硬件传输规范

引言

本文档基于AioT Hisi3861智能开发套件制作的自动化智能家居传感器，本组实验所做的智能家居传感器设计有关智能家居传感器，项目内容包括了基于人体红外感应传感器，温湿度感应传感器，OLED屏幕，风扇，光敏电阻，炫彩灯传感器的自动化智能家居管理系统，包括手动风扇挡位控制系统，温度风扇控制系统，OLED实时数据显示系统，门铃显示，人体红外感应系统，灯光控制系统，可燃气体检测和报警，火焰检测和报警系统，以下本文档将详细介绍该传感器的设计原理、功能以及使用方法。

设计原理：

本传感器主要采用了kaihong\_hi3861\_EVB\_01开发板以及kaihong \_Hi3961\_ T&H\_01模块，Kaihong\_Hi3861\_LCD\_01模块，kaihong \_Hi3961\_PIR\_01模块，kaihong \_Hi3961\_GAS\_01, ，kaihong \_Hi3961\_FIRE\_01并结合温湿度传感器，光敏电阻GL5537传感器,可燃气体检测，炫彩灯MHP5050RGBDTC传感器，风扇，人体红外感应器AS312传感器，火焰检测传感器等多种传感器，以实现对家居环境的监测和控制。具体来说，当房间有人，且光线不足的情况下，灯才会亮，实现灯光的自动控制，当按门铃时，在显示屏上会显示门铃状态，当温度达到一定值时，传感器会自动调节风扇，以调节室内的温度和通风，当需要手动通风时，可以通过前端来操控风扇的挡位控制，实现风扇的自动和手动相结合的控制系统。当可燃气体达到一定值时，可燃气体报警器会报警，当检测到火焰时，火焰报警器会报警，通过OLED显示屏，也可以实时查看当前温度和湿度的数据，房间状态，光照强度，灯光状态，可燃气体值，可燃气体报警器状态，火焰状态和火焰报警器状态，从而实现智能家居的控制。

数据规范

1. 数据上报：数据上报主题为“swu2\_temphum“

核心代码：

snprintf(payload,sizeof(payload),"{\"gasalarm\":%d,\"gas\":%d,\"fire\_alarm\":%d,\"fire\_state\":%d,}",alarmbell,gas,fire\_alarm,fire\_state);

snprintf(payload,sizeof(payload), "{\"swu2\_temp\":%d,\"swu2hum\":%d,\"fan\_level\":%d,\"room\_state\":%d,\"light\_state\":%d,\"intensity\":%d}", temp, hum,fan\_level,toilet\_state,toilet\_light\_state,intensity);

1. 数据下发：数据下发主题为“swu2\_fan”

核心代码：topicString.cstring = "swu2\_fan";

// topicString.cstring = "substopic";

len = MQTTSerialize\_subscribe(buf, buflen, 0, msgid, 1, &topicString, &req\_qos);

rc = transport\_sendPacketBuffer(mysock, buf, len);

if (MQTTPacket\_read(buf, buflen, transport\_getdata) == SUBACK) /\* wait for suback \*/

{

unsigned short submsgid;

int subcount;

int granted\_qos;

rc = MQTTDeserialize\_suback(&submsgid, 1, &subcount, &granted\_qos, buf, buflen);

if (granted\_qos != 0)

{

printf("granted qos != 0, %d\n", granted\_qos);

goto exit;

}

}

1. 客户端名称："swu2\_temphum1";和"swu2\_livingroom"
2. 传输时间：每两秒进行一次数据上报

技术方案

1. 硬件设备

传感器模块

1）温湿度感应传感器

系统采用了kaihong \_Hi3961\_ T&H\_01模块，能够准确地测量空气温度和湿度。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对房间的温度和湿度进行具体的评估。

代码示例：

获取空气温度和湿度

 uint32\_t SHT40\_GetMeasureResult(int \*temp, int \*humi)

uint32\_t retval = 0;

float t\_degC = 0;

float rh\_pRH = 0;

float t\_ticks = 0.0;

float rh\_ticks = 0.0;

if (temp == NULL || humi == NULL)

{

return -1;

}

uint8\_t buffer[SHT40\_STATUS\_RESPONSE\_MAX] = {0};

memset(buffer, 0x0, sizeof(buffer));

retval = SHT40\_Read(buffer, sizeof(buffer)); // recv status command result

if (retval != 0)

{

return retval;

}

t\_ticks = buffer[0] \* 256 + buffer[1];

rh\_ticks = buffer[3] \* 256 + buffer[4];

t\_degC = -45 + 175 \* t\_ticks / 65535;

rh\_pRH = -6 + 125 \* rh\_ticks / 65535;

if (rh\_pRH >= 100)

{

rh\_pRH = 100;

}

if (rh\_pRH < 0)

{

rh\_pRH = 0;

}

\*humi = (uint8\_t)rh\_pRH;

\*temp = (uint8\_t)t\_degC;

return 0;

}

1. 光照强度传感器

系统采用了kaihong \_Hi3961\_PIR\_01模块，能够准确地测量光照强度。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对房间的光线强度进行具体的评估。

代码示例

 //调用AdcRead读取值

        if (AdcRead(SOIL\_MOISTURE\_CHAN\_NAME, &data, WIFI\_IOT\_ADC\_EQU\_MODEL\_4, WIFI\_IOT\_ADC\_CUR\_BAIS\_DEFAULT, 0) == WIFI\_IOT\_SUCCESS)

        {

            intensity = 2000 - data;

        }

        sleep(2);

3）人体红外感应传感器

系统采用了kaihong \_Hi3961\_PIR\_01模块，能够感应是否有人。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对房间状态是否有人进行具体的评估。

代码示例

void toilet\_entry(void \*arg)

{

(void)arg;

init();

WifiIotGpioValue rel = 0;

while (1)

{

//读取人体红外传感器，

GpioGetInputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_7, &rel);

toilet\_state = rel;

//读取光敏电阻

GpioGetInputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_9, &rel);

//如果有人

if (toilet\_state)

{

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_10, (int)rel);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_11, (int)rel);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_12, (int)rel);

toilet\_light\_state = rel;

}

else

{

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_10, WIFI\_IOT\_GPIO\_VALUE0);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_11, WIFI\_IOT\_GPIO\_VALUE0);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_12, WIFI\_IOT\_GPIO\_VALUE0);

toilet\_light\_state = 0;

}

sleep(1);

}

}

1. 炫彩灯光传感器

系统采用了kaihong \_Hi3961\_PIR\_01模块，能够在光线弱且有人的时候灯自动打开。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对房间灯光状态评估。

代码示例

void init(void)

{

GpioInit();

//设置红色,蓝 色,绿色 LED IO为输出状态

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_10, WIFI\_IOT\_IO\_FUNC\_GPIO\_10\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_10, WIFI\_IOT\_GPIO\_DIR\_OUT);

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_11, WIFI\_IOT\_IO\_FUNC\_GPIO\_11\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_11, WIFI\_IOT\_GPIO\_DIR\_OUT);

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_12, WIFI\_IOT\_IO\_FUNC\_GPIO\_12\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_12, WIFI\_IOT\_GPIO\_DIR\_OUT);

//光敏电阻

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_9, WIFI\_IOT\_IO\_FUNC\_GPIO\_9\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_9, WIFI\_IOT\_GPIO\_DIR\_IN);

//人体红外感应

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_7, WIFI\_IOT\_IO\_FUNC\_GPIO\_7\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_7, WIFI\_IOT\_GPIO\_DIR\_IN);

IoSetPull(WIFI\_IOT\_IO\_NAME\_GPIO\_7, WIFI\_IOT\_IO\_PULL\_UP);

}

void toilet\_entry(void \*arg)

{

(void)arg;

init();

WifiIotGpioValue rel = 0;

while (1)

{

//读取人体红外传感器，

GpioGetInputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_7, &rel);

toilet\_state = rel;

//读取光敏电阻

GpioGetInputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_9, &rel);

//如果有人

if (toilet\_state)

{

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_10, (int)rel);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_11, (int)rel);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_12, (int)rel);

toilet\_light\_state = rel;

}

else

{

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_10, WIFI\_IOT\_GPIO\_VALUE0);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_11, WIFI\_IOT\_GPIO\_VALUE0);

GpioSetOutputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_12, WIFI\_IOT\_GPIO\_VALUE0);

toilet\_light\_state = 0;

}

sleep(1);

}

}

5）风扇

系统采用了风扇传感器，能够在温度达到一定值时风扇自动开启，也可以通过手动指令控制风扇挡位。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对风扇挡位状态进行评估。

代码示例

风扇控制

void fan\_task\_init(void)

{

GpioInit();

IoSetFunc(FAN\_PIN\_NAME, WIFI\_IOT\_IO\_FUNC\_GPIO\_8\_PWM1\_OUT);

PwmInit(WIFI\_IOT\_PWM\_PORT\_PWM1); //

// IoSetFunc(FAN\_PIN\_NAME, WIFI\_IOT\_IO\_FUNC\_GPIO\_8\_GPIO);

IoSetFunc(FAN\_POWER\_PIN, WIFI\_IOT\_IO\_FUNC\_GPIO\_12\_GPIO);

// GpioSetDir(FAN\_PIN\_NAME, WIFI\_IOT\_GPIO\_DIR\_OUT);

GpioSetDir(FAN\_POWER\_PIN, WIFI\_IOT\_GPIO\_DIR\_OUT);

}

void fan\_thread(void \*arg)

{

(void)arg;

fan\_task\_init(); //初始化

while (1)

{

if (fan\_level > 0 && fan\_level < 4)

{

GpioSetOutputVal(FAN\_POWER\_PIN, WIFI\_IOT\_GPIO\_VALUE1);

PwmStart(WIFI\_IOT\_PWM\_PORT\_PWM1, PWM\_FREQ\_DIVITION / 10 \* (fan\_level + 7), PWM\_FREQ\_DIVITION);

}

else if(temp>=30)

{

GpioSetOutputVal(FAN\_POWER\_PIN, WIFI\_IOT\_GPIO\_VALUE1);

PwmStart(WIFI\_IOT\_PWM\_PORT\_PWM1, PWM\_FREQ\_DIVITION / 10 \* (fan\_level + 7), PWM\_FREQ\_DIVITION);

}

else if (temp<30)

{

GpioSetOutputVal(FAN\_POWER\_PIN, WIFI\_IOT\_GPIO\_VALUE0);

PwmStop(WIFI\_IOT\_PWM\_PORT\_PWM1);

}

else

{

GpioSetOutputVal(FAN\_POWER\_PIN, WIFI\_IOT\_GPIO\_VALUE0);

PwmStop(WIFI\_IOT\_PWM\_PORT\_PWM1);

}

}

sleep(2);

}

}

6)OLED显示屏

系统采用了Kaihong\_Hi3861\_LCD\_01模块，能够在显示屏上显示实时的温度，湿度，光照强度，风扇挡位，房间状态，灯光状态的数据。传感器通过ADC芯片将模拟信号转换成数字信号，然后通过微处理器进行数据处理，通过量化的数据对房间各种数据状态进行评估。

**代码示例**

**温湿度风扇模块**

//该函数对GPIO管脚及OLED进行初始化

void oledTaskInit(void)

{

GpioInit();

OledInit();

}

//业务函数，完成数据在OLED上显示

void oledThread(void \*arg)

{

(void)arg;

oledTaskInit(); //初始化

OledFillScreen(0x00); //清屏，

//在左上角位置显示字符串Hello, HarmonyOS

OledShowString(0, 0, "SWU2 smart HOME", 1);//在(0, 0)位置显示

char line[32] = {0};

while (1)

{

//组装显示温度字符串

snprintf(line, sizeof(line), "temp: %3d du", temp);

OledShowString(0, 1, line, 1); //在（0，1）位置显示组装后的温度字符串

snprintf(line, sizeof(line), "hum : %3d%%", hum);

OledShowString(0, 2, line, 1); //在（0，2）//位置显示组装后的湿度字符串

snprintf(line, sizeof(line), "fan level: %03d", fan\_level);

OledShowString(0, 3, line, 1); //在（0，3）//位置显示组装后的风扇挡位【0-3】

snprintf(line, sizeof(line), "room\_state:%d", toilet\_state);

OledShowString(0, 4, line, 1); //在（0，4）//显示房间状态0or1

snprintf(line, sizeof(line), "light\_state:%d", toilet\_light\_state); //在（0，5）显示灯光状态0or1

OledShowString(0, 5, line, 1);

snprintf(line, sizeof(line), "intensity : %4d", intensity);

OledShowString(0, 6, line, 1); //在（0，6）//显示光照强度

sleep(1); //睡眠

}

}

**可燃气体火焰模块**

void oled\_thread(void \*arg)

{

(void)arg;

oledTaskInit(); //初始化

OledFillScreen(0x00); //清屏，

//在左上角位置显示字符串Hello, HarmonyOS

OledShowString(0, 0, "smart home 2.0", 1);

// printf("oled thread running ,led level:%d\r\n", fan\_level);

char line[32] = {0};

while (1)

{

//组装显示气体的字符串 单位是百万分之 ，检测浓度：300-10000ppm(可燃气体)

snprintf(line, sizeof(line), "fire\_alarm : %3s", fire\_alarm ? "on" : "off");

OledShowString(0, 1, line, 1); //在（0，1）位置显示组装后火焰报警器状态

snprintf(line, sizeof(line), "fire\_state : %2d", fire\_state);

OledShowString(0, 2, line, 1); //在（0，2）位置显示组装后的火焰状态

snprintf(line, sizeof(line), "temp: %.1f deg", temperature);

OledShowString(0, 3, line, 1); //在（0，3）位置显示组装后的温度字符串

//组装显示湿度的字符串

snprintf(line, sizeof(line), "humi: %.1f %%", humidity);

OledShowString(0, 4, line, 1); //在（0，4）位置显示组装后的湿度字符串

snprintf(line, sizeof(line), "gas\_state: %5d", gas);

OledShowString(0, 5, line, 1); //在（0，5）位置显示组装后的可燃气体值

snprintf(line, sizeof(line), "alarm bell:%3s", alarmbell == 0 ? "off" : "on");

OledShowString(0, 6, line, 1); //在（0，6）可燃气体报警器值

sleep(1); //睡眠

}

}

7）可燃气体检测传感器

**代码示例**

//对应管脚PIN11

#define GAS\_SENSOR\_CHAN\_NAME WIFI\_IOT\_ADC\_CHANNEL\_5

//气体报警阈值

#define GAS\_THRESHOLD 700

int alarmbell = 0;

int gas = 0;

void init(void)//初始化蜂鸣器

{

GpioInit();

// 蜂鸣器引脚 设置为 PWM功能

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_9, WIFI\_IOT\_IO\_FUNC\_GPIO\_9\_PWM0\_OUT);

PwmInit(WIFI\_IOT\_PWM\_PORT\_PWM0);

}

void kitchen\_entry(void \*arg)//可燃气体检测入口函数

{

(void)arg;

init();

while (1)

{

unsigned short data = 0; //保存读取到的燃气值

//调用AdcRead读取值

if (AdcRead(GAS\_SENSOR\_CHAN\_NAME, &data, WIFI\_IOT\_ADC\_EQU\_MODEL\_4, WIFI\_IOT\_ADC\_CUR\_BAIS\_DEFAULT, 0) == WIFI\_IOT\_SUCCESS)

{

gas = data;

if (gas > GAS\_THRESHOLD)

{

alarmbell = 1;

uint16\_t freqDivisor = 34052;

// 占空比， 频率

PwmStart(WIFI\_IOT\_PWM\_PORT\_PWM0, gas \* gas \* 3 / GAS\_THRESHOLD, freqDivisor);

}

else

{

alarmbell = 0;

PwmStop(WIFI\_IOT\_PWM\_PORT\_PWM0);

}

sleep(1);

}

}

}

8）火焰检测传感器

**代码示例**

int fire\_alarm = 0;

unsigned short fire\_state = 0; // 1，表示发生火灾，0表示未发生火灾

float humidity = 0.0f; //用于保存湿度的变量

float temperature = 0.0f; //用于保存温度的变量

void init\_living(void)

{

GpioInit();

// 蜂鸣器引脚 设置为 PWM功能

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_9, WIFI\_IOT\_IO\_FUNC\_GPIO\_9\_PWM0\_OUT);

PwmInit(WIFI\_IOT\_PWM\_PORT\_PWM0);

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_13, WIFI\_IOT\_IO\_FUNC\_GPIO\_13\_I2C0\_SDA);

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_14, WIFI\_IOT\_IO\_FUNC\_GPIO\_14\_I2C0\_SCL);

I2cInit(AHT20\_I2C\_IDX, AHT20\_BAUDRATE); //初始化I2c 的波特率

//火焰接收器引脚

IoSetFunc(WIFI\_IOT\_IO\_NAME\_GPIO\_5, WIFI\_IOT\_IO\_FUNC\_GPIO\_5\_GPIO);

GpioSetDir(WIFI\_IOT\_IO\_NAME\_GPIO\_5, WIFI\_IOT\_GPIO\_DIR\_IN);

}

void living\_room\_entry(void \*arg)

{

(void)arg;

init\_living(); //初始化IIC

uint32\_t retval = 0;

uint16\_t freqDivisor = 34052;

// 发送初始化校准命令

while (WIFI\_IOT\_SUCCESS != AHT20\_Calibrate())

{

printf("AHT20 sensor init failed!\r\n");

usleep(1000);

}

while (1)

{

// 发送 触发测量 命令，开始测量

retval = AHT20\_StartMeasure();

if (retval != WIFI\_IOT\_SUCCESS)

{

printf("trigger measure failed!\r\n");

}

else

{

// 接收测量结果，拼接转换为标准值

retval = AHT20\_GetMeasureResult(&temperature, &humidity);

printf("temp: %.2f, humi: %.2f\n", temperature, humidity);

}

//火焰探测

WifiIotGpioValue value;

GpioGetInputVal(WIFI\_IOT\_IO\_NAME\_GPIO\_5, &value); //获取火焰状态,0表示探测到火焰，1表示未探测到

//获取到的值转化为火灾状态

fire\_state = value ? 0 : 1;

if (fire\_state)

{

fire\_alarm = 1;

PwmStart(WIFI\_IOT\_PWM\_PORT\_PWM0, freqDivisor / 20, freqDivisor);

}

else

{

fire\_alarm = 0;

PwmStop(WIFI\_IOT\_PWM\_PORT\_PWM0);

}

sleep(1);

}

}

控制模块

控制模块采用了一款单片机芯片:kaihong\_hi3861\_EVB\_01，通过连接各种传感器可以实现传感器数据的输入、输出和控制。控制模块还集成了Wi-Fi模块，以便将数据上传到云端服务器对数据进行处理。。

执行模块

a.风扇模块，能够自动或手动控制风扇的开关。执行模块收来自控制模块的指令，根据指令自动控制风扇的开关以及风速

b.led灯模块能够自动控制led灯的开关。执行模块收来自控制模块的指令，根据指令自动控制led灯的开关以及风速

2. 软件系统

数据采集与处理

A.温湿度感应传感器

传感器模块测量的温湿度数据通过ADC芯片转换成数字信号，通过sth40检测环境温度，然后由控制模块进行数据处理和存储。控制模块将数据上传到云端服务器，并在需要时从服务器获取其他数据。

B.光照强度传感器

传感器模块测量的光线强度数据通过ADC芯片转换成数字信号，然后由控制模块进行数据处理和存储。控制模块将数据上传到云端服务器，并在需要时从服务器获取其他数据。

C.人体红外感应传感器

传感器模块通过感应是否有人通过ADC芯片转换成数字信号，然后由控制模块进行数据处理和存储。控制模块将数据上传到云端服务器，并在需要时从服务器获取其他数据。

D.OLED屏幕

传感器模块测量的房间各种数据通过ADC芯片转换成数字信号，然后由控制模块进行数据处理和存储，转到OLED屏幕上显示数据。

E.可燃气体检测传感器

传感器模块测量的可燃气体数值数据通过ADC芯片转换成数字信号，然后由控制模块进行数据处理和存储。控制模块将数据上传到云端服务器，并在需要时从服务器获取其他数据。

F.火焰检测传感器

传感器模块测量的火焰状态数据通过ADC芯片转换成数字信号，然后由控制模块进行数据处理和存储。控制模块将数据上传到云端服务器，并在需要时从服务器获取其他数据。

控制算法实现

A.温湿度感应传感器

系统通过预设的控制算法实现自动化控制功能。当空气的温湿度高于一定阈值时，控制模块会向执行模块发送指令，开启风扇对房间进行散热或除湿。等到空气的温湿度达到设定值后，执行模块会自动关闭风扇，根据空气的温湿度状况来确定风扇的风速。

B.光照强度传感器

系统通过预设的控制算法实现自动化控制功能。当光照强度低于一定阈值时，并且房间状态为有人时，控制模块会向执行模块发送指令，开启led灯进行照明，并上报灯光状态为1。等到光照强度变弱后，执行模块会自动关闭led灯，根据光照强度的情况来确定led灯的明亮程度，并输出光照强度的数值和房间状态。

C. 人体红外感应传感器

系统通过预设的控制算法实现自动化控制功能。当红外感应到有人时，控制模块会向执行模块发送指令，上报房间状态为1，为有人。

D可燃气体检测

系统通过预设的控制算法实现自动化控制功能。当可燃气体数值达到700时，控制模块会向执行模块发送指令，上报可燃气体数值，达到阈值时上报报警器状态为1。

E.火焰在检测传感器

系统通过预设的控制算法实现自动化控制功能。当感应到有火焰时，控制模块会向执行模块发送指令，上报火焰状态为1，为检测到火焰，并上报报警器状态为1。

远程监测与控制

用户可以通过手机APP或电脑浏览器远程监测和控制系统。用户可以查看当前温度，湿度，光照强度、当前时间，风扇挡位，房间状态，灯光状态，并可以设置阈值和定时任务。用户还可通过APP或浏览器手动控制风扇，led灯的开关和强度等级。

总结

该自动化智能家居传感器系统能够自动监测和控制房间的空气温湿度和光照强度，并进行自动化控制，如此发展下去能有诸多优势

1. 提供便利性：智能家居系统可以自动化执行日常任务，如控制灯光、调节温度、打开风扇等。可以使用手机应用程序来管理家居设备，从而提供更加方便的生活体验。
2. 能源效率：智能家居系统可以监测和控制能源使用，优化能源消耗。例如，它可以根据自动调节照明和温度，在无人的白天灯不会亮，只有有人和灯光暗时，灯才会亮，以减少不必要的能源浪费，从而降低能源账单。
3. 增强舒适度：智能家居系统可以根据偏好和日常行为自动调整环境条件，如温度、湿度和照明。可以根据自己的需求来创建个性化的舒适设置，使家居环境更加宜居。
4. 远程控制：智能家居系统能够远程监控和控制家中设备，无论身在何处。无论是在办公室、度假或外出，都可以使用手机应用程序远程打开灯光、监控家庭安全或控制其他智能设备。
5. 自动化协调：智能家居系统可以实现设备之间的自动协调。例如，当离开家时，系统可以自动关闭灯光并调整温度，以节省能源。这种自动化协调可以简化生活，节省时间和精力。

6.安全监控：可以检测家中的可燃气体数值，是否着火，及时进行报警提醒，提升家中安全。