第一届“麦格米特”

科技创新大赛

作品设计报告

题目：基于NBIOT的智能库房安全系统

参赛成员：姜浩 陈益潇 吉宇航

四川大学麦格米特科技创新设计大赛

姓　　名：姜浩 陈益潇 吉宇航

学　　院：电子信息学院

年　　级：2018 级

学　　号：2018141451127

手　　机：13159687605

邮　　箱：392320832@qq.com

2020 年 12 月 19 日

一、项目概述

现阶段，多数库房安全问题主要依仗于物理安全防范手段和人防，例如使用牢固的建筑结构，门窗防护以及安防人员的巡查等方式。这样的方式不仅会耗费大量的人力物力，同时容易出现死角，造成安防系统的漏洞，从而留下安全隐患。基于这样的背景，我们希望设计一套基于 NB-IoT 的智能库房安全系统，通过多个传感器，对库房周界，出入口，窗户及内部进行全方面防护，并利用 NB-IoT 将数据发送到云端实现远程监控。异常情况下通过短信和邮件及时上报。

基于上述特点，我们设计实现了以下的系统：

我们本次安防系统分为三大部分，首先带来我们的智能门禁系统，该门禁系统可显示此时的日期与时间。我们可以通过指纹开门，开门时电机会转动，此时手机也会收到开门者的信息。我们也可以通过刷卡开门，此时手机会收到开门的方式。我们也可以通过密码开门，手机同样会在开门后收到开门提醒。开门后，我们可以进行录入指纹，修改密码，修改时间等一系列调节操作。

其后是我们的智能传感器系统，此系统包括温度传感器、微动传感器以及超声波传感器。温度传感器实时采集库房的温度，并上传至云端，当温度超过阈值，就会触发高温预警。微动传感器可装放在窗户上，若窗户被异常打开，传感器被触发，此时手机端就会收到异常短信，此时我们的默认处于休眠后续指令就会被触发，我们此时可利用手机远程对设备下命令，在这里我们体现为点亮小灯。同理，若检测到非法人员活动，超声波传感器被触发，手机同样会收到报警短信，我们此时也可以远程点亮另一个小灯。

最后是我们的摄像头智能检测系统，为了便于识别合法人员在库房内的轨迹，我们实际识别的是标签的轨迹，但这在现实中也是合理的，例如我们要求每一个合法进入的成员都必须佩戴标签。我们可以在本地通过屏幕观察摄像头识别到的图像，识别到的轨迹我们会实时显示在串口屏上，而一旦轨迹进入我们的禁区，在这里我们设置的是屏幕中间，我们的手机端就会检测到异常路线的短信。

我们整套的设备具有极强的可移植性，且 NBIOT 云端与硬件是完全独立的，例如，我们可以将门禁的 NBIOT 拆下，与人体传感器相连接，实现了人体传感器的设备上云，此时我们设置人体传感器可与摄像头联动使用，因为摄像头会存在摄像死角，而在此状态下摄像头与人体传感器只要有一个被触发，就会发出异常路线的短信，这进一步增强了系统的准确性。

二、理论分析

在此板块中，我们主要对我们使用的各模块进行简介。

1. 智能门禁系统：

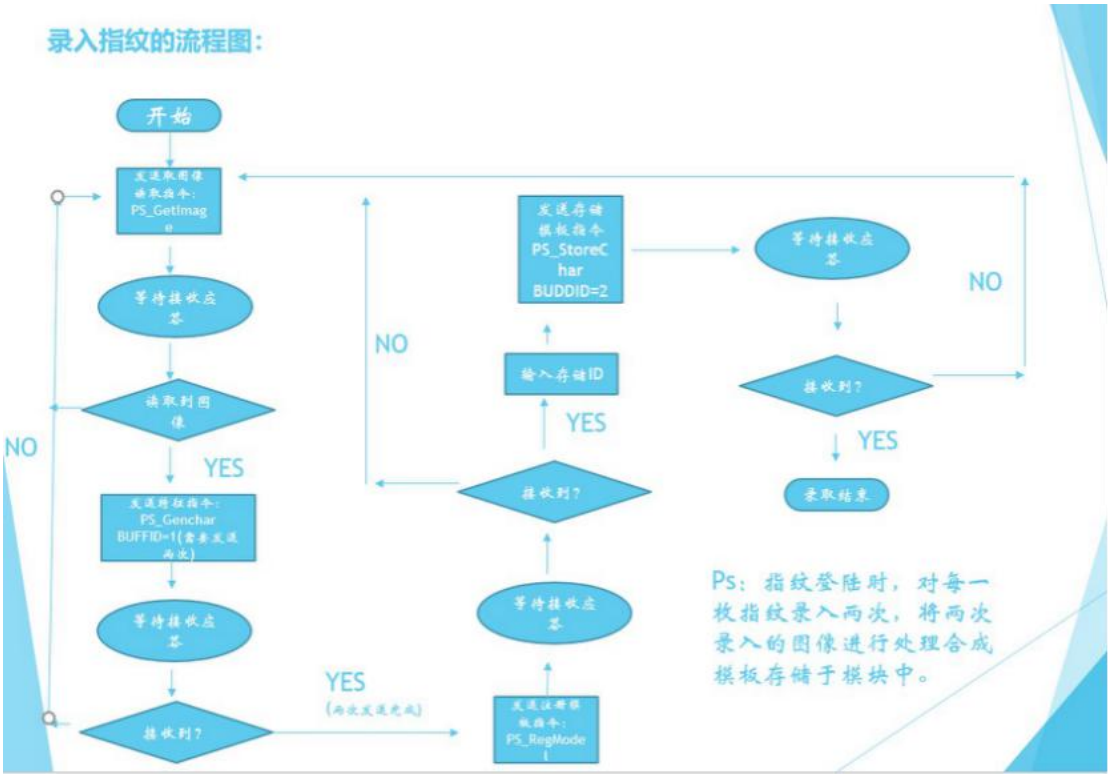
（1）AS608

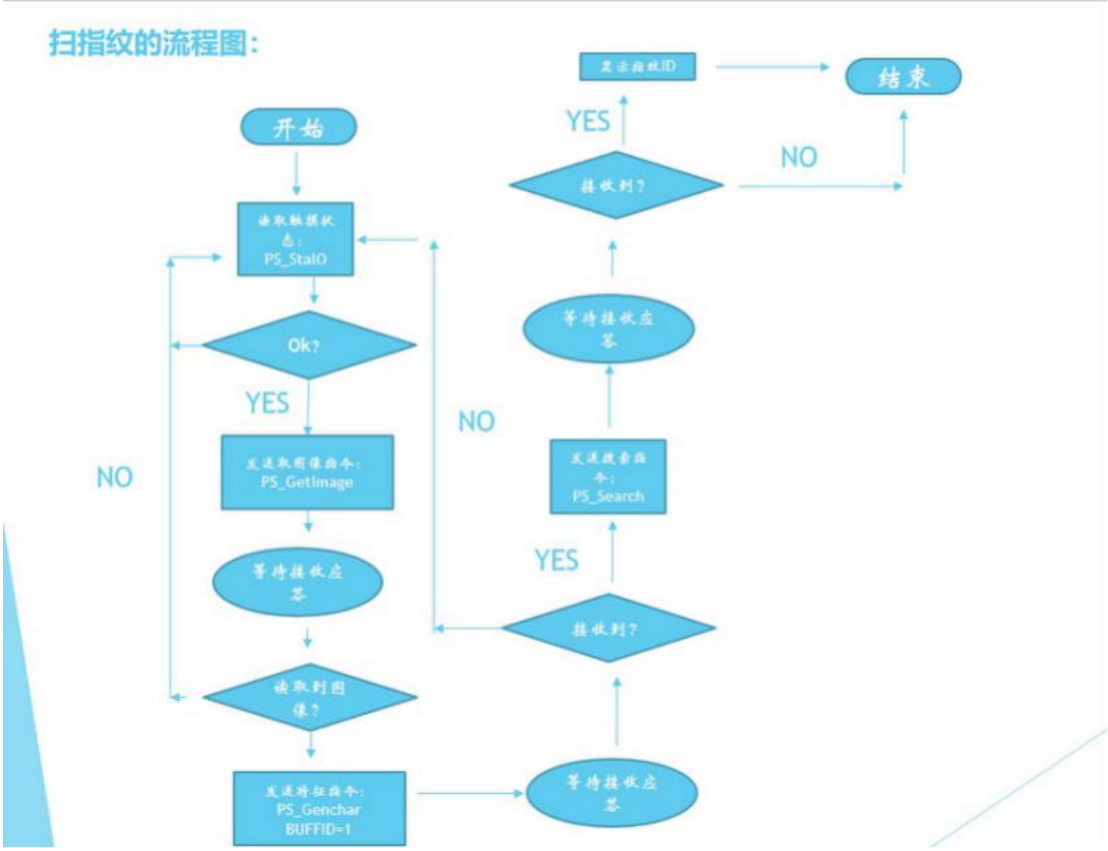
AS608 指纹识别模块主要是指采用了杭州晟元芯片技术有限公司(Synochip)的 AS608 指纹识别芯片而做成的指纹模块，集成一个可供 2 次开发的指纹模块；只要是基于 AS608 芯片的指纹模块，其控制电路及控制协议几乎是一样的。

具体原理则是利用光的折射和反射原理,光从底部射向三棱镜,并经棱镜射出,射出的光线在手指表面指纹凹凸不平的线纹上折射的角度及反射回去的光线明暗就会不一样,光学器件就会收集到不同明暗程度的图片信息,就完成指纹的采集。

AS608 的实物图如下图所示：

流程如下图所示：

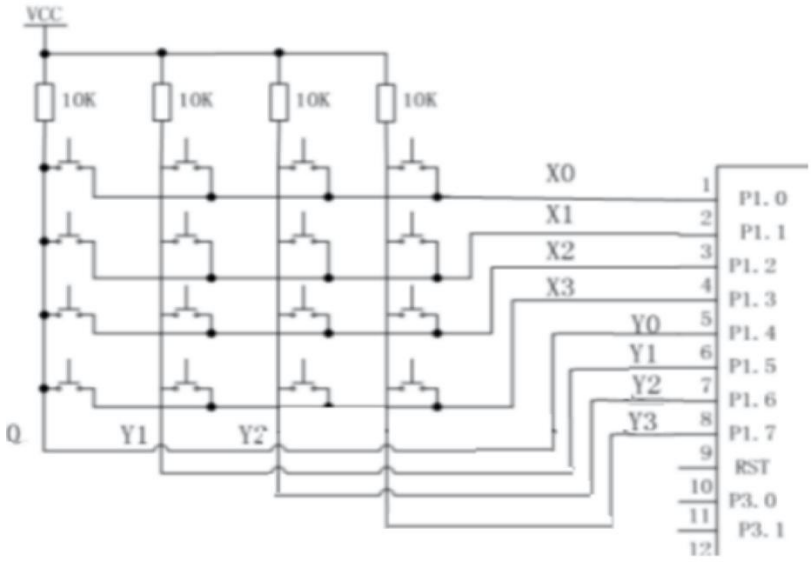




(2)矩阵键盘

采用行列扫描的原理，即通过高四位全部输出低电平，低四位输出高电平。当接收到的数据，低四位不全为高电平时，说明有按键按下，然后通过接收的数据值，判断是哪一列有按键按下，然后再反过来，高四位输出高电平，低四位输出低电平，然后根据接收到的高四位的值判断是那一行有按键按下，这样就能够确定是哪一个按键按下了。

以下图为例：



通过检测 X0~X3 是否输入为 0，检测是否有键按下，同时检测 Y0~Y3 中是否有 0 输入，根据两点交于一点，确定是哪个点按下。

(3)RC522

非接触式读写卡芯片，最大通信速率 13.56MHz；

支持接口： UART， SPI(Speed_max=10Mbit/s)， IIC(快速:400Kbit/s， 高速:3400Kbit/s)

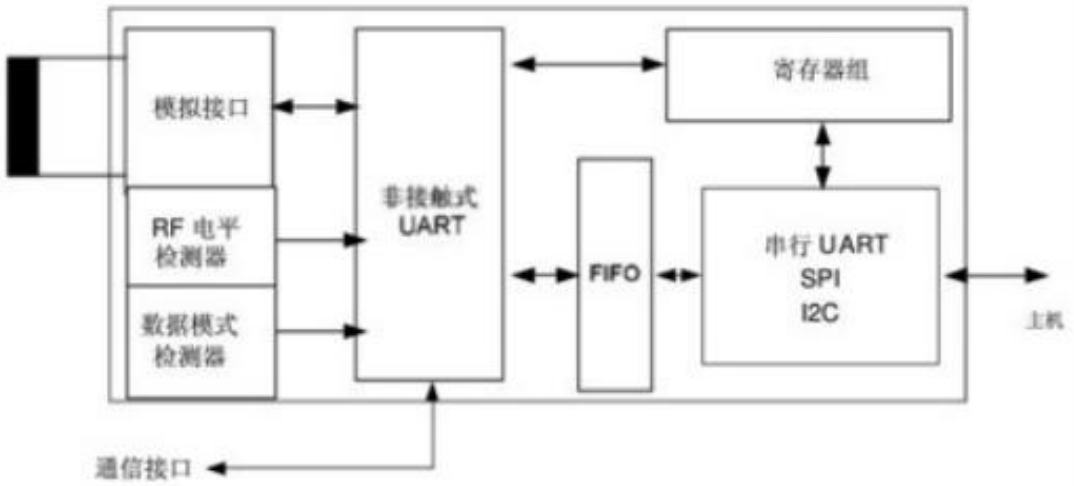
通信协议：ISO-14443A、7816 等

收发缓冲：64 字节

高度集成的非接触式读写芯片，将模拟信号通过模拟接口调制/解调，并通过 FIFO 和非接触式 UART 与主机进行通信。支持多种接口功能

并且该模块支持中断模式、可编程定时器、CRC 协处理器

模块原理图如下图所示：



模块框图

2.智能传感器系统

（1）HC-SR05

该模块的基本原理及流程如下：

1、给 TRIG 引脚最少 10us 的高电平信号来触发测距

2、模块自动发送 8 个 40kHz 的方波，并自动检测是否有信号返回

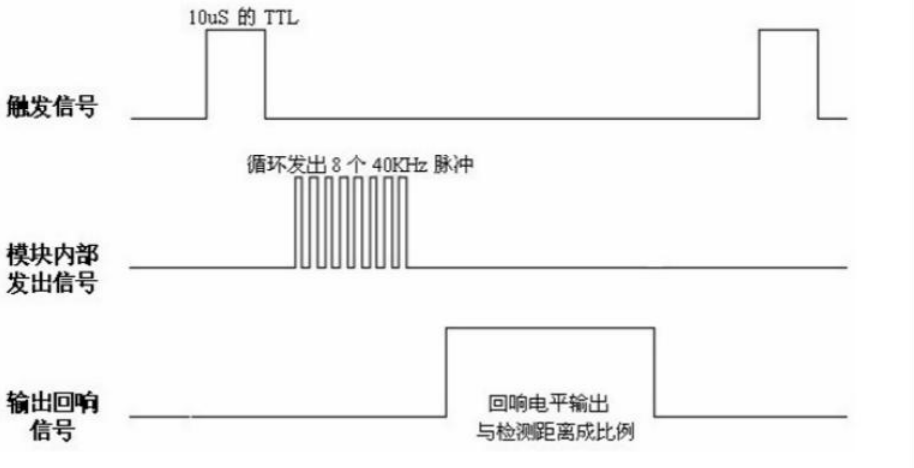
3、如果有信号返回，就通过 ECHO 引脚输出高电平，高电平持续的时间就是超声波，

发射到返回的时间。将超声波传播的时间代入公式距离=高电平持续时间 x 声速/2，可以算出测得的距离。

这里必须注意的是，两次测量之间的时间间隔最好在 60ms 以上，以防止发射信号对反射信号的影响。

模块实物图如下：

模块时序图如下：



(2) LMT70

LMT70 是一款带有输出使能引脚的超小型，高精度，低功耗互补金属氧化物半导体出传感器。模拟温度传感器 LMT70 几乎适用于所有高精度，低功耗的经济高效型温度感测应用，例如物联网 (IoT) 传感器节点。

LMT70 采用 I2C 通讯方式，在本工程中，因 32 内部的 AD 模块存在温漂且精度不够等原因，我们最终采用外部 ADC 对 LMT70 的输出进行采集。

LMT70 的实物图如下图所示：

(3) 微动传感器

基于电压陶瓷片的模拟振动传感器是一种逆变换过程，它使用压电陶瓷产生振动,当

压电陶瓷片振动时，产生电信号，特殊的传感器扩展板组合使用，模拟端口可以被弱感知。振动电信号可以实现与振动相关的交互式工作，如电子鼓互动作品。

ADO 输出:模拟电压陶瓷振动传感器根据程序链接到 UNO 控制器模拟端口 A0，当振动程度不同时,观察串口的输出值，可以实现与振动相关的交互工作，根据振动强度越大输出电压越高，振动幅度灵敏度可通过电位器调节，(左转 灵敏度低，右转灵敏度高)此产品具有高灵敏度可调，可以调节到用嘴巴吹起有响应或者用手敲击就有输出。

D0 输出:即 TTL 电平输出,输出信号 L ED 指示。TTl 输出有效信号为高电平输出控制电流可达 1A 可以外接大功率.ED 灯等。

微动传感器的实物图如下图所示：

3. 摄像头智能检测系统

（1）OpenMV 模块

模块实物图如下：

OpenMV 是一种非常易用和低价的机器视觉开发组件，可以通过编程调用图像处理的算法进行开发。内置算法包括滤波、颜色追踪、AptilTag、二维码、条形码、人脸识别、人眼追踪（瞳孔识别）、直线识别、圆形识别、矩形识别、数字识别、线性回归-巡线、模板匹配、特征点追踪、光流、边缘检测、录制视频、mavlink 等等。其具有性能非常强大的主控芯片 STM32F7 参数：主频 216MHZ、512KB RAM、2Mflash、色块追踪帧率可达 85~90 帧。内置 Micro Python 解释器，

可以直接用 Python(MicroPython)编程,高级数据结构使使用者很容易在机器视觉算法中处理复杂的输出，编写其所有的逻辑，可以调用 Python 库。

4.NBIOT

NBIOT 可以说是本项目的核心元件，他是硬件端与云端的桥梁，实现了设备上云的操作。

NBIOT 模块：即窄带物联网 (Narrow Band Internet of Things, NB-IoT)，构建于蜂窝网络，消耗带宽少，可与现有网络共存，降低部署成本。

在人与人通信之间逐渐达到天花板，万物互联要求与传统蜂窝网络有很大的不同，NBIOT 则在此背景下应运而生。

而作为一项应用于低速率业务中的技术，NB-IoT 的优势不难想象：

• 强链接：在同一基站的情况下，NB-IoT 可以比现有无线技术提供 50-100 倍的接入数。一个扇区能够支持 10 万个连接，支持低延时敏感度、超低的设备成本、低设备功耗和优化的网络架构。举例来说，受限于带宽，运营商给家庭中每个路由器仅开放 8-16 个接入口，而一个家庭中往往有多部手机、笔记本、平板电脑，未来要想实现全屋智能、上百种传感设备需要联网就成了一个棘手的难题。而 NB-IoT 足以轻松满足未来智慧家庭中大量设备联网需求。

• 高覆盖：NB-IoT 室内覆盖能力强，比 LTE 提升 20dB 增益，相当于提升了 100 倍覆盖区域能力。不仅可以满足农村这样的广覆盖需求，对于厂区、地下车库、井盖这类对深度覆盖有要求的应用同样适用。以井盖监测为例，过去 GPRS 的方式需要伸出一根天线，车辆来往极易损坏，而 NB-IoT 只要部署得当，就可以很好的解决这一难题。

• 低功耗：低功耗特性是物联网应用一项重要指标，特别对于一些不能经常更换电池的设备和场合，如安置于高山荒野偏远地区中的各类传感监测设备，它们不可能像智能手机一天一充电，长达几年的电池使用寿命是最本质的需求。NB-IoT 聚焦小数据量、小速率应用，因此 NB-IoT 设备功耗可以做到非常小，设备续航时间可以从过去的几个月大幅提升到几年。

• 低成本：与 LoRa 相比，NB-IoT 无需重新建网，射频和天线基本上都是复用的。以中国移动为例，900MHZ 里面有一个比较宽的频带，只需要清出来一部分 2G 的频段，就可以直接进行 LTE 和 NB-IoT 的同时部署。低速率、低功耗、低带宽同样给 NB-IoT 芯片以及模块带来低成本优势。模块预期价格不超过 5 美元。

不过，NB-IoT 仍有着自身的局限性。在成本方面，NB-IoT 模组成本未来有望降至 5 美元之内，但目前支持蓝牙、Thread、ZigBee 三种标准的芯片价格仅在 2 美元左右，仅支持其中一种标准的芯片价格不到 1 美元。巨大的价格差距无疑将让企业部署 NB-IoT 产生顾虑。同时，NB-IoT 因为带宽较低，所以无法实现音频，图像甚至是视频的传输但对于一款物联网模块而言，往往需求只是传送简单的状态信息，因此在这种情况下 NBIOT 是完全适用的。

三、设计部分

1.器件的选择及辨析。

（1）在选择主控芯片时，我们的可选方案有 stm32f1 以及 stm32f4 芯片，f1 芯片更为小巧，这样可以使得我们的系统更为小巧。但考虑到 f4 是专门为物联网设计的一款芯片，因此具有更多的串口以及资源，而我们的每一块板子都需要外接许多设备，这耗费了大量的资源，因此我们最终选用了 stm32f4 作为我们的主控芯片。

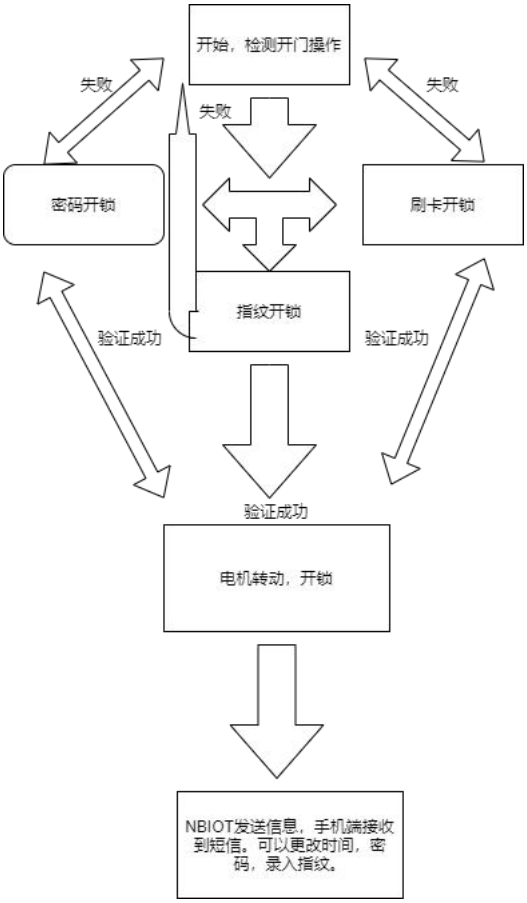
（2）在选择摄像头模组时，我们可以选择 CV 模组以及 OPENMV 模组，但考虑 OPENMV 内置 Micro Python 解释器，可以直接用 Python(MicroPython)编程,高级数据结构使使用者很容易在机器视觉算法中处理复杂的输出，编写其所有的逻辑，可以调用 Python 库，因此相对于 CV，OPENMV 直接调用库函数在一些简单的算法中反而会取得更好的效果，即使这消耗了部分的资源，但这对于我们 f4 芯片来说是完全可以接受的。

（3）在选用云平台时，华为云身为国内的第三大云平台，其实便捷性是不如阿里云的，最致命的缺点就在于该云平台无法直接生成 app，且托管服务是付费的。但考虑到出题老师的要求，我们最后克服了种种困难，；例如直接在手机端下载华为云 app 实现远程控制。

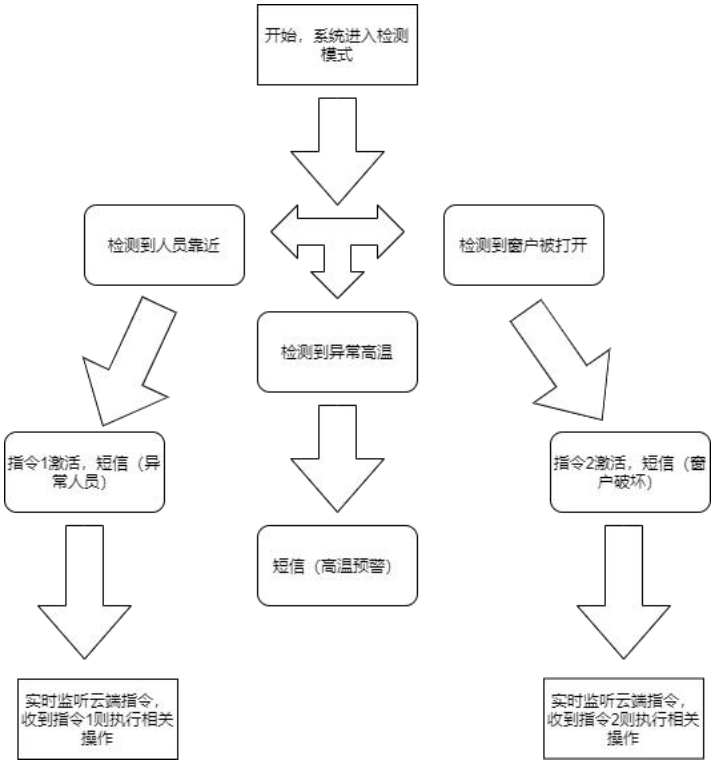
（4）在选用多主控还是单主控问题时，我们最初为了节约资源采用的是单主控，但因为本项目所用传感器较多，即使使用时分复用的方式也造成了及时性的大大损失，而事实上，在实际应用中我们采取的整套系统也一定不会只有整个主体组成，而必然是分布式的分布在环境的各个角落，也因此，我们最后设置出了三套系统，他们每套都可单独运行，合并到一起就完成了整个系统的搭建。

2.系统流程图

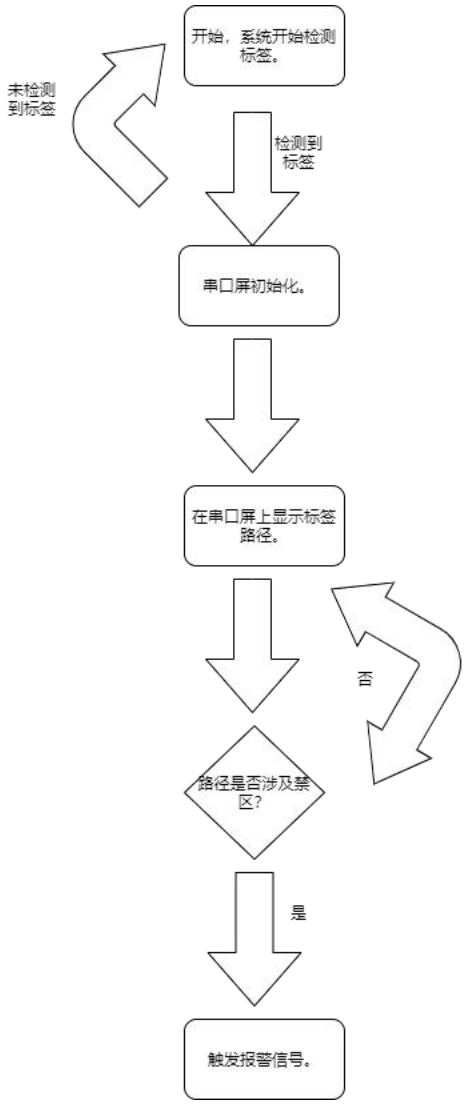
（1）智能门禁系统



(2) 智能传感器系统


开始，系统进入检测模式
检测到人员靠近
检测到异常高温
检测到窗户被打开
指令1激活，短信（异常人员）
短信（高温预警）
指令2激活，短信（窗户破坏）
实时监听云端指令，收到指令1则执行相关操作
实时监听云端指令，收到指令2则执行相关操作

(3) 摄像头智能检测系统



四、测试方案与测试结果

因本项目不设计电路的搭建，所有的参数都是预知的并良好实现，此外系统完整的工作流程可参考我们的视频演示，因此在此处不再赘述。

五、参考文献

[1] 谭浩强.C 语言程序设计[M].北京:清华大学出版社,2012

[2] 薛刘辉. 可穿戴多传感生理监测装置的设计与实现[D].电子科技大学,2020.

[3] TI LMT70+MSP430F5529 可穿戴设备温度传感器参考设计[J].世界电子元器件,2015(06):16-18.

六、总结与收获

囿于现有库房安全防护体系固有的缺陷，我们提出了上述的智能安防系统。通过 NB-IoT，实现单片机和华为云的通信。基于此，我们可以通过云端，从手机上实时监测库房的状态，很大程度上改善了现有库房系统费时费力同时存在监管死角的缺陷。不但如此，通过本次智能库房安防系统的设计，我们也更加深入地理解了云平台的搭建流程，并对嵌入式的原理有了更深入的理解。我们也希望通过这样的智能安防系统的设计，利用物联网和嵌入式的知识，对现有的体系缺陷做出一定的改变。

附录：源程序

注：因本工程过于庞杂，因此此处仅展示出各部分的主函数，限于版面要求对于库函数不予展示。

（1） 智能门禁系统

```
int main(void)
{
  u16 set=0;
  int key_num;
  int time1;
  int time2;    //锁屏时间
  char arrow=0;  //箭头位子

  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分组2
  My_RTC_Init();        //初始化RTC
  RTC_Set_WakeUp(RTC_WakeUpClock_CK_SPRE_16bits,0);   //配置WAKE UP中断,1秒钟中断一次
  delay_init(168);  //初始化延时函数
  uart_init(9600);  //初始化串口1波特率为9600. 用于支持NBIOT
  uart3_init(9600);// 蓝牙初始化
  usart2_init(usart2_baund);//初始化串口2,用于与指纹模块通讯
  PS_StaGPIO_Init();  //初始化FR读状态引脚
  LED_Init();          //初始化LED
  //  BEEP_Init();        //beep初始化
  Button4_4_Init();//矩阵按键初始化
  OLED_Init();       //oled初始化
  W25QXX_Init();        //初始化W25Q128

  Walkmotor_Init(); //步进电机初始化
  my_mem_init(SRAMIN);    //初始化内部内存池
  my_mem_init(SRAMCCM);   //初始化CCM内存池
  exfuns_init();          //为fatfs相关变量申请内存
  f_mount(fs[0],"0:",1);     //挂载SD卡
  f_mount(fs[1],"1:",1);     //挂载FLASH.
  starting();//开机信息  logo


  //0X08020004  密码1
  //0X08090004  密码2
  //0X080f0004  卡号1
  //  STMFLASH_Write(FLASH_SAVE_ADDR,(u32*)TEXT_Buffer,SIZE);
  STMFLASH_Read(0X08020004,(u32*)Pwd,2);  //读取密码1
  STMFLASH_Read(0X08090004,(u32*)Pwd2,2); //读取密码2
  STMFLASH_Read(0X080f0004,(u32*)cardid,1); //读取卡号1
```

⑵ 智能传感器系统

```
#include "sys.h"
#include "usart.h"
#include "delay.h"
#include "timer.h"
#include "oled.h"
#include "distimer.h"
#include "exti.h"
#include "led.h"
extern u8  TIM5CH1_CAPTURE_STA;   //输入捕获状态
extern u32  TIM5CH1_CAPTURE_VAL;  //输入捕获值
char b[5];
char b2[6];
int distancetime=0;
int main()
{
    float LMT70_data = 0;
  int xiaoshu;
  int zhengshu;
  int first=0;
  int secound=0;
  int tem_first;
  int tem_secound;
  long long temp=0,x_atual=0;
    delay_init(168);  //初始化延时函数
  uart_init(9600) ;
    my_app_init() ;
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
  TIM3_Int_Init(5000-1,8400-1); //定时器时钟84M，分频系数8400，所以84M/8400=10Khz的计数频率，计数5000次为500ms
  Trig_Init();
  TIM5_CH1_Cap_Init(0XFFFFFFFF,84-1); //以1Mhz的频率计数
  EXTIX_Init();
  OLED_Init();
  OLED_All(0);//清屏
  OLED_P8x16Str(0,0,(unsigned char*)"Tem:");
    OLED_P8x16Str(70,0,(unsigned char*)"0");
  OLED_P8x16Str(0,2,(unsigned char*)"Dis:");
  OLED_P8x16Str(30,2,(unsigned char*)"0");
    OLED_P8x16Str(70,2,(unsigned char*)"Win:");
  OLED_P8x16Str(100,2,(unsigned char*)"0");
```

```
while(1)
{
if(lmt70flag==3)
{

    LMT70_data=my_app();
//   if(LMT70_data>32){hottime=hottime+1;}
zhengshu=(int)LMT70_data;
secound=zhengshu%16;
first=zhengshu/16;
tem_first=zhengshu/10;
tem_secound=zhengshu%10;
    sprintf(b,"%d",tem_first);
sprintf(b2,"%d",tem_secound);

    OLED_P8x16Str(70,0,b);
OLED_P8x16Str(80,0,b2);

    USART_ITConfig(USART1, USART_IT_RXNE, DISABLE);
  printf("AT+NMGS=3,0000%X%X\r\n\r\n",first,secound);

lmt70flag=0;
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
}
delay_ms(100);
Trig1=1;
delay_us(25);
Trig1=0;
if(TIM5CH1_CAPTURE_STA&0X80)        //成功捕获到了一次高电平
{

  temp=TIM5CH1_CAPTURE_STA&0X3F;
  temp*=0XFFFFFFFF;                 //溢出时间总和
  temp+=TIM5CH1_CAPTURE_VAL;        //得到总的高电平时间
        x_atual=temp*340/20000;
//      printf("HIGH:%lld cm\r\n",x_atual); //打印总的高点平时间
  if(x_atual<10){    USART_ITConfig(USART1, USART_IT_RXNE, DISABLE);if(distancetime==0){LED2_Init();distancetime=1;  }OLED_P8x16Str(30,2,(unsigned char*)"1"); printf("AT+NMGS=3,000100\r\n\r\n");}
  TIM5CH1_CAPTURE_STA=0;            //开启下一次捕获
  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
}
```

（3）摄像头智能检测系统（OPENMV 源代码）

distance.py*

```
import sensor, image, time, math,pyb,time,lcd

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA) # we run out of memory if the resolution is much bigger...
sensor.skip_frames(30)
sensor.set_auto_gain(False)  # must turn this off to prevent image washout...
sensor.set_auto_whitebal(False)  # must turn this off to prevent image washout...
clock = time.clock()
lcd.init()
f_x = (2.8 / 3.984) * 160 # 默认值
f_y = (2.8 / 2.952) * 120 # 默认值
c_x = 160 * 0.5 # 默认值(image.w * 0.5)
c_y = 120 * 0.5 # 默认值(image.h * 0.5)

uart = pyb.UART(3, 115200)
uart.write("hello")
i=0

def lowpass(vi,vk_1,sample,cutfreq):
    rc = 1.0 / 2.0 / 3,1415926 / 5122
    cof1 = 1/(1 + 0.031830988 * 100)
    cof2 = (3.1830988) / (1+3.1830988)
    vk = vi * cof1 + vk_1 * cof2
    return vk

def degrees(radians):
    return (180 * radians) / math.pi

while(True):
    clock.tick()
    lcd.display(sensor.snapshot())
    img = sensor.snapshot()
    for tag in img.find_apriltags(fx=f_x, fy=f_y, cx=c_x, cy=c_y): # 默认为TAG36H11
        img.draw_rectangle(tag.rect(),color=(255,0,0))
        lcd.display(img)
        if i == 0:
            distance = (tag.x_translation()*15.3/200*80+80, tag.y_translation()*15.3/200*80+80, tag.z_translation()*15.3/200*80+80)
            distance_old = distance
            i = i+1
            uart.write("line %f,%f,%f,%f,RED\r\n" % (distance[0],distance[1],distance_old[0],distance_old[1]))
        else:
            distance = (tag.x_translation()*15.3/200*80+80, tag.y_translation()*15.3/200*80+80, tag.z_translation()*15.3/200*80+80)
            distancelp=(lowpass(distance[0],distance_old[0],100,5),lowpass(distance[1],distance_old[1],100,5),\
            lowpass(distance[2],distance_old[2],100,5))
            uart.write("line %d,%d,%d,%d,RED\r\n" % (distance[0],distance[1],distance_old[0],distance_old[1]))
            distance_old = distance
            if distance[0] < 80:
                uart.write("error\r\n")
    time.sleep(10)
```