基于 FPGA 的智能安全门系统

张杨豪 吴潇文 覃叶婷

第一部分 设计概述

1.1 设计目的

本作品针对现有安检系统缺乏对人体温度采用快速有效监测措施的问题，提出一种基于 FPGA 的智能安全门系统。

如今，新型冠状病毒在全球肆虐，世界各国都对重要场所、重大活动加强了人员体温安全检查。为了抑制病毒大规模的传播扩散，及时排查过往人员体温情况，实现人员抓拍以及温度记录以供信息查询，同时，结合高精度红外测温传感器、超声波测距传感器、触摸显示屏、报警电路等外围设备进行设计，并且采用 WIFI 模组与安卓手机 PC 端通信，完成数据的无线传输，便于记录和处理。

1.2 应用领域

常规安检系统一般设置在铁路、地铁、政府部门等地区，主要目的是排查出行人员是否携带危险物品，防恐防暴。本作品设计的智能安全门系统可运用于常规安检场所，也可以应用于学校、小区、商场、医院等人流较大的地点，针对出入人员进行体温检查，以及用于设备远程温度监控记录，实时监测人员体温有无异常情况。

1.3 主要技术特点

本作品的主要功能包括: 身份识别功能、测温报警拦截功能以及客户端查询记录功能等。

测温报警拦截功能通过 FPGA 与超声波进行串口通信,不断实时读取距离信息判断是否有人靠近，当有人靠近时 FPGA 驱动红外测温模块，进行数据处理后显示人体温度信息，若人体体温异常则驱动电机关闭舵机门禁止通行;

客户端记录查询功能分为两部分，一是上位机异常抓拍记录功能，可以实时记录所有通过人员的体温信息并可绘制图表和查询异常者体温的信息和当时拍摄的脸部照片，二是安卓手机客户端功能，IC 射频卡识别通过的人员身份信息并将体温信息上传到云端平台。

1.4 关键性能指标

（1）超声波测距最远距离为 720CM，分辨率可达 1cm;

（2）无线测温模块测量温度范围为人体温度 32-42.5 摄氏度，物体温度为 0-80 摄氏度。

（3）RGB 灯条报警闪烁频率为 2HZ。

第二部分　系统组成及功能说明

2.1 整体介绍

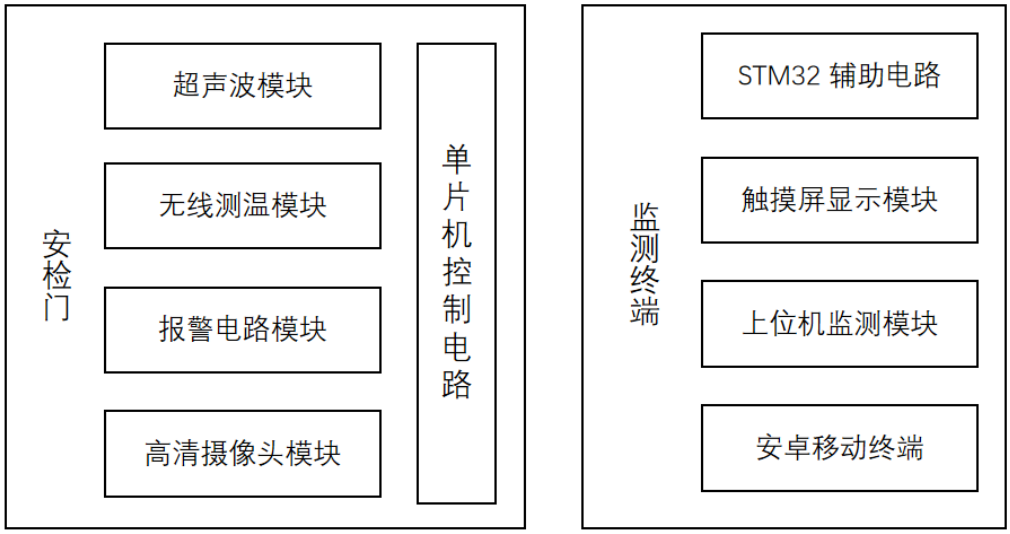

图 2.1 整体介绍

本系统硬件结构分为安检门和监测终端两部分，其中，安检门由单片机控制电路、超声波模块、无线测温模块、声光报警模块、高清摄像头模块组成；监测终端由 STM32 辅助电路、触摸屏显示模块、上位机监测模块、安卓移动终端组成。

首先安检门通过超声波模块检测是否有人靠近，当有人靠近时无线测温模块开始测温，若温度异常，则声光报警模块开始报警、高清摄像头模块拍摄人像图片。监测模块可将距离、温度、等参数显示在触摸显示屏模块上，安卓移动终端可查询温度记录。

2.2 各模块介绍

2.2.1 超声波模块

本系统超声波采用 GY-US42 测距传感器模块，功耗低、体积小、安装方便，其工作原理为，探头发射超声波，照射到被测物体或人体后，探头接收返回声波，利用时间差，计算出实际距离，从而得知是否有人体靠近。

2.2.2 无线测温模块

采用 FST-01 红外温度传感器，FST-01 模块采用高精度传感器和独特的光学结构，带有温度补偿，稳定性较好，可以通过串口完成相关功能。

2.2.3 高清摄像头模块

高清摄像头采用罗技 C270 高清网络摄像头，分辨率为 720P*1280，照片拍摄约 500 万像素。当无线测温模块测得温度高于正常范围时，PC 端上位机控制摄像头捕捉温度异常画面并保存。

2.2.4 报警模块

报警模块由 RGB 灯条和 LED 驱动组成，当超声波测距模块监测到人时报警模块亮绿灯，当无线测温模块测出人体体温超过额定温度时，报警模块亮红灯并快速闪速以警示。

2.2.5 STM32 辅助电路

STM32 辅助电路主要控制触摸显示屏模块、舵机控制门模块、WIFI 模块，触摸屏将温度以及人体是否正常等信息显示在触摸屏上，并且实现手动的舵机门的开关控制。STM32 与安卓手机端进行 WiFi 通信，讲人体体温传送到安卓手机端上查看数据，实现更加方便的温度查询与记录方法。

第三部分　完成情况及性能参数

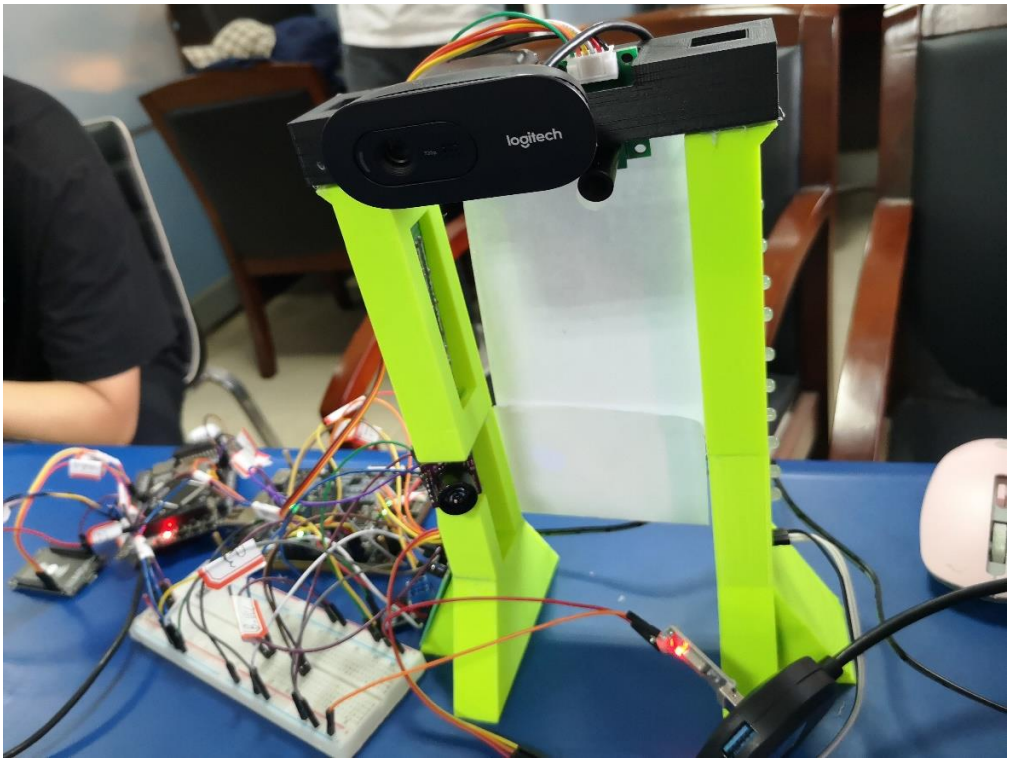

图 3 作品全貌

3.1　3D 打印框架

由 AutoCAD 自主设计并 3D 打印出来安检门的整体框架，并且预留无线测温模块、超声波模块、高清摄像头模块、报警电路等安装位置。填充密度为 100%，硬件整体框架抗压能力较强。

3.2　超声波测距

超声波用于测量距离来判断是否有人靠近，FPGA 通过超声波模块来判断是否开启 RGB 灯条亮灭。

3.3　无线测温

当超声波检测到有人靠近时无线测温模块开始测温，无人时测温模块不工作，以减少无人时的消耗。无线测温模块可测人体温度范围为 32-42.5 摄氏度，可测物体温度为 0-80 摄氏度，FPGA 根据测得温度判断 RGB 灯条对应的状态。

3.4 高清摄像头以及上位机监测模块

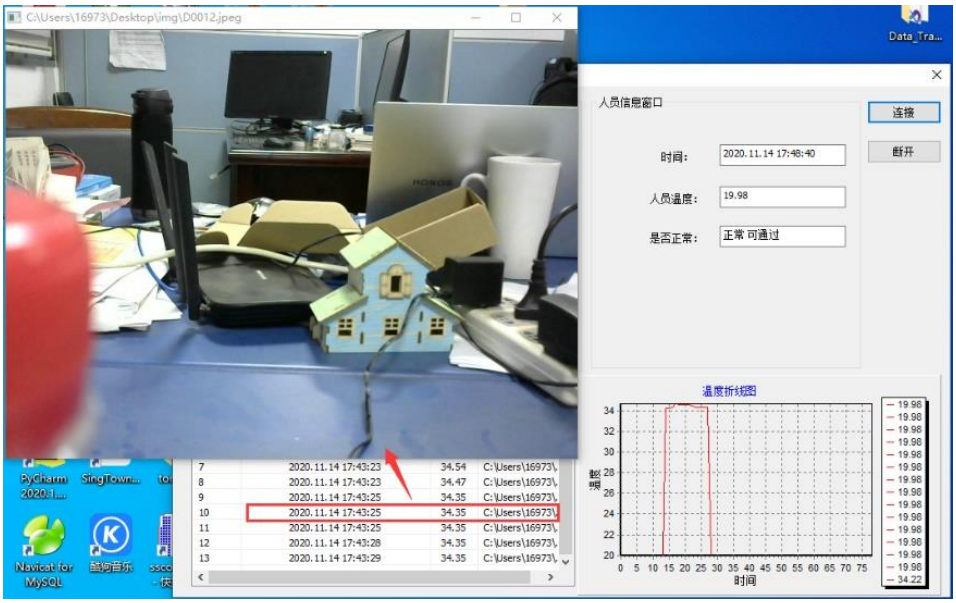

图 3.4 PC 端上位机

高清摄像头连接 PC 端上位机，上位机显示高清摄像头实时拍摄到的画面，并且 FPGA 检测到温度异常时，将温度异常数据的温度、时间、存储位置等信息用列表显示出来，可以通过点击查看某一时刻异常温度的捕捉画面，并且可以画出所测温度的实时波形。

3.5 安卓客户端

图 3.5 安卓客户端

安卓客户端通过 WiFi 接收 STM32 辅助电路传过来的温度和人员状态等信息，并且可以实现查询显示等功能。

第四部分　总结

截止目前，以及基本上完成本基于 FPGA 的智能安全门系统，可实现的基本功能有:

（1） 当人靠近时，两排报警灯亮绿灯并保持常亮，无人靠近时报警灯熄灭，当人体温度高于 37 摄氏度时两排报警灯亮红灯并且快速闪烁。

（2） PC 端上位机显示高清摄像头实时画面，记录异常人员时间、温度、捕捉图片、存储位置等信息，并且能将所测温度画出折线图。

（3） 安卓客户端通过 WiFi 通信接收 STM32 辅助电路传过来的温度、人员状态等信息，存储并显示。

4.1 主要创新点

（1） 数据通过 WIFI 模组上传到安卓手机端。

（2） 安检门的安检系统增加舵机控制门，更加安全。

4.2 可扩展之处

用 BP 神经网络实现人脸识别算法，将通过安检门的人员识别、检测、记录，通过 PC 端和安卓客户端进行监测与查询。

4.3 心得体会

通过这次大赛，我们感受到了 FPGA 的魅力，以及 FPGA 在我们生活中的广泛应用，同时，我们也在这次比赛当中看到了当前 FPGA 厂商的优势，看到了 FPGA 自身的优势所在。通过这次大赛，我们更加了解了 FPGA 板卡的相关模块特性，并且在本智能安全门系统中应用以前所学的安卓、超声波、无线测温等模块进行综合，对项目综合有了更进一步的经验学习。在制作本项目的过程中我们遇到的包括 3D 打印问题、传感器的使用问题、FPGA 板卡的学习等问题让我们团队每个人都收获良多，并且团队协作也有了极大的提高，每个人的工程实践能力都有了很大的进步。

第五部分　参考文献

[1] 许连望，陈冲．易燃易爆油气体的智能安检系统[J]．闽江学院学报,2019,40(05):41-48.

[2]李敏.无线红外测温系统的设计[J].电子测试,2020(19):27-28+13.

[3]沈毅,董浩明. 基于 ARM 单片机的人身安检装置的设计[A]. 湖北机械工程学会机械设计与传动专业委员会.武汉机械设计与传动学会第20届学术年会论文集[C].湖北机械工程学会机械设计与传动专业委员会:武汉机械设计与传动学会,2012:4.

第六部分　附录

重要代码、推导过程等不便于在正文中体现的内容

Dis.to.pc.v

```
module  dis_to_pc
(
    input           clk,
    input           rst_n,
    input   [7:0]   dis_byte_pc,
    input           send_en_pc,
    input   [2:0]   baud_set_pc,

    output  reg dis_tx,
    output  reg tx_done,
    output  reg uart_state
);

reg bps_clk;//波特率时钟
reg [15:0] BPS_DR;//分频计数最大值
reg [15:0] div_cnt; //分频计数器
reg [3:0]  bps_cnt;//波特率计数时钟
reg [7:0]  r_data_byte;

//reg [22:0] ns_cnt;

localparam START_BIT = 1'b0;
localparam END_BIT   = 1'b1;



always  @(posedge clk)
begin
    if(!rst_n)
        uart_state  <=  1'b0;
    else if(send_en_pc)
        uart_state  <=  1'b1;
    else if(tx_done)
        uart_state  <=  1'b0;
    else
        uart_state  <=  uart_state;
end


always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
```

```
        r_data_byte <=  1'b0;
    else if(send_en_pc)
        r_data_byte <=  dis_byte_pc;
    else
        r_data_byte <=  r_data_byte;
end

always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        BPS_DR <=  15'd5207;
    else
    begin
        case(baud_set_pc)
            0:BPS_DR <=  15'd5207;//9600
            1:BPS_DR <=  15'd2603;//19200
            2:BPS_DR <=  15'd1301;//38400
            3:BPS_DR <=  15'd867;//57600
            4:BPS_DR <=  15'd433;//115200
            default:BPS_DR <=  15'd5207;
        endcase
    end
end


always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        div_cnt <=  15'd0;
    else if(uart_state)
    begin
        if(div_cnt==BPS_DR)
            div_cnt <=  15'd0;
        else
            div_cnt <=  div_cnt + 1'b1;
    end
    else
        div_cnt <=  15'd0;
end


//always @(posedge clk or negedge rst_n)
//begin
//    if(!rst_n)
```

```
//        ns_cnt <=  23'd0;
//    else if(ns_cnt==23'd4)
//    begin
//        ns_cnt <=  23'd0;
//        send_en_pc <=  ~send_en_pc;
//    end
//    else
//        ns_cnt <=  ns_cnt + 1'b1;
//end


always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        bps_clk <=  1'b0;
    else if(div_cnt==1'b1)
        bps_clk <=  1'b1;
    else
        bps_clk <=  1'b0;
end

always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        bps_cnt <=  4'd0;
    else if(tx_done)
        bps_cnt <=  4'd0;
    else if(bps_clk)
        bps_cnt <=  bps_cnt + 1'b1;
    else
        bps_cnt <=  bps_cnt;
end

always @(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        tx_done <=  1'b0;
    else if(bps_cnt==4'd11)
        tx_done <=  1'b1;
    else
        tx_done <=  1'b0;
end

always @(posedge clk or negedge rst_n)
```

```
begin
     if(!rst_n)
        dis_tx <=  1'b1;
     else /*if(r_data_byte!=0 && r_data_byte!=8'h0d)*/
        begin
            case(bps_cnt)
                0:dis_tx <=  1'b1;
                1:dis_tx <=  START_BIT;
                2:dis_tx <=  r_data_byte[0];
                3:dis_tx <=  r_data_byte[1];
                4:dis_tx <=  r_data_byte[2];
                5:dis_tx <=  r_data_byte[3];
                6:dis_tx <=  r_data_byte[4];
                7:dis_tx <=  r_data_byte[5];
                8:dis_tx <=  r_data_byte[6];
                9:dis_tx <=  r_data_byte[7];
                10:dis_tx <= END_BIT;
                default:dis_tx <=  1'b1;
            endcase
        end
end

endmodule
```