肇庆学院

计算机科学院、软件学院、大数据学院

|  |
| --- |
|  |
| 嵌入式应用技术 |
| 实训报告 |

|  |
| --- |
| 题目： 实时监控  专业： ­­­­­­­­计算机科学与技术  班级： ­­­­­­­­19科技1班  学号：­­­­­­­­201924111149 2019241111150  姓名： 林鉴洲 曾澍铭 |
|  |

**目录**

[一．实训目的 1](#_Toc91506658)

[二．项目开发内容及项目组成员分工 1](#_Toc91506659)

[2.1开发内容 1](#_Toc91506660)

[2.2成员分工 1](#_Toc91506661)

[三．项目工作原理介绍和软硬件结构 2](#_Toc91506662)

[3.1系统结构 2](#_Toc91506663)

[3.1.1总体设计（流程图） 2](#_Toc91506664)

[3.1.2系统设计原理 2](#_Toc91506665)

[3.2软件结构 3](#_Toc91506666)

[3.2.1 Python-Opencv软件 3](#_Toc91506667)

[3.3.2树莓派开发语言——Python 3](#_Toc91506668)

[3.3硬件结构 3](#_Toc91506669)

[3.3.1树莓派 3](#_Toc91506670)

[3.3.2 树莓派3B+ raspberry pi3代b+CSI接口摄像头 4](#_Toc91506671)

[四．步骤 4](#_Toc91506672)

[4.1实时监控功能控制模块 5](#_Toc91506673)

[4.1.1摄像头的连接 5](#_Toc91506674)

[4.1.2摄像头实时监控 5](#_Toc91506675)

[4.1.3工作流程图 7](#_Toc91506676)

[4.1.4运行结果 9](#_Toc91506677)

[4.2代码 10](#_Toc91506678)

[4.2.1 PC端代码 10](#_Toc91506679)

[4.2.2树莓派端代码 18](#_Toc91506680)

[4.3视频流保存到本地功能模块 20](#_Toc91506681)

[4.3.1视频保存模块运行结果 20](#_Toc91506682)

[五．实训结果和工作总结 20](#_Toc91506683)

[5.1实训结果 20](#_Toc91506684)

[六．心得体会 21](#_Toc91506685)

[6.1收获 21](#_Toc91506686)

# 一．实训目的

1. 掌握Python语言的使用
2. 掌握树莓派上的Python库的安装和使用
3. 掌握嵌入式Linux系统开发过程和基于Linux应用程序开发
4. 实现实时监控方面的应用开发

# 二．项目开发内容及项目组成员分工

## 2.1开发内容

本次实训在嵌入式树莓派的基础上开发了一个PC端应用程序,主要硬件由树莓派和摄像头组成，摄像头负责图像采集，采集到的数据经由python socket通过局域网以jpg格式传输到PC端，在用户屏幕上实时显示监控录像。

该系统整体设计方案可以分为如下几个部分:实时监控、视频采集、数据传输、视频播放、视频录制、视频回放六个部分。因此整体设计过程中需要完成对系统环境的搭建，即硬件平台的开发，同时也需要在树莓派开发板上完成各种端口函数以及驱动程序等。

## 2.2成员分工

（1）林鉴洲（201924111149），主要负责这个项目中实时监控时，运用python-opencv对视频流进行截取，并保存到本地；以及实验报告中视频保存到本地功能模块、回放功能模块和实验结果的攥写。

（2）曾澍铭（201924111150），主要负责树莓派上摄像头驱动、opencv等硬件安装和配置，完成本项目的实时监控功能；查找资料和参考文献，绘制本项目所需各个功能模块流程图和系统结构流程图。

# 三．项目工作原理介绍和软硬件结构

## 3.1系统结构

### 3.1.1总体设计（流程图）

系统整体设计完成后，给系统上电，等系统运行状态正常后进行全自动运行测试，通过PC端在输入树莓派的ip地址和端口号与树莓派连接，实际上是在浏览树莓派上所提供的在线视频服务，看到的视频图像清晰，延时较小，监控系统性能良好。

图示

描述已自动生成

**图3-1-1系统工作流程图**

### 3.1.2系统设计原理

以树莓派3代B型 处理器为核心，结合RaspbianLinux 操作系统构建了视频监控系统的开发平台，并驱动摄像头采集视频图像，通过python socket建立连接，即可实现公网用户在线观看监控区域实时视频的需求。

用户通过PC端在输入树莓派的ip地址和端口号与树莓派连接，查看实时监控的视频或者图像，树莓派利用python socket传输相应的数据，发送给PC端，PC端接收到信息并通过python-opencv截取视频流，将视频转化为avi格式或者图像转化为jpg格式保存到本地。并可进行视频回放。

## 3.2软件结构

基于嵌入式系统软件设备主要由摄像头驱动、python -Opencv软件组成。

### 3.2.1 Python-Opencv软件

OpenCV-Python是OpenCV的python的API接口，它拥有OpenCV C++ API的功能，同时也拥有Python语言的特性。虽然Python语言运行起来比C++语言慢，但是Python语言具有简单、短小、学习快的特点，同时Python也很方便调用C++开发的组件，这样高性能要求的功能可以使用C++来实现。这样一来，我们就可以既运行像C++一样快的代码，又可以使用简单易用Python语言来开发软件的功能。

### 3.3.2树莓派开发语言——Python

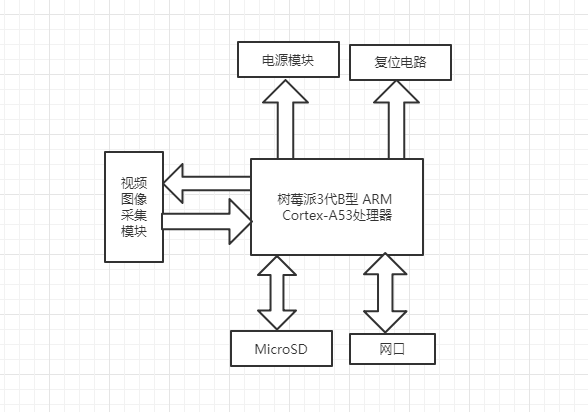
树莓派预搭载的编程开发环境是 Python 语言。 Python 是一种面向对象、直译式计算机编程语言，已有近 20 年的发展历史，包含了一组完善而且容易理解的标准库，能够轻松完成很多常见的任务，语法简捷、清晰，且具有垃圾回收功能，能够实现内存的自动管理，常用于处理系统管理任务和网络程序编写，也适合完成对图像和视频保存到本地。

## 3.3硬件结构

基于嵌入式系统硬件设备主要由树莓派、摄像头组成。

### 3.3.1树莓派

树莓派（Raspberry Pi）系统基于 Linux 研发，包括 40个 GPIO 口，可用于拓展模块，4个USB 端口，Micro SD 卡槽，支持视频音频的输出，网络接口，自带蓝牙 Wi Fi 模块，支持SPI，I2C等总线，是一款信用卡大小的小型电脑，支持linux操作系统，接口丰富，功能强大，价格低，可靠性高，扩展方便，开放性 好，成为开源硬件领域中较为高阶的硬件产品而备受欢 迎。 因此，本项目以树莓派为主板进行开发设计。



**图3-3-1树莓派硬件结构图**

### 3.3.2 树莓派3B+ raspberry pi3代b+CSI接口摄像头

参数: 镜头：1/4 5M 光圈：2.9 焦距：3.29 视场角：72.4度。

由树莓派与摄像头连接组成的系统如图 3-3-2所示。



**图3-3-2摄像头与树莓派的连接**

# 四．步骤

本项目主要分为几个步骤：①实时监控、②页面设计、③图片采集、④视频保存、⑤视频回放。具体如下所示：

## 4.1实时监控功能控制模块

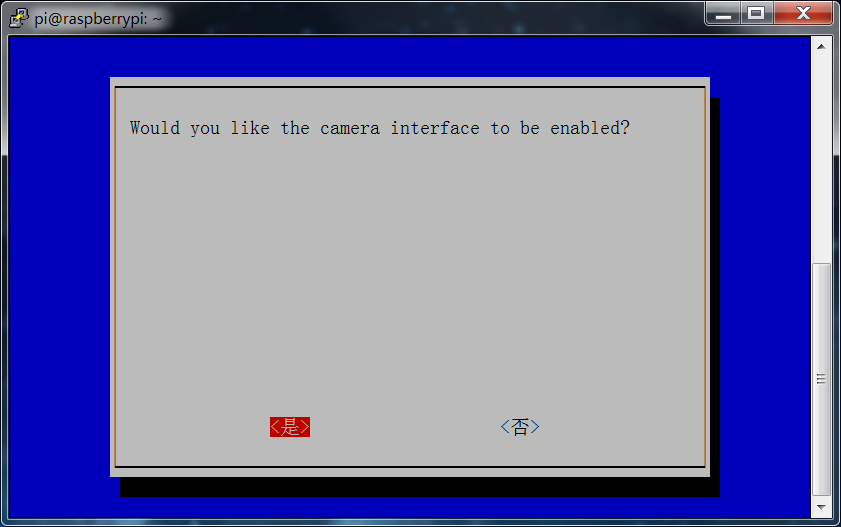
### 4.1.1摄像头的连接

树莓派有一个用于连接摄像头模块的专用CSI接口,是一种先进的外围设备控制器。可连接到树莓派的摄像头有两种,分别是USB摄像头和树莓派专用摄像头,考虑到成本和实际情况，我们采用的是前者的USB摄像头。连接好摄像头,需在配置窗口打开摄像头。

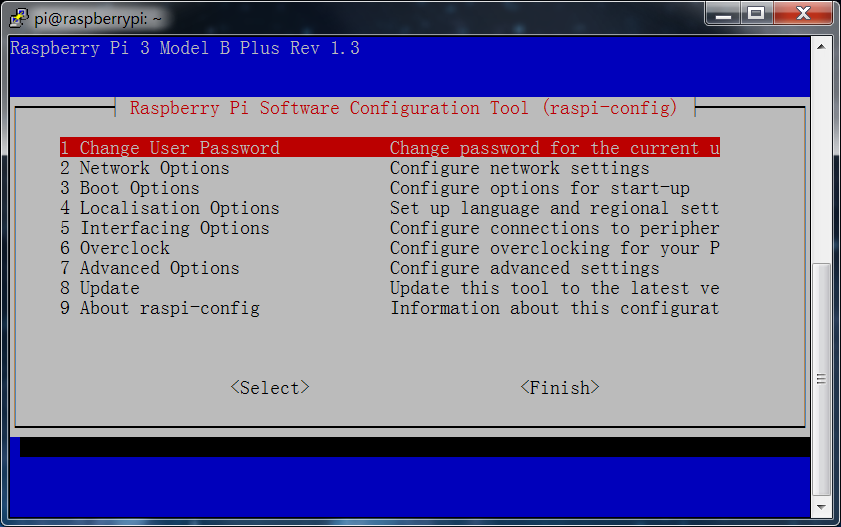
### 4.1.2摄像头实时监控

打开树莓派终端，输入sudo raspi-config,进入界面后选择Interfacing Options Configure connections to peripherals ,进入后选择Camera ，回车选择 yes。完成后退出界面，终端输入reboot 重启电脑。

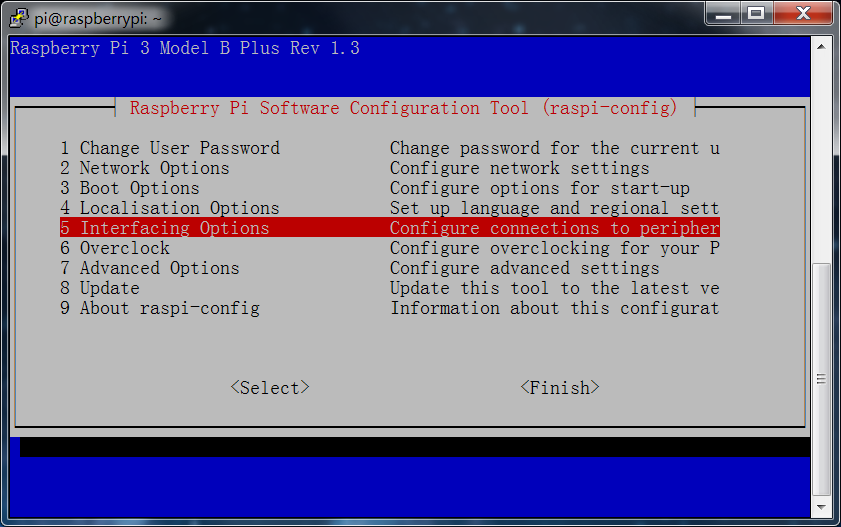
重启后，打开终端，输入sudo su选择root用户，输入raspistill -o image.jpg.测试拍照。



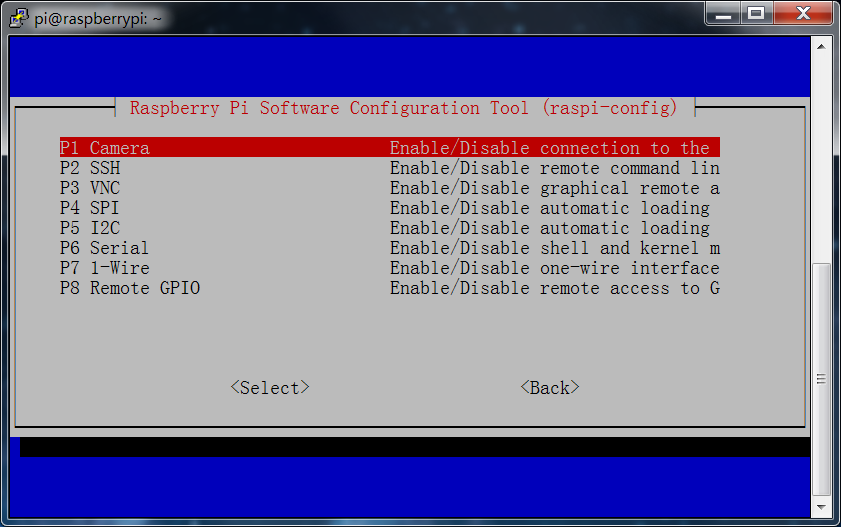
**图4-1-2-1进入树莓派打开摄像头选项**



**图4-1-2-2 配置选项**

****

**图4-1-2-3配置选项**



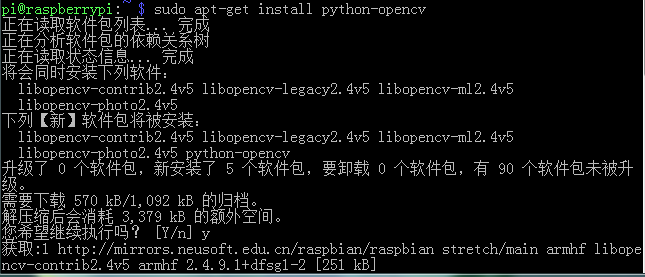
**图4-1-2-4配置选项**

安装opencv-python

pip install opencv-python

测试opencv的版本，在命令行输入python，进入python的命令模式，

然后import cv2,观察opencv2是否安装以及cv.\_\_version\_\_的版本是多少。 在树莓派中安装Python-opencv：



**图4-1-2-5 opencv的安装**

### 4.1.3工作流程图

摄像头将采集的图像通过opencv转换成jpg格式的视频流传输到内网，通过python socket将其视频数据转发到PC端上，PC端用户在输入树莓派的ip地址和端口号与树莓派连接，即可访问内网内的视频数据。

图示

描述已自动生成

**图4.1.3-1远程模块原理图**

### 4.1.4运行结果

手机屏幕的截图

描述已自动生成

在巴士上

低可信度描述已自动生成

**图4-1-4远程监控结果**

## 4.2代码

### 4.2.1 PC端代码

**monitor.py**

**from PyQt5 import QtCore, QtGui, QtWidgets**

**from PyQt5.QtGui import QPixmap**

**class Ui\_monitor(object):**

**def setupUi(self, monitor):**

**monitor.setObjectName("monitor")**

**monitor.resize(648, 620)**

**self.centralwidget = QtWidgets.QWidget(monitor)**

**self.centralwidget.setObjectName("centralwidget")**

**self.horizontalLayoutWidget = QtWidgets.QWidget(self.centralwidget)**

**self.horizontalLayoutWidget.setGeometry(QtCore.QRect(0, 490, 641, 41))**

**self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")**

**self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)**

**self.horizontalLayout.setContentsMargins(0, 0, 0, 0)**

**self.horizontalLayout.setObjectName("horizontalLayout")**

**self.pushButton\_3 = QtWidgets.QPushButton(self.horizontalLayoutWidget)**

**self.pushButton\_3.setObjectName("pushButton\_3")**

**self.horizontalLayout.addWidget(self.pushButton\_3)**

**self.pushButton\_2 = QtWidgets.QPushButton(self.horizontalLayoutWidget)**

**self.pushButton\_2.setObjectName("pushButton\_2")**

**self.horizontalLayout.addWidget(self.pushButton\_2)**

**self.pushButton = QtWidgets.QPushButton(self.horizontalLayoutWidget)**

**self.pushButton.setObjectName("pushButton")**

**self.horizontalLayout.addWidget(self.pushButton)**

**self.label = QtWidgets.QLabel(self.centralwidget)**

**self.label.setGeometry(QtCore.QRect(0, 0, 641, 481))**

**self.label.setObjectName("label")**

**self.horizontalLayoutWidget\_2 = QtWidgets.QWidget(self.centralwidget)**

**self.horizontalLayoutWidget\_2.setGeometry(QtCore.QRect(0, 530, 641, 51))**

**self.horizontalLayoutWidget\_2.setObjectName("horizontalLayoutWidget\_2")**

**self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget\_2)**

**self.horizontalLayout\_2.setContentsMargins(0, 0, 0, 0)**

**self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")**

**self.label\_3 = QtWidgets.QLabel(self.horizontalLayoutWidget\_2)**

**self.label\_3.setObjectName("label\_3")**

**self.horizontalLayout\_2.addWidget(self.label\_3)**

**self.lineEdit = QtWidgets.QLineEdit(self.horizontalLayoutWidget\_2)**

**self.lineEdit.setObjectName("lineEdit")**

**self.lineEdit.setText('127.0.0.1')**

**self.horizontalLayout\_2.addWidget(self.lineEdit)**

**self.label\_2 = QtWidgets.QLabel(self.horizontalLayoutWidget\_2)**

**self.label\_2.setObjectName("label\_2")**

**self.horizontalLayout\_2.addWidget(self.label\_2)**

**self.lineEdit\_2 = QtWidgets.QLineEdit(self.horizontalLayoutWidget\_2)**

**self.lineEdit\_2.setObjectName("lineEdit\_2")**

**self.lineEdit\_2.setText('6666')**

**self.horizontalLayout\_2.addWidget(self.lineEdit\_2)**

**self.pushButton\_4 = QtWidgets.QPushButton(self.horizontalLayoutWidget\_2)**

**self.pushButton\_4.setObjectName("pushButton\_4")**

**self.horizontalLayout\_2.addWidget(self.pushButton\_4)**

**monitor.setCentralWidget(self.centralwidget)**

**self.menubar = QtWidgets.QMenuBar(monitor)**

**self.menubar.setGeometry(QtCore.QRect(0, 0, 648, 23))**

**self.menubar.setObjectName("menubar")**

**monitor.setMenuBar(self.menubar)**

**self.statusbar = QtWidgets.QStatusBar(monitor)**

**self.statusbar.setObjectName("statusbar")**

**monitor.setStatusBar(self.statusbar)**

**self.retranslateUi(monitor)**

**QtCore.QMetaObject.connectSlotsByName(monitor)**

**def retranslateUi(self, monitor):**

**\_translate = QtCore.QCoreApplication.translate**

**monitor.setWindowTitle(\_translate("monitor", "MainWindow"))**

**self.pushButton\_3.setText(\_translate("monitor", "实时监控"))**

**self.pushButton\_2.setText(\_translate("monitor", "录制"))**

**self.pushButton.setText(\_translate("monitor", "回放"))**

**self.label.setText(\_translate("monitor", "请开始你的表演！"))**

**self.label\_3.setText(\_translate("monitor", "IP地址"))**

**self.label\_2.setText(\_translate("monitor", "端口"))**

**self.pushButton\_4.setText(\_translate("monitor", "监听"))**

**self.label.setPixmap(QPixmap('./load.jpg'))**

**PC.py**

**import os, sys, time, cv2, monitor, threading**

**import socket**

**import struct**

**import numpy**

**from PyQt5 import QtGui, QtWidgets**

**from PyQt5.QtWidgets import QMessageBox, QFileDialog**

**# 定义opencv图像转PyQt图像的函数**

**def cvImgtoQtImg(cvImg):**

**QtImgBuf = cv2.cvtColor(cvImg, cv2.COLOR\_BGR2BGRA)**

**QtImg = QtGui.QImage(QtImgBuf.data,**

**QtImgBuf.shape[1],**

**QtImgBuf.shape[0],**

**QtGui.QImage.Format\_RGB32)**

**return QtImg**

**class mainwin(QtWidgets.QMainWindow,monitor.Ui\_monitor):**

**def \_\_init\_\_(self):**

**super().\_\_init\_\_()**

**self.setupUi(self)**

**self.bClose = False**

**self.out = 1 # 视频文件**

**self.cap = 1 # 视频帧**

**# 当前状态**

**self.type = {'enable':1, 'display':2, 'ensave':3, 'relink':4}**

**self.status = self.type['enable']**

**# 信号与槽**

**self.pushButton\_3.clicked.connect(self.openMonitor) # 建立菜单点击的信号与方法openMonitor连接**

**self.pushButton.clicked.connect(self.openVideo) # 建立菜单点击的信号与方法openVideo连接**

**self.pushButton\_2.clicked.connect(self.saveVideo) # 建立菜单点击的信号与方法saveVideo连接**

**self.pushButton\_4.clicked.connect(self.monitor)**

**def runing(self):**

**self.client, self.addr = self.Server.accept()**

**fps = 24 # 设置帧率**

**while True:**

**tempdata = self.client.recv(8)**

**if len(tempdata) == 0:**

**print("+1")**

**continue**

**info = struct.unpack('lhh', tempdata)**

**buf\_size = int(info[0])**

**if buf\_size:**

**try:**

**# 把数据解码为opencv图像**

**self.buf = b""**

**self.temp\_buf = self.buf**

**while buf\_size:**

**self.temp\_buf = self.client.recv(buf\_size)**

**buf\_size -= len(self.temp\_buf)**

**self.buf += self.temp\_buf**

**data = numpy.fromstring(self.buf, dtype='uint8')**

**frame = cv2.imdecode(data, 1)**

**# 录制视频**

**if self.status == self.type['ensave']:**

**self.out.write(frame)**

**elif self.status == self.type['display']:**

**ret, frame = self.cap.read()**

**if not ret:**

**print('结束了')**

**# break**

**elif self.status == self.type['relink']:**

**self.status = self.type['enable']**

**break**

**# opencv图像转换pyqt图像**

**QtImg = cvImgtoQtImg(frame) # 将帧数据转换为PyQt图像格式**

**self.label.setPixmap(QtGui.QPixmap.fromImage(QtImg)) # 在ImgDisp显示图像**

**size = QtImg.size()**

**self.label.resize(size) # 根据帧大小调整标签大小**

**self.label.show() # 刷新界面**

**cv2.waitKey(int(1000 / fps)) # 休眠一会，确保每秒播放fps帧**

**except Exception as e:**

**print(e.args)**

**# 完成所有操作后，释放捕获器**

**cv2.destroyAllWindows()**

**# 实时监控**

**def openMonitor(self):**

**if self.status == self.type['display']:**

**self.status = self.type['enable']**

**t = threading.Thread(target=self.runing)**

**t.start()**

**# 录制视频**

**def saveVideo(self):**

**if self.status == self.type['enable']: # 录制**

**print('a')**

**self.status = self.type['ensave']**

**filename = time.strftime("%Y-%m-%d-%H-%M-%S.avi", time.localtime()) # 视频文件名**

**self.pushButton\_2.setText('保存')**

**fourcc = cv2.VideoWriter\_fourcc(\*'XVID')**

**self.out = cv2.VideoWriter(filename, fourcc, 20.0, (640, 480))**

**elif self.status == self.type['ensave']: # 保存**

**self.status = self.type['enable']**

**self.pushButton\_2.setText('录制')**

**self.out.release()**

**# 回放**

**def openVideo(self):**

**if self.status == self.type['ensave']: #打开文件**

**self.status = self.type['enable']**

**self.pushButton\_2.setText('录制')**

**self.out.release()**

**fileName\_choose, filetype = QFileDialog.getOpenFileName(self,**

**"选取文件",**

**os.getcwd(), # 起始路径**

**"avi Files (\*.avi)") # 设置文件扩展名过滤,用双分号间隔**

**if fileName\_choose == "":**

**print("\n取消选择")**

**return**

**self.status = self.type['display']**

**# 回放**

**self.cap = cv2.VideoCapture(fileName\_choose) # 播放视频**

**if not self.cap.isOpened():**

**print("Cannot open monitor")**

**QMessageBox.question(self, '警告', '找不到文件！')**

**exit()**

**def monitor(self):**

**# 上传服务器**

**self.TargetIP = ('127.0.0.1', 6666)**

**self.Server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # 生成新的套接字对象**

**self.Server.bind(self.TargetIP)**

**self.Server.listen(5)**

**# 线程**

**t = threading.Thread(target=self.runing)**

**t.start()**

**def closeEvent(self, event):**

**"""**

**重写closeEvent方法，实现dialog窗体关闭时执行一些代码**

**:param event: close()触发的事件**

**:return: None**

**"""**

**reply = QtWidgets.QMessageBox.question(self,**

**'本程序',**

**"是否要退出程序？",**

**QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No,**

**QtWidgets.QMessageBox.No)**

**if reply == QtWidgets.QMessageBox.Yes:**

**# 完成所有操作后，释放捕获器**

**self.cap.release()**

**cv2.destroyAllWindows()**

**self.Server.shutdown(2)**

**self.Server.close()**

**event.accept()**

**else:**

**event.ignore()**

**if \_\_name\_\_=='\_\_main\_\_':**

**app = QtWidgets.QApplication(sys.argv)**

**w = mainwin()**

**w.show()**

**sys.exit(app.exec\_())**

### 4.2.2树莓派端代码

**Camera.py**

**import socket**

**import struct**

**import sys**

**import time**

**import cv2**

**import numpy**

**class Config(object):**

**def \_\_init\_\_(self, TargetIP=('127.0.0.1', 6666)):**

**self.TargetIP = TargetIP**

**self.resolution = (640, 480) # 分辨率**

**self.img\_fps = 15 # 帧率**

**self.server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # 生成新的套接字对象**

**self.server.connect(self.TargetIP)**

**self.img = ''**

**self.img\_data = ''**

**def RT\_Image(self):**

**camera = cv2.VideoCapture(0) # 打开摄像头**

**img\_param = [int(cv2.IMWRITE\_JPEG\_QUALITY), self.img\_fps]**

**while True:**

**time.sleep(0.1) # sleep for 0.1 seconds**

**\_, self.img = camera.read()**

**self.img = cv2.resize(self.img, self.resolution) # 获取图像**

**# 把图像编码.jpg**

**\_, img\_encode = cv2.imencode('.jpg', self.img, img\_param)**

**img\_code = numpy.array(img\_encode)**

**self.img\_data = img\_code.tostring() # bytes data**

**try:**

**# 传输数据到客户端**

**packet = struct.pack(b'lhh', len(self.img\_data), self.resolution[0],**

**self.resolution[1])**

**self.server.send(packet)**

**self.server.send(self.img\_data)**

**except Exception as e:**

**print(e.args)**

**camera.release()**

**return**

**if \_\_name\_\_ == '\_\_main\_\_':**

**TargetIP = ('127.0.0.1', 6666)**

**if len(sys.argv) == 3:**

**TargetIP = (sys.argv[1], sys.argv[2])**

**config = Config(TargetIP)**

**config.RT\_Image()**

## 4.3视频流保存到本地功能模块

应用程序将将网络摄像机采集的.jpg 文件以数据流的形式。运行输入函数，与输出函数一样，二者都会创建从进程中独立出的线程，输入线程中创建了一个 抓图线程作为主体线程。抓图线程是一个循环体，没有用户的终止命令循环将一直持续。抓图函数以阻塞方式等待视频帧数据，当有一个视频帧到达，将 其 转 换为avi 格式，若该视频帧为YUYU格式则应先转换为RGB格式。通过opencv截取视频流以avi格式保存到本地。

### 4.3.1视频保存模块运行结果

文本, 信件

描述已自动生成

**图4-4-3-1保存视频到本地**

# 五．实训结果和工作总结

## 5.1实训结果

本次实训已经实现了以嵌入式树莓派为设计平台,结合树莓派专用摄像头、并运用Python语言编程实现视频实时监控与跟踪。为了验证此系统的可行性,在日常实验室测试多次,验证该系统为现代人的生活提供了一种简洁可靠的监控方式。主要完成了以下工作:

(1)本文在分析了一般嵌入式系统设计的基础上,对基于视频监控的嵌入式系统进行了总体设计,并选定了嵌入式树莓派为核心的设计。

(2)对视频采集处理模块进行了详细的软硬件设计。结合Open CV和python,使用软件获取闯入者信息,控制硬件对系统进行相应配置。

(3)最后,对视频实时监控与传输的网络接口模块和视频处理保存到本地分别进行了测试和分析,验证了系统设计方案的可行性。

# 六．心得体会

## 6.1收获

通过本次实训，我们掌握了树莓派系统的使用和Linux常用命令的使用。

本次实验的开头并不顺利，首先是树莓派端的环境配置，Opencv库的安装消耗了很多时间，从使用Python的安装模块进行安装到将Opencv源码下载到本地进行编译安装，尝试了很多方案，中途遇到一次系统崩溃，重装了一次树莓派系统后，根据前面出错的经验终于成功配置环境。

学习了python网络编程技术以及Opencv的使用，但还存在硬件知识方面的不足，今后在硬件知识方面需要继续学习和提高。