Img_To_Buffer1[7]={0x01,0x00,0x04,0x02,0x01,0x00,0x08}; //将图像放入到 BUFFER1
# code unsigned char FPM10A_Search[11]={0x01,0x00,0x08,0x04,0x01,0x00,0x00,0x00,0x64,0x00,0x72}; //搜索指纹搜索范围 0 - 999,使用 BUFFER1 中的特征码搜索

def recv(serial):
    while True:
        data = serial.read_all()
        if data == "":
            continue
        else:
            break
    return data

def fp_update():

```

```

myserial = serial.Serial('/dev/ttyUSB0', 57600,)  #/dev/ttyUSB0

if myserial.isOpen() :
    print("open success")
else :
    print("open failed")

while True:

    a = 'EF 01 FF FF FF FF 01 00 03 11 00 15'

    d = bytes.fromhex(a)

    myserial.write(d)

    time.sleep(1)

    data =recv(myserial)

    if data != b"":

        data_con = str(binascii.b2a_hex(data))[20:22]

        if(data_con == '00'):

            print("更新成功")

            data_con1 = str(binascii.b2a_hex(data))[22:26]

            print(data_con1)

            data_con2 = str(binascii.b2a_hex(data))[26:30]

            print("新的得分为: ")

            print(data_con2)

            myserial.close()

            return data_con1 + "号用户指纹更新成功， 新指纹得分为" + data_con2

        elif data_con != '03':

            return "指纹更新失败"

    if __name__ == '__main__':
        fp_update()

```

4.5 语音交互模块

Get_response.py 智能对话功能

```
import json
```

```
from tencentcloud.common import credential
from tencentcloud.common.profile.client_profile import ClientProfile
from tencentcloud.common.profile.http_profile import HttpProfile
from tencentcloud.common.exception.tencent_cloud_sdk_exception import TencentCloudSDKException
from tencentcloud.tbp.v20190627 import tbp_client, models

def get_response(question):
    try:
        cred = credential.Credential("AKIDD2d1CHbsLh8MTKwEQSJvJPFqDUBTd1jF", \
                                      "Yu7A0BEAxAd7g4IIJsNiusjuFpFTUt5")
        httpProfile = HttpProfile()
        httpProfile.endpoint = "tbp.tencentcloudapi.com"

        clientProfile = ClientProfile()
        clientProfile.httpProfile = httpProfile
        client = tbp_client.TbpClient(cred, "", clientProfile)

        req = models.TextProcessRequest()
        params = {
            "BotId": "1e821b18-dd0a-4ec1-90d1-e4258a4dcbb7",
            "BotEnv": "release",
            "InputText": question,
            "TerminalId": "None"
        }
        req.from_json_string(json.dumps(params))

        resp = client.TextProcess(req)
        res = eval(resp.to_json_string())
        print(res['ResponseMessage']['GroupList'][0]['Content'])
        return res['ResponseMessage']['GroupList'][0]['Content']

    except TencentCloudSDKException as err:
```

```
    print(err)
```

luyin.py 动态录音功能

```
# -*- coding: utf-8 -*-
# @Time    : 18-10-16 下午 12:20
# @Author  : Felix Wang

import pyaudio
import numpy as np
from scipy import fftpack
import wave

# 录音
# 录音必须安装 portaudio 模块，否则会报错
# http://portaudio.com/docs/v19-doxydocs/compile_linux.html

def recording(filename, time=0, threshold=500):
    """
    :param filename: 文件名
    :param time: 录音时间,如果指定时间, 按时间来录音, 默认为自动识别是否结束录音
    :param threshold: 判断录音结束的阈值
    :return:
    """

    CHUNK = 1024 # 块大小
    FORMAT = pyaudio.paInt16 # 每次采集的位数
    CHANNELS = 1 # 声道数
    RATE = 16000 # 采样率: 每秒采集数据的次数
    RECORD_SECONDS = time # 录音时间
    WAVE_OUTPUT_FILENAME = filename # 文件存放位置

    p = pyaudio.PyAudio()
    stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True,
frames_per_buffer=CHUNK)
```

```

print("* 等待语音输入中...")

frames = []
if time > 0:
    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)
else:
    stopflag = 0
    stopflag2 = 0
    begin = False
    while True:
        data = stream.read(CHUNK)
        rt_data = np.frombuffer(data, np.dtype('<i2'))
        # print(rt_data*10)
        # 傅里叶变换
        fft_temp_data = fftpack.fft(rt_data, rt_data.size, overwrite_x=True)
        fft_data = np.abs(fft_temp_data)[0:fft_temp_data.size // 2 + 1]

        # 测试阈值，输出值用来判断阈值
        # print(sum(fft_data) // len(fft_data))

        # 判断麦克风是否停止，判断说话是否结束，# 麦克风阈值，默认 7000
        if sum(fft_data) // len(fft_data) > threshold:
            stopflag += 1
        else:
            stopflag2 += 1
        oneSecond = int(RATE / CHUNK)
        if stopflag2 + stopflag > oneSecond:
            if stopflag2 > oneSecond // 3 * 2:
                if begin:
                    break
                else:
                    stopflag2 = 0

```

```

stopflag = 0
else:
    stopflag2 = 0
    stopflag = 0
    if not begin:
        begin = True
        print("* 录音中...")
    frames.append(data)
print("* 录音结束")
stream.stop_stream()
stream.close()
p.terminate()
with wave.open(WAVE_OUTPUT_FILENAME, 'wb') as wf:
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))

if __name__=='__main__':
    # recording('ppp.wav', time=5) # 按照时间来录音, 录音 5 秒
    recording('ppp.wav') # 没有声音自动停止, 自动停止

```

Speech_detected.py 语音唤醒功能

```

import speech_recognition
import speech_synthesis
import os
import get_response
import luyin
import Main
import LED
import RPi.GPIO as GPIO

```

```
import snowboydecoder
import sys
import signal

interrupted = False
detect_sleepy_thread = None

def signal_handler(signal, frame):
    global interrupted
    interrupted = True

def interrupt_callback():
    global interrupted
    return interrupted

def detected():
    global detect_sleepy_thread
    print("-----")
    detect_sleepy_thread = Main.func_choose(detect_sleepy_thread)

def safebound_init(sender, receiver):
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(sender,GPIO.OUT)
    GPIO.setup(receiver,GPIO.OUT)
    GPIO.output(receiver,GPIO.LOW)
    GPIO.output(sender,GPIO.HIGH)

LED.mythread_run()
# safebound_init(20, 21)
```

```
if len(sys.argv) == 1:  
    print("Error: need to specify model name")  
    print("Usage: python demo.py your.model")  
    sys.exit(-1)  
  
model = sys.argv[1]  
  
# capture SIGINT signal, e.g., Ctrl+C  
signal.signal(signal.SIGINT, signal_handler)  
  
detector = snowboydecoder.HotwordDetector(model, sensitivity=0.5)  
print('Listening... Press Ctrl+C to exit')  
  
# main loop  
detector.start(detected_callback=detected,  
               interrupt_check=interrupt_callback,  
               sleep_time=0.03)  
  
detector.terminate()
```

雪碧解码器内部解码函数

```
#!/usr/bin/env python  
  
import collections  
import pyaudio  
import snowboydetect  
import time  
import wave  
import os  
import logging  
from ctypes import *  
from contextlib import contextmanager
```

```
logging.basicConfig()
logger = logging.getLogger("snowboy")
logger.setLevel(logging.INFO)

TOP_DIR = os.path.dirname(os.path.abspath(__file__))

RESOURCE_FILE = os.path.join(TOP_DIR, "resources/common.res")
DETECT_DING = os.path.join(TOP_DIR, "resources/ding.wav")
DETECT_DONG = os.path.join(TOP_DIR, "resources/dong.wav")

def py_error_handler(filename, line, function, err, fmt):
    pass

ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

@contextmanager
def no_alsa_error():
    try:
        asound = cdll.LoadLibrary('libasound.so')
        asound.snd_lib_error_set_handler(c_error_handler)
        yield
        asound.snd_lib_error_set_handler(None)
    except:
        yield
        pass

class RingBuffer(object):
    """Ring buffer to hold audio from PortAudio"""

    def __init__(self, size=4096):
        self._buf = collections.deque(maxlen=size)
```

```

def extend(self, data):
    """Adds data to the end of buffer"""
    self._buf.extend(data)

def get(self):
    """Retrieves data from the beginning of buffer and clears it"""
    tmp = bytes(bytearray(self._buf))
    self._buf.clear()
    return tmp

def play_audio_file(fname=DETECT_DING):
    """Simple callback function to play a wave file. By default it plays
    a Ding sound.

    :param str fname: wave file name
    :return: None
    """
    ding_wav = wave.open(fname, 'rb')
    ding_data = ding_wav.readframes(ding_wav.getnframes())
    with no_alsa_error():
        audio = pyaudio.PyAudio()
        stream_out = audio.open(
            format=audio.get_format_from_width(ding_wav.getsampwidth()),
            channels=ding_wav.getnchannels(),
            rate=ding_wav.getframerate(), input=False, output=True)
        stream_out.start_stream()
        stream_out.write(ding_data)
        time.sleep(0.2)
        stream_out.stop_stream()
        stream_out.close()
        audio.terminate()

```

```

class HotwordDetector(object):
    """
    Snowboy decoder to detect whether a keyword specified by `decoder_model`
    exists in a microphone input stream.

    :param decoder_model: decoder model file path, a string or a list of strings
    :param resource: resource file path,
    :param sensitivity: decoder sensitivity, a float or a list of floats.

        The bigger the value, the more sensitive the
        decoder. If an empty list is provided, then the
        default sensitivity in the model will be used.

    :param audio_gain: multiply input volume by this factor.
    :param apply_frontend: applies the frontend processing algorithm if True.

    """

    def __init__(self, decoder_model,
                 resource=RESOURCE_FILE,
                 sensitivity=[],
                 audio_gain=1,
                 apply_frontend=False):

        tm = type(decoder_model)
        ts = type(sensitivity)

        if tm is not list:
            decoder_model = [decoder_model]

        if ts is not list:
            sensitivity = [sensitivity]

        model_str = ",".join(decoder_model)

        self.detector = snowboydetect.SnowboyDetect(
            resource_filename=resource.encode(), model_str=model_str.encode())
        self.detector.SetAudioGain(audio_gain)

```

```

        self.detector.ApplyFrontend(apply_frontend)

        self.num_hotwords = self.detector.NumHotwords()

        if len(decoder_model) > 1 and len(sensitivity) == 1:
            sensitivity = sensitivity * self.num_hotwords

        if len(sensitivity) != 0:
            assert self.num_hotwords == len(sensitivity), \
                "number of hotwords in decoder_model (%d) and sensitivity "% \
                "(%d) does not match" % (self.num_hotwords, len(sensitivity))

            sensitivity_str = ",".join([str(t) for t in sensitivity])

        if len(sensitivity) != 0:
            self.detector.SetSensitivity(sensitivity_str.encode())

        self.ring_buffer = RingBuffer(
            self.detector.NumChannels() * self.detector.SampleRate() * 5)

    def start(self, detected_callback=play_audio_file,
             interrupt_check=lambda: False,
             sleep_time=0.03,
             audio_recorder_callback=None,
             silent_count_threshold=15,
             recording_timeout=100):
        """
        Start the voice detector. For every `sleep_time` second it checks the
        audio buffer for triggering keywords. If detected, then call
        corresponding function in `detected_callback`, which can be a single
        function (single model) or a list of callback functions (multiple
        models). Every loop it also calls `interrupt_check` -- if it returns
        True, then breaks from the loop and return.
        """

```

Start the voice detector. For every `sleep_time` second it checks the audio buffer for triggering keywords. If detected, then call corresponding function in `detected_callback`, which can be a single function (single model) or a list of callback functions (multiple models). Every loop it also calls `interrupt_check` -- if it returns True, then breaks from the loop and return.

:param detected_callback: a function or list of functions. The number of items must match the number of models in `decoder_model`.

```

:param interrupt_check: a function that returns True if the main loop
    needs to stop.

:param float sleep_time: how much time in second every loop waits.

:param audio_recorder_callback: if specified, this will be called after
    a keyword has been spoken and after the
    phrase immediately after the keyword has
    been recorded. The function will be
    passed the name of the file where the
    phrase was recorded.

:param silent_count_threshold: indicates how long silence must be heard
    to mark the end of a phrase that is
    being recorded.

:param recording_timeout: limits the maximum length of a recording.

:return: None
"""

self._running = True

def audio_callback(in_data, frame_count, time_info, status):
    self.ring_buffer.extend(in_data)
    play_data = chr(0) * len(in_data)
    return play_data, pyaudio.paContinue

with no_alsa_error():
    self.audio = pyaudio.PyAudio()
    self.stream_in = self.audio.open(
        input=True, output=False,
        format=self.audio.get_format_from_width(
            self.detector.BitsPerSample() / 8),
        channels=self.detector.NumChannels(),
        rate=self.detector.SampleRate(),
        frames_per_buffer=2048,
        stream_callback=audio_callback)

```

```

if interrupt_check():

    logger.debug("detect voice return")

    return


tc = type(detected_callback)

if tc is not list:

    detected_callback = [detected_callback]

if len(detected_callback) == 1 and self.num_hotwords > 1:

    detected_callback *= self.num_hotwords


assert self.num_hotwords == len(detected_callback), \
    "Error: hotwords in your models (%d) do not match the number of " \
    "callbacks (%d)" % (self.num_hotwords, len(detected_callback))

logger.debug("detecting...")

state = "PASSIVE"

while self._running is True:

    if interrupt_check():

        logger.debug("detect voice break")

        break

    data = self.ring_buffer.get()

    if len(data) == 0:

        time.sleep(sleep_time)

        continue

    status = self.detector.RunDetection(data)

    if status == -1:

        logger.warning("Error initializing streams or reading audio data")

#small state machine to handle recording of phrase after keyword

    if state == "PASSIVE":

        if status > 0: #key word found

```

```

        self.recordedData = []
        self.recordedData.append(data)
        silentCount = 0
        recordingCount = 0
        message = "Keyword " + str(status) + " detected at time: "
        message += time.strftime("%Y-%m-%d %H:%M:%S",
                               time.localtime(time.time()))
        logger.info(message)
        callback = detected_callback[status-1]
        if callback is not None:
            callback()

        if audio_recorder_callback is not None:
            state = "ACTIVE"
            continue

    elif state == "ACTIVE":
        stopRecording = False
        if recordingCount > recording_timeout:
            stopRecording = True
        elif status == -2: #silence found
            if silentCount > silent_count_threshold:
                stopRecording = True
            else:
                silentCount = silentCount + 1
        elif status == 0: #voice found
            silentCount = 0

        if stopRecording == True:
            fname = self.saveMessage()
            audio_recorder_callback(fname)
            state = "PASSIVE"
            continue

```

```

        recordingCount = recordingCount + 1
        self.recordedData.append(data)

    logger.debug("finished.")

def saveMessage(self):
    """
    Save the message stored in self.recordedData to a timestamped file.
    """

    filename = 'output' + str(int(time.time())) + '.wav'
    data = b"".join(self.recordedData)

    #use wave to save data
    wf = wave.open(filename, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(self.audio.get_sample_size(
        self.audio.get_format_from_width(
            self.detector.BitsPerSample() / 8)))
    wf.setframerate(self.detector.SampleRate())
    wf.writeframes(data)
    wf.close()
    logger.debug("finished saving: " + filename)
    return filename

def terminate(self):
    """
    Terminate audio stream. Users can call start() again to detect.
    :return: None
    """

    self.stream_in.stop_stream()
    self.stream_in.close()
    self.audio.terminate()

```

```
self._running = False
```

snowboydetect.py snowboy 内部检测函数

```
# This file was automatically generated by SWIG (http://www.swig.org).
# Version 3.0.12
#
# Do not make changes to this file unless you know what you are doing--modify
# the SWIG interface file instead.

from sys import version_info as _swig_python_version_info
if _swig_python_version_info >= (2, 7, 0):
    def swig_import_helper():
        import importlib
        pkg = __name__.rpartition('.')[0]
        mname = '.'.join((pkg, '_snowboydetect')).lstrip('.')
        try:
            return importlib.import_module(mname)
        except ImportError:
            return importlib.import_module('_snowboydetect')
    _snowboydetect = swig_import_helper()
    del swig_import_helper
elif _swig_python_version_info >= (2, 6, 0):
    def swig_import_helper():
        from os.path import dirname
        import imp
        fp = None
        try:
            fp, pathname, description = imp.find_module('_snowboydetect', [dirname(__file__)])
        except ImportError:
            import _snowboydetect
            return _snowboydetect
        try:
            _mod = imp.load_module('_snowboydetect', fp, pathname, description)
        finally:
```

```

finally:
    if fp is not None:
        fp.close()
    return _mod

_snowboydetect = swig_import_helper()
del swig_import_helper

else:
    import _snowboydetect
del _swig_python_version_info

try:
    _swig_property = property
except NameError:
    pass # Python < 2.2 doesn't have 'property'.

try:
    import builtins as __builtin__
except ImportError:
    import __builtin__

def __swig_setattr_nondynamic(self, class_type, name, value, static=1):
    if (name == "thisown"):
        return self.this.own(value)
    if (name == "this"):
        if type(value).__name__ == 'SwigPyObject':
            self.__dict__[name] = value
            return
    method = class_type.__swig_setmethods__.get(name, None)
    if method:
        return method(self, value)
    if (not static):
        if __newclass:
            object.__setattr__(self, name, value)

```

```

        else:
            self.__dict__[name] = value
    else:
        raise AttributeError("You cannot add attributes to %s" % self)

def __swig_setattr(self, class_type, name, value):
    return __swig_setattr_nondynamic(self, class_type, name, value, 0)

def __swig_getattr(self, class_type, name):
    if (name == "thisown"):
        return self.this.own()
    method = class_type.__swig_getmethods__.get(name, None)
    if method:
        return method(self)
    raise AttributeError("%s object has no attribute '%s'" % (class_type.__name__, name))

def __swig_repr__(self):
    try:
        strthis = "proxy of " + self.this.__repr__()
    except __builtin__.Exception:
        strthis = ""
    return "<%s.%s; %s>" % (self.__class__.__module__, self.__class__.__name__, strthis)

try:
    _object = object
    _newclass = 1
except __builtin__.Exception:
    class _object:
        pass
    _newclass = 0

```

```
class SnowboyDetect(_object):
    __swig_setmethods__ = {}
    __setattr__ = lambda self, name, value: _swig setattr(self, SnowboyDetect, name, value)
    __swig_getmethods__ = {}
    __getattr__ = lambda self, name: _swig getattr(self, SnowboyDetect, name)
    __repr__ = _swig_repr

    def __init__(self, resource_filename, model_str):
        this = _snowboydetect.new_SnowboyDetect(resource_filename, model_str)
        try:
            self.this.append(this)
        except __builtin__.Exception:
            self.this = this

    def Reset(self):
        return _snowboydetect.SnowboyDetect_Reset(self)

    def RunDetection(self, *args):
        return _snowboydetect.SnowboyDetect_RunDetection(self, *args)

    def SetSensitivity(self, sensitivity_str):
        return _snowboydetect.SnowboyDetect_SetSensitivity(self, sensitivity_str)

    def SetHighSensitivity(self, high_sensitivity_str):
        return _snowboydetect.SnowboyDetect_SetHighSensitivity(self, high_sensitivity_str)

    def GetSensitivity(self):
        return _snowboydetect.SnowboyDetect_GetSensitivity(self)

    def SetAudioGain(self, audio_gain):
        return _snowboydetect.SnowboyDetect_SetAudioGain(self, audio_gain)
```

```

def UpdateModel(self):
    return _snowboydetect.SnowboyDetect_UpdateModel(self)

def NumHotwords(self):
    return _snowboydetect.SnowboyDetect_NumHotwords(self)

def ApplyFrontend(self, apply_frontend):
    return _snowboydetect.SnowboyDetect_ApplyFrontend(self, apply_frontend)

def SampleRate(self):
    return _snowboydetect.SnowboyDetect_SampleRate(self)

def NumChannels(self):
    return _snowboydetect.SnowboyDetect_NumChannels(self)

def BitsPerSample(self):
    return _snowboydetect.SnowboyDetect_BitsPerSample(self)
    __swig_destroy__ = _snowboydetect.delete_SnowboyDetect
    __del__ = lambda self: None

SnowboyDetect_swigregister = _snowboydetect.SnowboyDetect_swigregister
SnowboyDetect_swigregister(SnowboyDetect)

class SnowboyVad(_object):
    __swig_setmethods__ = {}
    __setattr__ = lambda self, name, value: __swig setattr__(self, SnowboyVad, name, value)
    __swig_getmethods__ = {}
    __getattr__ = lambda self, name: __swig getattr__(self, SnowboyVad, name)
    __repr__ = __swig_repr__

    def __init__(self, resource_filename):
        this = _snowboydetect.new_SnowboyVad(resource_filename)
        try:
            self.this.append(this)

```

```

except __builtin__.Exception:
    self.this = this

def Reset(self):
    return _snowboydetect.SnowboyVad_Reset(self)

def RunVad(self, *args):
    return _snowboydetect.SnowboyVad_RunVad(self, *args)

def SetAudioGain(self, audio_gain):
    return _snowboydetect.SnowboyVad_SetAudioGain(self, audio_gain)

def ApplyFrontend(self, apply_frontend):
    return _snowboydetect.SnowboyVad_ApplyFrontend(self, apply_frontend)

def SampleRate(self):
    return _snowboydetect.SnowboyVad_SampleRate(self)

def NumChannels(self):
    return _snowboydetect.SnowboyVad_NumChannels(self)

def BitsPerSample(self):
    return _snowboydetect.SnowboyVad_BitsPerSample(self)
__swig_destroy__ = _snowboydetect.delete_SnowboyVad
__del__ = lambda self: None

SnowboyVad_swigregister = _snowboydetect.SnowboyVad_swigregister
SnowboyVad_swigregister(SnowboyVad)

# This file is compatible with both classic and new-style classes.

```

speech_recognition.py 语音识别功能

```

#_*_ coding:UTF-8 _*_
# @author: jacky

```

```
# 百度云语音识别 Demo，实现对本地语音文件的识别。  
# 需安装好 python-SDK，录音文件不超过 60s，文件类型为 wav 格式。  
# 音频参数需设置为 单通道 采样频率为 16K PCM 格式 可以先采用官方音频进行测试  
  
# 导入 AipSpeech AipSpeech 是语音识别的 Python SDK 客户端  
from aip import AipSpeech  
import os  
import importlib,sys  
  
importlib.reload(sys)  
#sys.setdefaultencoding('utf8')  
  
''' 你的 APPID AK SK 参数在申请的百度云语音服务的控制台查看'''  
APP_ID = '23476240'  
API_KEY = 'XXH3nkO18oDqrerZTuvlMyc3'  
SECRET_KEY = 'PMf0mbamURxetU1aoxH5uXtvzi36uj7'  
  
# 新建一个 AipSpeech  
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)  
  
# 读取文件  
def get_file_content(filePath):    #filePath 待读取文件名  
    with open(filePath, 'rb') as fp:  
        return fp.read()  
  
# 语音识别  
def stt(filename, lan="中文"):      # 语音识别  
    # 识别本地文件  
    piddict = {"中文":1537, "英文":1737, "英语":1737}  
    print('语言码是', pidict[lan])  
    result = client.asr(get_file_content(filename),  
                        'wav',
```

```
        16000,
        {'dev_pid': piddict[lan]},      # dev_pid 参数表示识别的语言类型
1536 表示普通话
    )
# print(result)

# 解析返回值，打印语音识别的结果
if result['err_msg']=='success.':

    word = result['result'][0].encode('utf-8')      # utf-8 编码

    if word!="":

        if word[len(word)-3:len(word)]=='， ':

            #print(word[0:len(word)-3])

            with open('demo.txt','w') as f:

                f.write(word[0:len(word)-3])

            f.close()

            return word[0:len(word)-3]

        else:

            #print(bytes.decode(word))

            with open('demo.txt','w') as f:

                f.write(bytes.decode(word))

            f.close()

            return bytes.decode(word)

        else:

            print ("音频文件不存在或格式错误")

    else:

        print ("错误")

return None

# main 函数 识别本地录音文件 yahboom.wav
if __name__ == '__main__':
    stt('test.wav')
```

speech_synthesis.py 语音合成功能

```
#_*_ coding:UTF-8 _*_
# @author: zdl
# 百度云语音合成 Demo，实现对本地文本的语音合成。
# 需安装好 python-SDK，待合成文本不超过 1024 个字节
# 合成成功返回 audio.mp3 否则返回错误代码

# 导入 AipSpeech AipSpeech 是语音识别的 Python SDK 客户端
from aip import AipSpeech
import os

''' 你的 APPID AK SK 参数在申请的百度云语音服务的控制台查看'''
APP_ID = '23476319'
API_KEY = '6czQuWV4tavX9X0Gw2EHH4v5'
SECRET_KEY = 'LIpEcVha2ib082tlpoS0H9OjYZM3jGMp'

# 新建一个 AipSpeech
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)

# 将本地文件进行语音合成
def tts(filename, isstr=False):
    if isstr:
        word = filename
    else:
        f = open(filename,'r')
        command = f.read()
        if len(command) != 0:
            word = command
        f.close()
    if not word:
        return None
    result = client.synthesis(word,'zh',1, {
        'vol': 5,'per':0,
```

```

        })

    print('语音内容:', word)

# 合成正确返回 audio.mp3, 错误则返回 dict

    if not isinstance(result, dict):

        with open('tts_result.mp3', 'wb') as f:

            f.write(result)

            f.close()

        return 'tts_result.mp3'

    return None

# main

if __name__ == '__main__':

    #tts('demo.txt', False)

    tts('您好，请问有什么可以帮到您?', True)

    os.system('mplayer audio.mp3')

```

translation.py 中译英功能

```

import json

from tencentcloud.common import credential

from tencentcloud.common.profile.client_profile import ClientProfile

from tencentcloud.common.profile.http_profile import HttpProfile

from tencentcloud.common.exception.tencent_cloud_sdk_exception import

TencentCloudSDKException

from tencentcloud.tmt.v20180321 import tmt_client, models


def translation(content, source="zh", target="en"):

    try:

        cred      =      credential.Credential("AKIDD2d1CHbsLh8MTKwEQSJvJPFqDUbTd1fF",

"Yu7A0BEAxAd7g4IJsNiusjuFpFTUt5")

        httpProfile = HttpProfile()

        httpProfile.endpoint = "tmt.tencentcloudapi.com"

```

```

clientProfile = ClientProfile()

clientProfile.httpProfile = httpProfile

client = tmt_client.TmtClient(cred, "ap-shanghai", clientProfile)

req = models.TextTranslateRequest()

params = {

    "SourceText": content,
    "Source": source,
    "Target": target,
    "ProjectId": 0
}

req.from_json_string(json.dumps(params))

resp = client.TextTranslate(req)
res = eval(resp.to_json_string())
return res["TargetText"]

except TencentCloudSDKException as err:
    print(err)

```

weather.py 天气查询功能

```

import requests

def get_weather(city):

    try:

        url = 'http://wthrcdn.etouch.cn/weather_mini?city=' + city
        respj = requests.get(url).json()
        data = respj.get('data').get('forecast')
        content = '今天' + city + '的天气。' + data[0].get('type') + '。' + \
                  '最低温度' + data[0].get('low').split()[1] + '。' + \
                  '最高温度' + data[0].get('high').split()[1] + '。' + \
                  data[0].get('fengxiang') + str(data[0].get('fengli'))[9:11]
    
```

```

content2 = '明天' + city + '的天气。' + data[1].get('type') + '。' + \
    '最低温度' + data[1].get('low').split()[1] + '。' + \
    '最高温度' + data[1].get('high').split()[1] + '。' + \
    data[1].get('fengxiang') + str(data[1].get('fengli'))[9:11]

return content + '。' + content2

except:
    return None

if __name__ == '__main__':
    print(get_weather('杭州'))

```

chat.py 闲聊功能

```

import json

from tencentcloud.common import credential

from tencentcloud.common.profile.client_profile import ClientProfile

from tencentcloud.common.profile.http_profile import HttpProfile

from tencentcloud.common.exception.tencent_cloud_sdk_exception import
TencentCloudSDKException

from tencentcloud.nlp.v20190408 import nlp_client, models

def chat(query):

    try:
        cred      = credential.Credential("AKIDD2d1CHbsLh8MTKwEQSJvJPFqDUbTd1jF",
"Yu7A0BEAxAd7g4IIJsNiusjuFpFTUt5")

        httpProfile = HttpProfile()
        httpProfile.endpoint = "nlp.tencentcloudapi.com"

        clientProfile = ClientProfile()
        clientProfile.httpProfile = httpProfile
        client = nlp_client.NlpClient(cred, "ap-guangzhou", clientProfile)

        req = models.ChatBotRequest()
        params = {
            "Query": query

```

```
}

req.from_json_string(json.dumps(params))

resp = client.ChatBot(req)
res = eval(resp.to_json_string())
print(res["Reply"])
return res["Reply"]

except TencentCloudSDKException as err:
    print(err)
```

4.2 执行结果

详见同文件目录下的视频文件夹内的结果视频。

五、程序调试说明和实验感想

5.1 调试说明

5.1.1 指纹模块调试说明

- 调试说明：一开始采用串口通信进行了很长一段时间，但是串口通信始终接收不到收回包，可以之后再进行尝试，解决方案就是使用 ch340 改变思路通过 usb 而绕过了串口。在对端口配置的时候，如果因为不熟悉而输错了端口导致端口甚至不是一个可以启用的端口，那么 minicom 会因此不能正常启用，这时候就需要通过 minicom -s 直接绕过启动进行配置。
- 特点：采用上位机系统和树莓派两个系统联合操作，不同系统完成不同功能，很好地模拟了管理员和用户两个层面的设置。

5.2 实验特色

本次实验的主要特色有以下几点：

- 将语音模块与视觉模块相结合，使其能够提供更丰富且具有实际价值的功能。例如车宝所具有的疲劳检测功能，就是实际生活中非常实用的功能，既方便又安全。
- 通用功能高度可定制化，个性化。相比于目前市面上的车载控制设备，车宝支持一些特定

的实用功能，例如查询今天自己的汽车是否限行等，这是因为测报可以直接通过指纹，将用户的个人信息进行设定，从而使其成为用户专属的语音助手。

- **系统功能的拓展性强。**车宝在编程实现的工程中采用了模块化的思想，主函数完成了小派整体的运行逻辑，而其他的功能模块均通过类或函数的形式出现，用户可以十分简单地添加更多的拓展功能（例如使用其他的 API），系统整体的结构和维护性也更强。
- **系统鲁棒性强，用户体验较好。**系统的大多数模块中均加入了错误处理机制，同时设计了合理的使用逻辑，并在关键处添加了必要的延迟，以保证系统不会出现卡死的情况。
- **系统使用 python3 编写，考虑了对未来的支持。**在正文中已经指出在 2020 年 python 官方就将停止对 python2 的进一步维护工作，以推动用户从 python2 到 python3 的转变。目前可以参考到的绝大多数教程都是基于 python2 的，因此在本次实验中，所有的模块都在 python3 下做了相应的调整和重建构建。

5.3 总结与展望

- **语音模块**

语音模块最大的遗憾就是无法定制唤醒词为——车宝，所以展望部分便是希望未来能对唤醒词进行特制，以支持用户多元化的唤醒需求。

- **视觉模块**

由于树莓派运行一直照片进行识别分析的时间太长（硬件限制），使得实际测试中的视频帧率不到 30 帧，以后我们希望使用运行内存更大的树莓派板子，来更好地支持我们疲劳检测的功能实施。