基于树莓派和 OpenCV的人脸识别门禁

目录

1. 背景意义

随着社会经济快速发展，人民生活水平的不断提高，群众的安全防卫意识也逐步提升。由此，人们对安全防卫系统的要求越来越高。如何利用新的技术手段设计更加可靠的安防系统，增加居民生活的安全感，成为安防研究领域关注的焦点。传统门禁系统的身份验证方式主要有钥匙、口令、密码、磁卡等，然而这些验证手段与用户具有可分离性，容易被破译和盗用，已不能完全满足现代安防理念。而人体的生物特征由于其唯一性和不可分离性，在安全性能上能补充上述的不足，应用于门禁系统可以很大地提高其安全性能。随着计算机技术和生物特征识别技术（Biometric Identification Technology）的发展，人体的生物特征，譬如人脸、掌纹、虹膜、指纹等，可以与计算机技术、生物传感器技术等高科技手段相结合，从而实现对来访者身份的识别。

自动化技术的基本目的,是以弱电控制强电的方式实现对各种电气设备的控制，同时实现安全、方便的人机交互，以方便人们的日常工作和生活。作为自动化工程学院的学生，学好控制领域的专业知识以及各类软件是十分重要的。其中嵌入式计算机系统是自动化学科领域的一个重要分支,学习其相关知识将对我们以后的课程学习、专业深造、就业发展都有所帮助，这也能加深我们对自动化系统的概念认识，并锻炼自动化工程项目的开发本领。

本学期的嵌入式计算机系统课程，旨在让学生接触嵌入性系统，了解嵌入式系统的硬件组成、开发原理，并且通过动手设计嵌入式计算机系统作品，让学生们学习相关算法和嵌入式系统开发代码，以加深对嵌入式计算机系统的了解。本学期的嵌入式计算机系统课程选题是基于机器视觉和树莓派平台的智能硬件系统设计，正如前面所说，人脸识别门禁系统具有很大的学习及应用空间，故我们组本次嵌入式计算机系统课程设计选择制作一台人脸识别门禁的原型系统。

本设计的人脸识别部分将基于树莓派平台和 Python 开发语言。由于其的外观轻巧、易于扩展，以及其良好的运算能力，本次设计将采用树莓派作为人脸识别部分的硬件系统。Python 是一门功能非常强大的高级编程语言，由于其丰富的底层库函数，通过 Python 能够轻松编写出很多其他语言难以实现的功能。要想在 Python 环境下实现机器视觉，需要借助一些外部程序包。在本设计中使用的外部包是 opencv-contrib-python，其是基于 OpenCV 开发的 Python 语言包，能够通过函数调用实现图像的获取、分析、处理及智能操作。

由于本设计选题是制作人脸识别门禁，所以必须扩展的是门禁控制部分。由于树莓派本身的 GPIO 口数量有限，所以本设计的控制部分选用易于编程且外部扩展丰富的 C51 单片机开发板。C51 单片机将实现门禁继电器开闭、门锁状态显

示、密码键盘、门禁继电器驱动、单片机通信等功能。由树莓派和 C51 的配合，本设计作品能实现工作模式选择、人脸数据库建立、人脸识别开门、密码开门等功能。经测试，本系统能较好地实现预期效果。

在本次设计开发过程中，我们对嵌入式计算机系统组成和开发过程有了基本的认识，同时也锻炼了我们设计硬件电路、开发软件程序的能力。通过前期网上查阅资料、学习相关编程、环境搭建、系统设计与调试，我们最终得到了一个亲手参与制作的作品，这个过程大大提高了我们自主学习的能力，并给我们带来了很多收获。

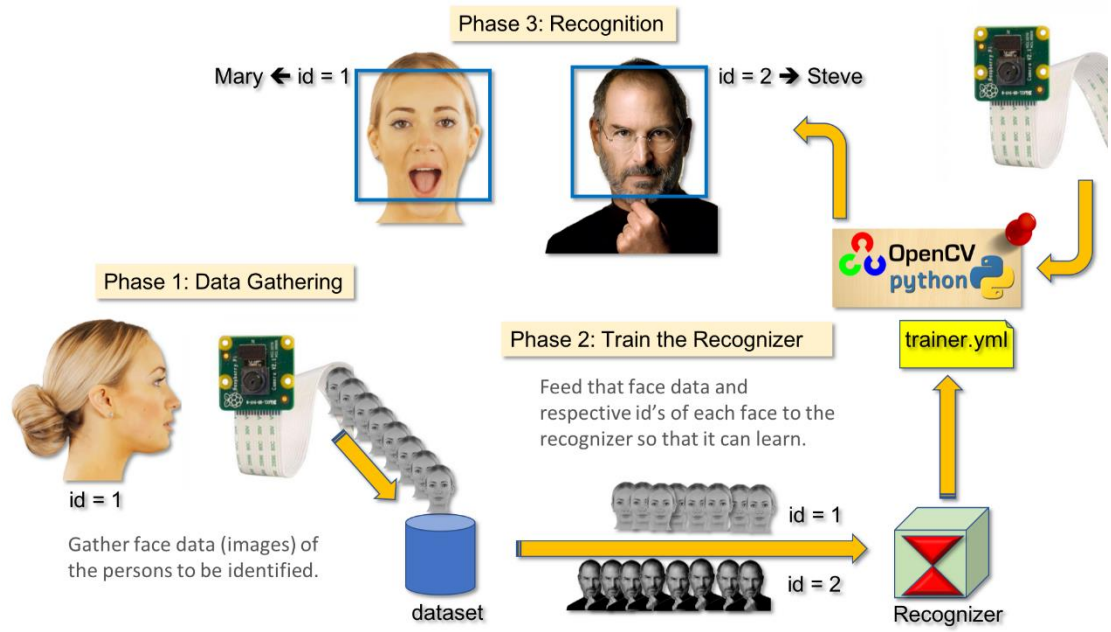


图 1 基于树莓派和 OpenCV 的人脸识别原理概念图

2. 系统设计功能

根据系统执行流程，我们将人类识别门禁系统分为三大模块和若干子模块。三大模块是：图像采集与人脸识别模块、控制器模块和门禁执行模块。其中，图像采集与人脸识别模块负责对人脸图像的采集、图像信息的编号、人脸图像库建立、人脸信息特征提取、识别模型建立以及人脸识别身份匹配功能。控制器模块负责模式选择、LCD 显示、矩阵键盘及密码操作和根据人脸识别结果执行电控锁等功能；门禁执行模块主要是执行控制器开关门信号以及门禁驱动功能。

考虑到本设计的难度和要求，我们组选定系统的预期实现的基本功能：

1、实现人脸检测和采集：图像采集设备获取访客的人脸图像，做出人脸的图像切片，保持其切片图像和位置坐标，并在屏幕上人脸位置处给出指示框。

2、防多人像干扰功能：开启人脸检测时将在屏幕上方显示出当前拍照的张数。当检测到两个人同时出现在摄像头面前时，将会给出错误信息，并停止计数，直到屏幕上检测到单一人脸后，从零开始重新拍照，最终对单一人脸信息将保持 100 张照片信息。

3、人脸图像库建立：将所有人脸信息设置编号并基于人脸识别模型进行机器学习，学习后生成人脸识别器模型 trainer.yml，并且该数据库和识别器模型可在后期进行删改。

4、人脸识别功能：识别人脸信息，与数据库中的照片进行比对，当比对成功时，显示出该人的序号和姓名，并向控制器模块发送识别成功信号。

5、控制器工作模式选择：采用按键来选择门禁相对应的工作模式，可以在人脸数据库建立、人脸识别开锁、密码开锁和关门中进行模式选择和切换。

6、密码开锁：采用密码开门的原因是模仿现实人脸识别产品，并防止人脸识别系统偶然故障，该功能基于既定的程序，可以实现密码的输入和修改，以及发出是否开门的指令。

7、状态信息提示：通过 LCD 模块实现对门禁当前工作状态的提示，包括显示需要执行的案件操作和当前正在执行的程序。

8、门禁驱动部分：由于该门禁的正常工作电压大于 3.3V，而两个单片机的 GPIO 高电平都在 3.3~2V，所以需要有专门的门禁驱动模块来放大信号对门禁的控制作用。

该系统功能及其实现控制器和模式实现如表 1 所示。

表 1　系统设计功能总述

组件	通信	功能
RaspberryPi	两根信号线、一根中止信号线与 C51 相连，开门信号线接继电器，HDMI 线和 Type-C 线与显示屏相连，PiCam 接口直连摄像头	作为机器视觉的实现控制器，主要用于人脸检测与采集，图片与视频的分析处理。通过 C51 的矩阵按键进行模式选择，通过信号线和中止线可以调用其人脸建库、模型训练、人脸识别等功能。同时具有摄像头视频显示，防多人数据冲突，错误报警提示，开门信号反馈等功能。
STC89C51	两根信号线、一根中止线与 RaspberryPi 相连，开门信号线、门禁控制线与继电器相连，LCD、矩阵键盘直连开发板	该组件为系统的主控制器，负责控制 LCD 模块显示系统状态信息及操作提示、控制 RaspberryPi 实现人脸识别与数据库建立。C51 模块通过两根控制信号线一根中止信号线、一根开门信号线和一跟门禁信号线及各供电、共地先组合，对整个系统进行控制。
继电器	继电器电源和地与公共 Vcc、GND 相连，两根信号线与 RaspberryPi、C51、电磁门禁相连	继电器主要功能是通过信号线高低电平控制输出端的电平输出，在本设计中继电器主要是解决电平不匹配和扰动的问题。
矩阵键盘	与 C51 直连	矩阵键盘从左至右，从上至下依次为“0, 1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9”数字键，“返回，确认”按键和“A, B, C, D”模式选择键。主要用于实现密码开门功能、模式选择功能和中止功能。

模式	通信	功能
模式 A	按键 A 被按下，信号线 1 发送高电平	模式 A 是建立人脸数据库，主要执行的操作有拍摄人脸，人脸照片切片、数据储存、人脸图片模型建立与训练、防冲突处理与报警等功能。同时 LCD 也会显示相应状态和提示信息。
模式 B	按键 B 被按下，信号线 2 发送高电平	模式 B 是人脸识别，主要执行的操作有调用人脸图片模型，人脸识别与检测，开门信号反馈等功能。同时 LCD 也会显示相应状态和提示信息。
模式 C	-	模式 C 是密码开门，主要执行的操作是读取键盘密码输入，发送开门信号，同时 LCD 提示开门成功或密码错误等信息。
模式 D	按键 D 被按下，中止线发生高电平	模式 D 是关门中断，主要负责在门禁进行其他功能操作（比如模式 A、B、C）时中止当前任务，发送关门信号并返回系统主界面。

3. 硬件系统组成及实现

为了实现拟定的系统设计目标，需要硬件平台和软件平台的相互支撑配合。其中，硬件平台为软件平台的运行环境提供物理支撑，而软件平台需要选择适合的硬件才能更好地实现系统拟定的各项目标。因此，硬件平台的选择关系到系统的运行效率和各项性能，可见硬件在整个系统中所发挥的作用中至关重要。本章围绕系统所采用的各个硬件设备对人脸识别门禁系统进行阐述。

3.1 系统硬件设计方案

嵌入式处理器作为硬件平台的核心，是衡量硬件是否满足系统开发需求的主要标志，本文结合当前传统的门禁系统，采用基于树莓派和 C51 处理器的开发板以及其他外部设备共同搭建系统开发的硬件平台，其具体的硬件选配在后续部分进行说明。系统硬件设计如图 2 所示。

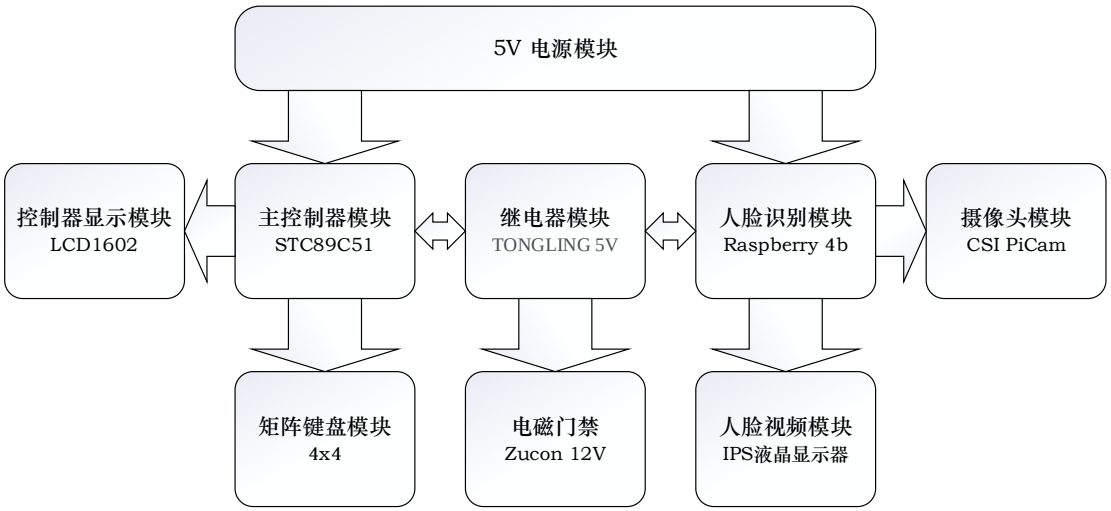


图 2 硬件系统连接示意图

3.2 Raspberry Pi 4B 概述

树莓派在本质上是一款袖珍电脑，功能强大的它集 CPU、GPU、DSP 和 SDRAM 为一体，以 SD 卡位内存硬盘，拥有网卡、USB 口（可直接连接键盘、鼠标、U 盘等外设），同时具备视频、音频模拟输出以及 HDMI 高清输出的能力。在外部接口上还具备了一般计算机设备不具备的 GPIO、SPI、UART 等硬件配置，为我们的人脸识别门禁系统提供了丰富的硬件条件。树莓派支持 Python 开发并且支持 Python 控制 GPIO 口。这使得树莓派的开发门槛降低。Python 作为新流行的编程语言，有着简单易学的优点。并且 Python 拥有十分庞大的标准库。这些库可以让我们的编程变得更加容易，同时让程序更加简洁。尽管基于 Python 环境的树

莓派相比于 C 语言系统来说，运行速度会相对较慢，但是我们可以直接看到程序运行给我们的反馈信息，所以说对于人脸识别门禁的开发来说，采用树莓派和 Python 环境会相对方便。

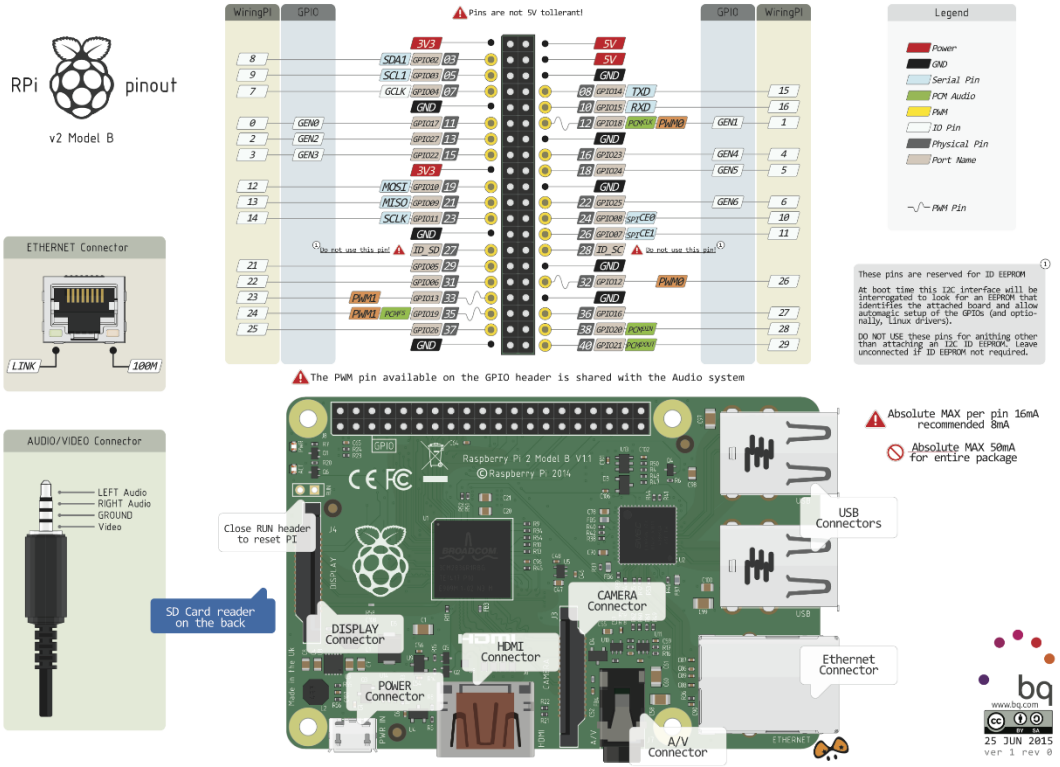


图 3 Raspberry Pi 结构图及其配件组成

本设计采用的树莓派型号是 Raspberry Pi 4B。相比上一代的树莓派 3B+，树莓派 4B 在处理器速度，多媒体性能，内存和连接方面提供了突破性的增长，同时保留了向后兼容性和类似的功耗。对用户来说，树莓派 4B 提供的桌面性能可与入门级 x86 PC 系统相媲美。树莓派 4B 的主要功能包括高性能 64 位四核处理器，通过一对 micro-HDMI 端口支持分辨率高达 4K 的双显示屏，高达 4Kp60 的硬件视频解码，可达 4GB 的 RAM，双频 2.4/5.0 GHz 无线局域网，蓝牙 5.0，千兆以太网，USB 3.0 和 PoE 功能等。

本次设计需要的树莓派系统，除了本身以外，还需要 SD 卡、HDMI 线，microHDMI 转接口，Picam 摄像机模块，USB 鼠标，USB 键盘和显示器。树莓派采用官方的 Linux Desktop 桌面环境，其需要烧录在 SD 卡上，并安装在树莓派上进行运行。摄像机模块直接连在对应的接口上，显示屏通过一根 HDMI 线和一根 TypeC 线进行连接，实现图像的输出和触屏信号的采集工作。

表 2 树莓派 Raspberry Pi 4B 硬件资源属性

名称	属性	名称	属性
SOC	Broadcom BCM2711	**供电接口**	Type-C 5V 3A
CPU	64-位 1.5GHz 四核（28nm 工	**Wifi 网络**	802.11 AC 无线 2.4GHz/5GHz 双

	艺）		频 Wifi
USB 接口	两个 USB2.0 和两个 USB3.0	**有线网络**	真千兆以太网
蓝牙	蓝牙 5.0	**以太网 PoE**	HAT 以太网 PoE
GPU	500MHz VideoCore VI	**电力需求**	3A 5V
HDMI	两个 Micro HDMI 支持 4K60	**多媒体**	H.265（4Kp60 decode） H264 encode（1080P30 / 1080P60） OpenGL，3.0graphics
内存	1-8G DDR4	**尺寸**	88 × 58 × 19.5mm
GPIO	40 个 GPIO 接口	**音频接口**	3.5mm 音频接口
DSI	一个 DSI 显示连接器	**摄像头**	Picam 摄像头接口

3.3 图像采集设备

本设计采用树莓派官方的 Raspberry Pi Camera，其支持树莓派 4B 开发板，同时也能兼容 3B+/3B/2B 等老版本树莓派。该 PiCam 的连接方式很简单，只需要将摄像头排线插入树莓派上摄像头接口就可以使用摄像功能。该款摄像头配有保护外壳，为亚克力材料制作，在此之上可以扩展其他支架和云台等设备。

图 4 Raspberry Pi Camera 及其支架

PiCam 的相关硬件参数为：

- 像素：500 万像素；
- 感光芯片：OV5647；
- 静态图片分辨率：2592 × 1944；
- 录像：支持 1080P30，720P60，640×480P60/90 视频录像；
- 尺寸：25 × 24 × 9mm；
- 角度：仰角 65°。

3.4 STC89C51 开发板概述

51 单片机是对所有兼容 Intel 8031 指令系统的单片机的统称。该系列单片机的始祖是 Intel 的 8004 单片机，后来随着 Flash rom 技术的发展，8004 单片机取得了长足的进展，成为应用最广泛的 8 位单片机之一，其代表型号是 ATMEL 公司的 AT89 系列，它广泛应用于工业测控系统之中。很多公司都有 51 系列的兼容机型推出，今后很长的一段时间内将占有大量市场。51 单片机是基础入门的一个单片机，还是应用最广泛的一种。需要注意的是 51 系列的单片机一般不具备自编程能力。

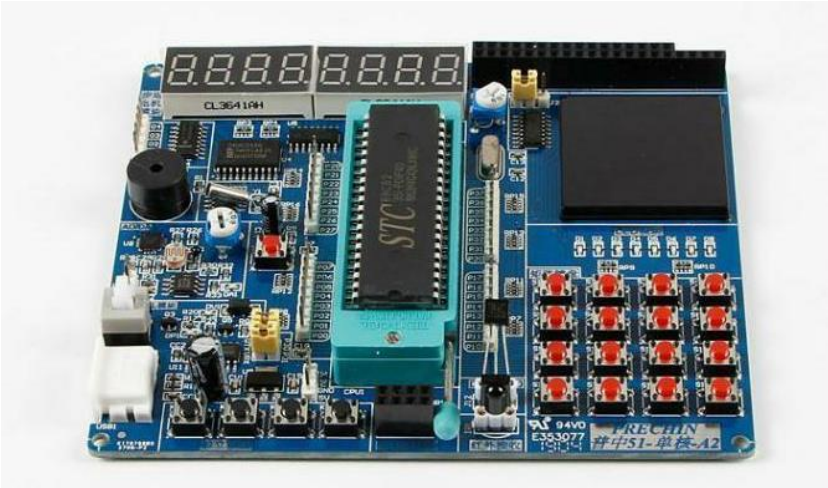


图 5 普中科技 STC89C51A2 单核开发板

STC89C51 单片机是当今广泛应用于学习、开发、科研领域中的一款单片机，其基于 C 语言进行编程，由于其性能良好而且简单易学，已经成为自动化、电子电信相关学科学生入门数字电路和学习电子设计相关课程的首选微处理器。尽管现在市面上有很多其他性能更好，具有更强扩展性和可开发性的微处理器，如 STM32、MSP430 等，但对于我们本设计作品的预期实现目标而言，C51 单片机的性能足够其使用。

STC89C51 单片机的硬件性能参数为：

- 时钟：6 时钟/机器周期和 12 时钟/机器周期可选；
- 工作电压：5.5~3.3V；
- 工作频率范围：0~40MHz；
- 程序空间：8K 字节；
- RAM：片上集成 512 字节 RAM；
- GPIO：40 个引脚，其中4×8Pin 可以自由定义；
- EEPROM 功能；
- 看门狗功能；
- 3 个 16 位定时器；
- 四路外部中断。

本设计所需的 C51 单片机开发板是采用普中科技的基础开发板，其上包含 C51 单片机最小系统（C51 微处理器、复位电路、时钟电路）、DC5V→3.3V 电源模块、ADC 模块、DAC 模块、LCD1602 接口模块、步进电机模块、DS1302 时钟模块、74HC138 译码器模块、USB 转 TTL 下载模块、矩阵按键模块等。以上模块能够充分满足本设计的硬件技术需求。

3.5 矩阵按键模块

独立键盘与单片机连接时，每一个按键都需要单片机的一个 I/O 口，若某单片机系统需较多按键，如果用独立按键便会占用过多的 I/O 口资源。单片机系统 I/O 口资源往往比较宝贵，当用到多个按键时为了减少 I/O 口引脚，引入了矩阵键盘。

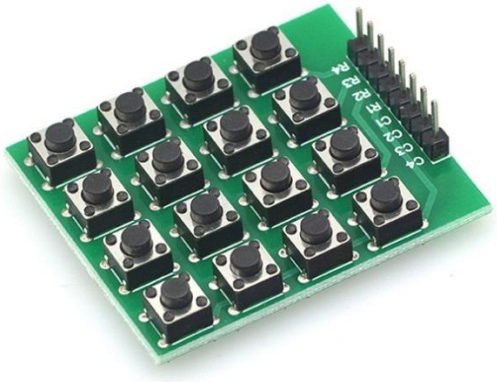


图 6　4×4 16 矩阵按键

本设计用到的矩阵键盘为4×4类型，共计 16 个按键。每一个按键的含义可以在代码中做定义。将 16 个按键排成 4 行 4 列，第一行将每个按键的一端连接在一起构成行线，第一列将每个按键的另一端连接在一起构成列线，这样便一共有 4 行 4 列共 8 根线，我们将这 8 根线连接到单片机的 8 个 I/O 口上，通过程序扫描键盘就可检测 16 个键。无论是独立键盘还是矩阵键盘，单片机检测其是否被按下的依据都是一样的，也就是检测与该键对应的 I/O 口是否为低电平。独立键盘有一端固定为低电平，单片机写程序检测时比较方便。而矩阵键盘两端都与单片机 I/O 口相连，因此在检测时需编程通过单片机 I/O 口送出低电平。检测方法有多种，最常用的是行列扫描和线翻转法。在本设计中采用行列扫描法进行按键检测，并配合延时消抖。

3.6 LCD1602 模块

1602 液晶也叫 1602 字符型液晶，它能显示 2 行字符信息，每行又能显示 16 个字符。它是一种专门用来显示字母、数字、符号的点阵型液晶模块。它是由若

干个5×7或者5×10的点阵字符位组成，每个点阵字符位都可以用显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此，所以它不能很好的显示图片。LCD1602 有 16 个管脚，分别起到了电源、显示偏压、命令选择、读写选择、使能和数据线的功能。详细的引脚功能说明如下：

• 1 脚：VSS，电源地，接单片机 GND 端。

• 2 脚：VDD，电源正极，接单片机 3.3V 端。

• 3 脚：VL，液晶显示偏压信号，用于调整 LCD1602 的显示对比度，一般会外接电位器用以调整偏压信号，注意此脚电压为 0 时可以得到最强的对比度。

• 4 脚：RS，数据/命令选择端，当此脚为高电平时，可以对 1602 进行数据字节的传输操作，而为电平时，则是进行命令字节的传输操作。命令字节，即是用来对 LCD1602 的一些工作方式作设置的字节；数据字节，即使用以在 1602 上显示的字节。值得一提的是，LCD1602 的数据是 8 位的。

• 5 脚：R/W，读写选择端。当此脚为高电平可对 LCD1602 进行读数据操作，反之进行写数据操作。

• 6 脚：E，使能信号，其实是 LCD1602 的数据控制时钟信号，利用该信号的上升沿实现对 LCD1602 的数据传输。

• 7-14 脚：8 位并行数据口，而 51 单片机一组 IO 也是 8 位，使得对 LCD1602 的数据读写大为方便。在 LCD1602 内部含有 80 个字节的 DDRAM，它是用来寄存显示字符的。

• 15 脚：BLA，背光源正极。

• 16 脚：BLK，背光源负极。

通过该 LCD1602 模块可以显示出该门禁当前的工作模式。

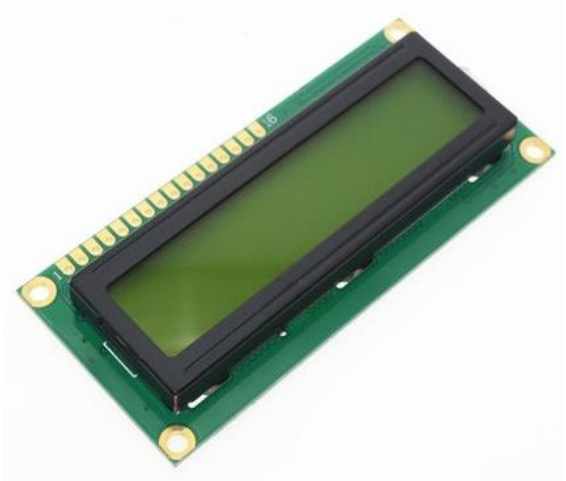


图 7 LCD1602 液晶16×2字符显示器

3.7 门禁及驱动模块

本设计所需的单门磁力锁采用电信号控制，磁力的强度收到输入信号电压的影响。其开关门的原理是电磁铁的电生磁现象，当电磁门禁中通入电流，信号会在其内部进行变换和传导，控制电磁铁芯产生磁性，吸住另一款磁吸实现关门；当电信号断开或者信号电流过低时，相应的磁性会减小，实现开门。

该门禁的硬件特性参数为：

- 工业材质：优质硅钢材料；
- 外壳：采用高强度铝合金锁体外壳；
- 工作电流：500mA；
- 工作电压：DC3.3~12V；
- 指示灯：LED；
- 信号反馈：COM、NO、NC、门磁信号反馈；
- 吸力：280KG；
- 类型：断电开锁；
- 尺寸：$250 \times 25 \times 45$mm。

图 8　ZUCON 5V 电控磁门禁

由于 C51 单片机 GPIO 口的输出高电平为 2.4V 以上，而往往在 3.3V 以下，而 C51 的流经电流通常不能超过$250mA$。C51 单片机主要是用来控制而非驱动，如果直接使用芯片的 GPIO 管脚去驱动大功率器件，要么将芯片烧坏，要么就是驱动不起来。故此电压难以驱动电控门禁，甚至可能门禁的 LED 指示灯都不能点亮，所以本次设计我们通过 C51 开发板上的板载电机驱动模块的放大电路对该电控门禁进行电流放大驱动。此电机驱动模块使用一片 ULN2003D 芯片，该芯片是一个单片高电压、高电流的达林顿晶体管阵列集成电路。其原本功能是驱动直流电机或五线四相步进电机，本设计仅用其电流放大功能。

ULN2003 是一个单片高电压、高电流的达林顿晶体管阵列集成电路。它是由 7 对 NPN 达林顿管组成的，它的高电压输出特性和阴极箝位二极管可以转换感应负载。单个达林顿对的集电极电流是 500mA。达林顿管并联可以承受更大的电流。此电路主要应用于继电器驱动器，字锤驱动器，灯驱动器，显示驱动器（LED 气

体放电），线路驱动器和逻辑缓冲器。ULN2003 的每对达林顿管都有一个 2.7k 串联电阻，可以直接和 TTL 或 5VCMOS 装置。

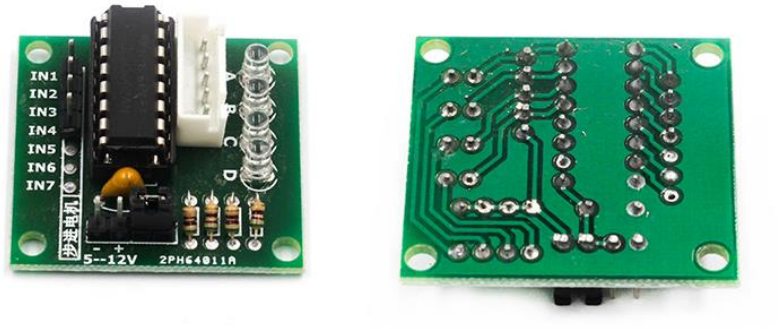


图 9 ULN2003 步进电机驱动模块/电流放大模块

其主要硬件性能参数是：

- 500mA 额定集电极电流（单个输出）
- 高电压输出：50V
- 输入和各种逻辑类型兼容
- 继电器驱动器

从上图可以很容易理解该芯片的使用方法，其内部实际上就相当于非门电路，即输入高输出为低，输入低输出高。

3.8 继电器模块

在本设计中使用了一组继电器，用于树莓派人脸识别信号的反馈。我们组使用的继电器为 TONGLING 5V 直流继电器模块。一般的继电器模块有六个端口，控制测有三个端口，VCC 是电源正极，GND 是电源负极，IN 是通断信号的输入引脚。被控端也有三个端口，NC 即常闭端（normal close），COM 即公共端，NO 即常开端（normal open）。在中学物理中已经学过，开路即通路、断路，闭合指的是开关闭合，也就是说，在没有任何上电之类的动作时，NC 和 COM 端相当于已经连通。

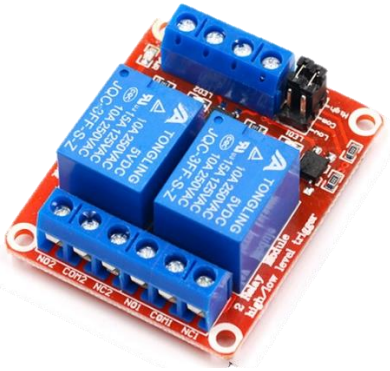


图 10 TONGLING 5V 两路继电器模块

本次设计用到的继电器模块为：

- 常开接口限大负载：交流 250V/10A，直流 30V/10A；
- 采用贴片光耦隔离，驱动能力强，性能稳定；
- 触发电流：5mA；
- 模块工作电压：DC 5V；
- 模块可以通过跳线设置高低电平触发
- 电源指示灯：绿色 LED；
- 继电器状态指示灯：红色 LED；
- 接口设计人性化，接口均可通过接线端子直线连线引出，非常方便；
- 设有 4 个固定螺栓孔，直径 3.1mm，间距44.5×35.5mm

继电器模块接口说明：

- DC+：接电源正极；
- DC-：接电源负极；
- IN1：1 路信号触发端，可以设置高低电平控制继电器吸合；
- IN2：1 路信号触发端，可以设置高低电平控制继电器吸合；
- NO1、NO2：1、2 路继电器常开接口，继电器吸合前悬空，吸合有与 COM1、COM2 短接；
- COM1、COM2：1、2 路继电器共用接口；
- NC1、NC2：1、2 路继电器常闭接口，继电器吸合前于 COM1、COM2 短接，吸合后悬空。

4. 软件算法及环境配置

软件算法是嵌入式计算机系统的灵魂所在，是计算机系统能够成功运行、执行需求功能的必须条件。而要在微处理器上使用相关的软件代码，有需要对相关软件和库函数进行配置，这些配置是高级语言与底层代码之间的交流环节，所以在嵌入式计算机系统设计中是起到奠基石的作用。由于本设计主要有两个微处理器，所以本章节将会从两方面来进行阐述相关的环境配置和库函数搭建，另外会对 opencv 的人脸识别算法以及我们采用的 Haar 分类器算法进行简要叙述。

4.1 树莓派系统环境配置

4.1.1 树莓派系统配置

前面已经提到过，一个可以开发的树莓派系统至少需要一个 SD 卡、一个具有烧录功能的读卡器、安卓充电线和适配器、HDMI 接口的显示屏、鼠标和键盘等。接下来应该安装树莓派系统，本设计的树莓派环境配置工作采用 RaspberryPi 官网的树莓派图形桌面系统：“Raspberry Stretch with desktop and recommended software”。下载后将.img 文件通过 SD 卡格式化软件烧录到 SD 卡上。将操作系统安装好的 SD 卡安装在树莓派上，开启电源即可以进入到树莓派桌面。

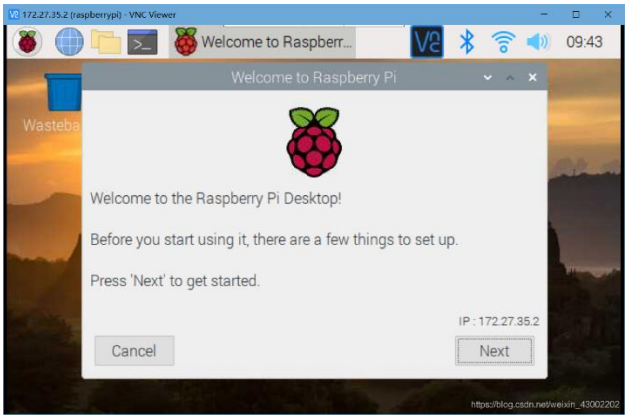


图 11 树莓派开机界面

接下来为了开发的方便，可以采用虚拟桌面远程连接的方式对树莓派进行连接。使用 putty 对树莓派进行网络配置。在红色箭头所指向的位置输入树莓派的 IP 地址。连接无线 wifi，必须通过查看路由器已连接树莓派设备的 IP 地址，同样，电脑也必须保持和树莓派在同一网段上。点击 Open，输入用户名及密码。

树莓派默认用户名：Pi，树莓派默认密码：raspberry。

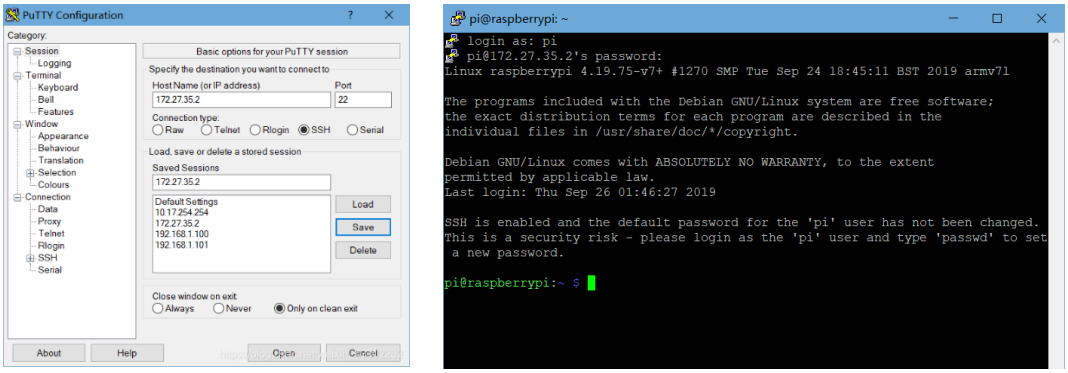


图 12　Putty 设置网络与配置远程操作

为了能得到图形界面的远程连接，我们还需要开启 VNC 功能。树莓派默认是允许 root 用户 ssh 登录，为了安全，建议关闭 root 用户 ssh 登录，方法为在树莓派的命令窗中输入：#sudo /etc/ssh/sshd_config。将“PermitRootLogin yes”改为“PermitRootLogin no”，若要重启 ssh 服务则输入：service ssh restart。

4.1.2 树莓派软件源更换

树莓派开发的过程中，避免不了的要从网络中下载相应的开发包。但是由于多数开发包的下载地址在国外服务器，造成下载速度很慢，所以我们需要对下载源进行必要的更换。其根本是要编辑系统源文件，在命令行中输入：sudo nano /etc/apt/sources.list。接下来要在弹出的页面中将初始源注释掉，在原位置添加两行清华的镜像源：

•deb http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main contrib non-free rpi

• deb-src http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main contrib non-free rpi

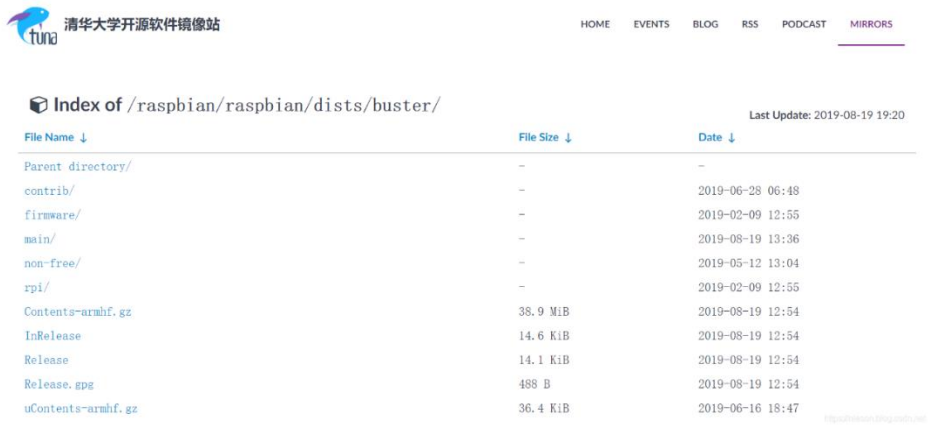


图 13 清华大学开源镜像网站

接下来要保持执行如下命令：sudo apt-get update，完成源的更新软件包索引。还要更新系统源，输入：sudo nano /etc/apt/sources.list.d/raspi.list，同样的在弹出的界面中注释掉原内容，用以下内容替代：

deb http://mirrors.tuna.tsinghua.edu.cn/raspberrypi/ buster main ui

deb-src http://mirrors.tuna.tsinghua.edu.cn/raspberrypi/ buster main ui

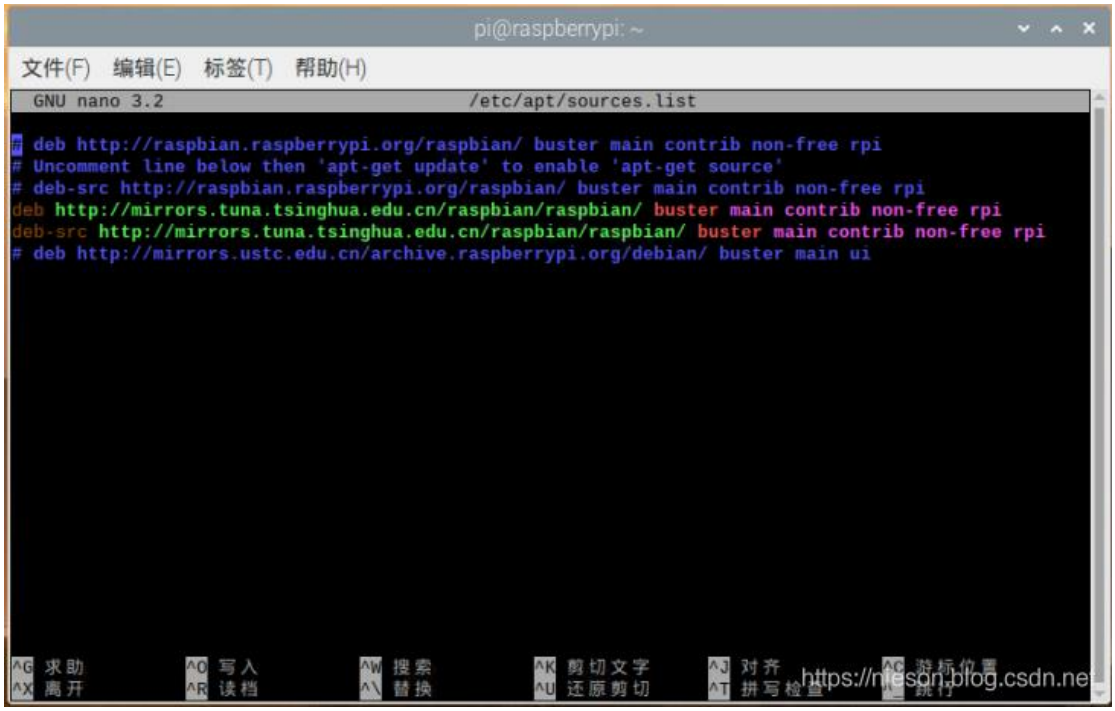


图 14 树莓派网络下载换源

当所有源都更换以后，系统下载的速度会变的很快。

4.1.3 Python 程序包安装

基于 Python 高级语言的通俗易懂、和扩展程序丰富的特点，Python 可以做很多程序和应用的开发。对于本次设计作品，也需要安装相关的程序包，有：

• opencv-contrib-python：opencv 的 python 程序包，以实现图像的采集、模型库的建立、机器视觉等功能。

• numpy：python 的数组计算处理程序包，能实现类似于 MATLAB 的矩阵及元素计算等功能。

• PIL：python 的图像处理程序包，能实现图像的分析和处理。

• RPi.GPIO：树莓派的 GPIO 定义和配置函数包，能实现树莓派 GPIO 引脚的配置和输入输出功能。

• time：树莓派的定时函数库，在本实验中主要负责延时和定时器的功能。

• os：Python 标准库，提供通用的、基本的操作系统交互功能。

上述的程序包树莓派的操作系统中都有预置，只有第一个也是最重要的一个

opencv-contrib-python 没有预置，所以我们要对此程序包进行安装。在命令行中输入：sudo pip install opencv-contrib-python == 3.4.3.18 即可。如果发现下载时间过长或者下载失败，则可以尝试重新安装，或者更换软件源，换源方法同 4.1.2。

4.2 C51 开发板环境配置

4.2.1 C51 驱动安装

开发 C51 程序的首要任务是按照开发板驱动，基于普中科技的 C51 开发板是通过 CH340G 芯片下载程序，进行 USB 到 TTL 的程序烧录，所以我们首先应该安装 CH340 的驱动程序对于大多数电脑系统，将 USB 线连接电脑和开发板的 USB 接口后会自动检测安装 CH340 驱动。如果没有安装驱动可以通过手动安装。手动安装时应该保持 USB 线将电脑 USB 口和开发板 USB 接口连接，当显示“驱动安装成功”后则代表安装成功。

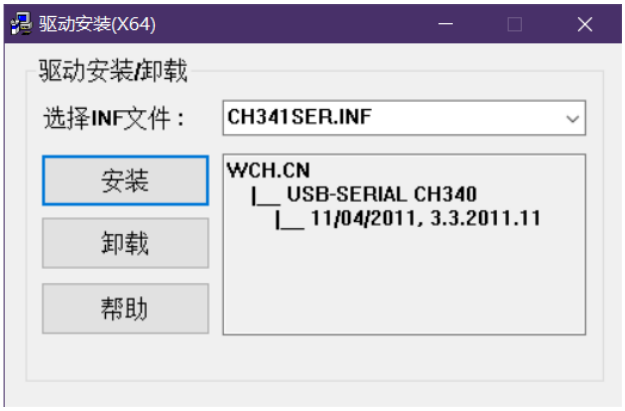


图 15 C51 驱动安装程序 CH340

4.2.2 C51 程序下载烧录

本设计采用普中科技的普中自动下载软件，当 CH340 驱动安装成功后将单片机和电脑进行连接。若设备连接正常，驱动程序安装正常，则在串口号一栏中会显示连接串口的编号。在我的电脑上的显示串口号是 COM7。一切正常后选择波特率 115200 和芯片类型 STC89Cxx（NEW），其他设置为默认设置。

配置好以后要打开文件，选择实验所需要的.HEX 文件，选择好后点击“程序下载”按钮即可以完成程序下载。当程序下载完成会提示程序下载成功。

图 16 普中科技自动下载烧录软件

4.2.3 C51 程序开发配置

C51 开发板的开发环境是 Keil uVision，其开发语言是 C 语言。一个 C51 程序的开发过程要有:新建工程、配置工程、调用库函数、编写程序、编译生成.HEX 文件等几个步骤。

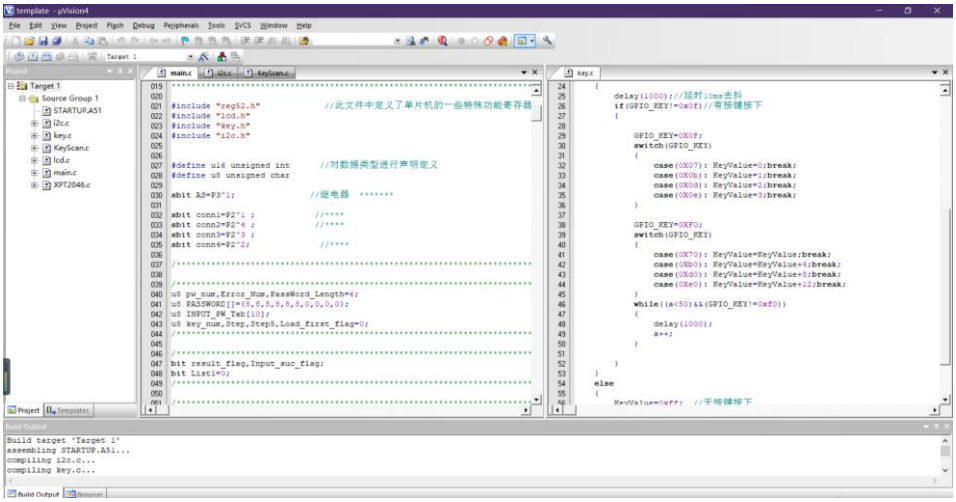
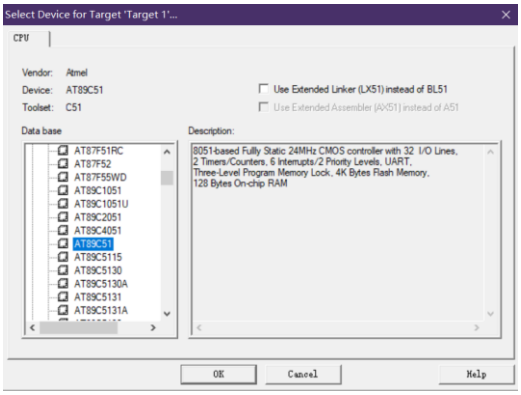


图 17 Keil 开发环境界面

首先打开 KEIL C51 软件，新建一个工程，工程名根据喜好命名，但是要注意使用英文，如果使用中文名可能会出现一些奇怪的错误，这里我们命名为 template，直接保存在你所要保存的文件夹下。接下来要选择根据开发板使用的 CPU 具体的型号来选择，这里选择 AT89C51。一个 51 单片机工程必须含有且仅有一个 main.c 函数，以及其他必要的库函数。选择 File/New...或者使用工具栏的图标来新建一个文件，后点击保存，系统会自动定位到我们工程目录，只需要在文件名栏输入新建的文件名即可。需要用到的库函数要复制其.c 文件和.h 文件并放到保存目录，然后导入到工程目录之中。选择工程组“Source Group 1”，

鼠标右键选择“Add Files to Group‘Source Group 1’...”,然后选择对应的xxx.c 文件，点击 Add 键后在点击 Close 关闭。这时工程中就显示已加入的文件。



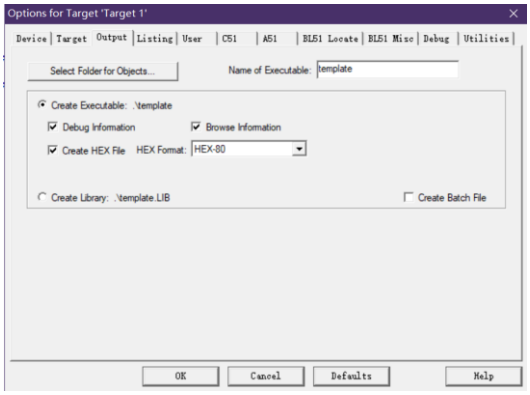


图 18 新建工程界面/ 图 19 配置输出界面

工程目录配置完成后下一步要配置魔术棒选项卡。选择魔术棒工具的Output 选项卡，勾选“Create hex file”，即程序编译成功后会产生 HEX 文件。其余设置保持默认即可。

4.3 AdaBoost 算法和 Haar 分类器概述

4.3.1 算法概述

基于 AdaBoost 的人脸检测算法是一个非常经典的算法，该算法是由 Viola 等于 2001 年在计算机视觉和模式识别会议上年发表的论文中提出。AdaBoost 全称为 Adaptive Boosting，是一种统计学习型的算法，给定大量的正样本和负样本，对人脸检测来说就是大量的“人脸”和“非人脸”样本，先训练成大量的弱分类器，形成弱分类器集。通过投票机制，按一定的规则，从弱分类器集中选出最优的弱分类器。把若干个最优弱分类器按照一定规则的权重组合起来形成了分类能力强的强分类器。最后利用 Cascade 的思想，把强分类器组合成级联分类器。Viola 等在计算 Haar 特征的过程中，创新性的引入了“积分图像”的概念，极大地提高在检测过程中计算人脸特征的速度。AdaBoost 的人脸检测算法的提出使得人脸检测技术取得了跨越性的进步，使实时人脸检测走向实际应用。AdaBoost 算法不仅可以用来检测人脸，而且可以检测其他目标，如人眼检测、嘴巴检测、车牌检测等。

所谓的分类器是一种计算机程序，它的设计目标是在通过自动学习后，可自动将数据分到已知类别。在这里人脸分类器指的是能够对人脸和非人脸进行区分的算法，Haar 分类器就是这样一种人脸分类器。Haar 分类器实际上是 Boosting 算法的一个应用，Haar 分类器用到了 Boosting 算法中的 AdaBoost 算法，将 AdaBoost 算法训练出的强分类器进行了级联，并且在底层的特征提取中采用了高效率的矩形特征和积分图方法。

表 3 Haar 分类器检测过程

（1）按照一定的比例对原图进行连续缩放；直到缩放得到的检测窗口小于 20×20 停止；
（2）穷尽每种尺寸图像的所有位置的20×20的子窗口；
（3）用分类器对所有子窗口进行人脸/非人脸的分类；
（4）计算分类得到的人脸子窗在原图中的矩形区域，并对每种尺寸图像检测到的重叠区域进行合并；
（5）输出检测结果。

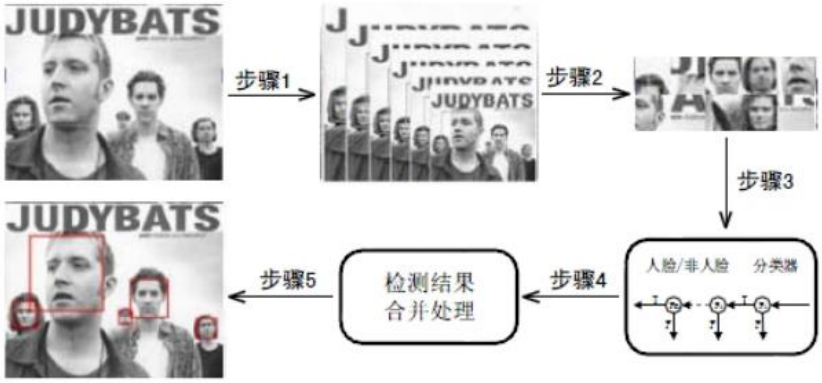


图 20　Haar 分类器检测过程示意图

Haar 分类器可分为训练与检测两个部分：训练部分是用给定正负样本集（一般为20×20尺寸的人脸与非人脸图像）对分类器进行训练，得到具有良好分类性能的分类器用于实际的人脸检测；而检测部分如表 3 和图 21 所示，检测过程是以实际的一幅图片作为输入，然后对图片中进行多区域，多尺度的检测，由于训练过程中使用的图片大小为20×20，因此需要对大尺寸的输入图像进行多尺度的检测。

Haar 分类器算法要点如下：

- 使用 Haar-like 特征做检测；
- 使用积分图对 Haar-like 特征求值进行加速；
- 使用 AdaBoost 算法训练区分人脸和非人脸的强分类器；
- 使用筛选式级联把强分类器级联到一起，提高准确率。

最早的 Haar-like 矩形特征库是由 Viola 等提出的，随后 Lienhart 等对 Haar-like 特征库进行进一步扩展。扩展后的特征主要分为 3 种类型：边缘特征、线特征、中心特征，如图 22 所示。将任意一个矩形放到待检测区域上，将白色区域的像素灰度值和减去黑色区域的像素灰度值和，得到的值便是 Haar-like 矩形特征值。一般人脸图像中头部外形基本一致并且五官分明，Haar-like 矩形特征能较好地描述人脸，并将人脸同背景或其他目标区分开来。

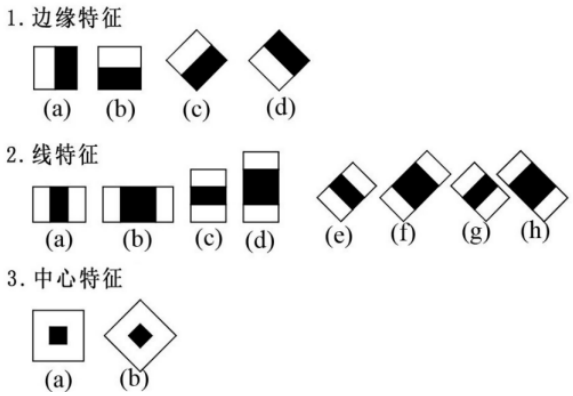


图 21 Haar-like 矩形特征

AdaBoost 算法能解决该问题，它在训练集上学习选出一系列弱分类器（如单个特征上基于阈值的分类器），并组合这些弱分类器构建成强分类器以实现稳健分类。其原理是通过迭代增加弱分类器来逐渐降低训练集的分类错误率。训练中的每个样本都有其权值，用于迭代训练弱分类器。若是某样本能被正确分类，则降低其权值;反之增加其权值,以使得在训练后续的弱分类器中起更大的作用。如此训练过程中算法将关注当前迭代中难以分类的样本,达到降低训练分类错误率的目的。

其基本训练过程为：平均设置样本的初始值；在每次迭代过程中依据权值训练出分类错误率最低的最佳弱分类器，然后更新权值，训练得到下一个最佳弱分类器，直至一定迭代次数或者错误率降低到一定的阈值后停止迭代，将所有训练得到的弱分类器组合成强分类器。弱分类器学习中，对每个特征首先搜索出区分所有正负样本的最优阈值，以保证误分样本数最少。以此特征及最优阈值建立该特征对应的弱分类器。

4.3.2 AdaBoost 人脸检测过程

通过训练学习大量的“人脸”和“非人脸”图像样本，最后得到了分类能力极强的级联分类器。检测时会遇到一个问题，待检测图像和训练得到的分类器的尺寸存在差别，这就需要把其中的一个进行缩放，因此可根据缩放目标将 AdaBoost 算法保持待检测图像保持不变两种情况。

保持待检测图像保持不变的情况是，对 Haar 特征检测窗口按照一定的比例进行缩放。这种方法不改变图像的大小，所以不用重新计算积分图，而 Haar 特征窗口的增大不会改变计算量，所以这种方法比第一种方法检测速度要快很多。这种方法在检测前需要提前设定好一个比例系数，为每一步 Haar 特征检测窗口缩放的比例。如果比例系数过大，搜索框的尺寸会增大的很快，从而导致人脸图像的漏检；如果比例系数设置的过小，不仅会降低检测的速度，还会导致运算量的增加。在对人脸图像进行搜索时，大部分的非人脸子窗口通过每一级的分类器

逐步排除掉。最终人脸图像即为通过了每一级分类器的窗口。当 Haar 特征检测窗口确定后，检测窗口在图像中移动搜索，每检测完一个位置后，按照一定的顺序移到下一个位置，移动的距离称为移动步长。当某个比例的 Haar 特征检测整个图像后，则按照比例系数改变 Haar 特征窗口的大小进行下一轮的搜索，直到 Haar 特征检测窗口的大小等于或大于图像的距离时结束。

对人脸检测，初始学习得出的弱分类器易于解释且分类性能好。如图 23 所示，第一次迭代出的矩形特征位于双眼区域，通过双眼和下方皮肤的灰度差别来体现，这个特征区域较大，因而对位移和尺度敏感度低。第二个特征则体现人眼灰度比鼻梁灰度暗。

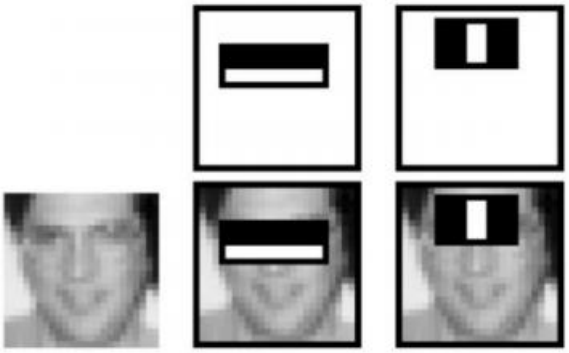


图 22 由 AdaBoost 训练得出前两个有效特征

Voila 等的正面人脸实验中，由 200 个特征组成的分类器在 14084 个子窗口的情况下，能达到 95%的检测率。检测效果良好，然而如果对每个窗口都提取 200 个特征并分类，会增加检测过程的计算量。若是增加弱分类器，在提升检测性能的同时也会增加计算时间。

在不同比例的 Haar 特征窗口检测时，有可能对同一个人脸区域进行重复的检测，或者检测区域会重合。检测结束后需要把不同比例下 Haar 检测窗口检测结果进行合并，把临近的人脸区域合并为一个区域，作为最终的结果。

4.4 基于 opencv-contrib-python 的机器视觉处理

4.4.1 OpenCV 关于人脸识别的相关功能

本文基于 AdaBoost 的人脸检测的实现是在 OpenCV(Open Source Computer Vision Library)的基础上完成的。opencv-contrib-python 是 OpenCV 对 Python 开发的程序包，其中集成了 Haar 分类器方法，它不仅适用于正面人脸，而且对侧面人脸、眼镜、嘴巴、鼻子等具有纹理近近似刚性的物体对象都适用。OpenCV 自带了一些训练好的 Haar 和 LBP 分类器，这样只需通过加载不同的级联分类器的 xml 文件就实现正面人脸、人脸轮廓（侧视）、眼镜或鼻子的检测了。

OpenCV 将 Haar 和 LBP 特征分类器的方法都封装在了 CascadeClassifier 类里，其中 CascadeClassifier::detectMultiScale()函数为检测函数，它的部分关键参数如下：

• minFeatureSize：检测窗口的最小尺寸，缺省值为20×20像素，这个值的大小可以根据实际的情景调整。例如，当在智能手机上执行人脸检测时，人脸一般总是很靠近镜头，可将该参数放大为80×80像素以得到更快的检测速度。而当人脸距离镜头较远的情境下则可将该参数大小设置为20×20像素。

• searchScaleFactor：在前后两次相继的扫描中，搜索窗口的缩放比例系数，通常设置为 1.1 就能得到较好的检测效果，若加大该参数值，虽然能加快检测速度但容易遗漏部分人脸。

• minNeighbors：构成检测目标的相邻矩形的最小个数（缺省值－1），如果组成检测目标的小矩形的个数和小于 minNeighbors-1 都会被排除。通常将该值设置为 3，如果需要所检测人脸的正确率更高，可将其设置得更大，但这样设置后会使部分人脸无法被检测到。

• flags：该参数指定是否要检测所有的人脸（默认）或者只查找最大的人脸，如果只查找最大的人脸，程序会运行的比较快。此外还有其他几个参数值可让人脸检测的速度提高1%~2%。

4.4.2 OpenCV 函数功能

在本设计中 OpenCV 主要用途是配置摄像头和视频、人脸图像捕捉、图片处理、人脸识别算法实现等。本部分将对实现上述功能的函数进行整理与介绍。

表 4 OpenCV 和其他机器视觉部分功能函数

函数	功能
cv2.VideoCapture(0)	用摄像头捕获视频，默认编号是 0，更换数字为选择不同的摄像头
cam.set()	设施摄像头捕捉视频长宽比
cv2.imread('image.jpg',0)	读入图像，image.jpg 为文件名
cv2.imwrite	保存一个图像，在本设计中用于人脸图像的储存
cv2.putText	在视频上添加文字，在本设计中用于显示拍摄信息和报警说明
cv2.CascadeClassifier	可以使用 Haar 或 LBP 的级联分析器函数，其中调用信息为“haarcascade_frontalface_default.xml”，其是一个用来识别人脸的 Haar 分类器模型，该模型为 OpenCV 自带模型库，为本设计能成功实现人脸识别的核心
cv2.cvtColor	转换颜色空间，在本设计中需要将图像转换为灰度图像
face_detector.detectMultiScale()	用于人脸检测的函数，用于识别一张图片中的人脸信息
cv2.rectangle	画矩形，在本设计中用于框选出人脸图像

cv2.namedWindow	为窗口命名
cv2.imshow('image',img)	显示图像
cv2.setWindowProperty	设置窗口格式
cv2.waitKey(0)	一个键盘绑定函数，接收键盘操作指令
cam.release()	释放摄像头
cv2.destroyAllWindows()	可以删除所有建立的窗口，主要用于视频节目的退出操作
cv2.face.LBPHFaceRecognizer_create()	该函数功能是生成 LBPH 识别器比例模型
cv2.face_FaceRecognizer.train()	该函数的功能是对生成的模型和捕捉到的图片数据库进行训练，为了生成用于人脸识别的识别库
cv2.face_FaceRecognizer.predict()	该函数的功能是对一个待测人脸图像进行识别判断，寻找与当前图像距离最近的人脸图像。与哪个人脸图像最近，就将当前待测图像标注为其对应的标签。当图像中没有合适的人脸数据后能返回无法识别信息。

通过以上函数，我们可以很轻松的使用 python 编写人脸识别算法。

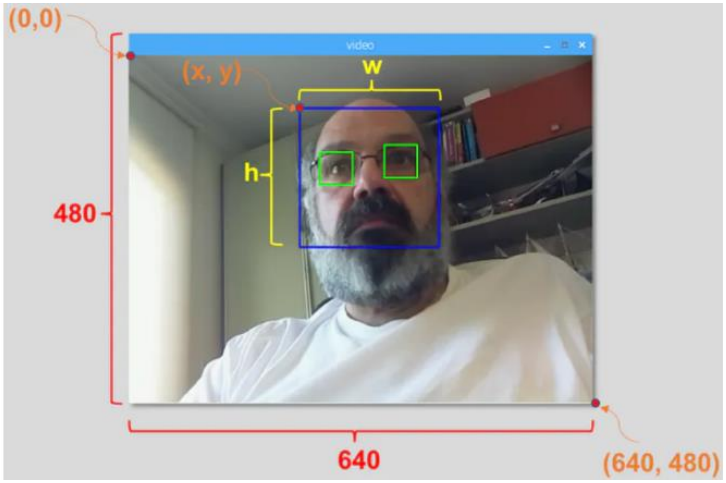


图 23 基于 python 和 opencv 的人脸图像处理案例

5. 图片集说明及标记

本设计是人脸识别门禁，所以顾名思义，本设计的图片集就是人脸。由于人脸识别算法模型 OpenCV 已经给出，所以我们只需要对人脸图像进行训练即可。本门禁的可通行人员是课程指导老师和 24 名班级同学，**共计 25 人。**图片集中预留出一位 id 为 0 的“NewGuest”用于新建人脸的图像储存，以及 4 位无关人员（图像来自网络），目的是增加训练样本，仅作为训练，不设置开门权限。不在库中的人脸和无权限人员都会被识别为“UnKnown”。

在图片库建立的过程中，会**对单一人员连续采集 100 张照片，并按序号进行储存**，并同时进行模型训练，在训练一次图片集后会对识别模型进行检验，若出现识别失败或者不匹配的情况，那么就对该人员进行重新图像采集。

图 24 人脸识别图片集（编号依次为 1-28）

图 24 为人脸图片集的所有对象。每一个人的图片集信息直接写在程序中，以数组的形式储存。所有的图片集序号及代号和对应人物整理如表所示。其中表示出红色的为无权限人员，表蓝色的为待采集人员。相关代码为：

```
# names 关联 ids: example ==> NewGuest: id=0,
names = ['0 NewGuest', '1 Niu Zhanhua', '2 Lu Sichen', '3 Feng Chengbo', '4 Bi Yifei', '5 Quan Wenxu','6 Liu Chenyu','7 Liu Nengdi','8 Su Shijia','9 Li Wang','10 Li Laiyuan','11 Yang Kuan','12 Di Weiran','13 Song Yuqing','14 Zhang Xuanhao','15 Zhang Ruitao','16 Chen Zihan','17 Chen Yang','18 Chen Runzhou','19 Chen Linfan','20 Jin Mingjun','21 Yao
```

Junhui','22 Gu Jiawen','23 Tang Kaixuan','24 Peng Hongyu','25 Xia Fei','UnKnown','UnKnown','UnKnown','UnKnown']

表 5 图片集序号及其代号

在人脸识别门禁系统中，若识别到在序号为 0-25（当 0 号人员被采集到信息之后）时，电磁门禁会开门；若识别到的是无权限人员或者不在图像库中的人员，则会统一报出“UnKnown”并且保持电磁门禁上锁。

6. 调试过程及成果展示

6.1 模块功能设计实现

本部分将分模块分步骤叙述系统程序、硬件连线的开发过程。主要将从四个部分进行描述：人脸识别库的建立、人脸识别与开门信息传递、密码门禁过程、主控制界面及 LCD 显示。整个系统的工作组成将在第四模块中进行叙述。在模块调试过程中遇到的问题将在第七章“问题讨论”中描述。

6.1.1 人脸识别库的建立

人脸识别库的建立是人类识别门禁能成功运行的关键，其依托的主要原理是的第四章已经介绍过的“AdaBoost 算法”和“Haar 分离器”。人脸识别库建立的所实现的主控制器是树莓派，配套硬件设备是显示器、PiCam 摄像头。显示器通过 HDMI 线传输视频数据，通过 Type-C 线传输控制和触屏指令。PiCam 通过排线与树莓派摄像头接口直接连接，摄像头可以在树莓派上通过 Python 指令直接调用。该部分的硬件连接图如图 25 所示。

图 25　树莓派部分硬件连接图

其功能实现程序设计方案如下步骤所示。

1. 导入函数包与模型库

根据本功能的要求，需要导入的函数包是与机器视觉、操作系统调用、矩阵计算和图片处理，所以需要导入的函数包为 OpenCV、OS、numpy、Pillow。其调

用程序为：

```
import cv2
import numpy as np
from PIL import Image
import os
```

2. 设置摄像头视频信息

首先第一步我们要开启摄像头和设置拍摄与视频信息。需要设置视频长宽、将视频数据存入数组，并设置显示界面的长宽。这里设置视频尺寸为640×480，定义视频数据名称为 img，窗口名称为 output_style_full_screen，屏幕为全屏显示。其相关程序操作为：

```
cam = cv2.VideoCapture(0)
cam.set(3, 640) # 设置视频宽度
cam.set(4, 480) # 设置视频高度
ret, img = cam.read()
out_win="output_style_full_screen"
cv2.namedWindow(o ut_win,cv2.WINDOW_NORMAL)
cv2.setWindowProperty(out_win,cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_
FULLSCREEN)
cv2.imshow(out_win, img)
```

3. 分配人脸编号

因为每一个人脸需要有一个特定的人脸标号，所以需要在每次读取新的人脸图片数据之前给定一个标号，这个标号可以在程序中制定，也可以在键盘中输入。

```
face_id = input('\n enter user id end press <return> ==>  ')
```

4. 人脸图像处理与储存

图像处理环节需要执行的操作有，将图像转换为灰度图像、根据人脸模型识别视频中的人脸、人脸图像标示、人脸图像储存等。视频每次可以读取到多个人脸，并同时将所有的人脸在视频上的位置坐标储存在元胞数组 faces 中，元胞数组一组数据的四位数据$[x, y, w, h]$代表的为人脸左上角横坐标、纵坐标、横向偏移量、纵向偏移量。程序在采集到一次人脸信息之后能将所有的人脸在屏幕上用矩形标示出来。同时标示出来的图像保存值本地的 dataset 文件夹，储存一张照片命名序号加 1,。人脸图像储存至 100 张照片结束采集操作。

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_detector.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
   cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
   count += 1
   cv2.putText(img,str(count),(40,50),cv2.FONT_HERSHEY_PLAIN,1.5,
```

```
(0,255,0),1)
        #将捕获的图像保存到数据集文件夹中
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) +
 ".jpg", gray[y:y+h,x:x+w])
```

图 26　多人脸识别过程

5 人脸识别库训练

人脸识别库的实现靠 Haar 分类器 haarcascade_frontalface_default.xml 来实现，人脸识别过程首先应该建立分类器，导入人脸图像数据库的路径，获取其图像与表签数据，将图像转换为灰度，在对每一组人脸图像带入分类器模型进行训练。全部的人脸图像训练完毕后将导出到/train 地址下的 train.yml 模型，最后报出训练完成信息提示。

```
    path = 'dataset'   # 人脸图像数据库的路径
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    detector = cv2.CascadeClassifier("haarcascade_frontalface_default.
xml");
    # 定义函数获取图像和标签数据
    def getImagesAndLabels(path):
        imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
        faceSamples=[]
        ids = []
        for imagePath in imagePaths:
            PIL_img = Image.open(imagePath).convert('L') # 将其转换为灰度
            img_numpy = np.array(PIL_img,'uint8')
            id = int(os.path.split(imagePath)[-1].split(".")[1])
            faces = detector.detectMultiScale(img_numpy)
            for (x,y,w,h) in faces:
                faceSamples.append(img_numpy[y:y+h,x:x+w])
                ids.append(id)
        return faceSamples,ids
```

```
    print ("\n [INFO] Training faces. It will take a few seconds. Wait
 ...")
    faces,ids = getImagesAndLabels(path)
    recognizer.train(faces, np.array(ids))
    recognizer.write('trainer/trainer.yml') # recognizer.save() worke
d on Mac, but not on Pi   # 将模型保存到 trainer/trainer.yml 中
    print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.
unique(ids))))    # 输出人脸训练数值，结束程序
```

6.1.2 人脸匹配与开门信息发送

人脸识别部分的功能是基于第一模块训练好的人脸识别库，对摄像头拍摄到的人脸图像进行比对，并进行开门信号的发出。这一部分也需要对摄像头和视频输出进行配置、调用第一个环节中的模型库和人脸识别库，以及同人脸采集中的图像处理部分等。由于第二模块所需的程序包同第一模块，本部分不再叙述。其主要程序步骤及效果演示如下所示。

1. 导入人脸识别模型、视频信息初始化

本模块首先需要建立分类器模型，然后导入前一个模块训练好的识别库，同时也要把 Haar 分类器导入，这一步用于对平幕上的人脸进行识别与采集，也就是说人脸匹配首先要对视频中的人脸进行识别。接下来同第一模块中的设定一样，需要对视频采集与显示信息进行参数设定。

```
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read('trainer/trainer.yml')
    cascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(cascadePath);
    # 导入人脸 ID 与名称信息（同第五章节）
    id = 0
    names = ['0 NewGuest', '1 Niu Zhanhua', '2 Lu Sichen', '3 Feng Chen
gbo',…（同第五章节）…,'UnKnown','UnKnown']
    # 初始化视频与窗口格式
    cam = cv2.VideoCapture(0)
    cam.set(3, 640) # 设置视频宽度
    cam.set(4, 480) # 设置视频高度
    minW = 0.1*cam.get(3)
    minH = 0.1*cam.get(4)
    # 视频窗口显示
    out_win="output_style_full_screen"
    cv2.namedWindow(out_win,cv2.WINDOW_NORMAL)
    cv2.setWindowProperty(out_win,cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_
FULLSCREEN)
```

```
cv2.imshow(out_win, img)
```

2. 图像处理与人脸匹配

在人脸识别与匹配过程中同样要重复第一模块中的人脸图像切片工作，并将该图像切片与库中的图像进行比对，其匹配的原理是根据模型匹配算法的置信度进行最优匹配。如果能与库中的数据进行匹配，则输出一个开门信号；如果识别出来的是无权限人员与不在库人员，则不输出开门信号。

```
ret, img =cam.read()
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(
   gray,
   scaleFactor = 1.2,
   minNeighbors = 5,
   minSize = (int(minW), int(minH)),
   )
for(x,y,w,h) in faces:
   cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
   id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
   # 检测是否置信度小于 100 ==> "0" 是完美匹配
   if (confidence < 100):
       id = names[id]
       confidence = "  {0}%".format(round(100 - confidence))
   else:
       id = "unknown"
       confidence = "  {0}%".format(round(100 - confidence))
```

6.1.3 密码门禁过程

本密码门禁模块和下面的主控制器模块均由 STC89C51 开发板来实现。密码门禁模块功能在前面方案总述中已经讲过，主要要实现的功能是输入密码，判断密码是否正确，若正确发送开门信号；若错误发送错误信息，并返回主界面。本部分的主要逻辑是 switch-case 结构，通过步骤标记变量 Step 来指示当前程序执行的步骤。

本模块及下个模块的硬件系统为一块 STC89C51 开发板，需要用到其上面的 C51 最小系统板（含 C51 微处理器、晶振时钟电路、复位电路和 GPIO 排针引脚）、LCD1602 驱动电路、4×4矩阵键盘、步进电机驱动模块等。其所有的模块都已经板载连接好，所以省去了连线的过程，使该开发板工作仅需要连接电源线并烧录好程序即可。

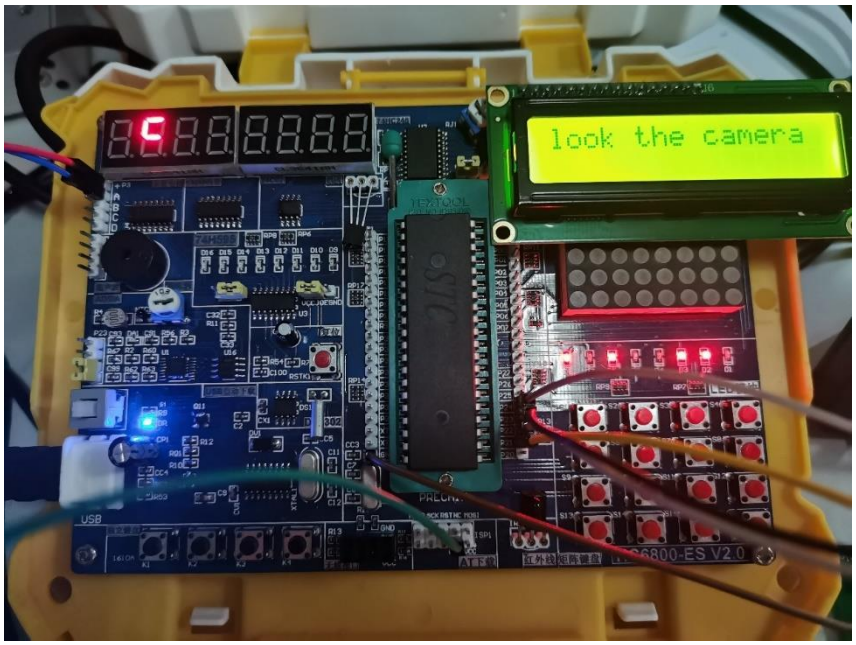


图 27　C51 连接及硬件结构

由于 C51 编程的开发特点，本模块程序算法描述部分不描述密码门禁的整体代码设计，只介绍密码门禁几个工作环节的程序代码设计，C51 部分的主要系统程序设计将在下一个模块“主控制系统”中做介绍和描述。本模块的实现程序由下面几个步骤组成，矩阵键盘的按键规则如表 xxx 所示，密码存储在数组 PASSWORD 中。

表 6　4×4矩阵键盘按键规则（“功能”-按键 16 进制值）

“0”-0x00	“1”-0x01	“2”-0x02	“3”-0x03
“4”-0x04	“5”-0x05	“6”-0x06	“7”-0x07
“8”-0x08	“9”-0x09	“返回”-0x0a	“确认”-0x0b
“A”-0x0c	“B”-0x0d	“C”-0x0e	“D”-0x0f

1. Step1：密码模式进入

本步骤是在 LCD 上做文字提示，按下任意键进入 Step2。

```
LcdWriteCom(0x01);          //清屏
ShowString(0x00,"Unlock");
ShowString(0x0f,"<");     //1602 第一行显示 unlock
ShowString(0x10,"Change Password");
ShowString(0x1f," ");     // 1602 第二行显示 Change Password
Step=2;
```

2. Step2：选择模式

本步骤的功能选取工作模式，分别是密码开门模式（按下“确认键”）和修改密码模式（按下“1”或“9”键）。密码开门模式进入Step3，修改密码模式进入Step5。其中key_num变量储存的是矩阵按键当前被按下的值，以十六进制的形式储存。

```
if(key_num!=0x0b) {
   if((key_num==0x01) ||( key_num==0x09))  //1 键或 9 键按下
   {   List1=~List1;                          //Change Password
       if(List1==0) {
           ShowString(0x0f,"<");    // Unlock
           ShowString(0x1f," ");       // Change Password
       }else{
           ShowString(0x0f," ");          // Unlock
           ShowString(0x1f,"<");       // Change Password
}}}
else      //确认键按下
{  if(List1==0){Step=3;}
   else   {Step=5;List1=0;}
}}
```

3. Step3：提示输入密码

这一步中，系统已经进入了密码开门模式，系统在LCD上做出输入密码的提示词。其中pw_num是密码长度，默认初始密码为888888。

```
Step=4;
pw_num=0;
LcdInit();
ShowString(0x00,"Pass Word:  ");
```

4. Step4：读取密码

这一步是主函数调用输入密码函数 imput_password()来对密码进行输入操作，当“确认”键被按下则完成输入，同时清除密码输入完成标志Input_suc_flag。imput_password()函数详见附录,其功能是读取键盘数字信息，并将读到的密码输入到数值INPUT_PW_Tab中，同时对在LCD上显示“*”，并修改密码长度变量pw_num。本程序中用到的pw_num，PASSWORD，INPUT_PW_Tab等都是全局变量。

```
input_password(0);     //输入密码并以*显示
if(Input_suc_flag==1){Step=6;}    //密码输入完成进入 Step6
Input_suc_flag=0;
```

图 28 步骤提示与密码输入显示

5.Step6：密码比对

这里先描述密码比对的过程。使用 CipherComparison()函数来对比前一个步骤中读到的密码与 PASSWORD 中的密码，如果正确，则结果标志 result_flag 为 1，发生开门信号，同时在 LCD 上做欢迎提示并延时约 5 秒。延时结束后关闭门禁，程序返回主页面。若密码错误则 LCD 显示错误提示并返回主页面。

```
u8 i=0;
CipherComparison();              //密码比对
if(result_flag==1)          //密码正确
{  LcdInit();
   ShowString(0x00,"    WELCOME!");
   for(i=0;i<60;i++)   //循环 100 次，也就是大约 5S
   {   AS=0;           //开门
       delay(5000);  //大约延时 50ms
   }AS=1;    //关门
}
else    //密码错误
{  LcdInit();
   ShowString(0x00,"Error 01!");
}Step=0;
```

其中的密码对比函数 CipherComparison()详细见附录，其功能是对比全局变量 PASSWORD 和 INPUT_PW_Tab 的值，如果正确则程序继续进行，如果错误则返回对比结果标志 result_flag=0（默认下 result_flag=1，即密码正确）。

6. Step5：修改密码

当在 Step2 中选择了修改密码模式，则可以对 PASSWORD 数组中的数字进行修改，这个修改过程又分为 Step5_0 ~ Step5_5 共计 6 个步骤，其执行的功能依次是：0.输入密码提示、1.输入旧密码、2.密码对比、3.密码错误重新输入、4.密码正确输入新密码、5.储存新密码。这里的主要程序与前面提到的 Step1~Step6 高度相似，所以这里主要描述 Step5_3 和 Step5_5 的程序，其他环节的代码见附录。本步骤下的分布骤标志变量为 Step5，其值 x:0~5 代表分程序

将指向分步骤 Step5_x。

Step5_3 的功能是若对比密码错误则触发重新输入指令，由变量 Error_Num 储存输入循环次数，输入次数若超过三次则直接返回主界面，若密码正确则进入 Step5_4 步骤。程序执行的同时 LCD 也给出相应的提示。

```
void Step5_3()
{  if(result_flag==0)          //密码错误
   {  if(Error_Num<3)          //输出错误次数小于 3
      {  Error_Num++;
         LcdInit();
         ShowString (0x00,"Error 01");
         delay(20000);
         Step5=0;
      }else                    //密码错误次数大于 3
      {  Error_Num=0;
         Step=0;
      }}
   else                        //密码正确
   {  LcdInit();
      ShowString (0x00,"New PassWord:");
      pw_num=0;
      Step5=4;
}}
```

Step5_5 储存在 Step5_4 中修改的密码，这里因为输入得到的密码储存在全局变量数组 INPUT_PW_Tab 中，而密码储存在 PASSWORD 中，这一步是将 INPUT_PW_Tab 逐位替换掉 PASSWORD。其中密码的最长位数为 PASSWORD 的定义长度 10 位。储存过程结束后返回主界面。

```
void Step5_5()
{  unsigned char j;
   PassWord_Length=pw_num;          //读取输入密码长度
   At24c02Write(0,Load_first_flag);delay(100);
   At24c02Write(1,PassWord_Length); delay(100); //保存密码长度
   for(j=0;j<PassWord_Length;j++)
   {  PASSWORD[j]=INPUT_PW_Tab[j];     //读取密码
      At24c02Write(j+2,INPUT_PW_Tab[j]);      //保存密码至 EEPROM
      delay(100);}
   Step5=0;Step=0;
}
```

6.1.4 主控制系统

前面的三个模块其实已经讲到了本设计的绝大部分设计功能的硬件连线与

程序设计，这一个部分是站在系统的角度上，以 C51 为系统主控制器，串联起全部的硬件，以及介绍 C51 实现对全部功能的控制程序。

该电路的硬件系统除了前面的树莓派机器视觉系统和 C51 的密码门禁系统之外，还需要与电磁门禁相连接，以及配置继电器。此外还有四根通信线和若干供电、共地线需要连接。该系统的控制器引脚分配与连线如表 7 所示，实物连接图如图 38 所示。

表 7　控制器引脚分配方案与连线规则

STC89C51 控制器

模块	GPIO 引脚连接（控制器-模块）		模块	GPIO 引脚连接（控制器-模块）	
矩阵键盘	P1.0~P1.7	0~7	电磁门禁	P1.0	门禁 +
继电器	P2.3	NO1		GND	门禁 GND
	GND	COM1	树莓派	P2.1	GPIO5
LCD	P0.0~P0.7	DB0~DB7		P2.2	GPIO19
	P2.5~P2.7	WR/RD/LCDE		P2.4	GPIO23
	VCC/GND	VCC/GND		GND	Board39

Raspberry Pi 4B 控制器

模块	GPIO 引脚连接（控制器-模块）		模块	GPIO 引脚连接（控制器-模块）	
C51	GPIO5	P2.1	继电器	GPIO26	NO1
	GPIO19	P2.2		Board2	DC+
	GPIO23	P2.4		Board6	DC-
	Board6	GND			

有了以上的硬件结构，接下来要对系统主控制器 C51 的程序进行分析。其主要逻辑采用和上一个密码门禁模块相同的 switch-case 步骤逻辑，事实上密码门禁的步骤是主控制器程序中的一部分，完整的控制代码的 switch-case 步骤逻辑共有 Step0~Step10，共计 11 个大步骤以及 Step5 下面的 6 个分步骤总计 17 个步骤。这里不再详细展开第三模块中的密码门禁部分了，主要介绍程序的主体部分和控制功能。

1. 程序头文件与库函数

一个 C 程序中首先要做的事情是声明头文件，即调用本设计中需要的库函数。由于我们使用了 LCD1602 模块和矩阵按键模块，所以需要调用两者的库函数 lcd.c 和 key.c。同时因为需要使用 C51 单片机的内部资源，所以需要调用库函数 reg52.c 和 i2c.c。由于所有的库函数都是由普中科技开发，所以在本设计程序中我们直接调用相关函数，库函数内容详见附录。其相关代码是：

```
#include "reg52.h"
#include "lcd.h"
```

```
#include "key.h"
#include "i2c.h"
```

同时由于C51单片机的特殊性，故要对其数据类型进行一个声明。本设计采用通用的定义规则，即：

```
#define u16 unsigned int
#define u8 unsigned char
```

2. 函数声明与全局变量

在写main函数之前，需要对其中自主定义的函数进行声明，以及对全局变量做一个定义。在C51单片机开发中还有一个重要过程就是定义引脚。需要声明的函数主要有Step0()~Step10()，中断服务函数，密码处理函数等。需要自主定义的GPIO是负责控制继电器、门禁、与通信的管脚，其他模块的GPIO定义过程在其对应的库函数头文件中已经有所声明。

```
//配置门禁与通信引脚
sbit AS=P3^1;
sbit conn1=P2^1;//模式A信号;
sbit conn2=P2^4;//模式B信号;
sbit conn3=P2^3;//门禁开门信号;
sbit conn4=P2^2;//中止信号
//配置密码全局变量
u8 pw_num,Error_Num,PassWord_
Length=6;
u8 PASSWORD[]={8,8,8,8,8,8,0,
0,0,0};
u8 INPUT_PW_Tab[10];
u8 key_num,Step,Step5,Load_fi
rst_flag=0;
bit result_flag,Input_suc_fla
g;
bit List1=0;
//步骤函数声明
void Step_0();
void Step_1();
void Step_2();
void Step_3();
void Step_4();
     void Step_5();
      void Step5_0();
      void Step5_1();
      void Step5_2();
      void Step5_3();
      void Step5_4();
      void Step5_5();
void Step_6()
void Step_7();
void Step_8();
void Step_9();
void Step_10();
void Int0Init();
//密码处理函数声明
void CipherComparison();
void input_password(bit m);
void Read_Password();
```

3. 主函数：程序初始化

在main函数中，首先需要对一些变量赋初值，并且要对LCD做初始化。其次是要对引脚电平做设定，进而开启外部中断。最后进入主界面程序，主程序是一个while(1)的无限循环,只要电源保持供电就能一直运行。其中conn1、conn2、conn4均为低电平，指C51不对树莓派有任何操作；门禁信号会保持高电平，即关门状态；函数KeyDown()是由库函数ksy.c定义的键盘值读取函数，代表C51

接收来自键盘的按键信息。Step0、Step9、Step10 为主界面步骤，Step1~Step6 为第三模块中实现密码门禁的步骤，Step7 是人类图像建库步骤，Step8 是人类识别门禁步骤。其主要的程序如下。

```
    u8 data1,a;
    LcdWriteCom(0x01);  //清屏
    for(data1=0;data1<PassWord_L
ength+2;data1++)
    {   a=At24c02Read(data1)+0x3
0;
        LcdWriteData(a);
        delay(1000);   }
    delay(1000);
    LcdInit();
    delay(1000);
    Step=0;
    Step5=0;
    Error_Num=0x00;
    Read_Password();
    Int0Init();
    conn2=0;
    conn1=0;
    conn4=0;
```

```
    while(1) {
        AS=1;
        key_num=KeyDown();  //读取
键盘输入值
        switch(Step)
        {case 0:{Step_0();break;}
        case 1:{Step_1();break;}
        case 2:{Step_2();break;}
        case 3:{Step_3();break;}
        case 4:{Step_4();break;}
        case 5:{Step_5();break;}
        case 6:{Step_6();break;}
        case 7:{Step_7();break;}
        case 8:{Step_8();break;}
        case 9:{Step_9();break;}
        case 10:{Step_10();brea
k;}
    }
```

4. 主函数：主界面程序

当前一个环节结束之后，程序自动进入 Step0。该环节中主要是一些 LCD 信息提示与模式选择功能。其逻辑是 Step0 中 LCD 给出欢迎信息，按下任意键进入 Step9，LCD 给出模型选择提示，再按下任意键进入 Step10，开始读取模型选择信息。共计三个模式选择“A”，“B”，“C”和一个退出选择“D”，退出即返回主界面 Step0 的状态。相关程序为：

```
    void Step_0()
    {   LcdInit();
        ShowString(0x00,"face-recognition");
        ShowString(0x10,"    WELCOME!     ");
        while(KeyDown()==0xff)Step=9;
    }

    void Step_9()
    {   LcdInit();
        ShowString(0x00," A:New B:facial ");
        ShowString(0x10," C;Key D:close ");
        while(KeyDown()==0xff)Step=10;
    }
```

```
void Step_10()
{   if(key_num==0x0c) {Step=7;}
    else  if(key_num==0x0d) {Step=8;}
    else if(key_num==0x0e) {Step=1;}
    else if(key_num==0x0f) {AS=1;Step=0;}
    else Step=10;
}
```

图 29 主系统界面/ 图 30 模式选择界面

5. A 模式：人脸图像建库

该部分是在 C51 的主控制下，树莓派配合操作。该功能的实现首先需要在树莓派上调试出待检测状态，这里就需要使用命令定义 GPIO，按照前面表格中给定的通信方式，定义相关引脚的程序为：

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(19,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(26, GPIO.OUT)
GPIO.output(26, 1)
```

上一段代码的含义是,选定引脚模式为 BCM 模式,设定 GPIO5、GPIO23、GPIO19 为输入模式，配置上拉电阻；设定 GPIO26 为输出模式，默认输出高电平。设定当 GPIO5 为高电平时开启人脸图像建库模式，当 GPIO23 为高电平时开启人脸识别门禁模式。当 GPIO5 和 GPIO23 均为低电平时，则程序会保持等待。在树莓派等待的同时,C51 一侧在 Step10 中按下“A”键进入人脸图像建库模式,即 Step7。首先通过通信通道 conn1 向树莓派 GPIO5 发送高电平,激活树莓派开始运行人脸图像建库。接下来 C51 会等待树莓派的训练成功信息。树莓派在训练完毕之后通过 GPIO26 向 C51 的 conn3 通道发送低电平，当 C51 接收到低电平就结束模式 A，返回主界面，同时将 conn1 置 0。

```
u8 i=0;
LcdWriteCom(0x01);  //清屏
conn1=1;
conn2=0;
LcdInit();
```

```
    ShowString (0x00,"Recognitng...")
    key_num=KeyDown();

    if(conn3==0) {
       Step=0;
       ShowString (0x10,"complete!");
       delay(10000);
       conn1=0;
       conn2=0;
       Step=0;
       break;
    }
```

程序中止功能是在按下“D”键之后能立即中止图像采集的工作，并且让 C51 系统返回主界面，其实现只需要多加一个判断即可。

```
    else if(key_num==0x0f)
    {  ShowString(0x10,"    Return!");
       AS=1;
       conn4=1;
       for(i=0;i<60;i++)  {delay(5000);}
       conn1=0;
       conn2=0;
       Step=0;
       conn4=0;
       break;
    }
```

图 31 人脸图像建库模式 LCD 提示

6. B 模式：人脸识别门禁

人脸识别门禁的功能与 A 模式在 C51 方面做的操作基本无异，conn2 发送信号给 GPIO23、检测反馈电平，并且也有中止操作。这一环节增加的功能是控制电磁门禁开闭，其实现代码与密码开门过程相同。

```
    u8 i=0;
    LcdWriteCom(0x01);  //清屏
    ShowString (0x00,"look the ca
mera");
                                    conn1=0;
                                    conn2=1;

                                    key_num=KeyDown();
```

```
    if(conn3==0)                          n!");
    {  ShowString(0x10,"    WELCO                 AS=1;
ME!");                                            conn4=1;
       AS=0;                                      conn1=0;
       for(i=0;i<100;i++) {AS=0;                  conn2=0;
delay(5000);}                                     for(i=0;i<60;i++)  { delay
       AS=1;                                (5000);}
       delay(60000);                              Step=0;
    }                                             conn4=0;
                                                  break;
    else if(key_num==0x0b)                     } Step=0;
    {  ShowString(0x10,"    Retur
```

图 32 人脸识别门禁模式 LCD 提示

7. C 模式：密码门禁

密码门禁部分的功能实现在前一个模块中已经全部介绍过，由于其只需要 C51 自己就可以实现，所以不需要进行额外的通信操作。

综合前面的主控制程序及功能逻辑，总结出主要程序流程图如下所示。图 33 为 C51 系统程序框图，图 34 为树莓派系统程序框图，图 35 为人脸图像采集程序框图，图 36 为人脸图像集训练程序框图，图 37 为人脸识别程序框图。

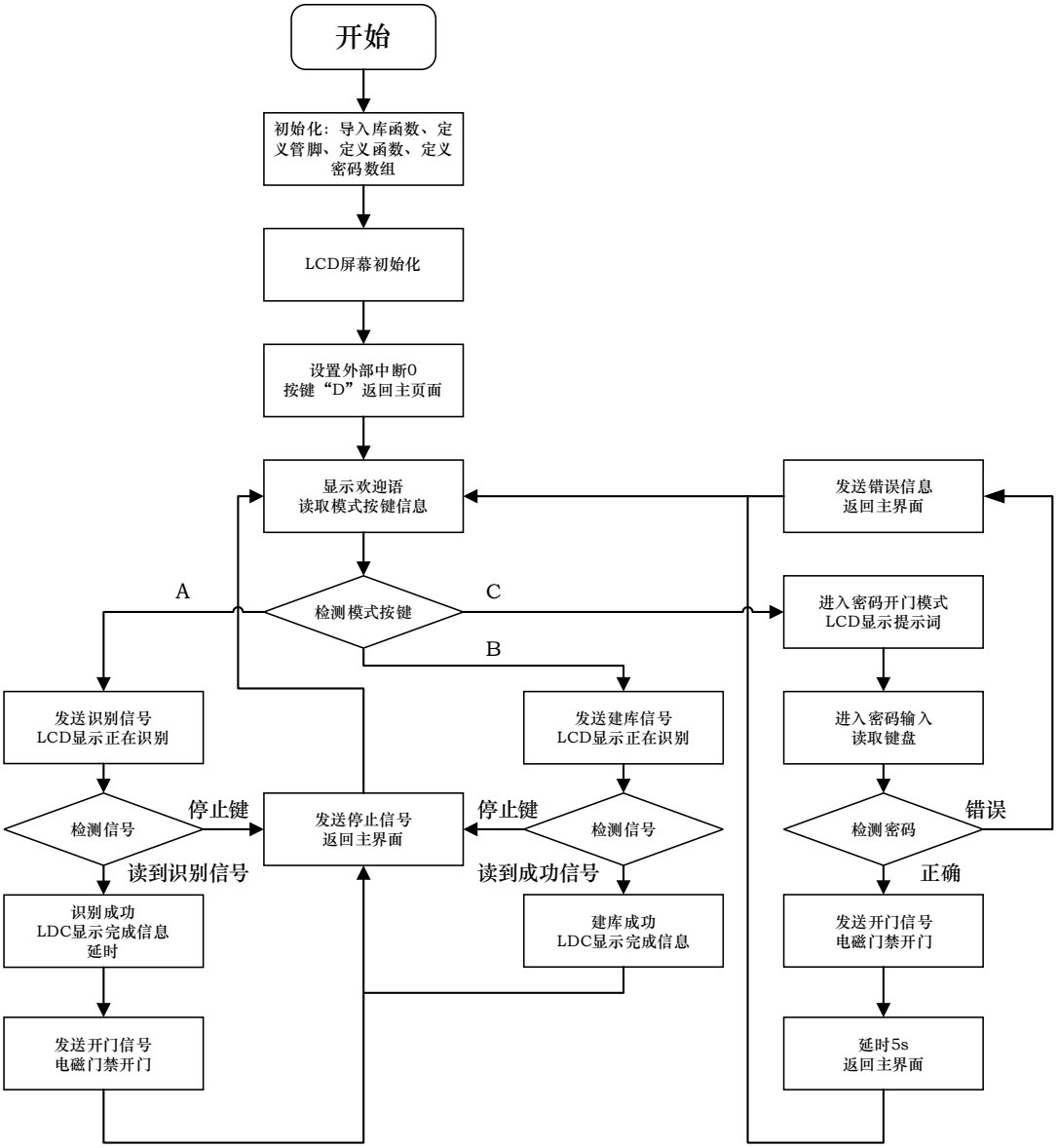


图 33　C51 主系统程序框图

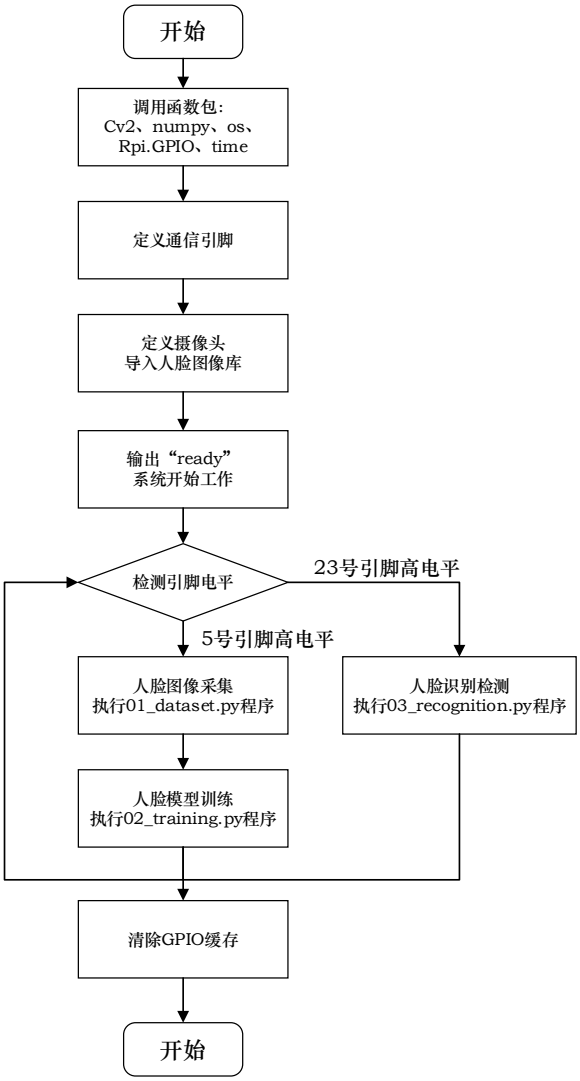


图 34　树莓派系统程序框图

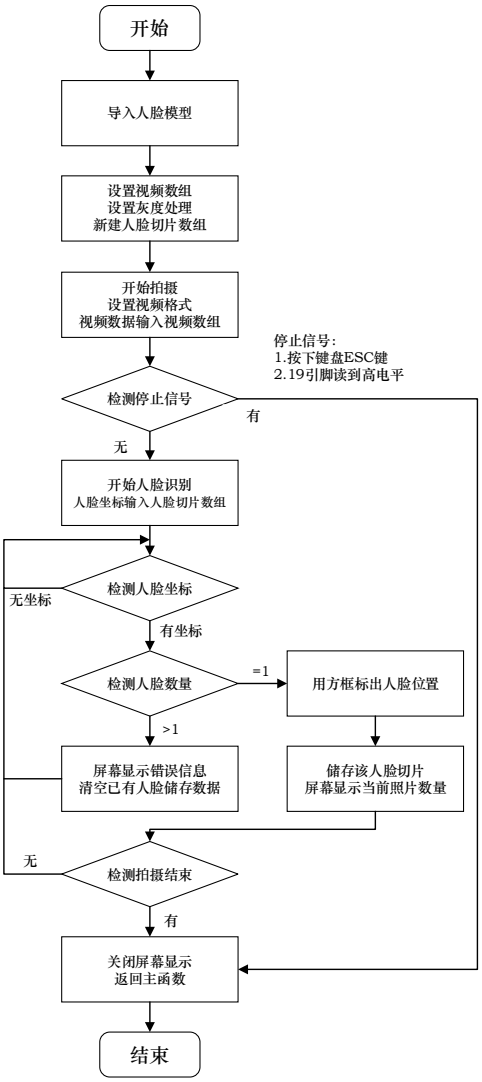


图 35　人脸图像采集程序框图

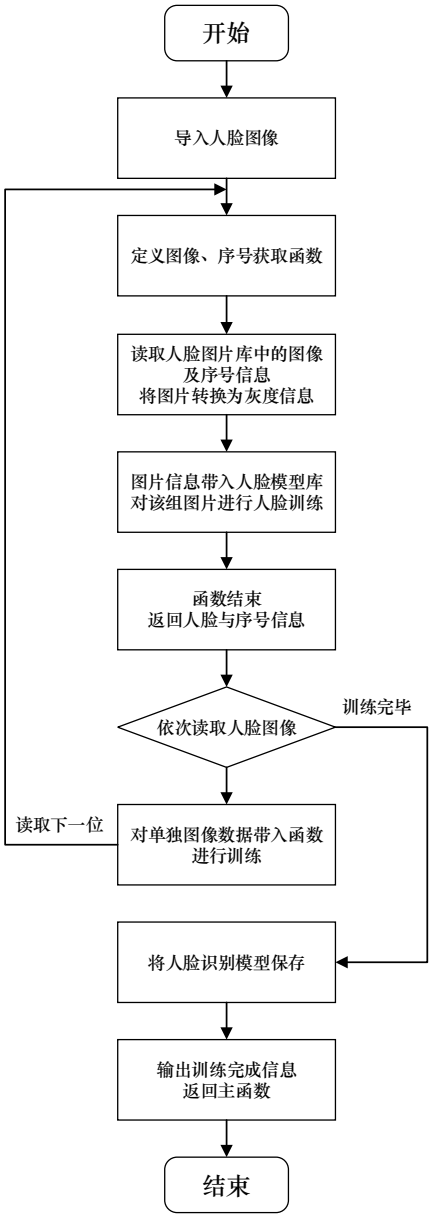


图 36　人脸图像集训练程序框图

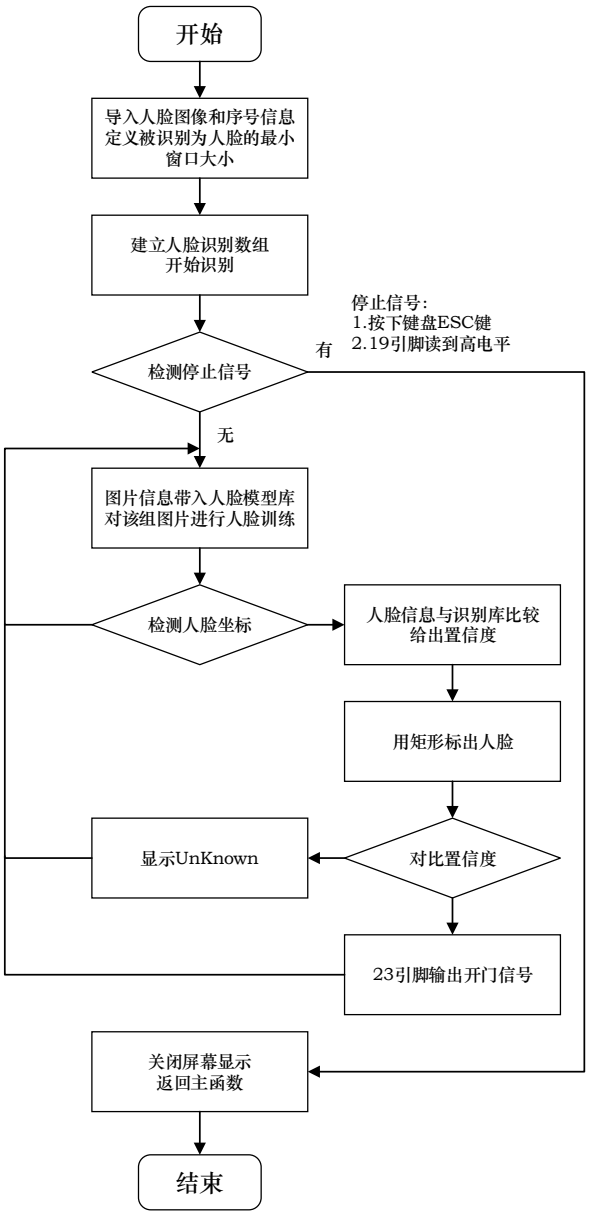


图 37　人脸识别程序框图

6.2 系统调试操作说明

程序全部设计完毕以后，可以进行系统调试工作。调试的目的是检测程序中设计的逻辑缺陷，寻找硬件连接上的错误并修正，提高工作性能。调试环节的思路如下：

• 连接好硬件系统，供电，开始运行程序，让两个控制器都保持待命状态；

• C51 进入主界面，按下按键“C”，先测试密码门禁功能。输入密码 888888，成功开启门禁，门禁开启并保持约 5 秒，随后自动关闭并返回主界面。接下来重新进入密码门禁功能，测试修改密码功能，修改的密码为 111111。修改密码后测试一遍新的密码。

• 按下按键“B”，测试人脸识别门禁功能（在测试前系统中预先有了图片集和识别库）。测试人脸识别门禁的效果，已经门禁开门与延迟的效果。最后按下中止键，C51 返回主界面，树莓派也保持待机。

• 按下按键“A”，测试人脸图像建库操作。系统默认的新人脸 id 为 0，名称为 NewGuest。系统读取完人脸数据之后自动训练模型，等到训练结束 C51 返回主界面，树莓派保持待机。建立新人脸数据后按下“B”键，测试该新人脸的开门效果。

• 最后返回主界面，关闭电源结束系统。

6.3 调试结果展示

将设计各模块按照前面的规则接线，并上电，其实物如图 38 所示

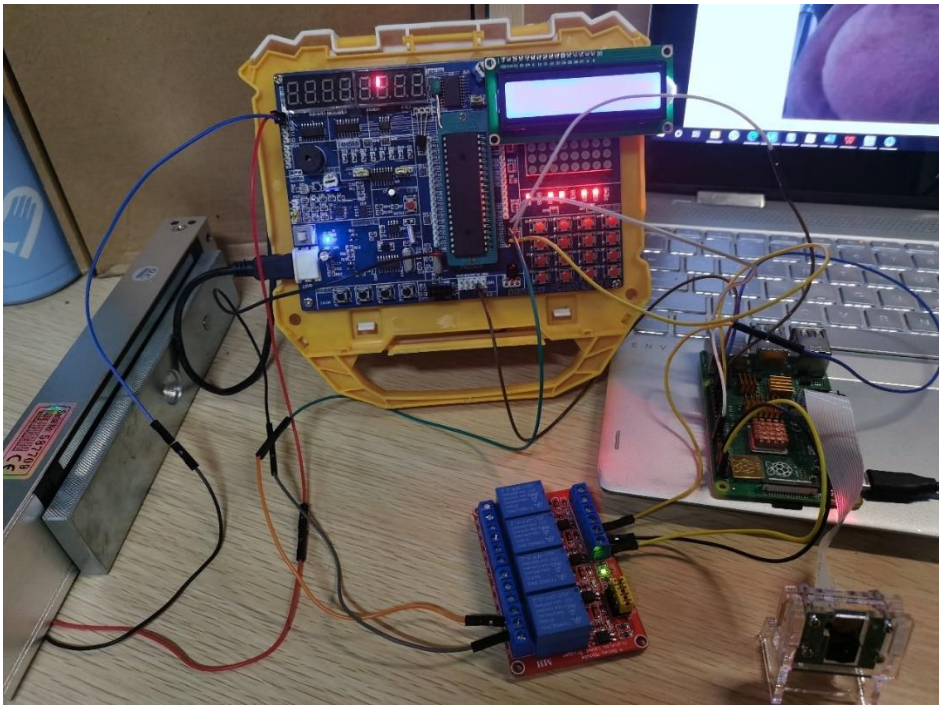


图 38 人脸识别门禁系统实物连接

调试过程中以图 39 为例进行建库与识别测试，其识别结果为图 40，设定名字为“Liu Chenuu”，密码测试与主界面 LCD 显示的调试图为图 41 至图 44，电磁门禁的开启状态与关闭状态如图 44 和图 45。

图 41 主系统界面/ 图 42 人脸识别成功界面

图 43 密码输入界面/ 图 44 密码修改成功界面

图 45 电磁门禁处于关门状态

图 46 电磁门禁处于开门状态

7. 问题讨论

本章节是问题讨论，主要内容是对前面设计方案中细节问题的补充，其次是对一些在设计与调试过程中遇到的错误的修正，以及系统的一些优化方案。经过一段时间的设计开发与调试工作，结合我设计中的遇到的各种问题，我们总结了其中最关键和典型的 7 个问题在本章节进行讨论。

7.1 函数库下载与导入

首先是在设计之初就遇见的函数库下载失败的问题。在电脑 PC 机中，我们的开发环境是 PyCharm，其用到的函数包可以直接从 setting 设置选项中直接下载，但是由于其下载源也在国外，所以同样会面临下载失败的问题。解决这一步问题我们可以采用同第四章环境配置中一样的换源方法，其需要在 python3 环境下使用 pip3 组件更换软件源和系统源。

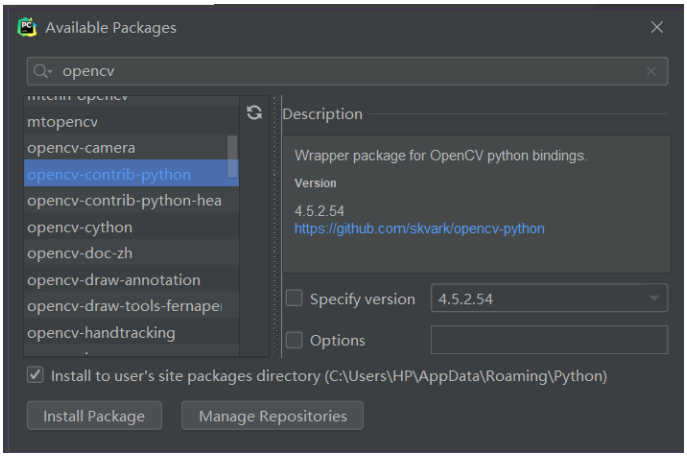


图 47 PyCharm 通过 setting 安装函数包

在电脑上如果所有在线安装方法均失效的情况下，还可以使用本地安装，即需要从 opencv 官网或者其他镜像源网站中下载其函数包相关版本的函数包，比如针对我的 win10x64 系统 python3.7 环境可以使用 opencv_python-3.4.2-cp35-cp35m-win_amd64.whl 的安装文件。下载到本地后从 cmd 中先找到这个文件所在目录，接下来使用 pip 安装即可：

```
pip install opencv_python-3.4.2-cp35-cp35m-win_amd64.whl
```

如果安装失败则使用管理员权限即可。

```
pip install --user opencv_python-3.4.2-cp35-cp35m-win_amd64.whl
```

图 48　opencv 镜像源网站与各安装版本

在树莓派上，当如果也遇到换源失败，或者受制于网络原因下载依然失败的话，也可以考虑通过离线下载 whl 文件并本地安装。具体方法与在 PC 端的本地下载方法类似，从 opencv-contrib-python 官网或者镜像源网站中下载相关版本的 whl 文件：opencv_python-3.4.3.18-cp37-cp37m-linux_armv7l.whl，将文件从 VNC 中传到树莓派上，然后在命令行中输入：**sudo pip3 install opencv_python-3.4.3.18-cp37-cp37m-linux_armv7l.whl**，当提示出如下信息，则说明 OpenCV 安装成功。

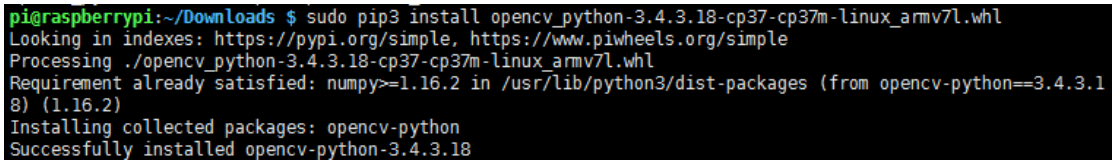


图 49　本地安装 opencv-contrib-python

最后安装五个 opencv-contrib-python 的依赖库，安装方法就是直接在命令行中使用 apt-get 直接安装即可。这几个库不需换源也能直接下载，当下载完成后在命令行中输入 python3 进入 python 环境，输入 import cv2 如果没有错误报警即代表安装成功。

- **sudo apt-get install libatlas-base-dev**
- **sudo apt-get install libjasper-dev**
- **sudo apt-get install libqtgui4**
- **sudo apt-get install libqt4-test**
- **sudo apt-get install libhdf5-dev**

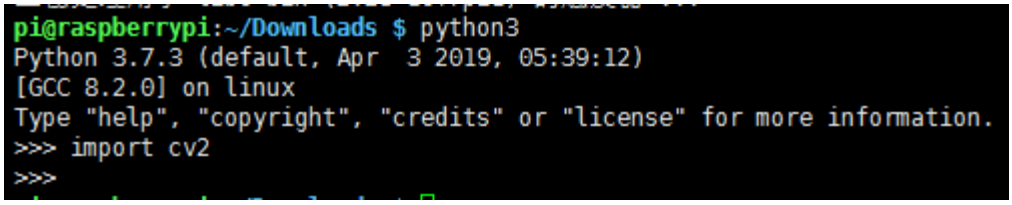


图 50 opencv-contrib-python 安装成功验证

7.2 C51 连接与烧录

由于 C51 单片机本身不具备程序下载功能，所以开发板是通过 CH340 芯片进行电平和信号转换，执行程序下载和烧录工作。下载时仅需要用 USB 线将 C51 和电脑向连接即可，然后电脑会安装相关的驱动软件，如果一切正常的话那就会在烧录软件上读取到已连接的串口信息，将 C51 工程生成的 hex 文件通过烧录软件下载到开发板中即可。

但是，C51 连接与烧录工作往往是在 C51 设计开发的初期阶段最容易出现问题的。在这一个环节中有可能出现故障的地方有，驱动安装失败、串口连接失败、Keil 环境编译失败等。首先是驱动安装失败，可以桌面右键我的电脑，在属性中点击设备管理器，当有串口正常联机时，可以在串口（COM 和 TPL）中看到相关信息。

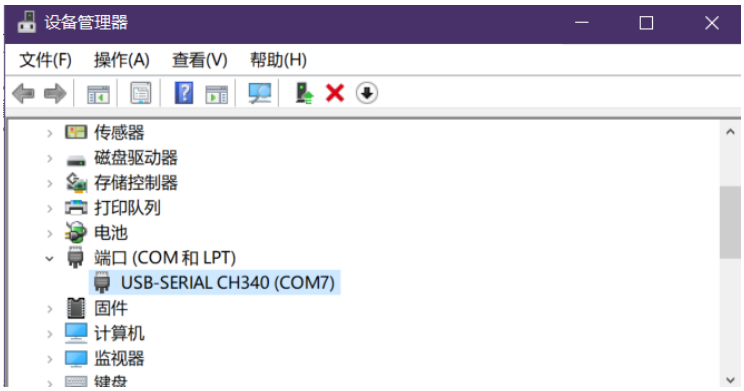


图 51 串口识别结果

如果按照第四章中的操作没有成功安装 CH340 驱动时，可以考虑更换兼容驱动安装，这一步就是针对不同的 PC 机操作系统选择对应不同的安装程序。一般来说 win10 系统可以兼容大多数的 CH340 驱动。当然在开发板的下载电路与最小系统中有一个环节有硬件上的问题，也会造成无法识别的问题，所以要对硬件连线进行检查，如果开发板的 power 指示灯没有点亮那么就是硬件损坏，需要跟换相关器件。

在 Keil 的程序编译过程中如果遇到报警提示 Failed to output Taggart，不会生成 hex 文件，那么就是该 Keil 软件没有注册，编译文件大小受限。这时仅需要对 Keil 重新注册即可。用到的软件是 Keil Generic Keygen，将 Keil 的序列号导入该软件中即可以生成匹配的注册码。注册好软件以后重启，即可以正常编译并导出 hex 文件。

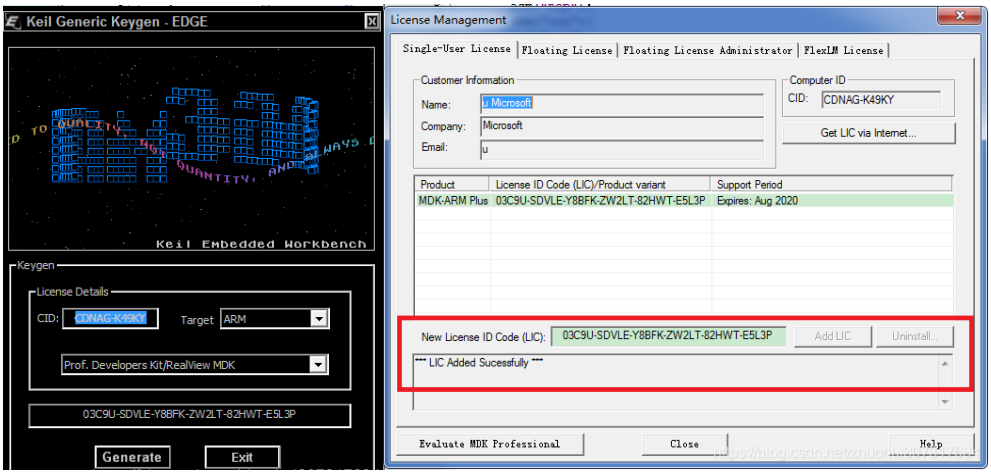


图 52 Keil 注册软件与注册页面

7.3 程序路径设置

由于两个控制器的程序和库函数都很多，且人脸识别部分还需要调用很多的图片集与模型库，所以说程序及其相关文件的路径是很重要的。在我们开发的前期就出现了路径错误问题。在主控制器 C51 一侧，各种库函数及其头文件统一放在 finnall/程序 文件夹下。并在 Keil 工程文件中将库函数导入到工程目录下。

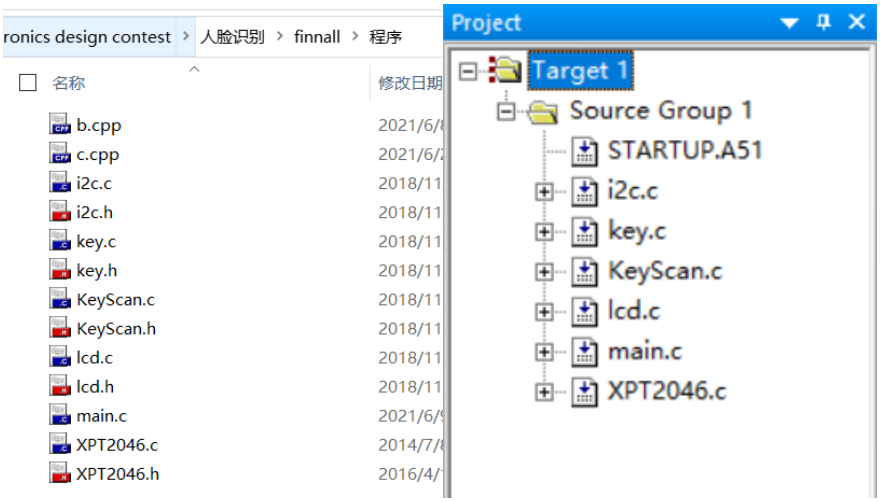


图 53 C51 程序路径配置

在树莓派一侧，需要对路径进行配置的是图片集、haar 分类器模型、训练集等相关文件。树莓派的所有文件都放到 FacialRecognition 文件夹下，图片集储存到 FacialRecognition/dataset 文件夹下，训练后得到的识别模型放到 FacialRecognition/trainer 文件夹下，haar 分类器模型直接放到 FacialRecognition 中。并且在主程序 finnal_main.py 中设置相关调用路径：

```
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  #配置人脸识别模型库地址
```

```
    cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".
jpg", gray[y:y+h,x:x+w])  #配置人脸照片保持路径
    path = 'dataset'     #配置待训练照片集地址
    recognizer.write('trainer/trainer.yml')     #配置人脸识别模型保存地址
    recognizer.read('trainer/trainer.yml')     #配置人脸识别模型读取地址
    cascadePath = "haarcascade_frontalface_default.xml"#配置人脸识别模型
库地址
```

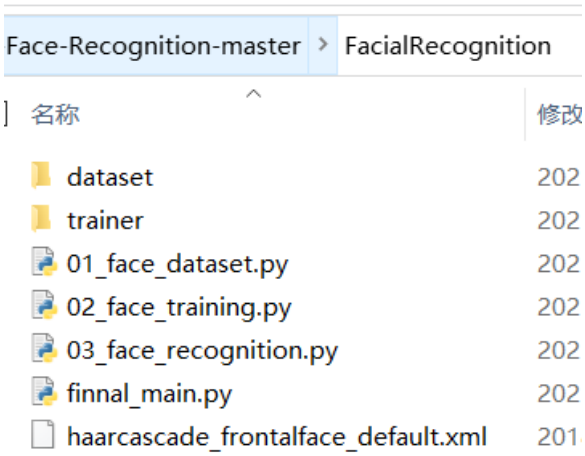


图 54 树莓派程序路径配置

7.4 人脸识别冲突检测

在建立识别库中最关键的一个就是保证原始图片集的准确性，如果原始图片集中出现了人与照片不匹配的情况，那么很有可能会造成后续人脸识别检测的结果混乱。由此来说我们要保证同一个编号下的人脸照片为同一人而且准确。这样我们就通过代码实现了一个人脸识别冲突检测的设计。

由于在图片采集过程中，faces 元胞数组会同时记录下同一时刻采集到的全部人脸的坐标信息，并依次储存，有多少个人脸，元胞数组的长度就是多少。也就是说我们可以在每一轮采集的过程中检测 faces 这个数组的长度，如果长度大于 1，那么就会发出报警信息，并修改计数变量 count=0，即从头开始拍照计数。直到 faces 的长度为 1 时，才开始正常的采集工作，直到 count 到 100。

```
    if len(faces)<=1:
        for (x,y,w,h) in faces:
            cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
            count += 1
            cv2.putText(img,str(count),(40,50),cv2.FONT_HERSHEY_PLAIN,
1.5,(0,255,0),1)
            cv2.imwrite("dataset/User." + str(face_id) + '.' + str(coun
```

```
t) + ".jpg", gray[y:y+h,x:x+w])
    elif len(faces)>1:
       cv2.putText(img,"Please do not stand in front of the camera",(4
0,50),cv2.FONT_HERSHEY_PLAIN,1,(0,0,255),1)
       cv2.putText(img,"with more than one person at a time",(40,70),c
v2.FONT_HERSHEY_PLAIN,1,(0,0,255),1)
       count=0
```

7.5 C51 读取低电平

单独讲述这一部分的原因是因为在设计之初我们根本就没有考虑过 c51 无法直接读取低电平。由于 C51 不同于 STM32、MSP430、K210、Raspberry Pi 等拥有专门的 GPIO 配置寄存器，也就是说我们无法通过软件程序定义一个 C51 引脚的输入输出模式，以及配置上拉下拉电阻。而在一般的 TTL 和 CMOS 微处理器中，未定义的引脚悬空往往是高电平，实测 C51 的悬空引脚为 3.3V 左右。也就是说，除非定义一个管脚是低电平输出，否则所有的 C51 管脚均为高电平。

这就出现了一个问题。conn3 作为开门信号输入引脚，负责接收来自树莓派的高低电平。实测得到树莓派的 GPIO（不含 5V，3.3V，GND 引脚）高电平为 3.3V 左右，低电平在 0.1V 左右。如果将 conn3 和 GPIO26 直接用杜邦线连接的话，那么由于 conn3 默认为高电平，即使在 GPIO26 处输出低电平，在 conn3 处依然为 2.5V 左右的电压，C51 依然认为其为高电平。这就导致无论树莓派一侧怎样做电平处理，C51 一侧完全不会接收到电平变化。

这里我们尝试过三种方案：

1. **采用基于 12 位 AD 芯片 XPT2046 的 ADC 模块**，将 GPIO26 的电压模拟值转换为数字量，通过程序判断该数字量是否低于一个阈值（如 2000，即 2.000V），若低于阈值则认为 GPIO26 给出低电平，C51 以收到低电平的操作去进行后续处理。但是尝试后放弃了该方案，因为 C51 的引脚资源有限，并且调试 AD 的开发工程量较大。

2. **采用一个硬件分压电路**，首先应该指出的是，这是一个错误的思路。我们设计这个电路的初衷是想提高硬件分压，将高电平通过一组并联电阻网络，降压为低电平再输入到 C51 中，以实现低电平的输入。但是这种方案的错误之处在于，只要是在 C51 正常工作电压范围之内（0~5V），没有特殊定义为低电平输出的引脚都会被视为高电平，所以该方案被放弃了。

3. **采用继电器控制 conn3 接地**。该方案是最后被采用的方案。其采用的是一路电磁继电器，控制端为 GPIO26，被控端为 GND 与 conn3。其实现的原理是，

当 GPIO26 处给到高电平，即人脸识别成功发出开门信号，则控制继电器电磁线圈吸合，使 conn3 接地，这样该引脚才能读到低电平。

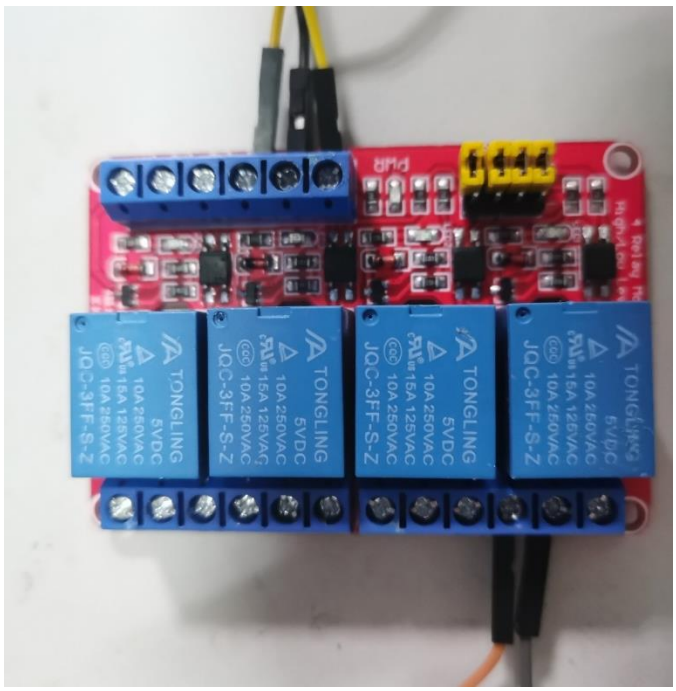


图 55 继电器连线

7.6 双处理器通信

由于本设计中用到了两个处理器，所以必然会有两个处理器的通信问题。那么在定义了两个处理器的输入输出引脚之后，只需要将对应的引脚通过杜邦线相连即可。但是为了让整个系统工作使减少电平的不匹配问题，所以还需要将两个处理器的 GND 端相连，并且两个处理器的供电也要来自同一个电源。两个处理器的引脚连接方式如第六章表 7。

7.7 外部中断、中止信号

C51 单片机外部中断有 2 个，分别是外部中断 0 和外部中断 1。中断指的是当中央处理机 CPU 正在处理某件事的时候外界发生了紧急事件请求，要求 CPU 暂停当前的工作，转而去处理这个紧急事件，处理完以后，再回到原来被中断的地方，继续原来的工作的过程。本设计用到中断的原因是在譬如人脸识别过程中按下按键引发外部中断，可以中止当前的功能，返回主界面，即“重启”的功能。

设置外部中断的相关函数为：设置外部中断 0 函数 Int0Init()、外部中断 0 的中断服务函数 Int0() interrupt 0。Int0Init()的功能是开启一些中断使能开关，设置外部中断 0；Int0() interrupt 0 的功能是当外部中断 0 被触发时

CPU 需要执行的功能。其函数为：

```
void Int0Init()
{   IT0=1;      //跳变沿出发方式（下降沿）
    EX0=1;      //打开 INT0 的中断允许
    EA=1;       //打开总中断
}
void Int0()   interrupt 0
{   delay(1000);
    if(key_num==0x0f)
    {   AS=1;
        Step=0;
}}
```

另外在树莓派中也有中止服务函数，其逻辑是 if-else 判断，并非中断。中止功能是否触发可以通过键盘 ESC 和 GPIO19 电平来判断。用 cv2.waitKey()其函数判断键盘状态，当收到 ESC 信号或者收到来自 C51 的中止信号，则关闭所有窗口，退出相关程序。相关代码如下。

```
k = cv2.waitKey(10) & 0xff    # 按下‘ESC’退出视频
if k == 27:
    GPIO.output(26,0)
    break
if (GPIO.input(19)==1):
    GPIO.output(26,0)
    cv2.destroyAllWindows()
    break
```

参考文献

[1] 陈佳林，智能硬件与机器视觉，机械工业出版社，2019

[2] 杨玉龙，人脸识别门禁系统的设计与实现，重庆大学计算机学院，2014 年 4 月

[3] 杨峰、唐华、毛昀、袁勇、刘文，基于 OpenCV 的人脸识别设计与实现，信息与电脑，新疆工程学院，新疆乌鲁木齐，2020 年 16 期

[4] 李杨、朱喆、金华，基于树莓派的学生寝室门禁系统设计，信息与电脑，延边大学工学院，吉林延吉，2020 年 11 期

[5] 兰鸽、刘文平，基于树莓派的实验室人脸识别门禁系统设计，信息记录材料，新疆工程学院，新疆乌鲁木齐，2021 年 2 月

[6] 高素文，基于 OpenCV 的实时人脸识别系统研究与实现，华北电力大控制与计算机工程学院，2015 年 3 月

[7] 深圳普中科技有限公司，普中 51 单片机开发攻略-A6，http://www.prechin.cn

[8] 深圳普中科技有限公司，普中 51-单核-A3&A4 开发板原理图，http://www.prechin.cn

[9] 张钰敏，叶传奇，张淑媛，王展，田雨薇，秦佳帅，树莓派人脸识别系统的开发与研究，电脑知识与技术，河南科技大学信息学院/软件学院

[10] https://www.hackster.io/mjrobot/real-time-face-recognition-an-end-to-end-project-a10826

[11] https://thecodacus.com/

[12] https://github.com/thecodacus/Face-Recognition

附录：实验程序及 C51 库函数

C51_main.c

main.c

```
#include "reg52.h"                    //此文件中定义了单片机的一些特殊功能寄存器
#include "lcd.h"
#include "key.h"
#include "i2c.h"

#define u16 unsigned int      //对数据类型进行声明定义
#define u8 unsigned char

sbit AS=P3^1;                 //继电器   *******

sbit conn1=P2^1 ;             //****
sbit conn2=P2^4 ;             //****
sbit conn3=P2^3 ;
sbit conn4=P2^2;              //****

u8 pw_num,Error_Num,PassWord_Length=6;
u8 PASSWORD[]={8,8,8,8,8,8,0,0,0,0};
u8 INPUT_PW_Tab[10];
u8 key_num,Step,Step5,Load_first_flag=0;

bit result_flag,Input_suc_flag;
bit List1=0;

void Step_0();
void Step_1();
void Step_2();
void Step_3();
void Step_4();
void Step_5();
        void Step5_0();
        void Step5_1();
        void Step5_2();
        void Step5_3();
        void Step5_4();
        void Step5_5();
void Step_6();
void Step_7();                        //****
void Step_8();
void Step_9();
void Step_10();                       //****
void Int0Init();                      //****

void CipherComparison();
void input_password(bit m);
void Read_Password();

void main()
{
    u8 data1,a;
    LcdWriteCom(0x01);  //清屏
    for(data1=0;data1<PassWord_Length+2;data1++)
    {
        a=At24c02Read(data1)+0x30;
        LcdWriteData(a);
        delay(1000);
    }
    delay(1000);
    LcdInit();
    delay(1000);
    Step=0;
    Step5=0;
    Error_Num=0x00;
    Read_Password();

    Int0Init();  //    设置外部中断 0 ****
    conn2=0;                          //****
    conn1=0;                          //****
    conn4=0;
    while(1)
    {   AS=1;
        key_num=KeyDown();
    //读取输入值
        switch(Step)
        {
            case 0:{Step_0();break;}
            case 1:{Step_1();break;}
            case 2:{Step_2();break;}
            case 3:{Step_3();break;}
            case 4:{Step_4();break;}
            case 5:{Step_5();break;}
            case 6:{Step_6();break;}
            case 7:{Step_7();break;} //按下黄键
            case 8:{Step_8();break;} //按下蓝键
            case 9:{Step_9();break;} //按下蓝键
            case 10:{Step_10();break;}
        }
    }
}

void Int0Init()
{
    //设置 INT0
    IT0=1;//跳变沿出发方式（下降沿）
    EX0=1;//打开 INT0 的中断允许。
    EA=1;//打开总中断
}

void Int0()interrupt 0
{
    delay(1000);        //延时消抖
    if(key_num==0x0f)
    {
        AS=1;
        Step=0;
    }
}

void Step_0()
{
```

```
            LcdInit();
            ShowString(0x00,"face-recognition");  //*
***
           ShowString(0x10,"       WELCOME!
  ");
            while(KeyDown()==0xff)Step=9;

}

void  Step_9()
{
        LcdInit();
       ShowString(0x00,"  A:New  B:facial  ");  //****

          ShowString(0x10,"  C;Key  D:close  ");
          while(KeyDown()==0xff)Step=10;
}

void  Step_10()
{

       if(key_num==0x0c)  {Step=7;}
       else    if(key_num==0x0d)  {Step=8;}
       else  if(key_num==0x0e)  {Step=1;}
       else  if(key_num==0x0f)  {AS=1;Step=0;}
       else  Step=10;
}

void  Step_1()
{
            LcdWriteCom(0x01);
          ShowString(0x00,"Unlock");
            ShowString(0x0f,"<");
          ShowString(0x10,"Change  Password");
            ShowString(0x1f,"  ");

          Step=2;
}

void  Step_2()
{
       if(key_num!=0x0b)
       {
            if((key_num==0x01)  ||(  key_num==0x0
9))
            {
                  List1=~List1;
                  if(List1==0)
                  {

                        ShowString(0x0f,"<");
                        ShowString(0x1f,"  ");
                  }
                  else
                  {
                        ShowString(0x0f,"  ");
                        ShowString(0x1f,"<");
                  }
            }
       }
       else                    //确认键按下
       {
            if(List1==0){Step=3;}
            else  {Step=5;List1=0;}
       }
}
```

```
void  Step_3()                              //
{
       Step=4;
       pw_num=0;
       LcdInit();
       ShowString(0x00,"Pass  Word:    ");
}

void  Step_4()

{
            input_password(0);              //输入密码并
以*显示
            if(Input_suc_flag==1){Step=6;}          //密
码输入完成进入下一步
          Input_suc_flag=0;                     //清除密码
输入完成标志
}

void  Step_5()                              //修改密码
{
       switch(Step5)
       {
            case  0:  {Step5_0();}    break;
            case  1:  {Step5_1();}    break;
            case  2:  {Step5_2();}    break;
            case  3:  {Step5_3();}    break;
            case  4:  {Step5_4();}    break;
            case  5:     {Step5_5();}       break;
       }
}

void  Step_6()
{
       u8  i=0;
       CipherComparison();          //密码比对
       if(result_flag==1)         //密码正确
       {
            LcdInit();
            ShowString(0x00,"       WELCOME!");
            AS=0;                   //开继电器
            delay(200000);
            for(i=0;i<60;i++)    //循环 5S
            {
                  AS=0;                       //开启电机
                  delay(5000);       //大约延时 50ms
            }
            AS=1;                   //关继电器
       }
       else              //密码错误
       {
            LcdInit();
            ShowString(0x00,"Error  01!");
       }
       Step=0;
}

void  Step5_0()
{
       LcdWriteCom(0x01);    //清屏
       ShowString  (0x00,"Input  PassWord:");
       Step5=1;
       pw_num=0;
}
void  Step5_1()
```

```
{
    input_password(0);      //    输入密码
    if(Input_suc_flag==1)    //密码输入完成
    {
        Step5=2;                    //
        Input_suc_flag=0;   //清除输入完成标志
    }
}
void Step5_2()                  //
{
    CipherComparison();   //密码比对
    Step5=3;
}
void Step5_3()                  //
{
    if(result_flag==0)          //      密码错误
    {
        if(Error_Num<3)      //输出错误次数小于 3
        {
            Error_Num++;
            LcdInit();
            ShowString (0x00,"Error 01");
            delay(20000);
            Step5=0;
        }
        else            //密码错误次数大于 3
        {
            Error_Num=0;
            Step=0;
        }
    }
    else                        //密码正确
    {
        LcdInit();
        ShowString (0x00,"New PassWord:");
        pw_num=0;
        Step5=4;
    }
}
void Step5_4()
{
    input_password(1);    //输入密码并显示
    if(Input_suc_flag==1)           //输入完成
    {
        Step5=5;
        Input_suc_flag=0;
        LcdWriteCom(0x01);   //清屏
        ShowString (0x00,"          OK!");
    }
}
void Step5_5()
{
    unsigned char j;
    PassWord_Length=pw_num;   //读取密码长度
    At24c02Write(0,Load_first_flag);
    delay(100);
    At24c02Write(1,PassWord_Length);  //保存长度
    delay(100);
    for(j=0;j<PassWord_Length;j++)
    {
        PASSWORD[j]=INPUT_PW_Tab[j];   //读取密码
        At24c02Write(j+2,INPUT_PW_Tab[j]);
```

```
        //保存密码至 EEPROM
        delay(100);
    }
    Step5=0;
    Step=0;
}


void Read_Password()
{
    unsigned char j;
    Load_first_flag=At24c02Read(0x00);
    {
        Load_first_flag=1;
        At24c02Write(0,0x01);
        delay(100);
        At24c02Write(1,0x06);                    //写默认密码长度 6 至 EEPROM
        delay(100);
        for(j=0;j<PassWord_Length;j++)
        {
            At24c02Write(j+2,8);   //写默认密码 888888 至 EEPROM
            PASSWORD[j]=INPUT_PW_Tab[j];       //读密码
            delay(100);
        }
    }
    Load_first_flag=At24c02Read(0x00);
    PassWord_Length=At24c02Read(0x01);   //读取密码长度
    for(j=0;j<PassWord_Length;j++)   //读取密码
    {
        PASSWORD[j]=At24c02Read(j+2);
    }
}

void input_password(bit m)
{
    unsigned char j;
    if(key_num!=0x0b)           //ok 键没有按下
    {
        if(key_num<0x0a)              //1-9 按下
        {
            INPUT_PW_Tab[pw_num]=key_num;   //保存至输入密码数组
            pw_num=pw_num+1;  //密码长度+1
            LcdWriteCom(0xc0);
            for(j=0;j<pw_num;j++)
            {
                if(m==0)  {LcdWriteData('*');
    }     //密码隐藏
                else  {LcdWriteData(INPUT_PW_Tab[j]+0x30);}     //显示密码
            }
        }
        if(key_num==0x0a)        //返回键按下
        {
            if(pw_num!=0)  {pw_num=pw_num-1;}
            else                {Step=0;}
            LcdWriteCom(0xc0);
            for(j=0;j<pw_num;j++)
            {
```

```
                        if(m==0) {LcdWriteData('*');
        }          //密码隐藏
                        else {LcdWriteData(INPUT_P
W_Tab[j]+0x30);}       //显示密码                        }
                    LcdWriteData(' ');
                }
        }
        else              //ok 键按下
        {
                if(pw_num==0)
                {
                        Step=0;
                        LcdWriteCom(0x01);
                        ShowString (0x00,"Error 02!");
                        delay(10000);
                }
                else
                {
                        Input_suc_flag=1;
                }
        }
}

void CipherComparison()
{
        u8 i,j=0;
        if(PassWord_Length==pw_num) //长度比对
        {
                for(i=0;i<PassWord_Length;i++)   //密码
比对
                {
                        if(PASSWORD[i]!=INPUT_PW_Ta
b[i])
                        {
                                result_flag=0;break;     //密码
错误
                        }
                        else
                        {
                                result_flag=1;     //密码正确
                        }
                        INPUT_PW_Tab[i]=0XFF;
        //清除密码缓存数组
                }
        }
        else {result_flag=0;}
}

void Step_7()
{
        u8 i=0;
        LcdWriteCom(0x01);   //清屏
        conn1=1;
        conn2=0;
        LcdInit();
        ShowString (0x00,"Recognitng...");         //160
2 显示：正在识别
        while(1)
        {
                key_num=KeyDown();
                if(conn3==0)
                {
                        Step=0;
                        ShowString (0x10,"complete!");   //
1602 显示：识别成功
                        delay(10000);
                        conn1=0;
                        conn2=0;
                        Step=0;
                        break;
                }
                else if(key_num==0x0b)
                {
                        ShowString(0x10,"       Return!");
                        AS=1;
                        conn4=1;
                        for(i=0;i<60;i++)   {delay(5000); //
延时 50ms}
                        conn1=0;
                        conn2=0;
                        Step=0;
                        conn4=0;
                        break;
                }
        }
}

void Step_8()
{
        u8 i=0;
        LcdWriteCom(0x01);   //清屏
        ShowString (0x00,"look the camera");        //
1602 显示：正在识别
        conn1=0;
        conn2=1;
        while(1)
        {
                key_num=KeyDown();
                if(conn3==0)              //密码正确
                {
                        ShowString(0x10,"       WELCOM
E!");
                        AS=0;                  //开继电器
                        for(i=0;i<100;i++)         // 5S
                        {
                                AS=0;       //开启电机
                                delay(5000);      //延时 50ms
                        }
                        AS=1;
                        delay(60000);
                }
                else if(key_num==0x0b)
                {
                        ShowString(0x10,"       Return!");
                        AS=1;
                        conn4=1;
                        conn1=0;
                        conn2=0;
                        for(i=0;i<60;i++) {delay(5000);   //
大约延时 50ms}
                        Step=0;
                        conn4=0;
                        break;
                }
        }
        Step=0;
```

Lcd.c

```
#include  "lcd.h"
```

```
uchar i;

void Lcd1602_Delay1ms(uint c)    //误差 0us
{
    uchar a,b;
    for (; c>0; c--)
    {
        for (b=199;b>0;b--)
        {
            for(a=1;a>0;a--);
        }
    }

}

#ifndef    LCD1602_4PINS        //当没有定义这个 LCD1602_4PINS 时
void LcdWriteCom(uchar com)     //写入命令
{
    LCD1602_E = 0;      //使能
    LCD1602_RS = 0;     //选择发送命令
    LCD1602_RW = 0;     //选择写入

    LCD1602_DATAPINS = com;     //放入命令
    Lcd1602_Delay1ms(1);        //等待数据稳定

    LCD1602_E = 1;              //写入时序
    Lcd1602_Delay1ms(5);        //保持时间
    LCD1602_E = 0;
}
#else
void LcdWriteCom(uchar com)     //写入命令
{
    LCD1602_E = 0;      //使能清零
    LCD1602_RS = 0;     //选择写入命令
    LCD1602_RW = 0;     //选择写入

    LCD1602_DATAPINS = com;   //由于 4 位的接线是接到 P0 口的高四位，所以传送高四位不用改
    Lcd1602_Delay1ms(1);

    LCD1602_E = 1;      //写入时序
    Lcd1602_Delay1ms(5);
    LCD1602_E = 0;

    LCD1602_DATAPINS = com << 4; //发送低四位
    Lcd1602_Delay1ms(1);

    LCD1602_E = 1;      //写入时序
    Lcd1602_Delay1ms(5);
    LCD1602_E = 0;
}
#endif

#ifndef    LCD1602_4PINS
void LcdWriteData(uchar dat)            //写入数据
{
    LCD1602_E = 0;      //使能清零
    LCD1602_RS = 1;     //选择输入数据
    LCD1602_RW = 0;     //选择写入

    LCD1602_DATAPINS = dat; //写入数据
    Lcd1602_Delay1ms(1);

    LCD1602_E = 1;      //写入时序
    Lcd1602_Delay1ms(5);    //保持时间
    LCD1602_E = 0;
}
#else
void LcdWriteData(uchar dat)            //写入数据
{
    LCD1602_E = 0;      //使能清零
    LCD1602_RS = 1;     //选择写入数据
    LCD1602_RW = 0;     //选择写入

    LCD1602_DATAPINS = dat;     //由于 4 位的接线是接到 P0 口的高四位，所以传送高四位不用改
    Lcd1602_Delay1ms(1);

    LCD1602_E = 1;      //写入时序
    Lcd1602_Delay1ms(5);
    LCD1602_E = 0;

    LCD1602_DATAPINS = dat << 4; //写入低四位
    Lcd1602_Delay1ms(1);

    LCD1602_E = 1;      //写入时序
    Lcd1602_Delay1ms(5);
    LCD1602_E = 0;
}
#endif

 void ShowString (unsigned char Coordinate,char *ptr)
 {

        if(Coordinate<=0x0f)    //高四位为 0，显示第一行
                {
                    LcdWriteCom((Coordinate&0x0f)+0x80);
                    while(*ptr!='\0')
                    {
                        LcdWriteData(*ptr);
                        ptr++;
                    }
                }
            else                        //高四位为 1，显示第 2 行
                {
                    LcdWriteCom((Coordinate&0x0f)+0xc0);

                    while(*ptr!='\0')
                    {
                        LcdWriteData(*ptr);
                        ptr++;
                    }
                }
```

```
  }


#ifndef          LCD1602_4PINS
void  LcdInit()                                    //LC
D 初始化子程序
{
      LcdWriteCom(0x38);    //开显示
      LcdWriteCom(0x0c);    //开显示不显示光标
      LcdWriteCom(0x06);    //写一个指针加 1
      LcdWriteCom(0x01);    //清屏
      LcdWriteCom(0x80);    //设置数据指针起点
}
#else
void  LcdInit()                                    //LC
D 初始化子程序
{
      LcdWriteCom(0x32);    //将 8 位总线转为 4 位
总线
      LcdWriteCom(0x28);    //在四位线下的初始化
      LcdWriteCom(0x0c);    //开显示不显示光标
      LcdWriteCom(0x06);    //写一个指针加 1
      LcdWriteCom(0x01);    //清屏
      LcdWriteCom(0x80);    //设置数据指针起点
}
#endif
```

Key.c

```
#include  "key.h"

u8  KeyValue=0;

void  delay(u16  i)
{
      while(i--);
}

u8  KeyDown(void)
{
      char  a=0;
      GPIO_KEY=0x0f;

      if(GPIO_KEY!=0x0f)//有按键按下
      {
            delay(1000);//延时 10ms 去抖
            if(GPIO_KEY!=0x0f)//有按键按下
            {
                  GPIO_KEY=0X0F;
                  switch(GPIO_KEY)
                  {
                        case(0X07):       KeyValue=0;
break;
                        case(0X0b):       KeyValue=1;
break;
                        case(0X0d):  KeyValue=2;brea
k;
                        case(0X0e):       KeyValue=3;
break;
                  }
                  GPIO_KEY=0XF0;
                  switch(GPIO_KEY)
                  {
                        case(0X70):       KeyValue=K
eyValue;break;
                        case(0Xb0):       KeyValue=K
eyValue+4;break;
                        case(0Xd0):  KeyValue=KeyV
alue+8;break;
                        case(0Xe0):       KeyValue=K
eyValue+12;break;
                  }
      while((a<50)&&(GPIO_KEY!=0xf0))
      {
            delay(1000);
            a++;
      }
      }
}
      else
      {
            KeyValue=0xff;    //无按键按下
      }
      return  KeyValue;    //返回 KeyValue
}
```

I2c.c

```
#include"i2c.h"

void  Delay10us()
{
      unsigned  char  a,b;
      for(b=1;b>0;b--)
            for(a=2;a>0;a--);

}

void  I2cStart()
{
      SDA=1;
      Delay10us();
      SCL=1;
      Delay10us();//建立时间是 SDA 保持时间>4.7u
s
      SDA=0;
      Delay10us();//保持时间是>4us
      SCL=0;
      Delay10us();
}

void  I2cStop()
{
      SDA=0;
      Delay10us();
      SCL=1;
      Delay10us();//建立时间大于 4.7us
      SDA=1;
      Delay10us();
}

unsigned  char  I2cSendByte(unsigned  char  dat)
{
      unsigned  char  a=0,b=0;//最大 255，一个机器
周期为 1us，最大延时 255us。
      for(a=0;a<8;a++)//要发送 8 位，从最高位开始
      {
            SDA=dat>>7;     //起始信号之后 SCL=
0，所以可以直接改变 SDA 信号
            dat=dat<<1;
            Delay10us();
            SCL=1;
            Delay10us();//建立时间>4.7us
```

```
            SCL=0;
            Delay10us();//时间大于 4us
        }
        SDA=1;
        Delay10us();
        SCL=1;
        while(SDA)//等待应答，也就是等待从设备把
SDA 拉低
        {
            b++;
            if(b>200)    //如果超过 2000us 没有应答
发送失败，或者为非应答，表示接收结束
            {
                SCL=0;
                Delay10us();
                return  0;
            }
        }
        SCL=0;
        Delay10us();
        return  1;
}

unsigned  char  I2cReadByte()
{
        unsigned  char  a=0,dat=0;
        SDA=1;                  //起始和发送一个字节
之后 SCL 都是 0
        Delay10us();
        for(a=0;a<8;a++)//接收 8 个字节
        {
            SCL=1;
            Delay10us();
            dat<<=1;
            dat|=SDA;
            Delay10us();
            SCL=0;
            Delay10us();
        }
        return  dat;
}

void  At24c02Write(unsigned  char  addr,unsigned  ch
ar  dat)
{
        I2cStart();
        I2cSendByte(0xa0);//发送写器件地址
        I2cSendByte(addr);//发送要写入内存地址
        I2cSendByte(dat);       //发送数据
        I2cStop();
}

unsigned  char  At24c02Read(unsigned  char  addr)
{
        unsigned  char  num;
        I2cStart();
        I2cSendByte(0xa0);  //发送写器件地址
        I2cSendByte(addr);  //发送要读取的地址
        I2cStart();
        I2cSendByte(0xa1);  //发送读器件地址
        num=I2cReadByte();  //读取数据
        I2cStop();
        return  num;
}
```

RaspberryPi.main

```
import cv2
import numpy as np
import os
import RPi.GPIO as GPIO
import time

#5 channal 1, 23 channal 2

GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(19,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(26, GPIO.OUT)
GPIO.output(26, 0)
ledStatus = 1
print("ready")
while True:
    if (GPIO.input(5)==1):
        print("dataset!")
        cam = cv2.VideoCapture(0)
        cam.set(3, 640) # 设置视频宽度
        cam.set(4, 480) # 设置视频高度
        face_detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        face_id = 0
        print("\n [INFO] Initializing face capture. Look the camera and wait ...")
        count = 0
        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_detector.detectMultiScale(gray, 1.3, 5)
            if len(faces)<=1:
                for (x,y,w,h) in faces:
                cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
                count += 1
            cv2.putText(img,str(count),(40,50),cv2.FONT_HERSHEY_PLAIN,1.5,(0,255,0),1)
            cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
            elif len(faces)>1:
                cv2.putText(img,"Please do not stand in front of the camera",(40,50),cv2.FONT_HER
SHEY_PLAIN,1,(0,0,255),1)
                cv2.putText(img,"with more than one person at a time",(40,70),cv2.FONT_HERSHEY
_PLAIN,1,(0,0,255),1)
                count=0
            out_win="output_style_full_screen"
            cv2.namedWindow(out_win,cv2.WINDOW_NORMAL)
            cv2.setWindowProperty(out_win,cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FULLSCR
EEN)
            cv2.imshow(out_win, img)
            k = cv2.waitKey(100) & 0xff # 按'ESC'退出视频
            if k == 27:
                break
            elif count >= 100: # 取 30 张脸样本，停止录像
                break
            elif (GPIO.input(19)==1):
                cv2.destroyAllWindows()
            break
        print("\n [INFO] Exiting Program and cleanup stuff")
        cam.release()
        cv2.destroyAllWindows()

        path = 'dataset'
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
        def getImagesAndLabels(path):
            imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
            aceSamples = []
```

```
                ids = []
                for imagePath in imagePaths:
                    PIL_img = Image.open(imagePath).convert('L')
                    img_numpy = np.array(PIL_img, 'uint8')
                    id = int(os.path.split(imagePath)[-1].split(".")[1])
                    faces = detector.detectMultiScale(img_numpy)
                    for (x, y, w, h) in faces:
                        faceSamples.append(img_numpy[y:y + h, x:x + w])
                        ids.append(id)
                    return faceSamples, ids
            print("\n [INFO] Training faces. It will take a few seconds. Wait ...")
            faces, ids = getImagesAndLabels(path)
            recognizer.train(faces, np.array(ids))
            recognizer.write('trainer/trainer.yml')  # recognizer.save() worked on Mac, but not on Pi
            print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
            GPIO.output(26,1)
            time.sleep(1)
            GPIO.output(26,0)

        if (GPIO.input(23)==1):
            print("Recgonition!")
            recognizer = cv2.face.LBPHFaceRecognizer_create()
            recognizer.read('trainer/trainer.yml')
            cascadePath = "haarcascade_frontalface_default.xml"
            faceCascade = cv2.CascadeClassifier(cascadePath);
            font = cv2.FONT_HERSHEY_SIMPLEX
            id = 0

            names = ['0 NewGuist', '1 Niu Zhanhua', '2 Lu Sichen', '3 Feng Chengbo', '4 Bi Yifei', '5 Qua
n Wenxu','6 Liu Chenyu','7 Liu Nengdi','8 Su Shijia','9 Li Wang','10 Li Laiyuan','11 Yang Kuan','12 Di W
eiran','13 Song Yuqing','14 Zhang Xuanhao','15 Zhang Ruitao','16 Chen Zihan','17 Chen Yang','18 Chen Ru
nzhou','19 Chen Linfan','20 Jin Mingjun','21 Yao Junhui','22 Gu Jiawen','23 Tang Kaixuan','24 Peng Hongy
u','25 Xia Fei']
            cam = cv2.VideoCapture(0)
            cam.set(3, 640) # 设置视频宽度
            cam.set(4, 480) # 设置视频高度
            minW = 0.1*cam.get(3)
            minH = 0.1*cam.get(4)

            while True:
                ret, img =cam.read()
                gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                faces = faceCascade.detectMultiScale(
                    gray,
                    scaleFactor = 1.2,
                    minNeighbors = 5,
                    minSize = (int(minW), int(minH)),
                )
                GPIO.output(26, 0)
                for(x,y,w,h) in faces:
                    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
                    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
                    if (confidence < 100):
                        id = names[id]
                        confidence = "  {0}%".format(round(100 - confidence))
                        GPIO.output(26, 1)
                        print("gooood!")
                    else:
                        id = "unknown"
                        confidence = "  {0}%".format(round(100 - confidence))
                        GPIO.output(26, 0)
                    cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
                    cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)
                out_win="output_style_full_screen"
                cv2.namedWindow(out_win,cv2.WINDOW_NORMAL)
                cv2.setWindowProperty(out_win,cv2.WND_PROP_FULLSCREEN,cv2.WINDOW_FULLSCR
EEN)
```

```
                cv2.imshow(out_win, img)
                k = cv2.waitKey(10) & 0xff # 按下‘ESC’退出视频
                if k == 27:
                        GPIO.output(26,0)
                        break
                if (GPIO.input(19)==1):
                        GPIO.output(26,0)
                        cv2.destroyAllWindows()
                        break
        print("\n [INFO] Exiting Program and cleanup stuff")
        cam.release()
        cv2.destroyAllWindows()
GPIO.cleanup()
```