本科毕业论文(设计)

基于树莓派与OpenCV的驾驶员疲劳检测系统

学　　院	物理与电子信息工程学院
专　　业	电子信息工程专业
班　　级	2018级B班
学生姓名	许超
学　　号	20183911436
指导教师	张悦（讲师）
答辩日期	2022年5月18日

基于树莓派与 OpenCV 的驾驶员疲劳检测系统

摘要：在如今这样交通发达，路况复杂的时代，人们取得驾驶执照已不再是保证道路安全的唯一保障，我们更关注驾驶员驾车时的状态和车辆是否具有预警系统。根据道路安全相关部门的统计，在中国约 20%的交通事故是由驾驶员疲劳引起的[1]。由于人在疲劳时相关特征可以通过检测人脸特征进行判断，所以研究一款能有效判断出人脸疲劳状态和预防驾驶员疲劳驾驶的嵌入式系统是可行的。

从处理器的性能和价格上对比了各大嵌入式系统，本系统将选用树莓派第三代（Raspberry Pi 3）作为嵌入式开发平台，使用 Python 的第三方视觉库将人脸转换为 68 个特征点，使用 OpenCV 图像处理库和 Dlib 深度学习算法进行人脸特征值提取，再使用多线程编程配合语音和警报灯模块达到定时和预警的目的。本方法具有易于实现、易于编程、识别率高的特点，经过多次实验与调试，本系统人脸识别准确率较高且硬件运行稳定。

关键词：树莓派；OpenCV；疲劳检测；预警系统

Driver fatigue detection system based on Raspberry Pi and OpenCV

Abstract: In today's era of developed traffic and complex road conditions, obtaining a driver's license is no longer the only guarantee for road safety. We pay more attention to the driver's driving state and whether the vehicle has an early warning system. According to the statistics of relevant departments of road safety, about 20% of traffic accidents in China are caused by driver fatigue [1]. Since the relevant features of people when they are fatigued can be judged by detecting the facial features, it is feasible to develop an embedded system that can effectively judge the fatigue state of the face and prevent the driver from fatigued driving.

This system will choose Raspberry Pi 3 as the embedded development platform, and use Python's third-party vision library to convert face into 68 feature points. OpenCV image processing library and Dlib deep learning algorithm are used to extract face feature values, and multithreaded programming is used to achieve the purpose of timing and warning with voice and alarm light module. The method is easy to implement, easy to program and high recognition rate. After many experiments and debugging, the system has high accuracy and stable hardware operation.

Keywords: Raspberry pi; OpenCV; Fatigue detection; Warning system

目　录

摘要.....I
Abstract.....II
目　录.....III
1 绪论.....1
1.1 疲劳驾驶的研究背景与现状.....1
1.2 嵌入式 Linux 系统以及树莓派简介.....2
1.3 Python 语言简介.....5
2 基于人脸检测的疲劳驾驶检测技术.....7
2.1 开源计算机视觉库 OpenCV 简介.....7
2.2 疲劳驾驶检测算法.....7
3 疲劳驾驶检测与预警功能实现.....11
3. 1 疲劳检测.....11
3.2 音频实现.....17
3.3 疲劳驾驶检测系统的硬件整体实现.....17
4 系统测试.....22
4.1 程序调试.....22
4.2 系统性能测试.....23
4.3 本章小结.....23
5 结论与展望.....24
5.1 全文总结.....24
5.2 后续工作展望.....24
参考文献.....25
附录 A 程序代码.....26
附录 B 系统硬件连接原理图.....32
致 谢.....33

1 绪论

1.1 疲劳驾驶的研究背景与现状

1.1.1 交通安全与疲劳驾驶

随着现代化城市的发展，道路行驶车辆数量逐年增加，交通事故一直是各国面临的严重问题。据不完全统计，过去十年每年全世界约有 50 万左右的老百姓死于交通事故[2]。根据欧美国家更详细的交通事故统计，人为因素导致的交通事故占所有交通事故的 80%-90%[3]。美国印第安纳大学在对交通事故产生原因做了更深入的研究，其数据表明有 85%的事故由人为因素导致，有约 10%的因素来自车辆因素，最后的 5%左右是环境因素造成的如图 1-1 所示。因此，减少人为因素导致的交通事故能有效减少死亡人数，而其中预防疲劳驾驶则是本系统要解决的问题。

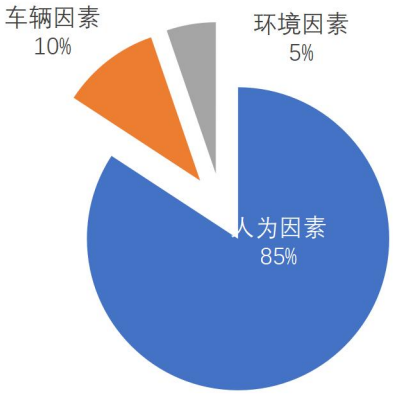


图 1-1 交通事故原因统计

对于疲劳的研究至今仍无一个准确定义，主要说的是驾驶员连续行驶一定时间后，在生理和心理功能方面上积极性的减退，客观地显示动作方面的迟缓或误操作的行为。如果驾驶员最近睡眠质量差或不足，且车辆行驶时间长，则此时容易疲劳。疲劳影响范围包括但不限于注意力、视觉、思维、判断、决定和运动等。最终可导致车辆轨道偏离、追尾、急刹以及出现影响其它车辆行驶的动作。

据英国一家研究机构统计，在疲劳驾驶导致的交通事故里，有 43%是由于连

续驾驶将过长[4]，Lisperetal. 的研究结果表明了连续驾驶时间长度和事故发生次数的关系如图 1−2。

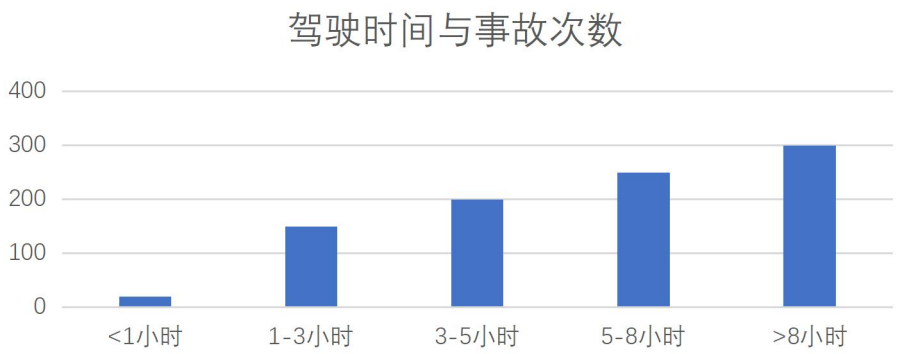


图 1-2 连续驾车时间与交通事故次数

本系统的设计旨在预防疲劳驾驶，在驾驶员出现打哈欠和闭眼时间过长的情况下予以提醒，并在检测到驾驶员严重疲劳时开启危险警报灯以对周边车辆起到警示作用，以避免更加严重的交通事故。

1.1.2 疲劳驾驶应对策略

早在上个世纪 60 年代，西方国家已开始应对驾驶疲劳，一直到 90 年代，各国取得了不同的进展与成果，其中具有代表性的日本 DAS 路面预警系统和美国 DDDS 系统等[5]，但由于其嵌入性或性价比都不很理想，没有广泛应用。

经调查相关文献及官方文档，驾驶员对车辆操控能力的下降的主要因素之一是驾驶员处于疲劳状态，主要体现在对挡位的切换、油门的控制、方向盘的调整等，驾驶员驾驶行为的变化将直接导致车辆行为的变化。然而检测驾驶员行为技术复杂且难实现，但由于人脸疲劳信息丰富且易检测，对驾驶员人脸检测的算法也相对成熟，且可行性较高，本系统通过模拟实验，分析出人脸疲劳特征，并在此特征基础上设计了可以对驾驶员预警和对周边车辆预警的嵌入式疲劳检测系统。

1.2 嵌入式 Linux 系统以及树莓派简介

自 1991 年 Linux 操作系统由林纳斯·托瓦兹发布以来，其底层代码开源，

任何机构甚至个人均可以自由地修改和再发布，自其发布以来 Linux 操作系统已被设计得功能强大且更加完善。可以运行在众多硬件平台上如：X86、NEC、Raspberry Pi OS 等[6]，加上可以支持定制和源代码开放的特点，在与各大操作系统的竞争中占有一席之地，Linux 在嵌入式操作系统上取得巨大进展离不开各大研发机构或者各大高校的参与研发。

树莓派（英文全称“Raspberry Pi”，简称 RPI）是树莓派基金会与博通联合开发的一系列小型单板计算机（SBC）[7]，最初应用在学校计算机课程上，是性价比高的教学设备。由于其价格亲民、如信用卡般大小如图 1-3，且有强大的性能。再搭配上 HDMI 和 USB 设备，它通常被计算机和电子爱好者使用。

图 1-3 卡片大小的树莓派单板计算机

树莓派目前发展到了第四代，由于第三代性能足以支持本系统，且性价比较高，本实验将采用树莓派第三代。众所周知，硬件是系统的基础。各种想法的实现离不开硬件，且树莓派最初为学校教学所设计的一台小型计算机，因此其性能上不输其它嵌入式设计平台。

第三代树莓派 Model B 与 2016 年 2 月发布，配备 1.2 GHz 64 位四核 ARM Cortex-A53 处理器、板载 802.11n Wi-Fi、蓝牙和 USB 启动功能[8]，可以更顺利地处理多线程任务。它还有四个 usb2.0 接口和百兆网络端口，便于访问数据网络。作为嵌入式设备，还搭载了 40 个 GPIO 针脚和 SPI 总线用于各传感器的接入如图 1-4、1-5，提供了摄像头和 TFT 显示器接口可接入红外补光摄像头如图

1-6，以及具有 HDMI 高清视频输出接口和 3.5 英寸音频接口等。

综上，树莓派第三代已具有本系统所需的图像处理器性能和丰富的接口，加上其价格并不昂贵，足以胜任人脸检测加预警的功能。

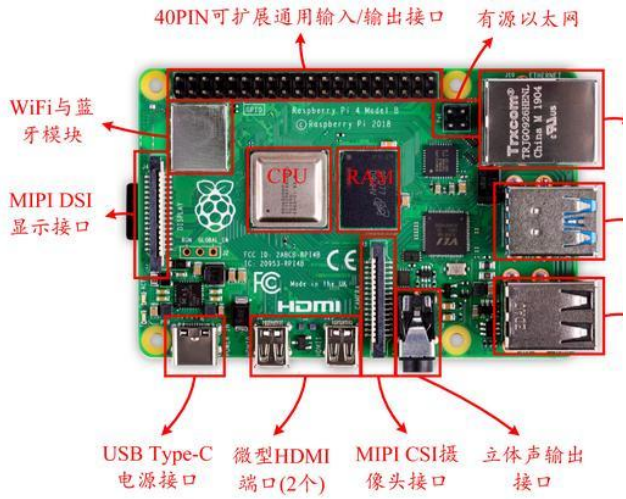


图 1-4 树莓派结构图

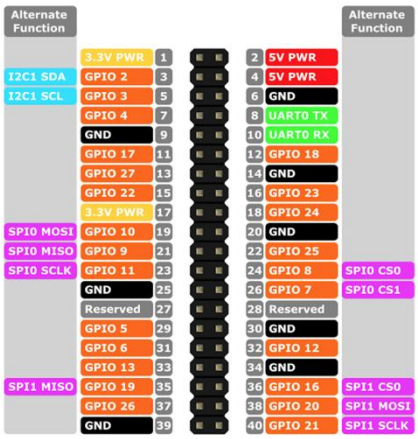


图 1-5 树莓派针脚图

图 1-6 红外摄像头模块

1.3 Python 语言简介

1.3.1 Python 语言的特点

Python 语言是一种面向对象的、解释型的、且更开源的脚本语言[9]。与其他嵌入式编程语言相比，Python 的语法更短，学习成本更低。整个代码是严格控制缩进的。标准库和第三方库相对完整，代码是开源的，可以创建业务应用程序。Python 可以充分利用大数据，并拥有许多用于处理大数据的开源库，使整体代码更简洁明了，为本系统视频流数据处理环节提供了便捷。

1.3.2 Python 的应用

图形处理：处理视频流实质上是处理视频流的每个帧，一帧即为一张特定格式的图形。Python 还拥有许多可用于处理图形的第三方库，如 OpenCV、Tkinter、PIL 等均为可以用于图形变换的高效率工具。

数据计算：目前有大量的数据处理库，例如 SciPy、NumPy。适合对视频流此类大数据的处理及计算。

系统编程：Python 有许多库用来系统处理。在 Linux 操作系统平台下提供许多用于系统维护和管理的 API 集成。本系统集成桌面环境为 Thonny，其支持 Windows、MAC、Linux 和其他平台，内嵌开发图形库，且 Thonny 对于刚接触 Python 编程相对友好，拥有更简洁的工作界面，如图 1-7 集成环境工作界面所示该窗口由顶部代码编辑器和底部 Shell 两个主窗口组成，在这里将直接显示代码结果输出。支持调试、语法错误检测等功能。

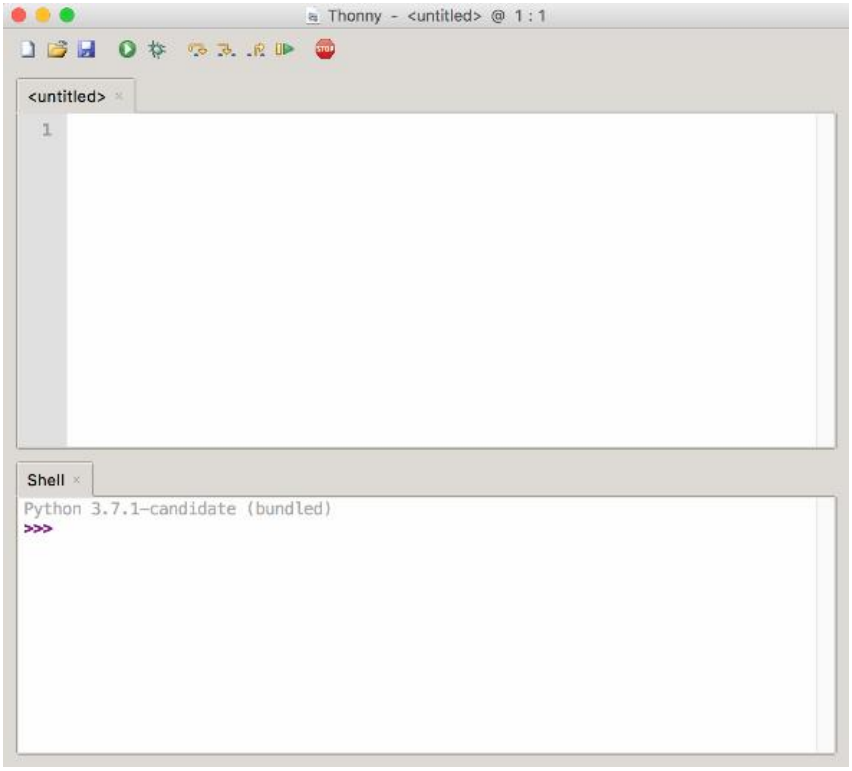


图 1-7 Thonny 工作界面

1.3.3 Python 版本历史

Python Software Foundation（PSF）曾支持两个主要版本，Python 2. x 和 Python 3. x，PSF 支持 Python 2，因为大量现有代码无法转发到 Python 3，因此在 2020 年 1 月后，PSF 已停止支持。

Python 3. 0 版本于 2008 年 12 月发布，为纠正早期版本的某些缺陷，此版本与以前的版本将不完全向后兼容，尽管如此它的许多主要功能已经被向后移植到 Python 2. 6. x 和 2. 7. x 版本系列中。Python 3 的发行版包括用于促进将 Python 2 代码自动转换为 Python 3 的实用程序。

由于人脸识别算法的迭代频次高，虽然 Python 语言不像其他语言向下兼容，但考虑到时效性，绝大多数算法类都是基于 Python 3. x 的，所以本系统采用 Python 3. 9 版本。

2 基于人脸检测的疲劳驾驶检测技术

2.1 开源计算机视觉库 OpenCV 简介

OpenCV（Open Source Computer Vision Library）是一个同时兼容多个操作系统并且代码开源的计算机视觉和机器学习软件库[10]。OpenCV 可用于实时图像处理、计算机视觉和模式识别程序。它不仅具有 Python 和 MATLAB 接口，还支持 Linux 操作系统。

该库拥有 2500 种优化算法的同时，还揽括了一整套先进的计算机视觉和机器学习算法。众所周知，大多数机器学习算法要求输入本质上是定量的，即数字。OpenCV 允许我们将机器学习技术应用于图像，但是，通常我们需要预处理和准备原始图像，以便将它们转换为机器学习算法有用和可用的特征。

2.2 疲劳驾驶检测算法

通常在疲劳事故前驾驶员的眨眼行为将会提前发生变化，且这些变化可识别，即可以通过分析每个帧来进行疲劳判断。因此测量眼睛开合状态能有效判别出驾驶员是否处于疲劳状态。同理，打哈欠时也标志着驾驶员开始进入疲劳与困倦的状态，因此驾驶员的嘴的特征也是疲劳检测的对象。本系统基于以上两个特征，对驾驶员的眼睛和嘴部进行综合的疲劳状态评判，若检测出驾驶员疲劳将触发后续的预警系统。本系统大致工作流程如图 2-1。

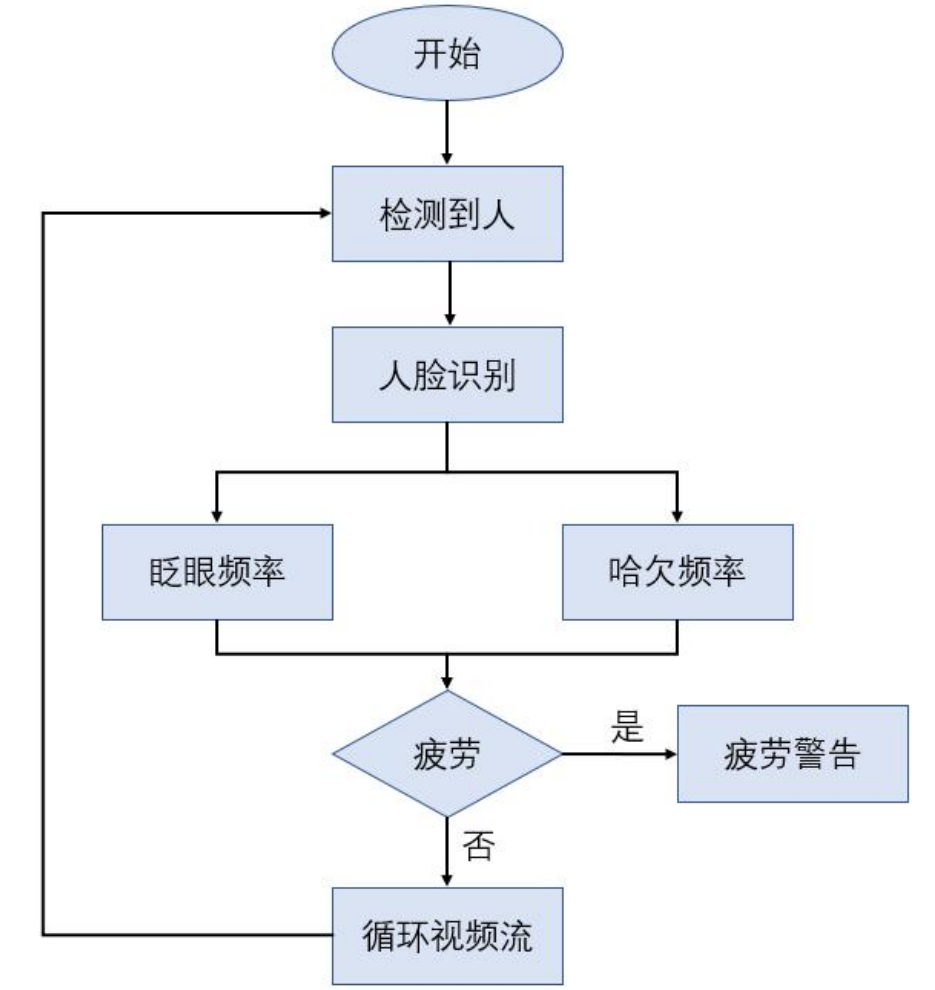


图 2-1 软件工作流程

2.2.1 基于 Dlib 库的 68 个关键点采集

Dlib 库作为一个开源的工具箱，装配了机器学习（ML）模块、深度学习（DL）模块和图像处理（GP）模块等[11]。本系统采用了该库中人脸关键点模型库 ShapePredictor68FaceLandmarks. dat 以及在实时摄像头 webcam 中检测出的人脸，随后返回人脸各个点对应的坐标和人脸框，68 个人脸关键点如图 2-2。

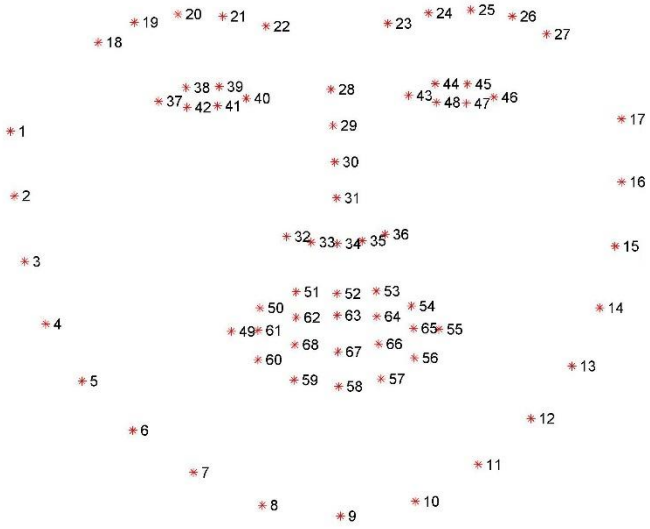


图 2-2 Dlib 人脸关键点 ids

利用这 68 个关键点计算眼睑长宽比（EAR），参照人眼关键点位图 2-3，相应的计算出计算嘴的开合比（YAWN），摄像头参数初始化大体流程如下：

（1）打开面部检测器并标记面部预测关键点；

（2）获取两个位置的检测器 get_Forward_face_predictor；

（3）获取脸部位置的特征检测器 Dlib_shape_predictor；

（4）获取左、右眼索引号；

（5）获取嘴部标志索引号；

（6）打开 cv2 本地摄像头。

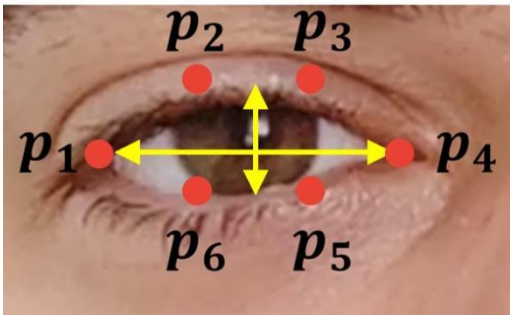


图 2-3 睁眼长宽比

2.2.2 Imutils

（1）Imutils 功能简介：

Imuutils 是一个基于 OpenCV 的软件包，用于 OpenCV 界面。可以简单地移动、缩放、旋转、缩放和其他图像操作。安装的前提是系统中已经安装了 NumPy、SciPy、Matplotlib 和 OpenCV，在树莓派终端使用 pip 安装，命令如下：

pip install imutils

随着 OpenCV 4 发布后主版本的升级，后台兼容性问题尤为突出，因此本系统使用了相应的函数和方法，以与主版本融合。

2.2.3 SciPy

SciPy（Scientific Python）作为用于工程领域常用的第三方软件包，利用该工具包可以对图像处理插值、积分、优化等。它能准确有效地计算 Numpy 矩阵，使 Numpy 和 SciPy 协同且高效地解决问题。

由众多不同科学计算领域子模块组成的 SciPy 其中包括但不限于：聚类算法（cluster）、快速傅里叶变换（fftpack）、输入输出（io）、线性代数（linalg）和空间数据结构和算法（spacial）等，本系统使用了其中矩阵距离计算函数 scipy. spatial. distance 以计算距离。

2.2.4 NumPy

NumPy（Numeric Python）的定义大致可以分为一下三种：

（1）NumPy 是 Python 的包装器。库由多个实例组成。使用 NumPy，以执行以下数组的数学和逻辑的运算。

（2）傅里叶变换和图形运算程序。

（3）一个综合函数，由线性代数和随机数生成的线性代数计算函数。

Numpy 通常与 SciPy 和 Matplotlib 一起使用。这种组合经常被用来取代较为流行的计算机平台 MATLAB。作为 MATLAB 的替代品，Python 现在是一种更现代、更全面的编程语言。

系统必须计算一个 n 维矩阵，并使用 Ndaray 对象。可用于访问空索引集中的项的缩放表和相同类型的项。类型是对象的实例，每个对象都有唯一的属性，类型可以是布尔类型，浮点数类型等。

2.2.5 Argparse

Argparse 为 Python 命令行分析包，该工具包可以轻松读取并解析命令行参数。使用时需人为添加定义必要的参数，Argparse 将找出并解析这些参数并自动生成一条帮助消息并发送给用户。

3 疲劳驾驶检测与预警功能实现

3. 1 疲劳检测

眼睛和嘴唇的形态是驾驶员疲劳的重要指标，本系统通过关键点坐标提取与计算，分别获得眼睛、嘴横纵比 EAR（Eye aspect ration）与 MAR（Mouth aspect ratio），与所设定的阈值作比较，最后确认驾驶员是否疲劳。当检测到疲劳时，系统报警模块将开始执行。

梯度直方图（HOG）是一种特征描述符，它可以将图像分割为多个单元，并通过获得像素直方图来创建特征描述器[12]。即使光线和背景发生变化，性能仍能保持在一个良好的状态。本系统中通过 Dlib 库获取人脸位置和特征，并将信息转换成数组形式，再使用 OpenCV 用于标记 68 个人脸关键点。

3.1.1 人眼信息识别

对已经获得的 68 个关键点位置的坐标，分别获取如图 2-3 所示的 p1、p2、p3、p4、p5、p6，这用于计算眼睛纵横比的 EAR 值[13]。

$$EAR = \frac{||p2 - p6|| \pm ||p3 - p5||}{2||p1 - p4||}$$

下面是返回单个眼睛的 EAR 值函数：

```
def EAY(eye)::
    p2_minus_p6 = dist.euclidean(eye[1], eye[5])
    p3_minus_p5 = dist.euclidean(eye[2], eye[4])
    p1_minus_p4 = dist.euclidean(eye[0], eye[3])
    ear = (p2_minus_p6 + p3_minus_p5) / (2.0 * p1_minus_p4)
    return ear
```

人眼状态处于如上一章节图 2-3 所示那样，处于正常睁开状态时，经过实验验证 EAR 值会处于正常的波动范围。在驾驶员闭眼过程中 EAR 值骤降，完全闭上眼时理论上如图 3-1，人眼完全闭合时 EAR 为零。

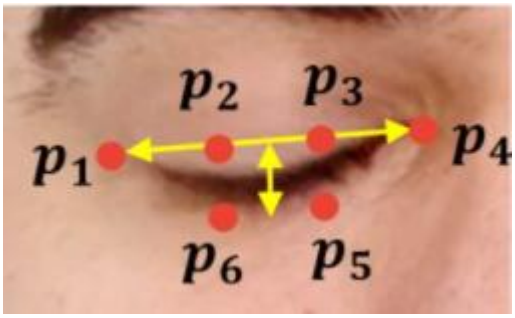


图 3-1 闭眼状态人眼长宽比

首先引入一些必要的 python 库开始：

```
import cv2
import dlib
import imutils
from imutils import face_utils
from scipy. spatial import distance as dist
```

然后声明一些必要的变量，这些变量将在后续的代码中经常会使用：

```
FACIAL_LANDMARK_PREDICTOR=
"shape_predictor_68_face_landmarks. dat"
MINIMUM_EAR = 0. 2
MAXIMUM_FRAME_COUNT = 10
```

其中 FACIAL_LANDMARK_PREDICTOR 为 Dlib 库提前训练好的人脸关键点预测器。其中模型获取可以借助 Dlib 官方网站下载。

以上 MINIMUN_EAR 值代表的是最小的人眼长宽比阈值，检测出人眼长宽比大于此阈值时，人眼将被标记为正常睁眼状态，否则将被标记为睁眼。由于个体差异以及光照强度的变化，此参数可以根据实际情况进行调优以达到准确检测的目的。此外，EAR 值并不代表着单只眼的长宽比，它是两个眼睛 EAR 取的平均结果。

对于 MAXIMUM_FRAME_COUNT，因为 EAR 值变化幅度大且快，其实驾驶员眨眼也会导致 EAR 值骤降，然而检测到 AER 值小于设定的阈值并不代表驾驶员处于疲劳状态。本系统经过多次试验，完成一次眨眼动作大概需要 1 至 3 帧。调查相关文献可知当驾驶员处于疲劳状态时，眨眼频率由于大脑活跃度下降相应变慢[14]。系统设定当十帧内 EAR 值仍小于最小阈值时，系统判定驾驶员为疲劳状态，并触发语音模块提醒驾驶员谨慎开车。

再实例化 Dlib 的 faceDetector 和 landmarkFinder，它们的作用分别是：检测图像中的人脸图像和在已检测到人脸的图像里找出这 68 个点，相关代码如下所示：

```
faceDetector = dlib.get_frontal_face_detector()
landmarkFinder =
dlib.shape_predictor(FACIAL_LANDMARK_PREDICTOR)
webcamFeed = cv2.VideoCapture(0)
```

接下来查找两只眼的关键点唯一标识 ids，如图 3-2 所示我们既能手动添加标识，也能使用 face_utils，只需要给它传入左眼或者右眼的参数即可。

图 3-2 人眼关键点 ids

```
(LeftEyeStart, leftEyeEnd) =
    face_utils.FACIALMARKS_IDXS["left_eye"]
```

（rightEyeStart, rightEyeEnd）=

face_utils. FACIAL_LANDMARKS_IDXS［“right_eye”］

最后我们再声明了 EYE_CLOSED_COUNTER 变量来记录当 EAR 值小于最小阈值的连续帧数。

```
EYE_CLOSED_COUNTER = 0
try:
    while True:
        (status, image) = webcamFeed.read()
        image = imutils.resize(image, width=800)
        grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        faces = faceDetector(grayImage, 0)
        for face in faces:
            faceLandmarks = landmarkFinder(grayImage, face)
            faceLandmarks = face_utils.shape_to_np(faceLandmarks)
            leftEye = faceLandmarks[leftEyeStart:leftEyeEnd]
            rightEye = faceLandmarks[rightEyeStart:rightEyeEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)
            ear = (leftEAR + rightEAR) / 2.0
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(image,[leftEyeHull],-1,(255,0,0), 2)
            cv2.drawContours(image,[rightEyeHull],-1,(255,0,0), 2)
            if ear < MINIMUM_EAR:
                EYE_CLOSED_COUNTER += 1
            else:
                EYE_CLOSED_COUNTER = 0
```

```
                cv2.putText(image, "EAR: {}".format(round(ear, 1)), (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                if EYE_CLOSED_COUNTER >=
                MAXIMUM_FRAME_COUNT:
                cv2.putText(image, "Drowsiness", (10, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.imshow("Frame", image)
                cv2.waitKey(1)
    except:
        pass
```

以上程序里我们对图像进行了如下操作：

（1）适当的放缩并转换为灰度图像。

（2）使用 faceDetector(grayImage, 0)来检测图像里的所有人脸。

（3）使用 landmarkFinder 在每张图上不断循环一检测出 68 个关键点 faceLandmarks = landmarkFinder(grayImage, face)。

（4）获取左眼和右眼的关键点坐标并且作为参数传入 eye_aspect_ratio()以获得左眼和右眼的 EAR 值：

```
leftEye = faceLandmarks[leftEyeStart:leftEyeEnd]
rightEye = faceLandmarks[rightEyeStart:rightEyeEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
```

（5）取 EAR 平均：

```
ear = (leftEAR + rightEAR) / 2.0
```

（6）如果当前帧下 EAR 值的平均小于最小阈值 MINIMUM_EAR，则增加变量 counter 计数加 1，因为我们只考虑连续帧的情况，如果有一帧大于 MINIMUM_EAR，则重置 counter 计数器：

```
If ear < MINIMUM_EAR:
    EYE_CLOSED_COUNTER += 1
Else:
```

EYE_CLOSED_COUNTER = 0

（7）相应的，如果 EAR 值小于 MINIMUM_EAR 并持续帧数大于等于指定的值，则系统判定驾驶员为疲倦状态：

if EYE_CLOSED_COUNTER >= MAXIMUM_FRAME_COUNTER:

cv2. putText(image, ”Drowsiness”, (10, 50), cv2. FONT_HERSH

EY_SIMPLEX, 0. 7, (0, 0, 255), 2)

当运行此程序，系统运行结果如图 3-3 所示：

图 3-3 EAR 值小于 MINIMUM_EYE 持续十帧，判断困倦

3.1.2 嘴部信息识别

在疲劳检测中，驾驶员嘴部同样能获取相关疲劳信息，打哈欠则属于嘴部重要的特征之一[15]。特别的，若驾驶员连续打哈欠或者打哈欠时间持续时间长更能说明此时驾驶员处于疲劳状态。相比于眼睛特征提取，嘴部特征提取相对来说较容易，即使外界光影环境复杂的情况下，本系统依然具有较高的识别率。

本系统通过设定双阈值即张口时间与张口度的方法来实现对打哈欠的行为判断。与人眼判别过程同理，利用嘴部关键点坐标计算长宽比 MAR：

$$MAR = \frac{||M_2 - M_6|| + ||M_3 - M_5||}{2||M_1 - M_4||}$$

3.2 音频实现

方式一：espeak 文本转语音输出，人声显得较为生硬，打开树莓派终端输入以下命令：

$sudo apt-get install espeak

数据目录：/usr/lib/arm-linux-gnueabihf/eapeak-data

$espeak -zh ‘Hello World’

方式二：百度 AI，由于百度 AI 语音需要 api 安装并且不是免费，为减少本系统的成本，暂不对百度 AI 予以应用。

方式三：相较于 espeak 语音库，libhaisantts. so 语音库人声更柔，避免驾驶员造成惊吓从而导致更严重的交通事故, 语音包下载指令如下：

wget http://simcommander. cn/download/haisantts-py. gz

解压压缩文件：

$tar xvfz haisantts-py. gz

跳转到该文件目录：

$cd haisantts-py

语音包的使用：

$python haisantts. py

#coding=utf-8

import ctypes

lib = ctypes. cdll. LoadLibrary(“. /libhaisantts. so”)

lib. startHaisanTTS(“请注意休休息”)

3.3 疲劳驾驶检测系统的硬件整体实现

本系统硬件部分的开发基于 Proteus 软件。该软件为 Labcenter 开发的用于电路分析与模拟系统，可模拟各种硬件电路。特别地，proteus8. 9 不仅支持 MCU 编程模拟，还支持第三代树莓派模拟。器件完整且易于使用。

该软件功能强大，具有众多优势：电路仿真、由单片机，树莓派及相关外围电路组成的系统仿真；虚拟设备有多种，如信号发生器、示波器和逻辑分析仪。

版本 8. 9 支持的 MCU 类型包括 68000 系列、8051 系列、AVR 系列、PIC16 系列、PIC18 系列、Z80 系列、HC11 系列和第三代树莓派及外围模拟设备。简而言之该软件版本集成器件丰富为本系统硬件模拟提供了可靠地支持。

3.3.1 硬件连接

本系统预警灯硬件部分连接如附录二所示。硬件主要由五个部分组成：时钟震荡（基于 555 集成定时器）、译码电路（基于 74ls38 译码器）、继电器与 LED 灯组、树莓派和 9V 直流电源模块。

（1）时钟震荡模块：本系统硬件部分时钟震荡模块是 NE555 集成电路组成的时钟脉冲发生器如图 3-4。通过改变图中电容器 C4 的电容大小可获得一个低频脉冲。若想获得目标频率的脉冲信号可通过调节电位器 RV1 的组织大小，如秒脉冲、1kHz、10kHz 等。

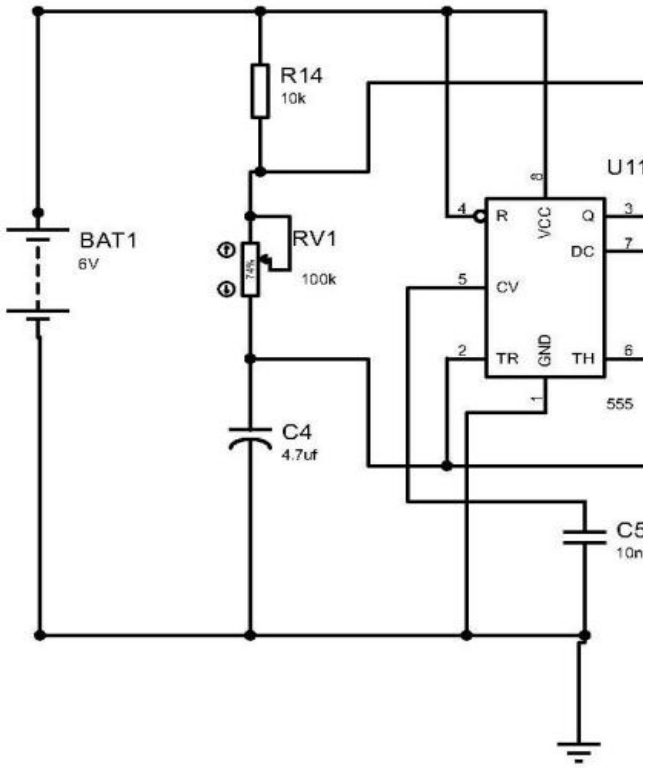


图 3-4 时钟震荡模块连线

（2）译码电路模块：本系统主要模拟汽车尾灯仅预警功能，未对汽车紧急刹车、左转向与右转向以及雾灯进行模拟。当树莓派实时摄像模块检测到驾驶员打哈欠频率高于阈值时，则触发危险警报灯，汽车尾灯将随脉冲信号闪烁，硬件原理设计如图 3-5。

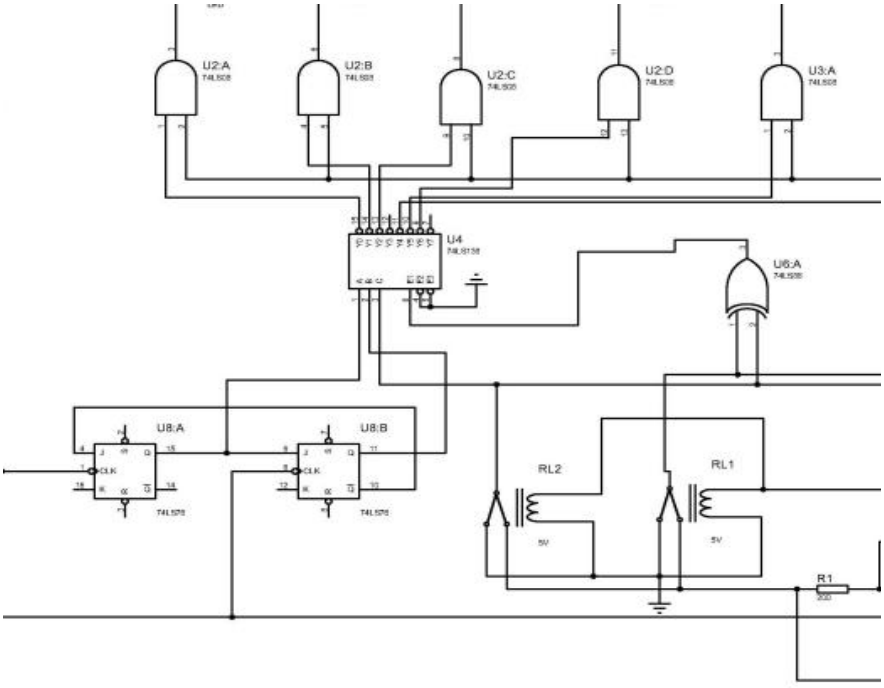


图 3-5 译码电路连线

电路工作原理：74LS138 的三个输入端子 A、B 和 C 分别连接到 Q_0、Q_1 和继电器高电位输出端口。当使能信号 E_1=0，74ls86 输出为 1，74ls138 输出端全为 1，Y_0、Y_1、Y_2、Y_4、Y_5、Y_6 输出也全为 1，指示灯全灭灯；当 E_1=0，74ls86 输出为 CP 时钟震荡信号时，指示灯随相应频率闪烁。

（3）继电器与 LED 灯组：继电器模块如图 3-6，继电器采用 5V 直流控制能直接响应树莓派 GPIO 输出的高点平（树莓派输出高电位约为 3. 3V），控制端口有三个接口，其中包含一个公共接口 COM 口，一个常闭接口 NC，一个常开接口 NO。本系统采用红、黄、白各两个 LED 灯，LED 发光效率高成本低使用寿命通常为 50000 小时, 以模拟真实的汽车尾灯设计，一组 LED 灯如图 3-7，

图 3-6 继电器模块

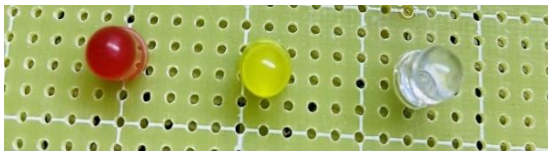


图 3-7 一组 LED

（4）树莓派与直流电源模块：树莓派模块已在本论文前面进行了详细的论述，对于直流电源模块，本系统采用 9V 直流电池替代供电，实物如图 3-8，而在真实的汽车电路系统中，可以通过变压器来控制电路电压。

图 3-8 9V 直流电源

3.3.2 参数阈值设置

本系统经过多次实验校准，设置眼睛长宽比阈值：EYE_AR_THRESH = 0. 25，设置连续帧帧数以判断闭眼时间阈值：EYE_AR_CONSEC_FRAME = 10，实时检测驾驶员眼部长宽比，当小于 EYE_AR_THRESH 并持续了 EYE_AR_CONSEC_FRAME 的帧数时，系统判定驾驶员疲劳并触发语音提示；设置嘴部长宽比阈值：YAWN_THRESH = 25，当驾驶员嘴部 YAWN 值大于 YAWN_THRESH 时，系统判定驾驶员已疲劳并触发语音提醒，特别地当一段时

间内检测到驾驶员打哈欠频次过高时，将触发汽车危险警报灯，灯组将随时钟振荡频率闪烁，以此来对周边车辆起到警示作用。

3.3.3 语音提示功能实现

对于瞌睡警告本系统设置两种语音，一个是驾驶员闭眼警告语音，创建线程并设置提示语：t1 = Thread(target=warning, args=('醒醒，醒醒！',))；第二个是哈欠警告提示语：t2 = Thread(target=warning, args=('困了吧，停车休息一下吧！',)。

3.3.4 连续驾驶时间监测

本系统创建了一个线程，该线程在系统通电程序运行开始即在树莓派后台运行，每个固定的时间提醒一次，根据《中国道路交通安全法》的规定：连续驾驶机动车超过四小时需在就近休息停车区休息，一般休息时间不少于 20 分钟，驾驶员必须遵守相关规定，这也有利于道路其它驾驶员的安全。本系统演示设置 20 秒为一个周期，以模拟定时提醒功能，创建线程如下：

```
t = LoopTimer(20.0, LTWarning)
t.start()
```

在此之前定义了一个语音提醒函数 LTWarning 函数设置响应提醒语句：

```
def LTWarning():
    ClockWarning("开了很久了，去休息吧！")
```

3.3.5 报警器设置

报警器输出语音主要有两种方式，一个是通过树莓派 3.5 英寸音频接口外界音响，无需额外安装驱动或者对树莓派音频的设置；第二种是连接外部音响以输出语音。本实验模拟语音警报输出采用 YBL-202 音响如图 3-9。该音响支持蓝牙连接与 3.5 英寸音频连接线连接，足以满足本实验报警语音输出功能。

图 3-9 报警语音输出音响

4 系统测试

4.1 程序调试

采用 Windows 远程桌面程序连接树莓派，将树莓派连接局域网后输入树莓派 IP 地址，登陆界面如图 4-1 所示。程序调试使用 Raspberry OS 自带 IDE：Thonny。正常程序运行时在 Shell 终端结果如图 4-2 所示。

图 4-1 远程连接树莓派

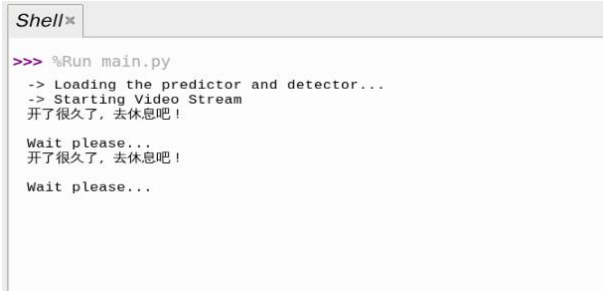


图 4-2 输出终端 shell 显示程序运行结

4.2 系统性能测试

在系统测试项目中，影响检测准确率的因素有人脸与摄像头的距离，光照强度以及驾驶员是否佩戴眼镜，实验通过控制变量实验结果如表 4-1 与 4-2：

表 4-1 光强度与距离影响下的识别准确率%（未佩戴眼镜）

	0-50 cm	50-100 cm	100-150 cm
10-40 LUX	50	60	80
40-80 LUX	80	90	90
80-120 LUX	70	80	90

表 4-2 光强度与距离影响下的识别准确率%（佩戴眼镜）

	0-50 cm	50-100 cm	100-150 cm
10-40 LUX	50	50	80
40-80 LUX	70	90	80
80-120 LUX	60	80	80

4.3 本章小结

在系统程序眼睛长宽比阈值：EYE_AR_THRESH = 0.25，设置嘴部长宽比阈值：YAWN_THRESH = 25 的情况下，经过多次实验得出：在光强一定的情况，摄像头与驾驶员最佳距离在 40-80cm 之间；在距离一定时，光强大于 50LUX 均有较好的识别准度以及较低的延时，有效模拟了多数驾车时驾驶室的光照情况。

5 结论与展望

5.1 全文总结

本系统采用多线程程序结构，利用 OpenCV 与 Dlib 人脸检测算法将人脸划分为 68 个关键点，用 Dlib 人脸视觉库实时检测驾驶员的疲劳状态，检测到疲劳时触发语音输出提醒驾驶员安全驾驶；由于多线程的程序结构，系统将定时告知驾驶员已连续驾驶一定的时间应注意休息；为增加道路交通安全性，本系统模拟了若驾驶员打哈欠频率过高（十秒内连续哈欠检测数超过三个）时，将触发汽车危险双闪灯，具体来讲树莓派接收到哈欠频率过高的信号，通过定义的 GPIO 口向外接硬件电路传输高电位触发 LED 灯组闪烁，同时语音模块输出警报语音提醒驾驶员安全驾驶，此时由于模拟车辆的危险警报灯闪烁将引起周边车辆的注意，最大程度上避免严重交通事故的发生。

5.2 后续工作展望

在系统调试过程中，若环境光照强度较低可以采用红外补光摄像头，同时也相应存在延时问题，延时体现在检测到人脸到标出人脸 68 个关键点最后给出预警语音这段时间响应不迅速。经济有限时选择树莓派 3 代不失为性价比最优的选择，后续工作中可以选择树莓派第 4 代，其 CPU 核心数量以及图像处理能力均有所提高增强，相应延时也将随之减小。

在未来实践生产中，对驾驶员疲劳检测的指标将越来越多，不仅仅是对驾驶员人脸的检测，此外例如对驾驶员的血压，心跳速率，有无其它突发疾病以及对瞳孔大小的检测等[16]。从疲劳识别成功率以及使用成本上来讲，对人脸特征的分析的系统将逐步普及，应用范围将越来越广泛，人脸识别算法也将逐步优化，此外人工智能技术也将加入道路疲劳检测与预警中，届时疲劳驾驶造成的交通事故将有效减少。

参考文献

[1] 黄涛. 基于深度学习的驾驶员视觉疲劳检测研究[D].重庆邮电大学,2021.

[2] G.W.特丽卡, I.R.约翰斯顿等，减少交通事故——一项全球性的挑战，上海科学技术出版社，1993

[3] Parker, D., Reason, J.T., Driving errors, driving violations and accident prevention, Ergonomics, 1995,38:1036-1048

[4] 张开冉. 机动车驾驶员疲劳问题研究[D].西南交通大学,2002.

[5] 孙显彬，唐洪伟，文妍.疲劳驾驶预警系统的研究现状和发展趋势[J].青岛理工大学学报，2007，28（3）：91-94.

[6] 王卓,包杰.嵌入式Linux系统及其应用前景[J].单片机与嵌入式系统应用,2004(05):5-8.

[7] "Raspberry Pi Foundation - About Us". Raspberrypi.org. Retrieved 23 August 2020.

[8] "Eben Upton talks Raspberry Pi 3". The MagPi Magazine. 29 February 2016.

[9] 孔德民. 基于Python开发预警机系统检测设计与研究[D].哈尔滨理工大学,2017.

[10]AMODIO A，ERMIDORO M，MAGGI D，et al.Automatic detection of driver impairment based on pupillary light reflex [J].IEEE transactions on intelligent transportation systems，2019，20（8）：3038-3048.

[11]Gary Bradski, Adrian Kaehler. 学习 Open CV 中文版[M]. 北京:清华大学出版社,2009.

[12]杨云,盛成峰,朱立文,等. 基于人脸识别技术的疲劳监测系统设计[J]. 仪表技术,2020(8):5-6,39.

[13]苏锦瑾,霍春宝,王特特.基于面部特征的驾驶员疲劳检测系统[J].信息技术与信息化,2021(09):114-116.

[14]张伯辰,施鑫杰,霍梅梅. 基于OpenCV的树莓派人脸识别疲劳驾驶检测系统[J]. 现代计算机,2021,27(23):129-132. DOI:10.3969/j.issn.1007-1423.2021.23.023.

[15]黄斌. 基于人脸特征的驾驶员疲劳检测算法与实现[D].南京航空航天大学,2017.

[16]罗元, 云明静, 王艺, 赵立明. 基于人眼信息特征的人体疲劳检测[J]. 计算机应用, 2019, 39(07):2098-2102.

附录 A 程序代码

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import *
from threading import Timer
import RPi.GPIO as GPIO
from ctypes import *
import numpy as np
import argparse
import threading
import imutils
import ctypes
import time
import dlib
import cv2
lib = ctypes.cdll.LoadLibrary("./libhaisantts.so")
lib.startHaisanTTS.argtypes=[POINTER(c_char)]
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
LED = 4         #physical 7
PIN = 17        #physical 11
GPIO.setup(LED,GPIO.OUT)
GPIO.setup(PIN,GPIO.IN,pull_up_down = GPIO.PUD_DOWN)
GPIO.add_event_detect(PIN,GPIO.RISING)
```

```
def ClockWarning(msg):
    TTS=(c_char * 100)(*bytes(msg,'utf-8'))
    cast(TTS, POINTER(c_char))
    lib.startHaisanTTS(TTS)
def LTWarning():
    ClockWarning("开了很久了，去休息吧！")
class LoopTimer(Timer):
    def __init__(self, interval, function, args=[], kwargs={}):
        Timer.__init__(self,interval, function, args, kwargs)
  def WarningLed():
    print("led binked")
def warning(msg):
    global alarm_status
    global alarm_status2
    global saying
    while alarm_status:
        print('call')
        TTS=(c_char * 100)(*bytes(msg,'utf-8'))
        cast(TTS, POINTER(c_char))
        lib.startHaisanTTS(TTS)
    if alarm_status2:
        print('call')
        saying = True
        TTS=(c_char * 100)(*bytes(msg,'utf-8'))
        cast(TTS, POINTER(c_char))
        lib.startHaisanTTS(TTS)
        saying = False
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
```

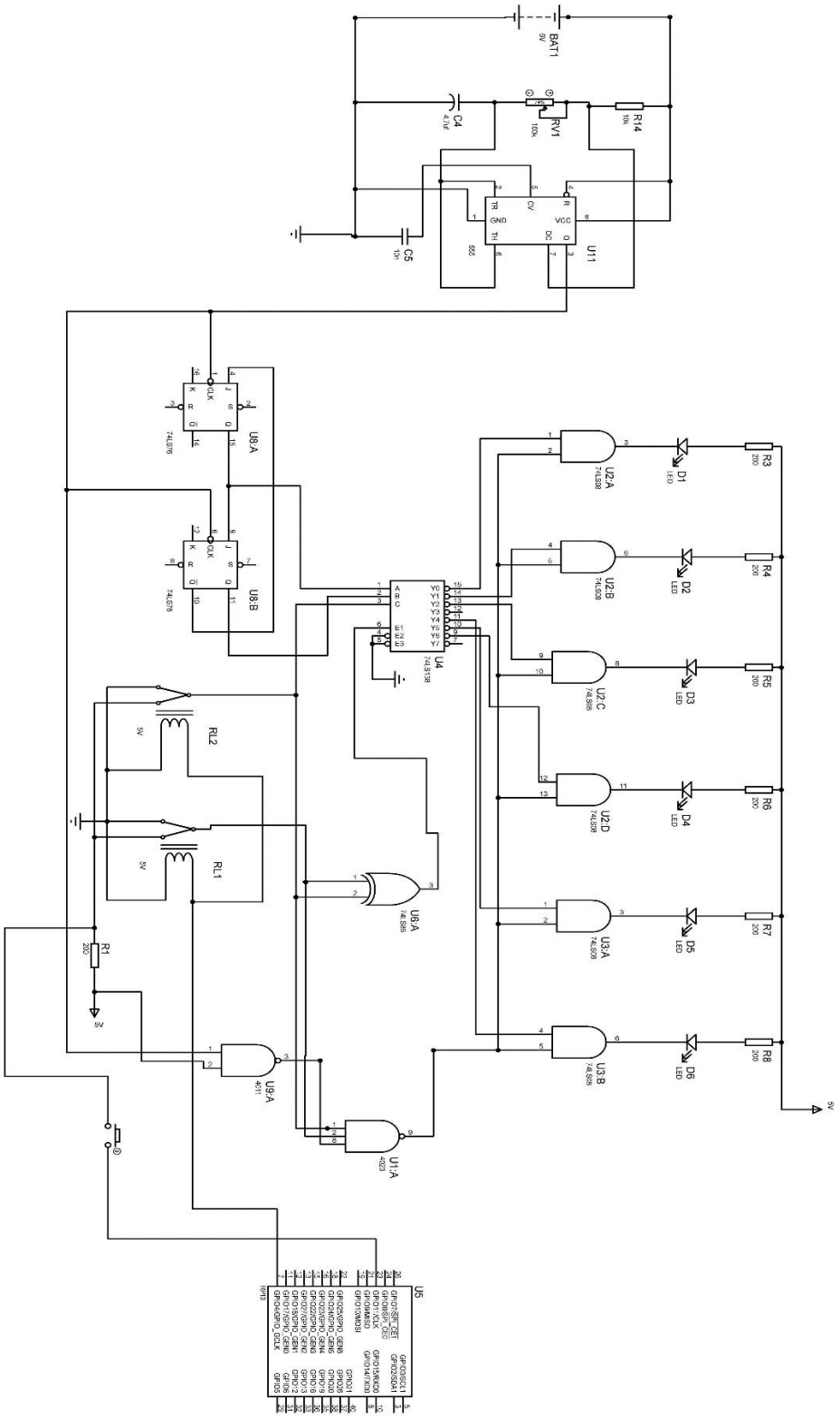
```
        B = dist.euclidean(eye[2], eye[4])
        C = dist.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear
def final_ear(shape):
        (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
        (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        return (ear, leftEye, rightEye)
def lip_distance(shape):
        top_lip = shape[50:53]
        top_lip = np.concatenate((top_lip, shape[61:64]))
        low_lip = shape[56:59]
        low_lip = np.concatenate((low_lip, shape[65:68]))
        top_mean = np.mean(top_lip, axis=0)
        low_mean = np.mean(low_lip, axis=0)
        distance = abs(top_mean[1] - low_mean[1])
        return distance
ap = argparse.ArgumentParser()
ap.add_argument("--webcam", type=int, default=0,help="index of webcam on system")
args = vars(ap.parse_args())
EYE_AR_THRESH = 0.25
EYE_AR_CONSEC_FRAMES = 10
YAWN_THRESH = 25
alarm_status = False
```

```
alarm_status2 = False
saying = False
COUNTER = 0
print("-> Loading the predictor and detector...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
print("-> Starting Video Stream")
vs= VideoStream(usePiCamera=True).start()
time.sleep(1.0)
t = LoopTimer(20.0, LTWarning)
t.start()
flag = True
start = 0
end = 0
time_dif = 0
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = detector(gray, 0)
    for rect in rects:
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye [1]
        rightEye = eye[2]
        distance = lip_distance(shape)
        leftEyeHull = cv2.convexHull(leftEye)
```

```
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
lip = shape[48:60]
cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)
if ear < EYE_AR_THRESH:
    COUNTER += 4
    if COUNTER >= EYE_AR_CONSEC_FRAMES:
        if alarm_status == False:
            alarm_status = True
            t1 = Thread(target=warning, args=('醒醒，醒醒！',))
            t1.daemon = True
            t1.start()
        cv2.putText(frame, "warning!", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
else:
    COUNTER = 0
    alarm_status = False
if (distance > YAWN_THRESH):
        cv2.putText(frame, "Drawning alart!", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        if alarm_status2 == False and saying == False:
            if flag == True:
                start = time.time()
                flag = False
            elif flag == False:
                end = time.time()
                time_dif = end - start
                flag = True
```

```
                    if time_dif < 20:
                        GPIO.output(LED,GPIO.HIGH)
                    print(flag)
                    alarm_status2 = True
                    t2 = Thread(target=warning, args=('困了吧，停车休息一下吧！',))
                    t2.daemon = True
                    t2.start()
            else:
                alarm_status2 = False
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            time.sleep(1)
            break
        if GPIO.event_detected(PIN):
            GPIO.output(LED,GPIO.LOW)
            continue
cv2.destroyAllWindows()
GPIO.output(LED,GPIO.LOW)
vs.stop()
```

附录 B 系统硬件连接原理图



致 谢

大学经历将成为我一生中最为宝贵的财富。

日常生活中，首先感谢父母对我的精神和经济上的支持，让我在大学知识的海洋里遨游；感谢辅导员刘老师对我的教导与关怀，让我有勇气面对生活遇到各种挫折；感谢同班同学和寝室室友，没有你们我不会像今天这样积极乐观，努力进取。

对于学习方面，感谢专业各科老师的谆谆教诲，是你们的辛勤付出让我能牢牢掌握专业知识，成为对社会有用之才；感谢张悦老师在百忙之中对我毕业设计的耐心指导，从课题的选择到论文的撰写，让我学会了如何从提出问题到最后解决问题。

最后，感谢在背后默默支持我的亲戚朋友们，无论是学习和生活中感谢你们无时不刻都在鼓励我，为我出谋划策。

逝者如斯夫，不舍昼夜。大学四年稍纵即逝，作为 21 世纪主力军应积极响应国家号召，珍惜时间努力奋斗，为实现社会主义现代化贡献一己绵薄之力。