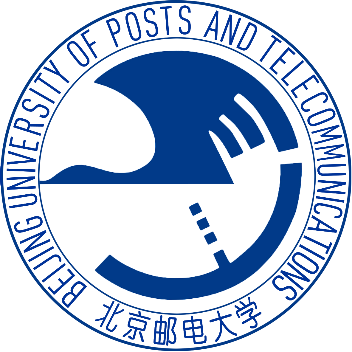
信息工程综合实验

期末报告

****

题目：基于ZYNQ\_MINI开发板的人脸识别项目设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 成员 | 姓名 | 学号 | 班级 | 贡献度 |
| 1 | 黄自锴 | 2020212262 | 2020219105 | 0.5 |
| 2 | 林子川 | 2020212327 | 2020219106 | 0.4 |
| 3 | 常海博 | 2020212102 | 2020219103 | 0.1 |

**北京邮电大学**

**人工智能学院**

**2023年12月**

目录

[一. 系统简介 3](#_Toc155306239)

[二. 总体设计 3](#_Toc155306240)

[2.1 整体设计框架 3](#_Toc155306241)

[2.2 系统处理流程 4](#_Toc155306242)

[三. 模块的设计与实现 5](#_Toc155306243)

[3.1 镜像模块 5](#_Toc155306244)

[3.1.1 功能 5](#_Toc155306245)

[3.1.2 外部接口 12](#_Toc155306246)

[3.1.3 组成框图/工作流程 12](#_Toc155306247)

[3.2 socket通信模块 12](#_Toc155306248)

[3.2.1 功能 12](#_Toc155306249)

[3.2.2 工作流程 14](#_Toc155306250)

[3.3 人脸检测模块 15](#_Toc155306251)

[3.3.1 功能 15](#_Toc155306252)

[3.3.2 工作流程 16](#_Toc155306253)

[3.4 按键控制模块 17](#_Toc155306254)

[3.4.1 功能 17](#_Toc155306255)

[3.4.2 工作流程 19](#_Toc155306256)

[四. 系统联调与测试 19](#_Toc155306257)

[五. 遇到的问题及解决方案 23](#_Toc155306258)

[六. 总结与分析 24](#_Toc155306259)

[七. 项目分工 24](#_Toc155306260)

[参考文献 24](#_Toc155306261)

# 系统简介

本项目实现了一个人脸检测系统，它能够识别出摄像头视频范围内的数张人脸并用矩形框标识出来进行提示。

# 总体设计

本章节包含系统整体设计框架和系统处理流程两个部分。

## 整体设计框架

本项目整体设计框架如下图所示。



整体设计框架图

整个系统由镜像制作、输入处理、数据传输、人脸检测、输出处理、按键检测等组成。各步骤的主要功能说明如下：

**镜像制作：为ZYNQ\_MINI开发板制作自定义PYNQ镜像，满足项目个性化需求，使得可以使用python语言和库对嵌入式处理器和overlay库进行编程。**

**输入处理：主机摄像头采集实时视频流作为人脸检测模型的输入。**

**数据传输：**通过socket通信将主机采集的实时视频流传输到开发板上，作为人脸检测模型的输入。

**人脸检测：实时检测视频流中的人脸并用矩形框框出。**

**输出处理：**通过socket通信将实时检测结果视频流传输到主机上，并打开窗口展示实时检测结果。

**按键检测：程序开始运行后，LED灯流水，监测key1键是否被按下。若key1键被按下，关闭开发板上的服务器端程序，LED灯熄灭。**

## 系统处理流程

该系统采用服务器-客户端通信模式，主机作为客户端，ZYNQ\_MINI开发板作为服务器。主机摄像头采集实时视频流并通过网口通信将其传输到具有PYNQ框架的ZYNQ\_MINI开发板上，通过人脸检测模型实时识别视频流的人脸并框出，并把实时识别结果回传到主机上显示。检测过程中开发板的LED灯以流水形式表示程序正在运行，当按下开发板上的key1键时，终止程序，LED灯全部熄灭。系统处理流程图如下：



系统流程图

# 模块的设计与实现

## 镜像模块

### 功能

编译镜像使用版本：Vitis2020.1 PYNQ2.6.0 PetaLinux2020.1 Ubuntu18.06

Vivado工程使用版本：Vitis2020.1 LinuxMint21.1

在构建Vivado工程的时候可以不使用特定的Ubuntu版本。

**下载Vitis2020.1。**下载地址：[BYRBT :: 种子详情 "[Windows/Linux][编程开发][][Xilinx Vivado Design Suite + Vitis Unified Software Platform][2020.1][英文][压缩包]" - Powered by NexusPHP](https://byr.pt/details.php?id=298198&hit=1)

**下载PetaLinux2020.1** 。下载地址：[Release xilinx-v2020.1: embeddedsw: Update commit ids for 2020.1 release · Xilinx/meta-petalinux (github.com)](https://github.com/Xilinx/meta-petalinux/releases/tag/xilinx-v2020.1) （建议国际联网）

**下载PYNQ。**下载方法（命令行）：

git clone https://github.com/Xilinx/PYNQ.git

cd PYNQ

git checkout v2.6.0

git checkout -b fzu3

（建议国际联网）

**下载编译使用的适配于ZYNQ开发板arm架构的镜像：**

注意不是下载aarch64架构的！

[BYRBT :: 种子详情 "[Linux][其他][][PYNQ prebuilt board-agnostic root filesystem][arm v2.6][英文][压缩包]" - Powered by NexusPHP](https://byr.pt/details.php?id=339914&hit=1)

原官网地址：

为ZYNQ\_MINI开发板定制一个可使用PYNQ框架进行开发的满足项目个性化需求的镜像文件。镜像的制作与烧录过程如下：

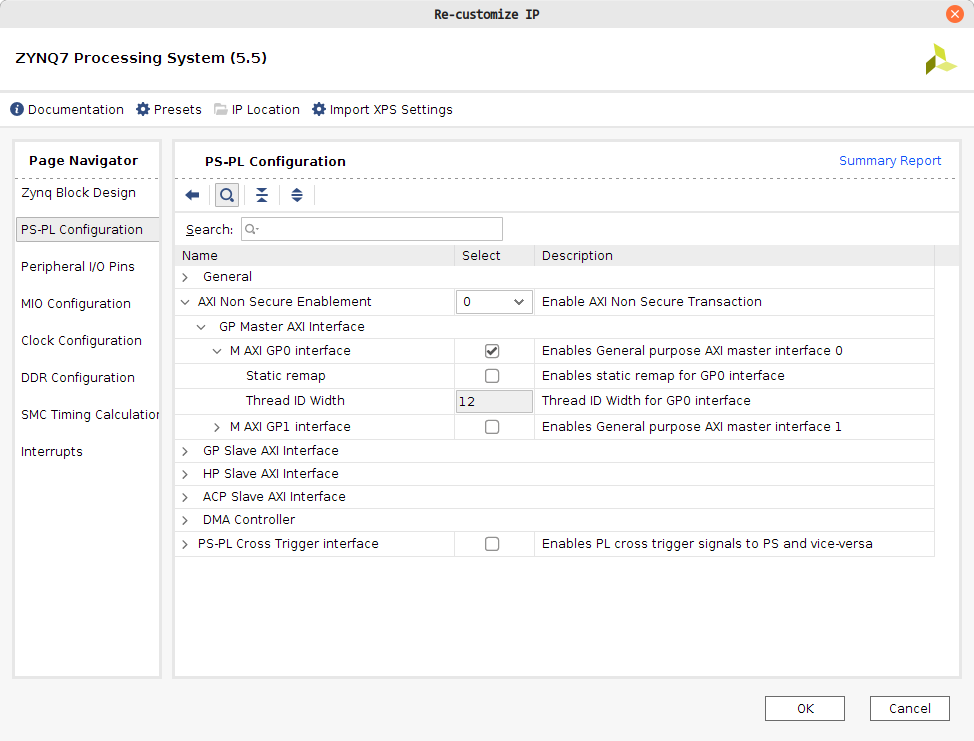
#### vivado硬件设计部分



针对ZYNQ模块，按照原理图，根据Bank0和Bank1的PS\_MIO管脚编号，设定为使用以下编号的MIO接口，并分别设置两个Bank的电平：



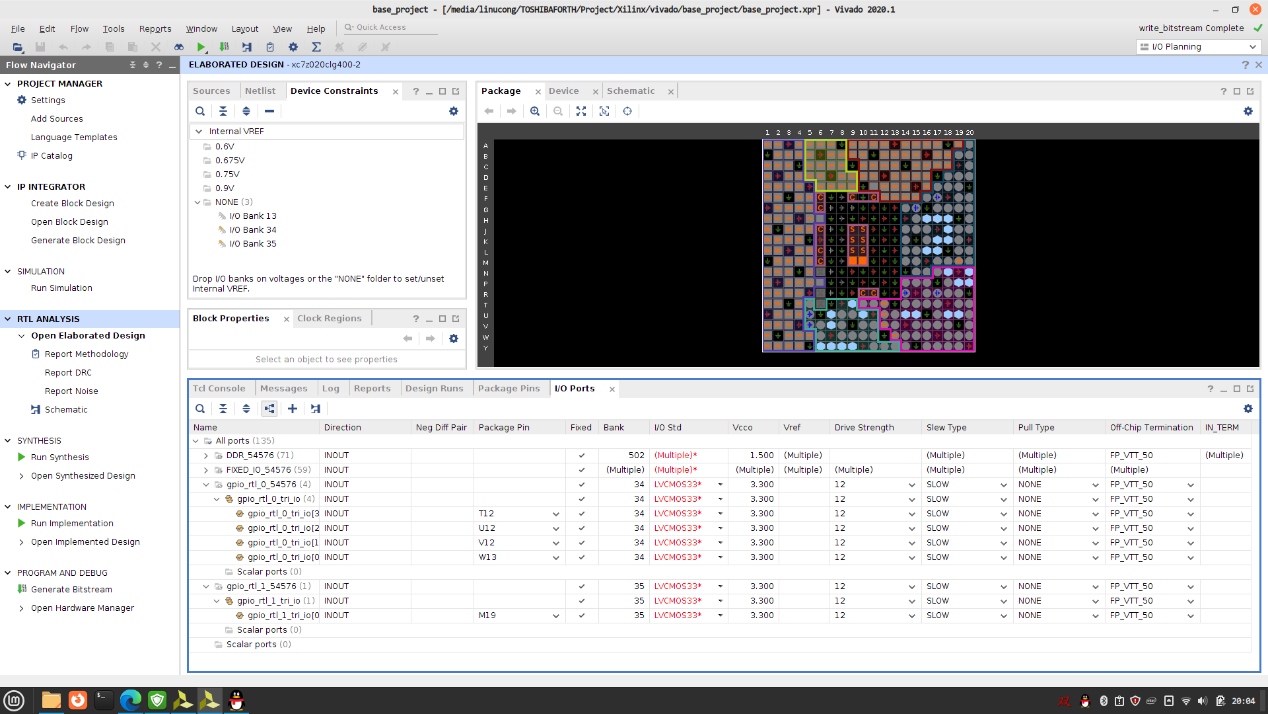
为了控制外设，使用AXI-GP总线控制PL中GPIO的方式，对板上的灯进行检测，读取按键是否按下。参考：AXI 总线详解-CSDN博客



一共有4个灯和一个连接PL的按钮，因此设置axi\_gpio\_0宽度为4位，axi\_gpio\_1为1位。

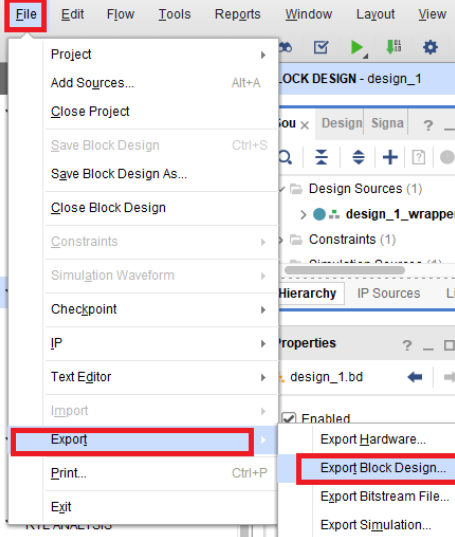
|  |  |
| --- | --- |
|  |  |

之后配置GPIO连接外设对应管脚。按照原理图，根据BANK34和BANK35的管脚编号，设置以下管脚。同时注意电平。



单击综合，完成硬件设计模块。

随后导出模块设计并准备好四个文件备用，分别是.xsa、.hwh。其中.xsa在硬件导出之后（勾选导出bit流）可以在linux下解压，另外获得.bit、.tcl文件。



#### ubuntu制作镜像部分

参考：自定义单板创建PYNQ镜像V2.6\_米联客 pynq-CSDN博客

在制作镜像时首先**将虚拟机Ubuntu进行国际联网。**对于虚拟机，如果Windows使用V2rayN，可以在.NAT模式下设置虚拟机代理端口号和Windows下v2rayN软件使用端口号一致。同时建议设置v2rayN的路由规则，使得内外路由分离。

预留虚拟机有300G以上空间。

可以使用VSCode在下载的PYNQ文件夹中，全局搜索检查是否包含有“gitsm://”字段。如有，请将其更改为“https://”或者“http://”

参考自定义单板创建PYNQ镜像V2.6\_米联客 pynq-CSDN博客，

在~/.bashrc中加入如下代码，以后每次打开一个新终端，都会自动加载。

source /tools/Xilinx/Vivado/2020.1/settings64.sh

source /tools/Xilinx/PetaLinux/2020.1/settings.sh

source /tools/Xilinx/Vitis/2020.1/settings64.sh

export PATH=/opt/qemu/bin:/opt/crosstool-ng/bin:$PATH

对于前三行命令，需要根据Vivado、PetaLinux、Vitis安装/放之路径进行更改。建议下载完毕Vivado、PetaLinux、Vitis后，将其放在同一个文件夹下。

**将编译使用的适配于ZYNQ开发板arm架构的镜像**bionic.arm.2.6.0\_2020\_10\_19**复制**到/sdbuild/prebuilt

在 ./PYNQ/boards/ 目录下新建文件夹 fzu3

在 fzu3 中再新建两个文件夹分别为 base 和petalinux\_bsp

在 petalinux\_bsp 中再建立文件夹 hardware\_project

将工程中的bit文件重名为base.bit拷贝到base

根据教程将.xsa文件重命名为fzu3.xsa，拷贝到hardware\_project

同时参考：那些年我们定制PYNQ需要避免的坑 - 知乎 (zhihu.com)

在fzu3文件夹中新建fzu3.spec文件，在fzu3.spec中填入内容如下：

ARCH\_fzu3 := arm

BSP\_fzu3 :=

BITSTREAM\_fzu3 := **base/base.bit**

FPGA\_MANAGER\_fzu3 := 1

STAGE4\_PACKAGES\_fzu3 := pynq ethernet

**对ubuntu sudo进行免密码：**

sudo visudo

在%sudo ALL=(ALL:ALL) ALL下面添加一行username ALL=(ALL) NOPASSWD: ALL即可。

我们使用的Ubuntu的版本号为18.06。对于不符合要求的Ubuntu版本会编译失败。版本对应参考：PYNQ SD Card image — Python productivity for Zynq (Pynq)。

**需要检查国际联网流量是否充足，剩余流量应至少大于15G。**

**在sdbulid文件夹运行命令开始构建：**

make BOARDS=fzu3 PREBUILT=./prebuilt/bionic.arm.2.6.0\_2020\_10\_19.img

可以使用make checkenv命令对下载的版本进行确认和检查。

如果2000余个任务存在编译错误，则可以使用make delete删除编译错误的部分而不必重新开始下载和编译。在2000余个任务编译完成之后，如果发生了编译错误，那么需要make clean。

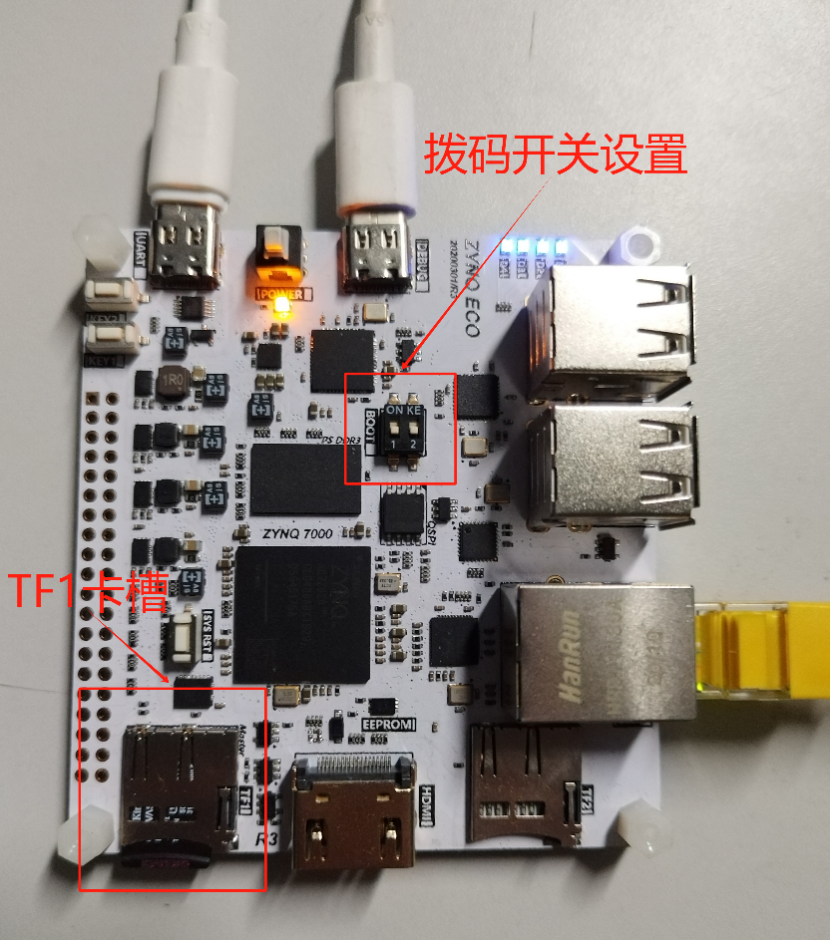
#### 使用Win32DiskImager烧写镜像部分

将编好的镜像复制到Windows操作系统下。

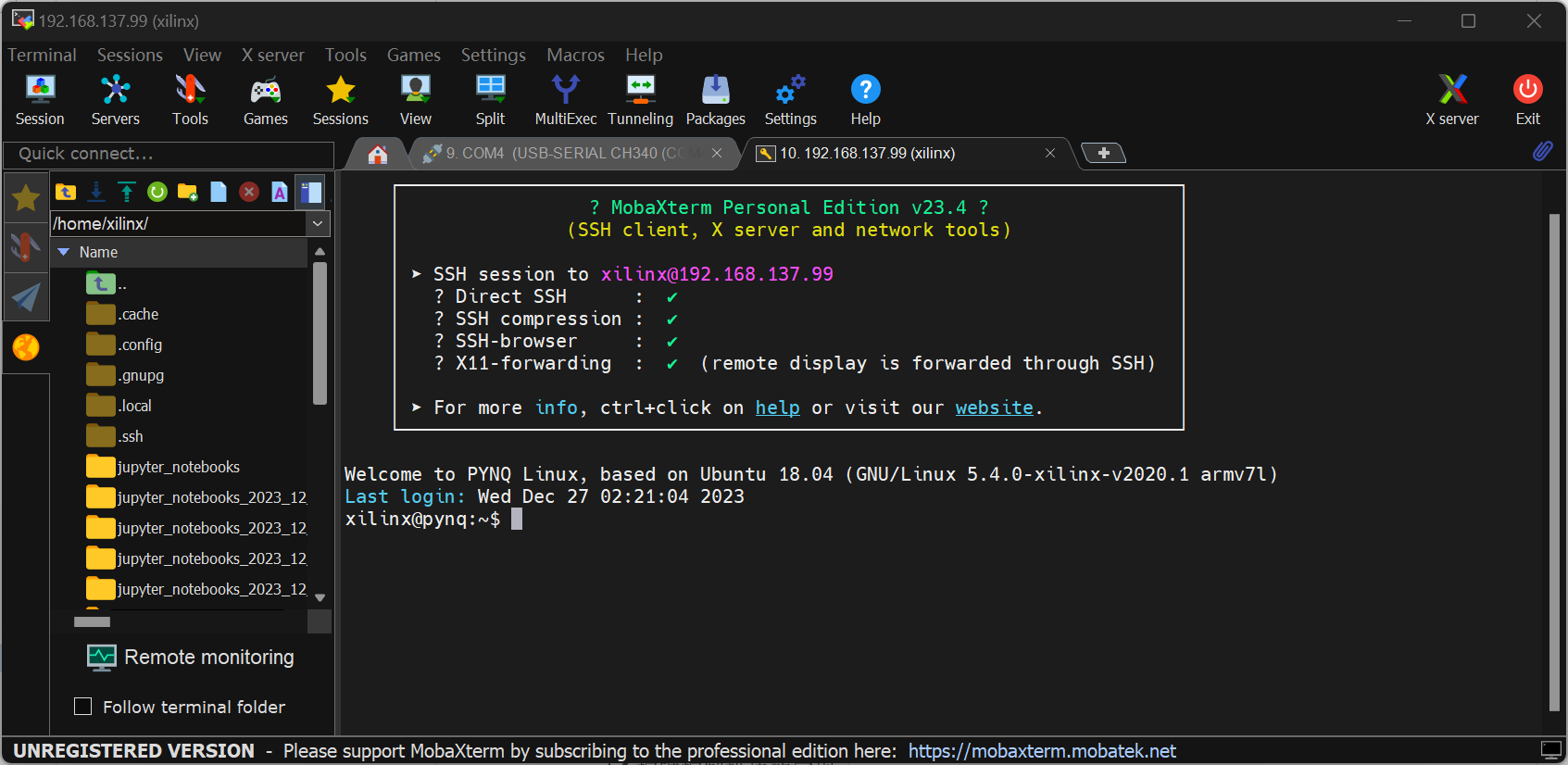
使用TF卡读卡器，在Windows操作系统下，进入磁盘管理，对TF卡进行格式化之后，即可使用Win32DiskImager工具将镜像烧入TF卡。

#### 操作流程

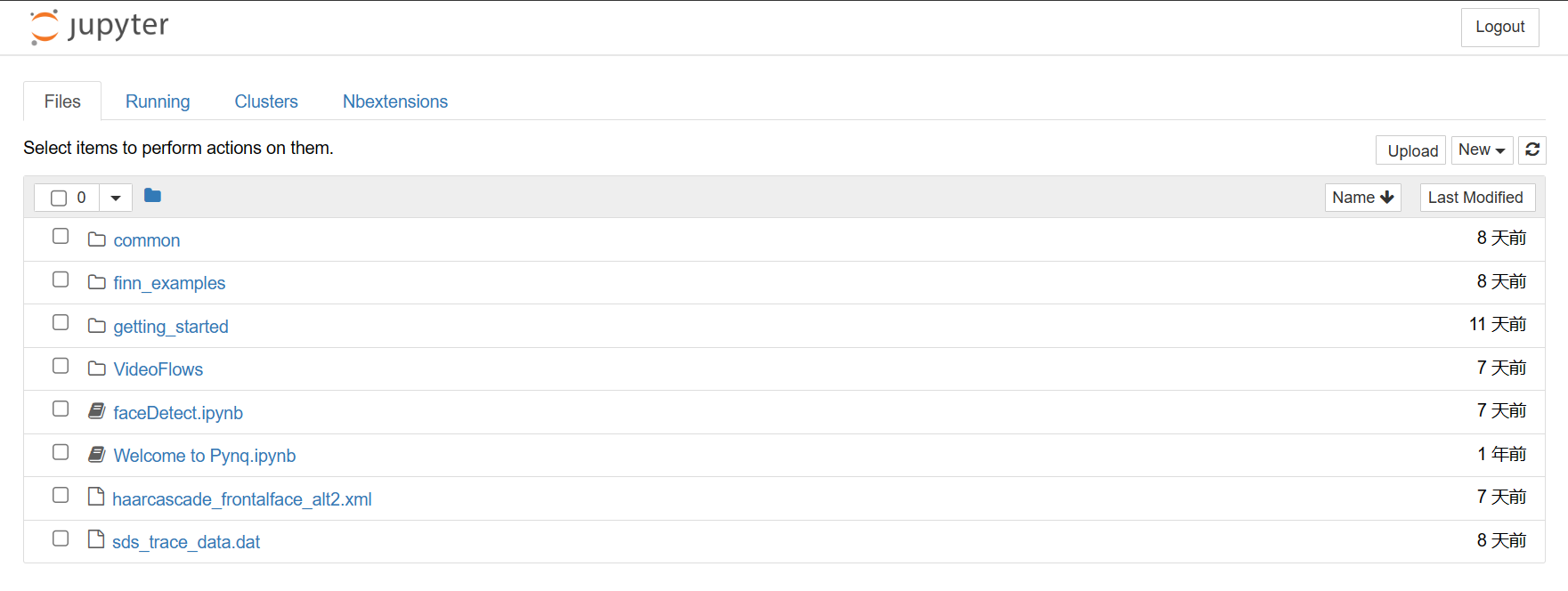
将TF卡插入开发板的TF1卡槽里，设置BOOT为TF卡启动。开发板实物图如下所示。



关闭Windows防火墙，打开MobaXterm，使用网口调试登录。



打开主机浏览器，输入网址<http://pynq>，或输入开发板的IP地址，在出现的界面中输入开发板系统登录密码，即可进入jupyter notebook开发页面，进行python代码的编写，如下图所示。如果Python代码需要使用Overlay库进行灯的控制和按键的读取，则还需要在.py文件同一文件夹下放置.hwh、.bit、.tcl文件。注意需要将.bit文件重命名至文件名和另外两个文件名字相同。



### 外部接口

编译的镜像相当于在开发板上搭建了一个软件平台。可以通过Overlay库，通过AXI总线，控制板上的外设。

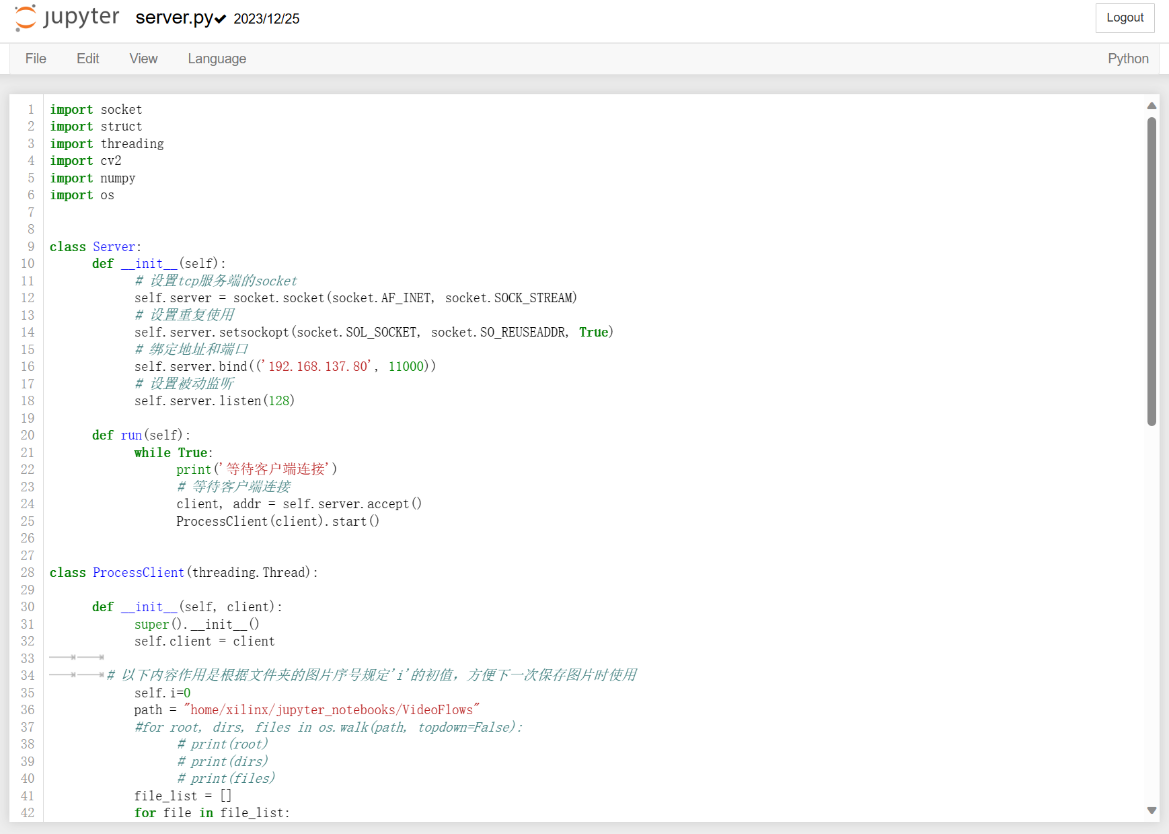
### 组成框图/工作流程

## socket通信模块

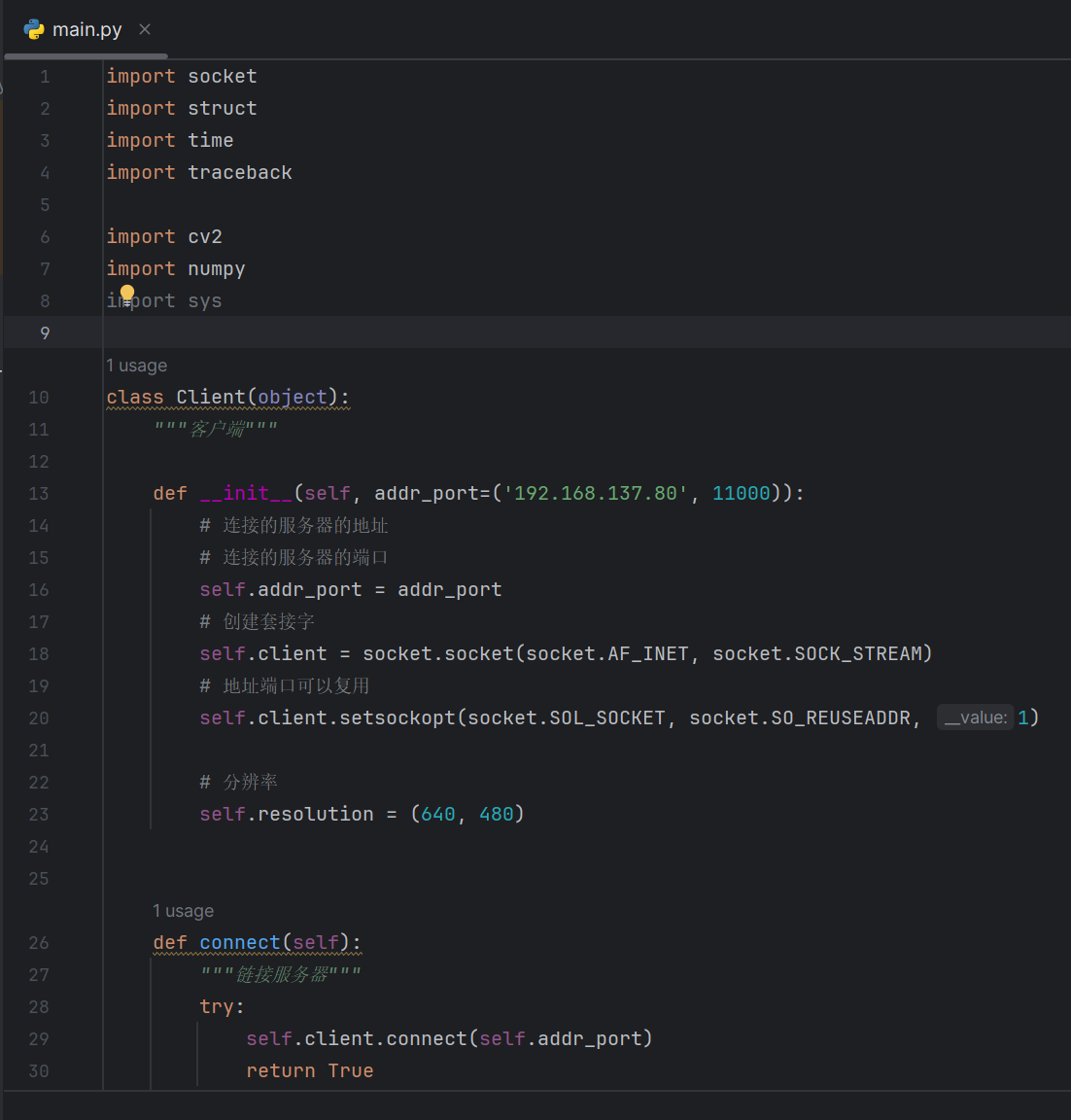
### 功能

实现主机与ZYNQ\_MINI开发板之间的数据传输，包含摄像头输入的实时视频流、人脸检测结果实时视频流和开发板按键控制信号等。实现主机和开发板之间的socket通信过程如下：

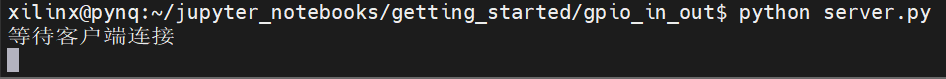
（1）使用PYNQ提供的jupyter notebook进行服务器端代码的编写并存放到开发板上，如下图所示。



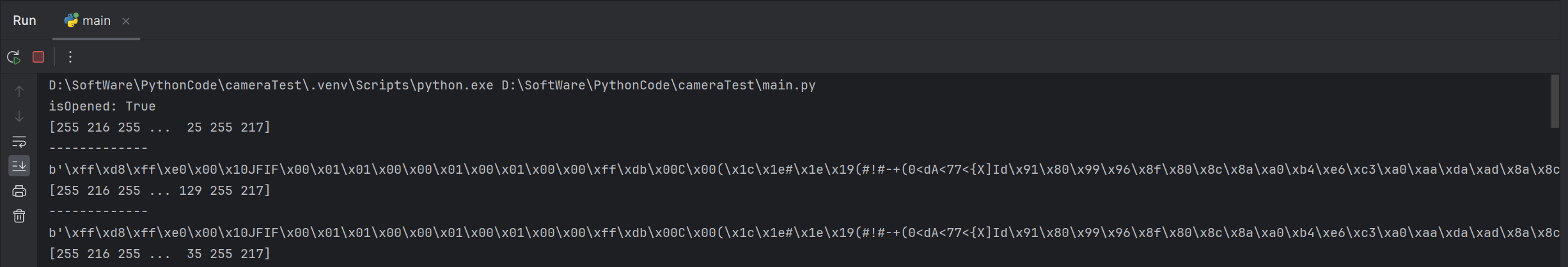
（2）在主机上编写客户端代码，如下图所示。



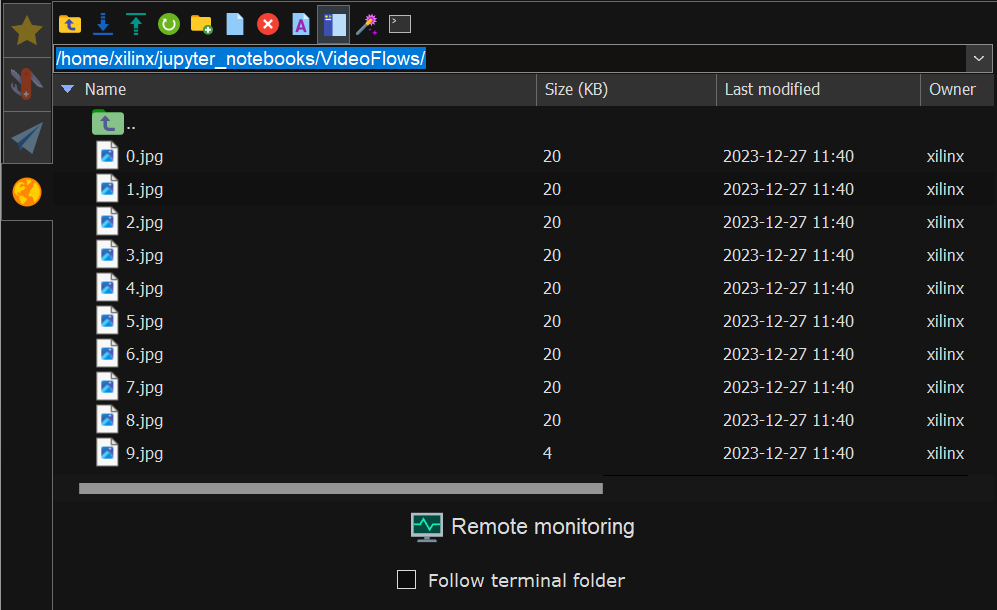
1. 在MobaXterm终端上运行服务器端代码，如下图所示。



（4）在主机上运行客户端代码，如下图所示。



（5）在MobaXterm上打开视频流在开发板上的存放目录，可以看到正实时生成每帧图像，如下所示。



### 工作流程

Socket通信模块的工作流程如下图所示。

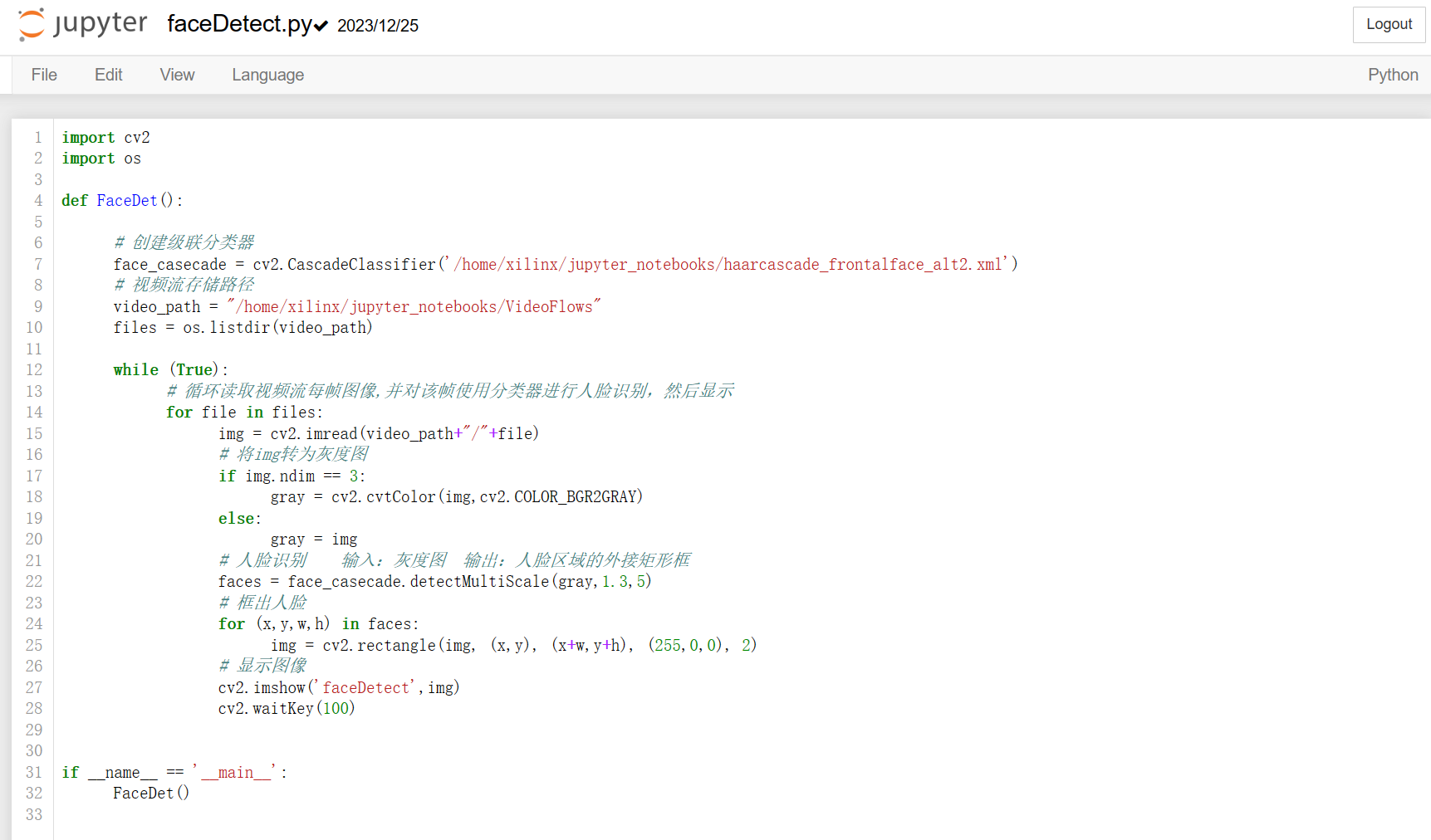


工作流程图

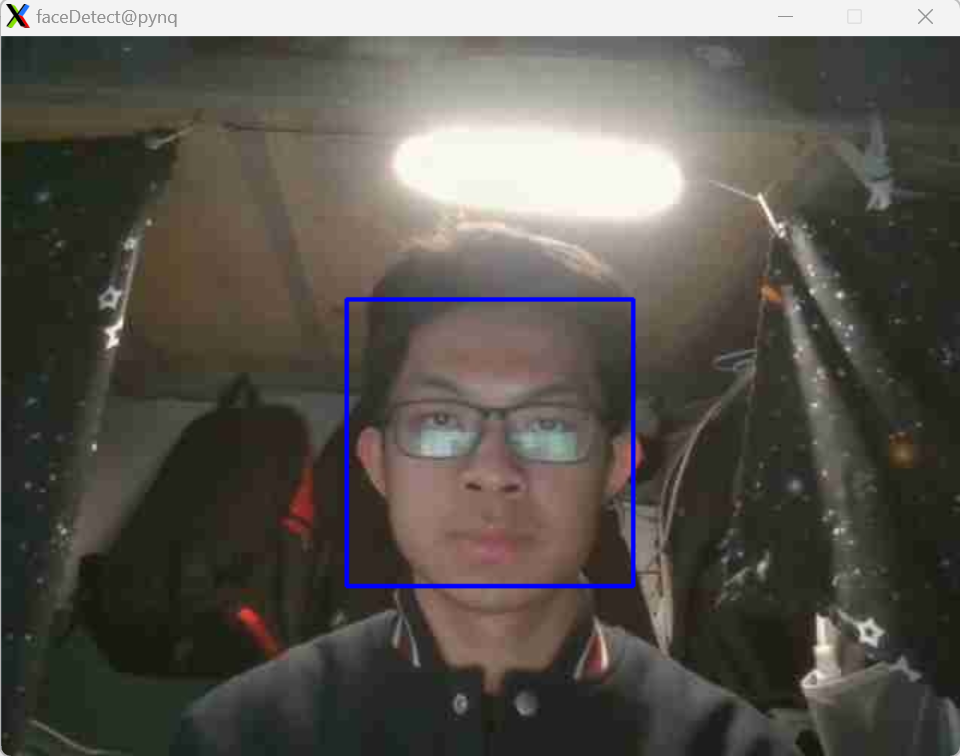
## 人脸检测模块

### 功能

按帧对输入的实时视频流进行循环读取，并逐帧使用分类器进行人脸识别，输出框出人脸的识别结果。实现过程如下：

使用PYNQ提供的jupyter notebook进行人脸检测代码编写，如下图

1. 检测效果如下图所示。



### 工作流程

人脸检测模块的工作流程如下图所示。



工作流程图

## 按键控制模块

### 功能

由control.py文件和key.py文件实现。在人脸识别程序工作过程中，使用流水灯表明正在工作。需要停止工作时，持续按下按键1秒，即可终止人脸识别程序工作，同时灯将全部熄灭。

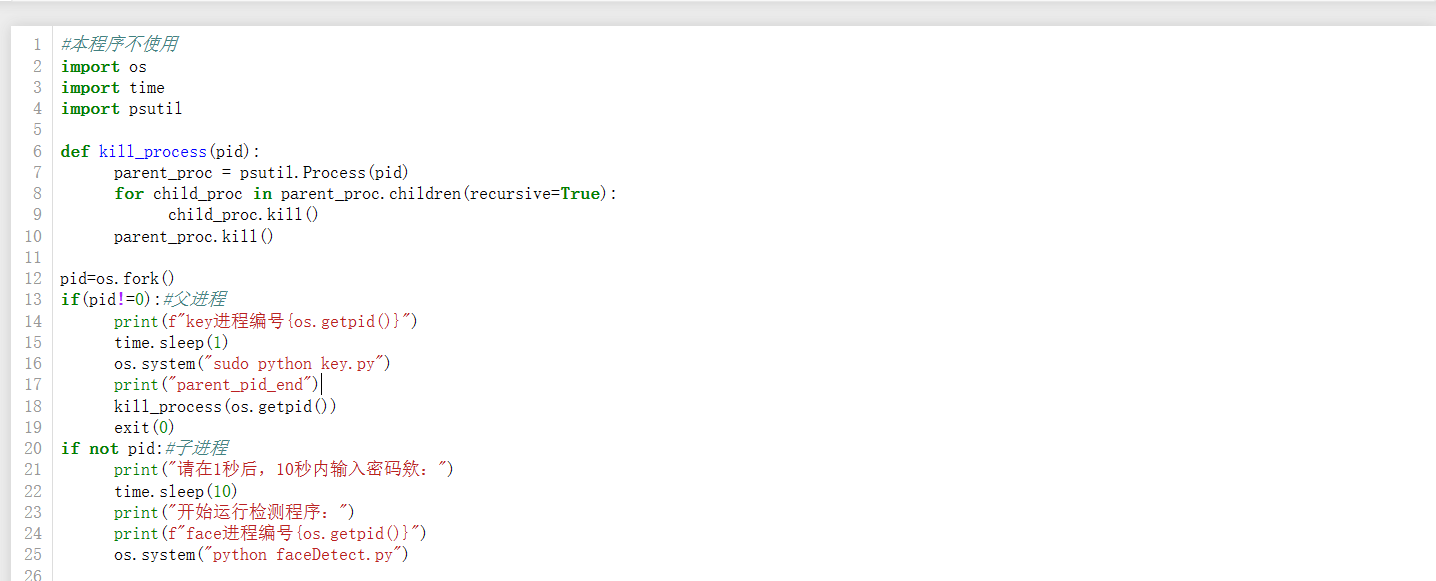


图 1 control.py

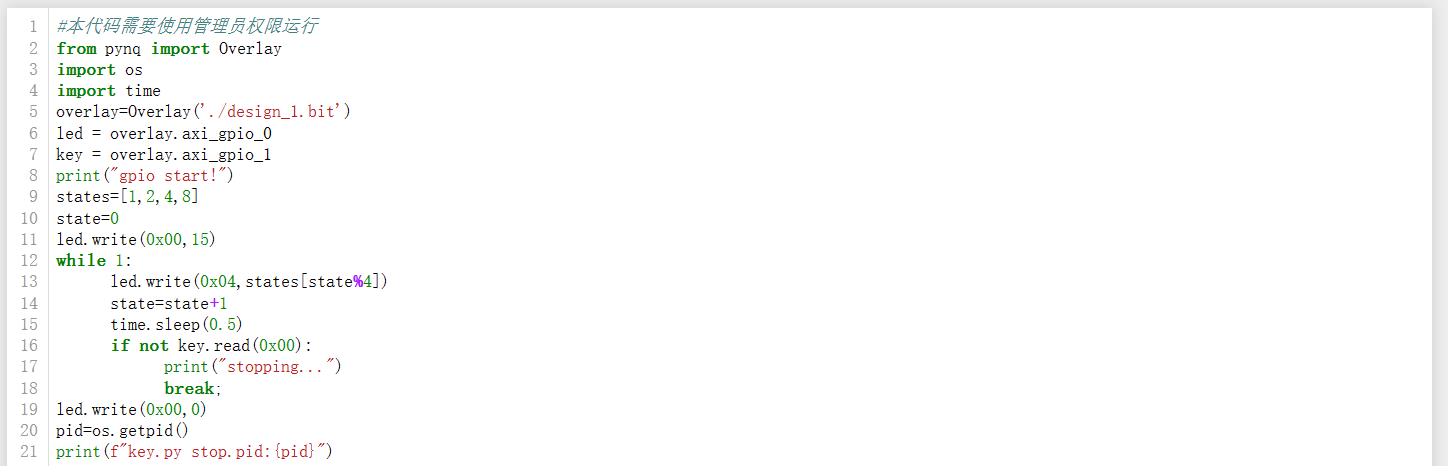


图 2 key.py

control.py文件运行将分为两个进程。父进程将执行key.py，控制流水灯，以及检测按键是否按下。子进程运行按键检测程序。

按键按下后，key.py结束运行，随后control.py的父进程会杀死本进程及所有的子进程，包括人脸检测模块所在的进程。

考虑到控制流水灯和检测按键状态需要使用管理员权限，人脸检测模块不能使用管理员权限运行（这和MobaXterm相关）。因此需要使用非管理员权限运行control.py。在control.py中使用非管理员权限运行人脸检测模块，同时使用管理员权限运行key.py。

由于Jupyter Notebook运行程序自带使用管理员权限，因此在编码时均不采用.ipynb格式编写，而是使用.py并用命令行运行。

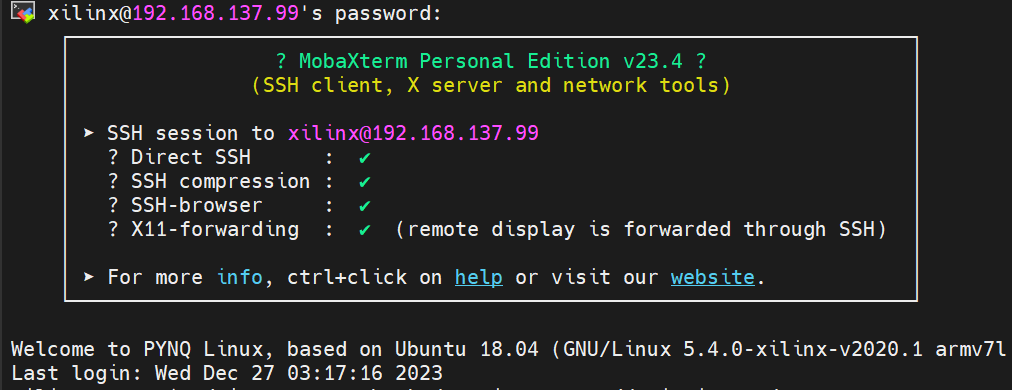
### 工作流程

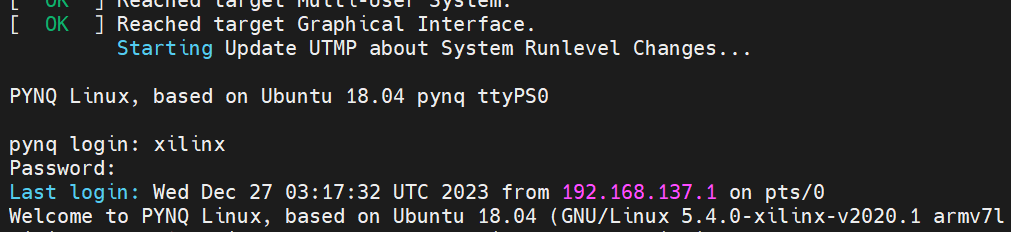


图 3 按键控制模块流程图

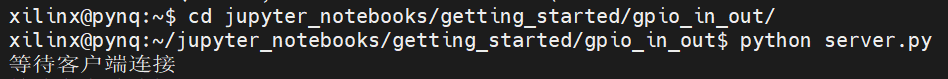
# 系统联调与测试

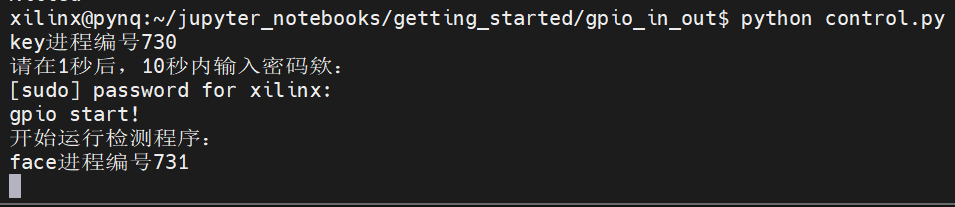
1. 使用MobaXterm同时使用网口和串口连接开发板，如下图所示。



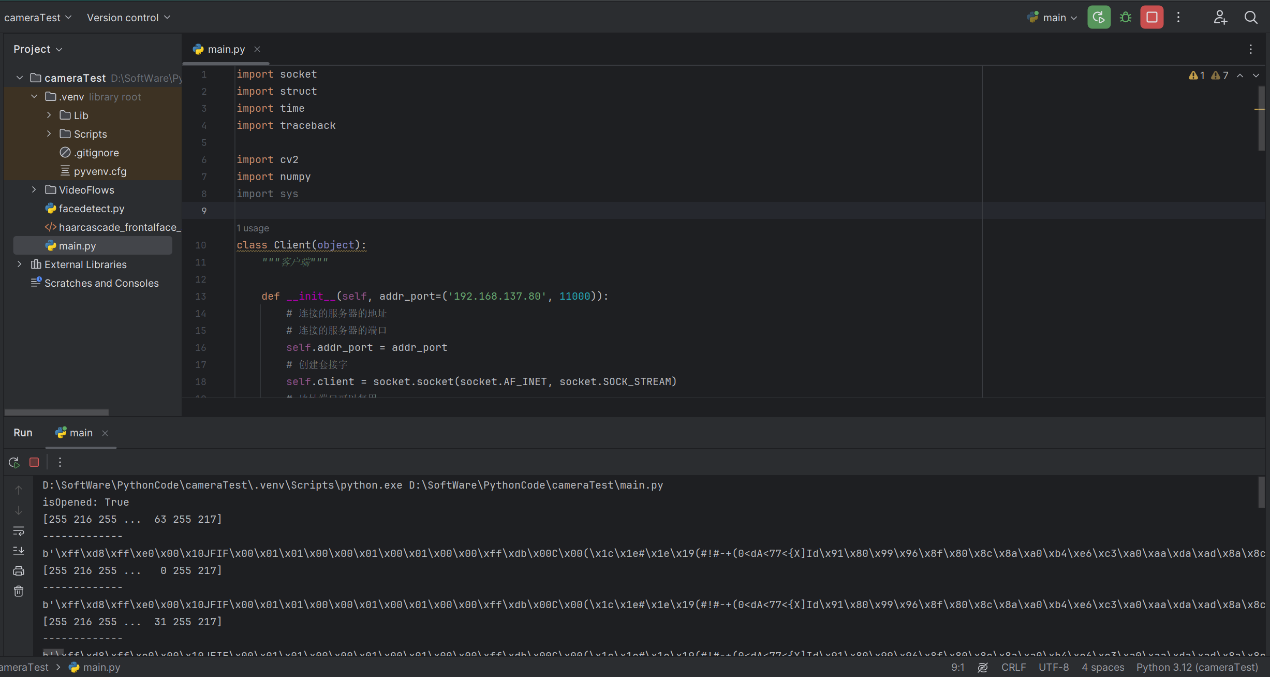


2、使用串口终端和网口终端分别运行server.py和control.py，如下图所示。

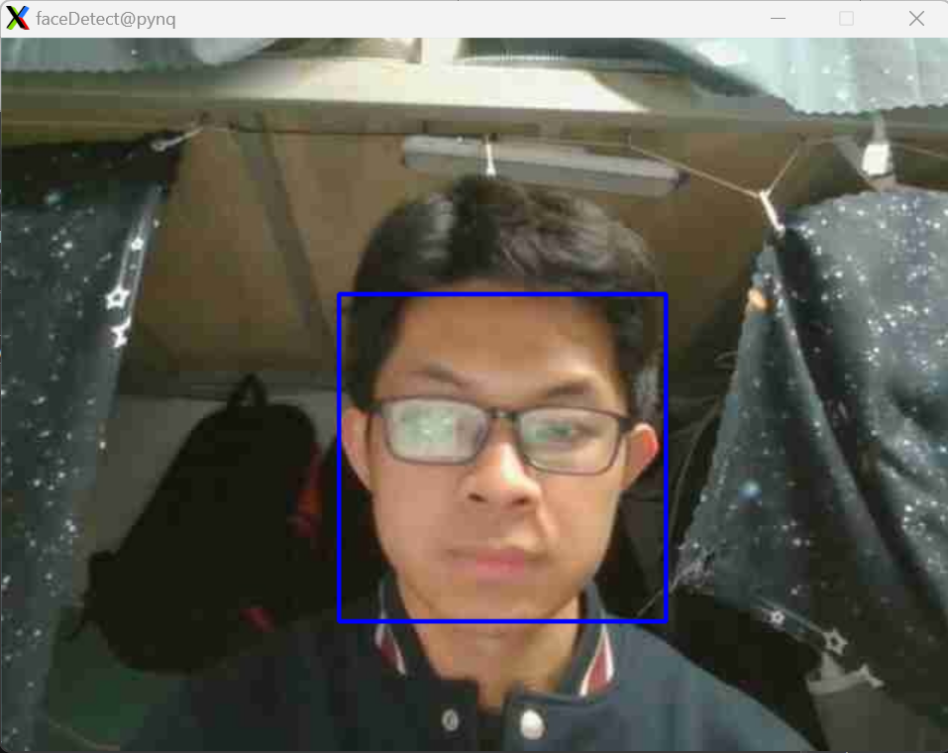


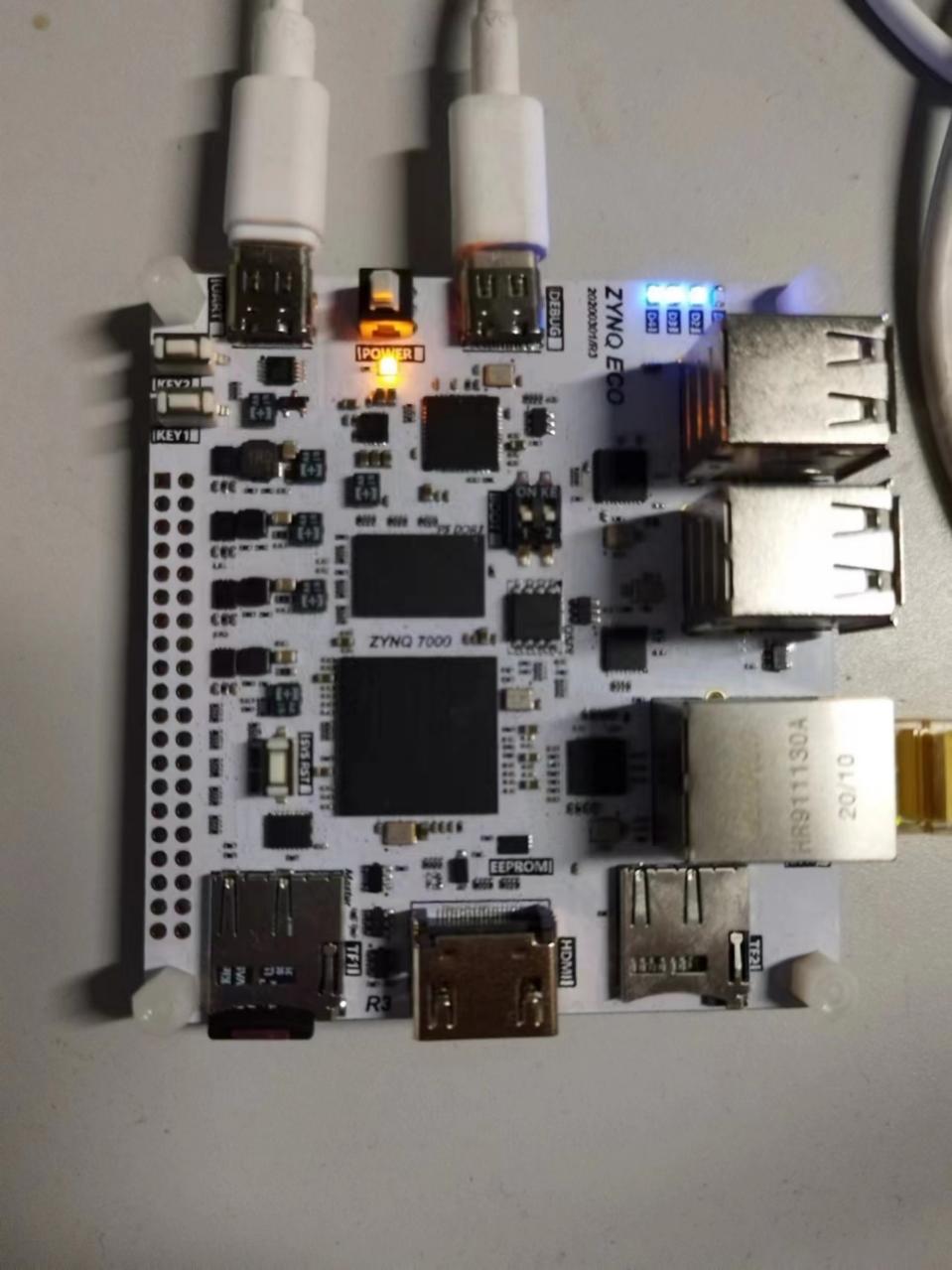


3、打开主机上的客户端代码并运行，如下图所示。

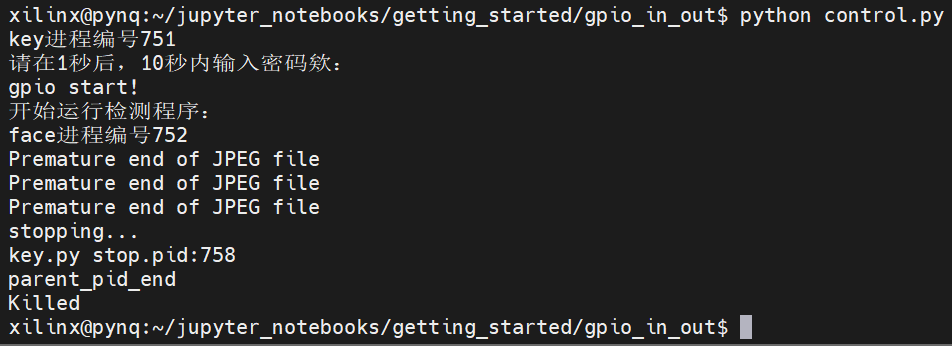


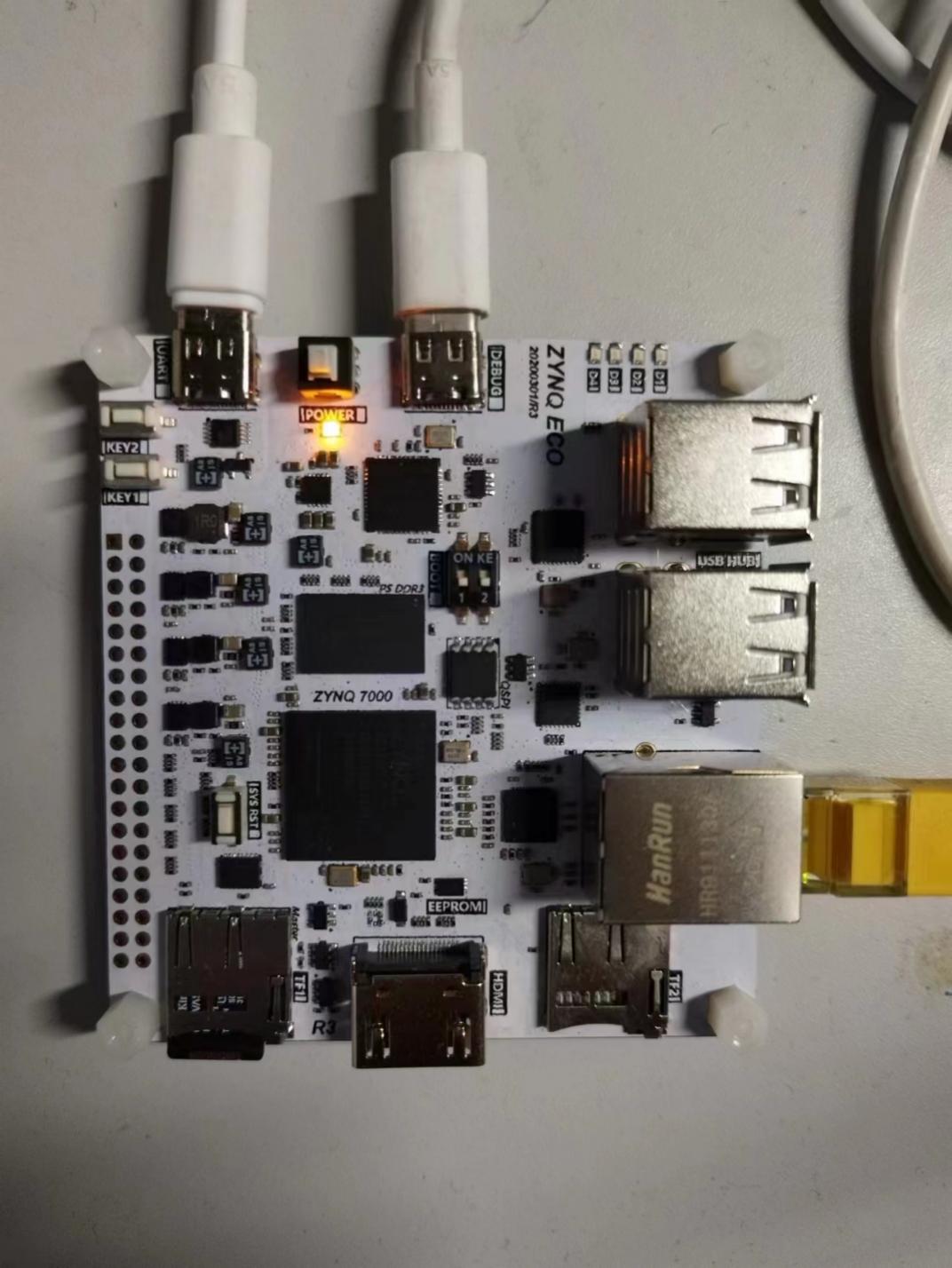
4、检测窗口显示人脸检测结果，开发板上的LED为流水效果，如下图所示。





5、按下开发板的key1键，人脸检测程序关闭，LED灯全部熄灭，如下图所示。





# 遇到的问题及解决方案

|  |  |  |
| --- | --- | --- |
|  | 问题 | 解决方法 |
| **1** | 镜像编译失败 | 使用arm版本的镜像、国际联网、设置不用密码登录root用户、将操作系统的语言换成英文、将gitsm换成https |
| **2** | 视频流传输时延偏大 | 调整cv2.imencode()参数，设置合理的图片质量参数，平衡时延和图片质量 |
| **3** | 显示检测图片失败 | 原因在于MobaXterm和管理员权限的冲突。需要用普通用户身份运行人脸检测模块。 |
| **4** | LinuxMint下有线连接开发板导致无法无线连接外网 | 使用route系列命令，更改路由表，调高有线连接Metric字段数值，降低有线连接优先级。  参考：Linux笔记：Ubuntu18.04有线网络和无线网络的优先级设置\_ifmetric-CSDN博客 |
| **5** | LinuxMint下开发板无法通过个人电脑连接外网 | 在网络连接设置中，对于有线与开发板组网方面，选择与其它计算机共享（无线联网不需要更改） |
| **6** | 对开发板设置DNS 时/etc/resolv.conf被覆盖的问题 | 在/etc/systemd/resolved.conf修改DNS。  参考：Ubuntu 20.04 设置 DNS 的方法 - 刘应杰 - 博客园 (cnblogs.com) |

# 总结与分析

本实验主要目的为熟悉镜像制作与使用PYNQ框架进行嵌入式软硬件的控制与开发。使用主机摄像头捕获实时视频流，通过socket通信传输到ZYNQ\_MINI开发板上，将板上人脸检测模型的输出视频流再回传主机进行窗口显示，同时控制当模型运行时LED灯流水，按下开发板key1键可以关闭模型，并使LED灯全部熄灭。

本实验难点主要在为ZYNQ\_MINI开发板定制PYNQ镜像以及PYNQ框架中Overlay库的使用。在使用Vivado配置硬件文件时，需要清楚ZYNQ\_MINI开发板的管脚分布、硬件配置和硬件参数。使用Overlay库控制开发板按键，需要清楚Overlay库的内容，以及熟悉python多进程编程。

# 项目分工

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PPT与实验报告 | 镜像制作 | Python开发 | 外设控制 |
| 黄自锴 | √ | √ | √ | √ |
| 林子川 | √ | √ | √ | √ |
| 常海博 | √ |  |  |  |

# 参考文献

1. [PYNQ - Python productivity for Zynq - Community](http://www.pynq.io/community.html)
2. [Overlay Tutorial — Python productivity for Zynq (Pynq)](https://pynq.readthedocs.io/en/v2.5.1/overlay_design_methodology/overlay_tutorial.html)
3. [自定义单板创建PYNQ镜像V2.6\_米联客 pynq-CSDN博客](https://blog.csdn.net/clj609/article/details/114438810)
4. [计算机视觉——利用openCV与Socket结合进行远程摄像头实时视频传输并保存图片数据\_opencv连接远程摄像头-CSDN博客](https://blog.csdn.net/weixin_47460769/article/details/116300354)
5. [OpenCV——通过摄像头捕获实时视频并探测人脸、准备人脸数据\_imooc opencv-CSDN博客](https://blog.csdn.net/cyj972628089/article/details/113354720)
6. [AXI 总线详解-CSDN博客](https://blog.csdn.net/weixin_49617016/article/details/108398026)
7. <https://www.bilibili.com/video/BV1jF411c7cu/>
8. [PYNQ SD Card image — Python productivity for Zynq (Pynq)](https://pynq.readthedocs.io/en/latest/pynq_sd_card.html#pynq-sd-card)
9. [那些年我们定制PYNQ需要避免的坑 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/256365015)
10. [Linux笔记：Ubuntu18.04有线网络和无线网络的优先级设置\_ifmetric-CSDN博客](https://blog.csdn.net/weixin_44444810/article/details/124768090)
11. [Ubuntu 20.04 设置 DNS 的方法 - 刘应杰 - 博客园 (cnblogs.com)](https://www.cnblogs.com/mouseleo/p/14976527.html)