**基于树莓派的红外安防**

**监控装置**

**入职作业项目报告**

**团队成员：董培方、杜光磊、黄晶晶**

**团队名称：三号路**

# 1．项目功能：

摄像头自动监控拍照。通过三个90度夹角安装的红外感应模块检测人体，舵机控制摄像头在180度范围内扫描监控，当红外模块感应到人体时，舵机转动到对应位置，摄像头拍照保存。通过QT界面进行可视化控制，具备视频监控、拍照、自动/手动模式切换、调节扫描速度、监控历史照片等功能。

# 2．使用到的基本知识

C/C++、git、PWM、舵机控制、solidwork三维建模、红外感应原理、Linux指令操作、Linux下开发环境搭建、OpenCV2.4.9、Qt编程、多线程编程、定时器

# 3.git仓库地址：

项目地址：[https://github.com/Lei0126/ No.3\_Road.git](https://github.com/Lei0126/DJI_No.3_Road.git)

# 4.项目分工：

董培方：主要负责零部件设计，硬件搭建以及传感器控制。

杜光磊：主要负责树莓派上openCV+QT开发环境搭建、QT编程

黄晶晶： Qt界面编写，整个系统的软件优化

# 5.开发流程：

1）根据小组讨论的方案选择红外传感器和舵机。采用180度的数字舵机作为摄像头的转动云台，传感器选择人体红外感应模块，在0度、90度、180度三个方向上对称安装，人体感应的距离小于7m。

2）根据传感器的安装位置和树莓派的尺寸设计了连接件，采用solidworks进行三维建模，然后用实验室的3d打印机打印出来，零件如图1、2所示：

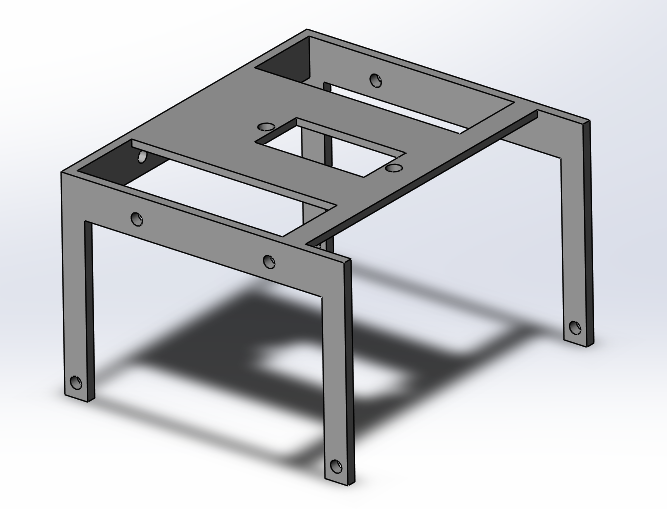


图1. 传感器安装架

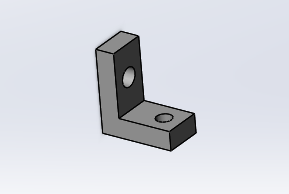


图2. 树莓派连接件

3）选择树莓派gpio26口作为舵机的控制口，gpio27、gpio28、gpio29作为红外感应模块的信号输入口，采用蛇形管摄像头，利用以前设计的支架进行安装，摄像头支架和舵机臂之间采用502胶水进行固定，实物图如图3所示。



图3. 树莓派监控设备实物

4）设计底层软件，主要包括舵机转角的控制和红外感应模块信号处理，采用WiringPi开源库进行开发，根据舵机的控制方式，设置PWM控制周期20ms，高电平0.5~2.5ms对应舵机0~180度转角。通过读取红外感应模块的信号输出来判断对应位置是否检测到人。

5）针对整个系统功能，设计了手动和自动控制模式：手动即通过输入具体角度来控制舵机旋转。自动模式分为自动扫描和静止两个状态：在自动扫描状态，没检测到人的情况舵机带动摄像头在0~180度范围内，以设定的速度来回旋转。一旦红外模块检测到人，则控制舵机转动到对应的位置。静止状态下只有传感器检测到人之后才转动到对应位置。

6）在树莓派上Enable树莓派摄像头，安装摄像头相关的驱动，利用MJPG\_streamer监控软件通过8080端口进入软件界面，观测摄像头工作情况，如图4所示。摄像头可以正常工作。

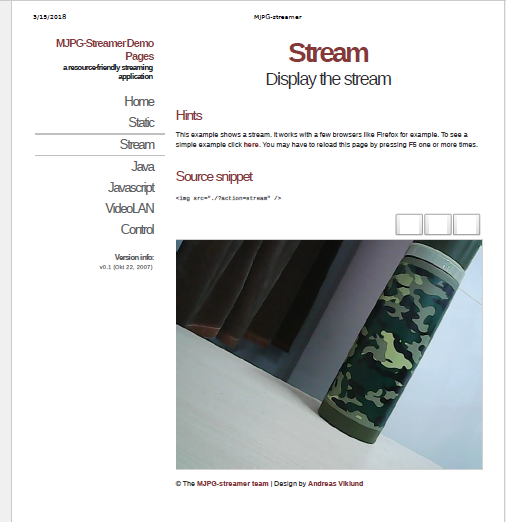


图4. MJPG\_streamer监控界面

7）在树莓派上安装Qt5.7，配置Qt编译器。配置OpenCV安装环境，下载并解压OpenCV，利用cmake生成的Makefile文件对OpenCV编译。编译完成后，安装OpenCV。修改QT的.pro文件，增加QT对OpenCV库的支持。

8）设计应用层软件。主要通过调用OpenCV库实现图像信息的处理。通过设置定时器的初始值，定时读取摄像头信息，并将每一帧数据图片显示在label上。设计拍照功能，可以自动或手动将拍摄照片显示在label\_2上，并以“Monitor+时间”的命名格式，jpg的图片格式存储在默认的路径下。增加了图片另存为，历史照片查看的功能。利用Designer工具，设计显示视频监控和拍照图片的两个label控件和摄像头打开、关闭、拍照、保存图片的button控件，并利用connect函数将这些控件的信号和槽函数连接起来。完成以上功能的界面如图5所示。

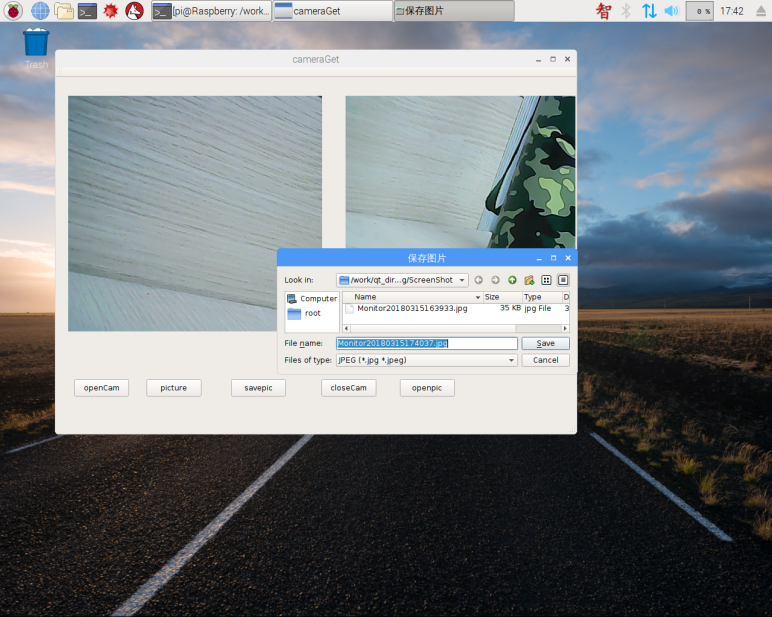


图5.视频监控&拍照界面

9）将编写的舵机控制&红外感应的程序sensor.c添加到Qt工程中，使得程序中的函数可以被Qt程序调用。利用Designer工具调加自动/手动切换按钮，周期运转功能按钮，红外检测功能按钮。添加两个vertical Slider分别用于调整手动模式下舵机转动的角度和自动模式下舵机扫描的速度。增加一个spinBox用于显示手动模式下舵机的转角（0°~ 180°）。所设计的界面如图6所示。通过界面直接为这些控件添加槽函数。通过Qt程序编写，将红外监控和舵机自动扫描，手动调节的功能加入到原有视频监控的程序中。初步实现了利用设计界面与监控装置进行人机交互的效果。

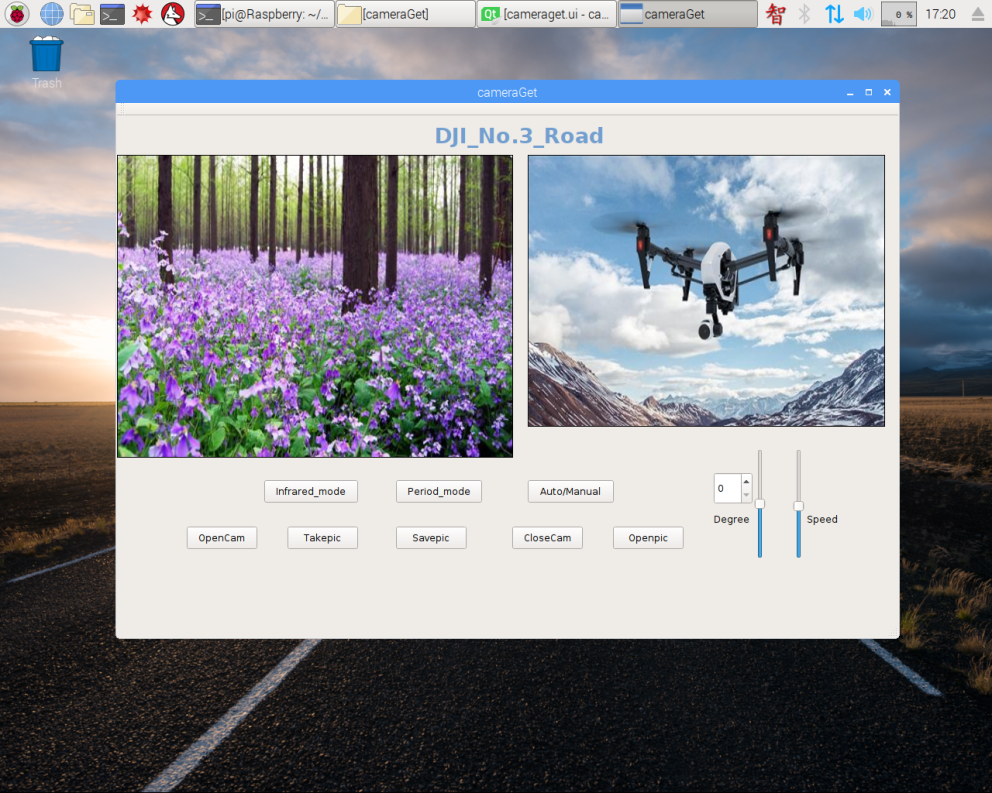


图6.人机交互界面

10）依据设计要求对前文设计的Qt界面以及Qt程序作进一步优化。优化后的界面布局更加合理，且可以将多张历史照片实时显示。

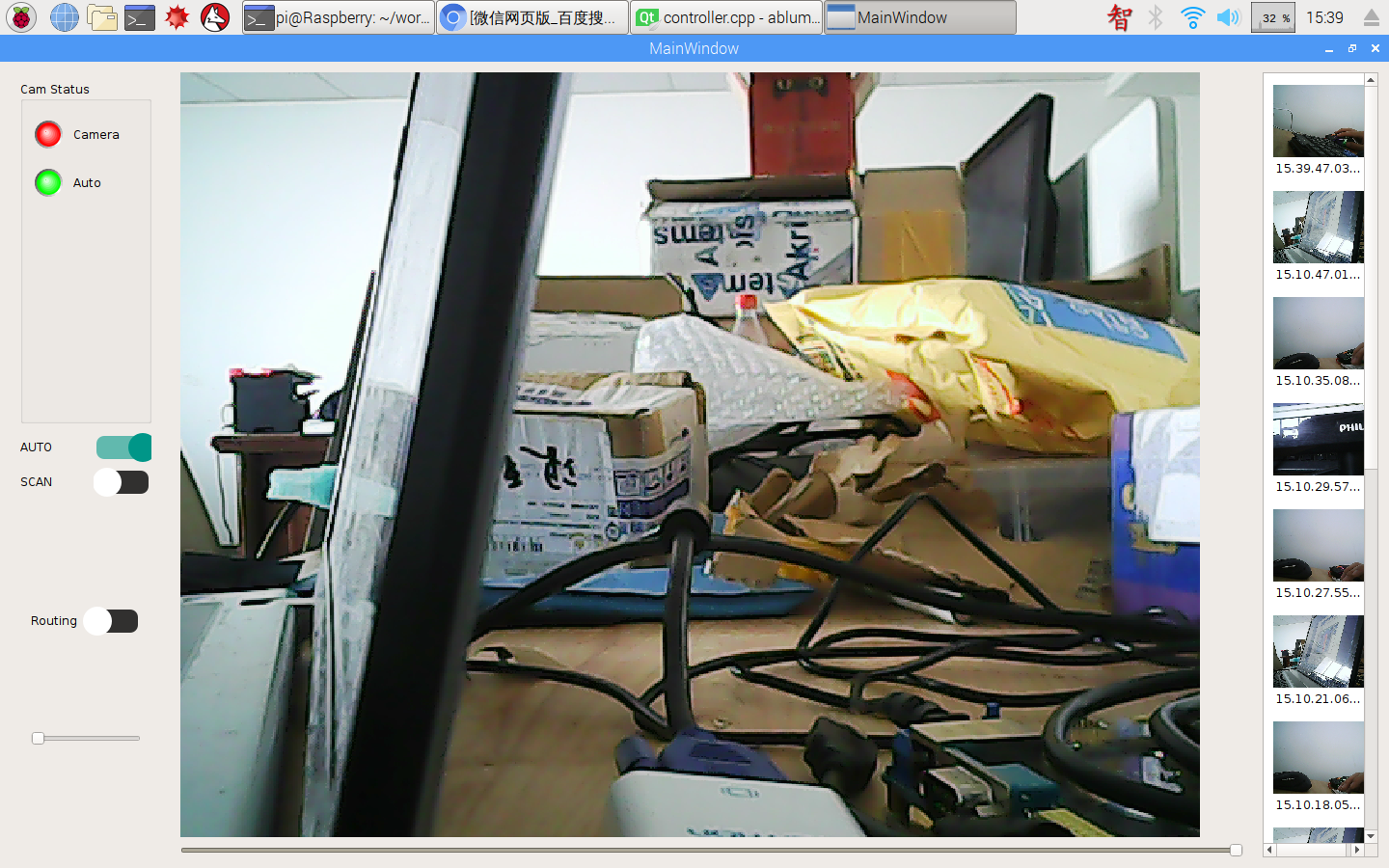


图7.优化后的界面

图中：左边一列为监控摄像头的状态显示、监控的自动/手动切换、历史浏览开关，中间为摄像头图像的实时显示，其下滑块用于手动控制摄像头的转向，而右边一栏则是历史图片的略缩图，方便实时浏览。

11）采用多线程对之前的程序进行优化，将整个程序分为三个线程：界面显示线程、摄像头拍摄线程以及舵机与红外的控制线程，并进行分别实现。

# 6.遇到的问题及解决方法：

1）由于红外感应模块安装位置较为接近，且模块的识别角度较大，所以经常会出现互相干扰，误识别的情况出现。解决方法：设计挡板将传感器隔开。

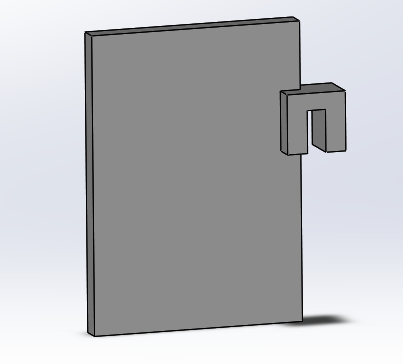


图8.挡板三维图

2）当控制引脚给舵机发送PWM信号后，此时摄像头立刻捕捉图像信息，电机还在转动且未到预定位置，导致图像模糊不清。

解决办法：在周期函数最后增加延时函数delay(50);

3）当红外探测到人体后，舵机带动摄像头转到预定位置，拍照后将捕获图片显示在label\_2上，然而图片显示总是大大滞后于监控的画面。

解决办法：将通过拍照自动保存的默认格式由png改为所占内存更小的jpg格式,大大缩短了捕获图片在label\_2上显示的时间。

4）舵机总是会不断抖动，即使PWM的占空比没有改变。

解决办法：该情况是由于控制舵机的PWM通过软件产生，而由软件产生的PWM占空比不稳定，因此改用硬件方式产生PWM，问题解决。

5）Qt程序响应较慢，模式切换以及拍摄照片时总是会存在一定程度的滞后。

解决办法：由于摄像头线程以及运动控制线程均由界面线程中的定时器进行触发、执行，导致摄像头线程和运动控制线程出现堵塞而导致线程滞后，因而将属于每个线程的定时器封装入自己的线程中，通过每个线程的started()信号触发每个线程的定时器从而使线程运作，问题解决。