
四川大學

# 大学生创新创业训练计划
# 中期检查报告

项目名称：基于树莓派的远程实时监控系统

项目负责人：邢国浩

所在学院：软件学院

专业年级：软件工程 2018 级

学　　号：2018141463075

手　　机：15053392805

电子邮箱：1542620885@qq.com

指导教师：李辉

项目起止年月：2020.1-2020.10

项目参与学生人数：4

四川大学

2020 年 6 月 5 日

# 填 写 说 明

1、中期验收报告由正文和附件两部分组成，正文部分请按表格要求填写，并可根据需要加页，要求层次分明，内容准确。项目执行过程中的进展或研究成果、计划调整情况等，须在报告中如实反映。

2、对不按要求填报《中期验收报告》，或项目执行不力，或研究内容调整不当而影响项目顺利进展的，中止拨款。

3、一式一份，纸质版请用 A4 双面打印。

## 一、基本情况

<table>
<tr><td colspan="2">项目名称</td><td colspan="5">基于树莓派的远程实时监控系统</td></tr>
<tr><td colspan="2">项目类别</td><td colspan="5">☑ 创新训练计划 ☐ 创业训练计划<br>☐ 创业实践计划 ☐ 科研训练计划</td></tr>
<tr><td colspan="2">项目级别</td><td colspan="5">☐国家级 ☐省级 ☐校级 ☑院级 ☐重点 / ☑一般</td></tr>
<tr><td colspan="2">立项时间</td><td colspan="2">2019.12</td><td>立项经费</td><td colspan="2">3000 元</td></tr>
<tr><td colspan="2">项目执行情况<br>（自评）</td><td colspan="5">☐提前完成 ☑按期完成 ☐终止<br>预计完成时间： 2020 年 10 月 31 日之前</td></tr>
<tr><td rowspan="6">项目主要研究人员</td><td>序号</td><td>姓 名</td><td>学号</td><td>专业</td><td>所在学院</td><td>备注<br>（异动情况）</td></tr>
<tr><td>1</td><td>邢国浩</td><td>2018141463075</td><td>软件工程 2018 级</td><td>软件学院</td><td></td></tr>
<tr><td>2</td><td>段士童</td><td>2018141463005</td><td>软件工程 2018 级</td><td>软件学院</td><td></td></tr>
<tr><td>3</td><td>熊伟</td><td>2018141463047</td><td>软件工程 2018 级</td><td>软件学院</td><td></td></tr>
<tr><td>4</td><td>郦千辰</td><td>2018141463058</td><td>软件工程 2018 级</td><td>软件学院</td><td></td></tr>
<tr><td colspan="2">指导教师</td><td>李辉</td><td>-</td><td>软件工程</td><td>软件学院</td><td></td></tr>
<tr><td colspan="2">联系电话</td><td colspan="2">13008187268</td><td>-</td><td colspan="2">-</td></tr>
</table>

## 二、项目意义

本项目是基于树莓派实现的远程实时监控系统；目的之一是考虑到中国目前日渐庞大的监控系统发展趋势。2010-2017 年期间，我国视频监控市场规模从 242 亿元增长到 1124 亿元，年均复合增长率达 24.53%，2018,19 年更是飞速增长。可见监控产品在中国的需求量很大，且广泛应用于道路交通公共基础设施等相关领域。目前国内监控系统具有成熟且稳健的发展趋势，但是仍然存在较大的问题，归结起来可以总结为以下几点：

A 兼容性方面问题。当前我国各生产厂家所生产的产品兼容性较低。在实际应用过程中往往容易出现受到某些厂商制约的问题，或是只能进行二次开发，这样就给系统用户和集合商，尤其是研究者造成较大的困扰。

B 继承性方面问题。监控的目的重点在于看清楚被监控的目标，所以首先要解

决的问题便是图像清晰度。当前，市面上已经出现了 720P/1080P、百万像素等高清摄像机，并且也已经将红外夜视技术、可视性增强技术、高速高灵敏 ExmorCMOS 技术以及超宽动态技术等加入其中，使得视频采集水平获得大幅提高。不过出于成本角度的考虑，在建设某些实验系统时，尤其是本项目目标指向的识别实验平台，不可能在短期内实现全高清，这是我们要考虑的因素

C 技术方面问题。比如：网络视频监控系统的中心监控端主要依赖人工操作。就目前来看，安保人员最主要的就是监控视频屏幕，如何能将这项重要工作任务解放出来由智能计算机来完成是当前面临的技术难题。

D 安装规划方面问题。在进行具体安装作业前应当对摄像头安装位置做好相应的统筹和规划，但当前在横向规划（设置位置、角度等方面）上面还存在着某些问题。本系统的设计目的之一在于为不同用户群体探索更加方便、更加廉价的远程监控系统，可以在保护财产安全、创建一个安全的环境方面起到作用。

本项目是基于树莓派实现的远程实时监控系统，分为远程端和客户端部分，该系统实现对多个远程摄像头的控制，在客户端内根据用户的需求进行切换。客户端实现实时显示，接受图像和录屏，查看远程端历史视频功能，并带有调焦和切换等辅助功能。远程端可以根据安装摄像头的种类不同实现不同的监控目的。并根据环境光的强度来切换摄像模式。

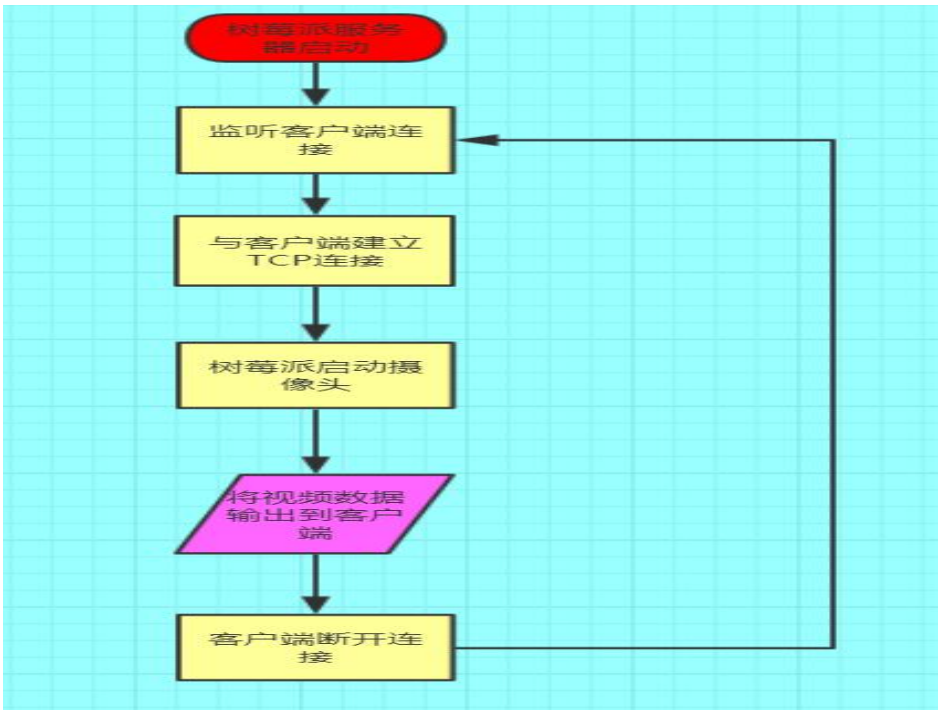
该系统成本十分低廉，实现完整远程监控功能的基础上，具有极强的可拓展性，应用场景广泛。能够作为各类技术的实验平台，为研究监控技术收集大量的图片、视频信息和探索新的道路。

## 三、项目主要进展

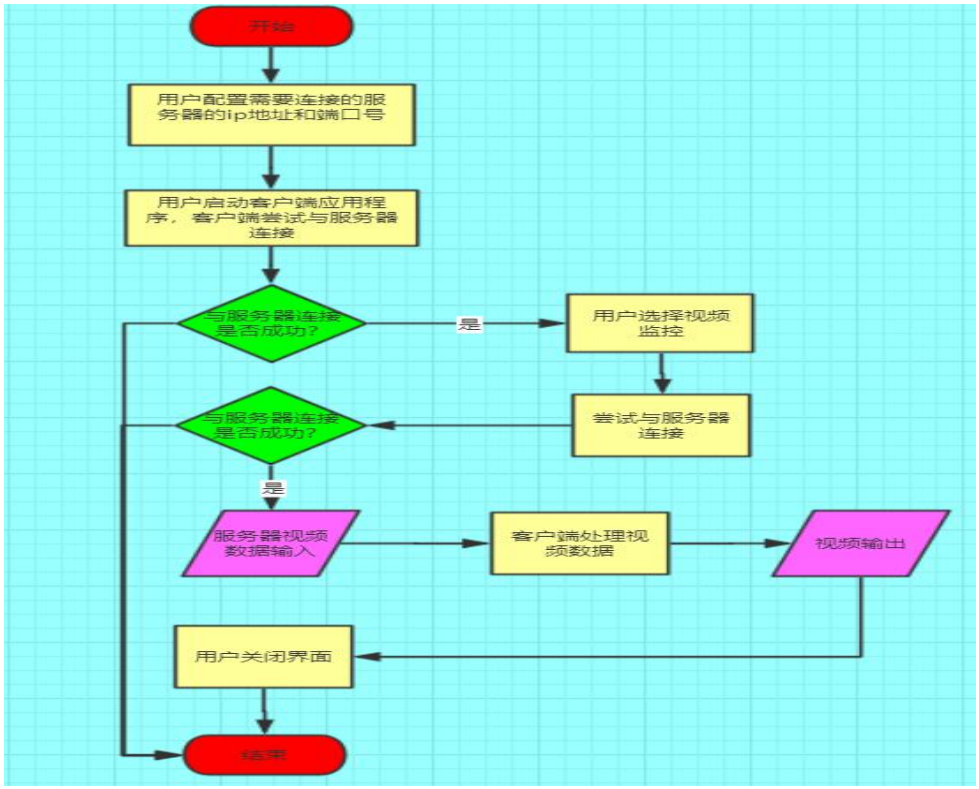
1. 项目设计

项目设计完成应用端服务器和客户端的视频实时传输，在申报书设计的基础上细化设计，以下是部分设计核心的展示：

远程监控功能服务器流程图：



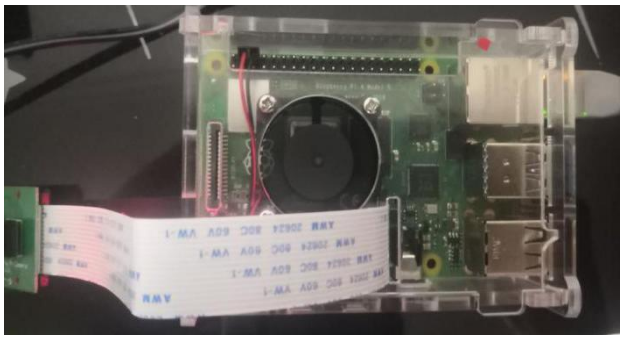

远程监控系统客户端流程图



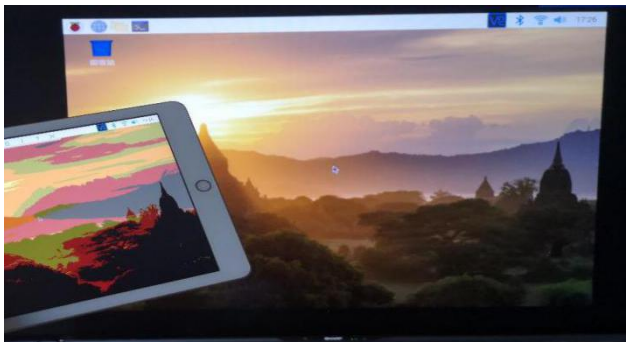
2.项目实现

2.1 完成树莓派环境的搭建和配置

根据指示连线，安装风扇、板子、摄像头。使用官方工具烧录 raspberry 系统。将 TF 卡插到树莓派上，完成通电启动，硬件主机配置完成：



获取树莓派动态 ip。使用 SSH 登录，更改用户名和密码，配置树莓派网络文件 dhcpcd.conf，设置树莓派的静态 ip 节约每次访问时间。之后下载 vnc viewer 或者使用远程桌面连接，得到树莓派系统桌面：



在树莓派上使用命令：

raspivid -o mykeychain.h264

sudo apt-get install -y gpac

MP4Box -fps 30 -add keychain.h264 keychain.mp4

调试摄像机。得到一个 mp4 文件和一个 h264 文件(长度 5 秒，分辨率 1920x1080，比特率 17Mbps)。确定摄像工作正常。

在树莓派上安装中文字库和中文输入法，安装 python3 和 pycharm，创建虚拟环境，下载导入 opencv 和 pillow，在 github 上创建项目仓库邀请项目成员（https://github.com/CC1AH/REMOTE-MONITORING-SYSTEM）实现项目管理；资料收集；完成编码准备工作。

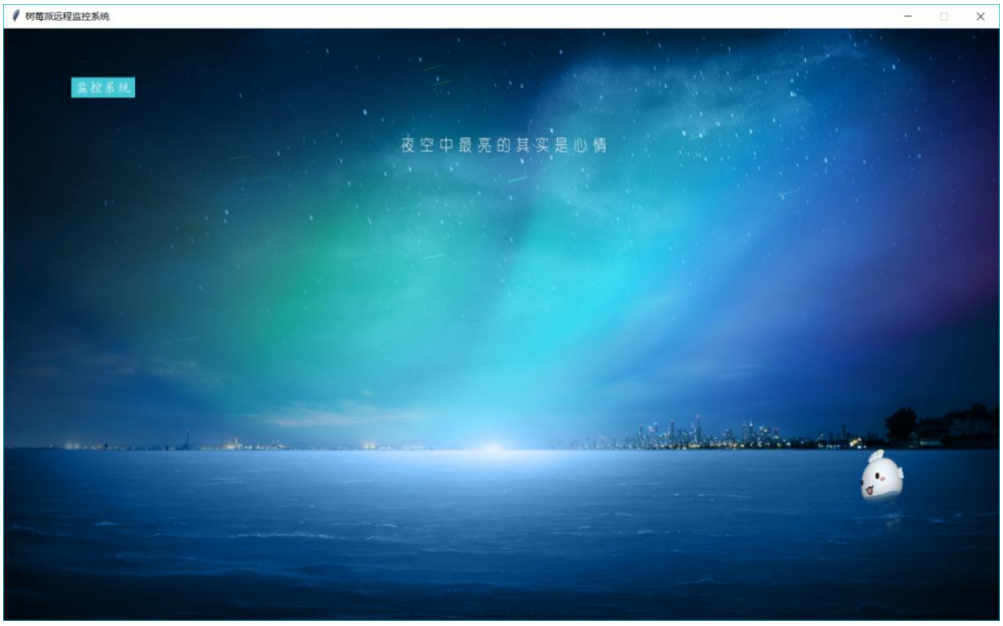
2.2 应用程序版本内容实现

项目完成了基于 TCP 的应用程序版本的树莓派系统的初步搭建，整个系统由服务器和客户端两部分组成。树莓派作为服务器在远端，始终处于开启状态，监听客户端的连接，并根据响应数据，传输相应的数据到客户端；用户的电脑作为客户端，应用启动前用户需要配置树莓派服务器的 ip 地址和端口，连接上服务器后，用户选择功能，例如视屏监控、拍照、录屏等。

2.2.1 UI 设计

本项目的 UI 实现用到了 python 的 tkinter 库。Tkinter 是 Python 的标准 GUI 库。Python 使用 Tkinter 可以快速的创建 GUI 应用程序。GUI 的编写可以分为以下四个部分：导入 Tkinter 模块、创建控件、指定这个控件的 master 和控件置入。

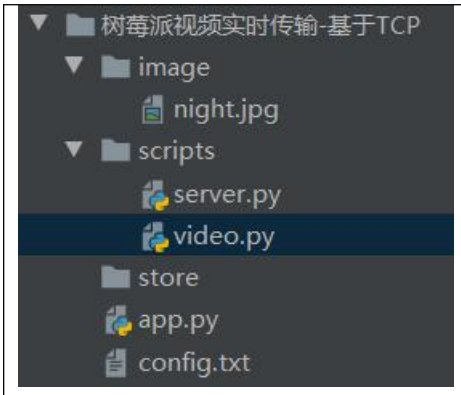
目前，我们完成了远程视频监控功能，后面也将实现其他功能。我们利用 tkinter 制作了初始的客户端应用界面，进入界面展示如下：




2.2.2 python 核心逻辑代码

项目组成员在自己的电脑里面进行客户端和服务器端的 python 编程，然后将服务器端的代码移植到树莓派中，在树莓派中启动服务器，然后通过花生壳进行内网穿透。经过测试，外网能够连接到服务器，并且能够实现远程监控功能。

代码目录：




示例实现代码展示：

app.py 定义了基本界面的构成，同时提供基本视频显示等的接口：

app.py

```
# coding=utf-8
from tkinter import *
import tkinter.messagebox as msg
from PIL import Image, ImageTk
import cv2
import time
import os
import socket


# 用于获取图片
def get_image(fp, width, height):
    img = Image.open(fp).resize((width, height))
    return ImageTk.PhotoImage(img)


def runVideo(event):
    os.system('python scripts/video.py')


if __name__ == "__main__":
    app = Tk()
    app.title("树莓派远程监控系统")
    app_width = 1280
    app_height = 760
    app.geometry('%dx%d+%d+%d' % (app_width, app_height,
                                  (app.winfo_screenwidth() - app_width) / 2,
                                  (app.winfo_screenheight() - app_height) / 2 - 20))
    app.resizable(False, False)
```

```
# 读取配置文件，获取服务器ip地址和端口号
try:
    with open('config.txt', 'r+') as f:
        ip = f.readline().split(':', 2)[1].replace('\n', '')
        port = f.readline().split(':', 2)[1].replace('\n', '')
        port = int(port)
        f.close()
except FileNotFoundError:
    msg.showerror('错误', '配置文件缺失!')
    exit(0)
except Exception:
    msg.showerror('错误', '配置文件内容格式错误!')
    exit(0)
try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((ip, port))
    # sock.connect(('127.0.0.1', 9999))
    print("connect successfully.")
    sock.close()
except Exception:
    msg.showerror('错误', '与远程服务器连接失败！')
    exit(0)

# 主界面
main_frame = Frame(app)
img_bg = get_image('image/night.jpg', app_width, app_height)
background = Label(main_frame, width=app_width, height=app_height, image=img_bg)
btn_video = Button(main_frame, text="监控系统", font=('楷体', 13), bg='#44C7D3', fg='white'
btn_video.bind("<Button-1>", runVideo)
background.pack()
btn_video.place(relx=0.1, rely=0.1, anchor=CENTER)
main_frame.pack()
app.mainloop()
```

下面 server.py 部分代码展示了 TCP 服务器的搭建（本地测试，关于内网穿透的部分后面提及），同时提供视频传输服务：

server.py

```python
# coding=utf-8
# 服务器端
import socket
import cv2
from PIL import Image
from io import BytesIO
import time

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind(('127.0.0.1', 9999))
sock.listen(2)  # 监听端口

# 等待目标端连接
dst, dst_addr = sock.accept()
print("Destination Connected by", dst_addr)
cap = cv2.VideoCapture(0)
cap.set(3, 440)
cap.set(4, 330)

while True:
    ret, img = cap.read()
    if ret is False:
        print("can not get this frame")
        continue
    pi = Image.fromarray(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    buf = BytesIO()
    pi.save(buf, format='JPEG')
    jpeg = buf.getvalue()
    buf.close()
    transfer = jpeg.replace(b'\n', b'\-n')
    print(len(transfer), transfer[-1])
```

Video.py 展示了从服务器获取视频的部分

```python
# coding=utf-8
# 客户端
import cv2
import socket
from PIL import Image
from io import BytesIO
import numpy as np
import tkinter.messagebox as msg

# 获取TCP连接外网地址和端口号
try:
    with open('config.txt', 'r+') as f:
        ip = f.readline().split(':', 2)[1].replace('\n', '')
        port = f.readline().split(':', 2)[1].replace('\n', '')
        port = int(port)
        print(ip, port)
except FileNotFoundError:
    msg.showerror('错误', '配置文件缺失!')
    exit(0)
except Exception:
    msg.showerror('错误', '配置文件内容格式错误!')
    exit(0)

# 连接到服务器
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((ip, port))
# sock.connect(('127.0.0.1', 9999))
print('connect successfully.')
f = sock.makefile(mode='rb')

winname='camera'
```

```python
print("connect successfully.")
f = sock.makefile(mode='rb')

winname='camera'

while True:
    # 从服务器读取数据，一行的结尾是'\n'，注意我们前面已经将每一帧数据的'\n'替换成'\-n'，而结尾就是'\n'
    msg = f.readline()
    if not msg:
        break
    print(len(msg), msg[-2])
    # 将'\-n'换回来成'\n'
    jpeg = msg.replace("\-n".encode('utf8'), "\n".encode('utf8'))
    buf = BytesIO(jpeg[0:-1])  # 缓存数据
    buf.seek(0)
    pi = Image.open(buf)  # 使用PIL读取jpeg图像数据
    img = cv2.cvtColor(np.asarray(pi), cv2.COLOR_RGB2BGR)  # 从PIL的图像转成opencv支持的图像
    buf.close()
    img = cv2.resize(img, (800, 720))
    cv2.resizeWindow(winname, 800, 720)
    cv2.imshow("camera", img)  # 实时显示
    if cv2.waitKey(10) == 27:
        break
    if cv2.getWindowProperty(winname, cv2.WND_PROP_AUTOSIZE) < 1:
        break

sock.close()
cv2.destroyAllWindows()
```

2.2.3 核心界面展示（获取视频）：


2.3 网页版内容实现

项目除了计划了应用端的视频传输功能基本实现之外，还考虑到开发网络端的应用内容,实现了基本前端界面和视频实时传输显示。并且尝试在应用端应用网络端的技术拓展。

2.3.1 flask 框架

项目开发网页版，经过讨论选择使用 flask 框架。Flask 是一个用 Python 编写的 Web 应用程序框架。Armin Ronacher 带领一个名为 Pocco 的国际 Python 爱好者团队开发了 Flask。Flask 基于 Werkzeug WSGI 工具包和 Jinja2 模板引擎。两者都是 Pocco 项目。Flask 也被称为“microframework” ，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。使用 Flask 目的是通过使用应用网络框架来增加健壮、稳定、可拓展性。

2.3.2 python 核心逻辑代码展示

视频显示界面核心代码 appCam.py：

```python
@app.route('/')
def index():
    """回报主页"""
    return render_template('index.html')


def gen(camera):
    """生成函数流"""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/video_feed')
def video_feed():
    """视频流路由. 将该属性放在img src中."""
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(host='192.168.1.10', port=8080, debug=True, threaded=True)


video_feed()
```

硬件模块核心代码 pi_camera.py：

```python
class Camera(object):
    thread = None
    frame = None
    last_access = 0

    def initialize(self):
        if Camera.thread is None:
            Camera.thread = threading.Thread(target=self._thread)
            Camera.thread.start()

            #等待直到帧可用
            while self.frame is None:
                time.sleep(0)

    #更新摄像时间 返回桢 在appCam的生成函数流方法中被调用
    def get_frame(self):
        Camera.last_access = time.time()
        self.initialize()
        return self.frame

    @classmethod
    def _thread(cls):
        with picamera.PiCamera() as camera:
            # camera setup
            camera.resolution = (320, 240)
            camera.hflip = True
            camera.vflip = True

            # let camera warm up
            camera.start_preview()
            time.sleep(2)
```

```python
        return self.frame

    @classmethod
    def _thread(cls):
        with picamera.PiCamera() as camera:
            # camera setup
            camera.resolution = (320, 240)
            camera.hflip = True
            camera.vflip = True

            # let camera warm up
            camera.start_preview()
            time.sleep(2)

            stream = io.BytesIO()
            for foo in camera.capture_continuous(stream, 'jpeg',
                                                 use_video_port=True
                stream.seek(0)
                cls.frame = stream.read()
                stream.seek(0)
                stream.truncate()
                if time.time() - cls.last_access > 5:
                    break
        cls.thread = None
```

### 2.3.3 核心实现界面：

使用浏览器打开进行测试，核心的视频显示界面如下展示：


2.4 跨局域网的传输

为实现跨局域网的视频传输，经讨论我们决定使用花生壳内网穿透工具或者在路由器上开启端口映射（组员可以选择），从而实现和外网的通信。

花生壳是动态域名解析服务商，在 2013 年 11 月 11 号正式发布了花生壳（内网穿透软件）。其颠覆性的技术创新，无需公网 IP，适用各种复杂的网络环境；无需路由器设置端口映射，简化了大量设置流程，给用户提供了方便。我们使用了花生壳生成的一个域名 http://i31a202966.wicp.vip。

## 四、下一步工作计划

1.完善两个版本其他功能和接口，网页版提供应用程序版本的下载接口

完成两个版本的录屏、存储功能，同时充实界面内容。项目组成员对于树莓派硬件模块和 flask 框架的熟悉程度还不足，应该继续根据教程学习讨论 picamera 编程和网络编程的更多技术细节：

FLASK:

https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world

PICAMERA:

https://picamera.readthedocs.io/en/release-1.13

2.研究分辨率和延迟的解决方案

经过实验，当分辨率调高时会出现明显的卡顿现象。下一步应该继续讨论研究使用其他视频传输协议等方法提高视频流畅度的手段。

## 五、经费使用情况和经费安排计划

**1 经费使用情况**

| 支出 | 费用 |
|---|---|
| 总计 | 2072 |

**2 经费安排计划**

| 支出 | 费用 |
|---|---|
| 树莓派相关硬件购买/熊伟 | 480 |
| 树莓派相关硬件购买/邢国浩 | 485 |
| 树莓派相关硬件购买/段士童 | 540 |
| 树莓派相关硬件购买/郦千辰 | 567 |
| 总计 | 2072 |

## 六、存在问题、建议及需要说明的情况

因为疫情原因无法到校，为保证每个人可以进行到项目，通知了每个人买了树莓派。5 月又通知经费削减，超出预算了。

**项目负责人（签字）：邢国浩**

**2020 年 6 月 4 日**

## 六、审核意见

<table>
<tr><td colspan="3">指导教师对该项目研究进展情况的意见：<br><br>指导教师签名：<br>年　月　日</td></tr>
<tr><td colspan="3">中期检查评价意见：<br><br>专家组组长签名：<br>年　月　日</td></tr>
<tr><td rowspan="4">学校（学院）对项目进展审核意见：<br><br>学校（学院）负责人签名（盖章）：<br>年　月　日</td><td colspan="2">中期检查结果类别（请打√）</td></tr>
<tr><td>按期完成</td><td></td></tr>
<tr><td>限期整改</td><td></td></tr>
<tr><td>终止</td><td></td></tr>
</table>

注：如表格空间不够请另附纸张装订在一起。