# 智能家居系统开发文档

## 一、项目概述

项目名称: 基于树莓派的智能家居系统

项目目标: 实现家庭环境的实时监测和控制，提高居家安全性和舒适度。

主要功能:

实时监测温湿度、烟雾、火焰、光照和雨滴。

控制LED灯光。

## 二、系统架构

硬件组成:

树莓派（主控单元）

传感器:

温湿度传感器

烟雾传感器

火焰传感器

光敏传感器

雨滴传感器

控制模块:

LED灯光控制模块

继电器控制模块

软件组成:

操作系统: Raspberry Pi OS、window

编程语言：c++

库:wiringPi、标准库、POSIX线程库

界面: Web界面

3.BOM

树莓派:

型号: 树莓派 4B

存储: 32GB microSD卡

电源: 5V 3A 电源适配器

树莓派扩展板PI GPIO-T Plus和线

杜邦线、跳线

鼠标、键盘

HDMI采集器

面包板

传感器和模块:

###### 温湿度传感器DHT11

DHT11数字温湿度传感器是一款含有已校准数字信号输出的温湿度复合传感器。它应用专用的数字模块采集技术和温湿度传感技术，确保产品具有可靠性与卓越的长期稳定性，成本低、相对湿度和温度测量、快响应、抗干扰能力强、信号传输距离长、数字信号输出、精确校准。传感器包括一个电容式感湿元件和一个NTC测温元件，并与一个高性能8位单片机相连接。可用于暖通空调、除湿器、测试及检测设备、消费品、汽车、自动控制、数据记录器、气象站、家电、湿度调节器、医疗、其他相关湿度检测控制。

模块资料：https://blog.csdn.net/lzh201864031/article/details/130791061

###### 烟雾传感器MQ-2

MQ2传感器对氨气、硫化物、苯系蒸汽的灵敏度高，对烟雾和其它有害的监测也很理想。这种传感器可检测多种有害气体，是一款适合多种应用的低成本传感器。

模块资料：https://telesky.yuque.com/bdys8w/01/ao4hdpqcfwt6wfkf

###### 火焰传感器

模块资料：https://blog.csdn.net/m0\_46152793/article/details/118313463?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=%E7%81%AB%E7%84%B0%E4%BC%A0%E6%84%9F%E5%99%A8%E6%A8%A1%E5%9D%97%E8%B5%84%E6%96%99&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-0-118313463.142^v100^pc\_search\_result\_base8&spm=1018.2226.3001.4187

###### 光敏电阻传感器模块

模块资料：https://blog.csdn.net/m0\_64562382/article/details/135830532?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%252208EABDCA-115A-4C70-B66B-EC7922BF8657%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=08EABDCA-115A-4C70-B66B-EC7922BF8657&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~top\_positive~default-1-135830532-null-null.142^v100^pc\_search\_result\_base8&utm\_term=%E5%85%89%E6%95%8F%E4%BC%A0%E6%84%9F%E5%99%A8%E6%A8%A1%E5%9D%97&spm=1018.2226.3001.4187

###### 雨滴传感器

模块资料：https://pan.baidu.com/s/10bjbsmcOh2N7YGDS3PquPw 提取码：psfm

###### LED灯光模块

连接:

MQ2烟雾传感器：GPIO4

DHT11温湿度传感器：GPIO17

火焰检测传感器：GPIO18

光敏电阻：GPIO27

蜂鸣器：GPIO22

LED灯1：GPIO23

LED灯2：GPIO24

LED灯3：GPIO25

雨滴检测传感器：GPIO5

#### 三、软件配置

操作系统安装:

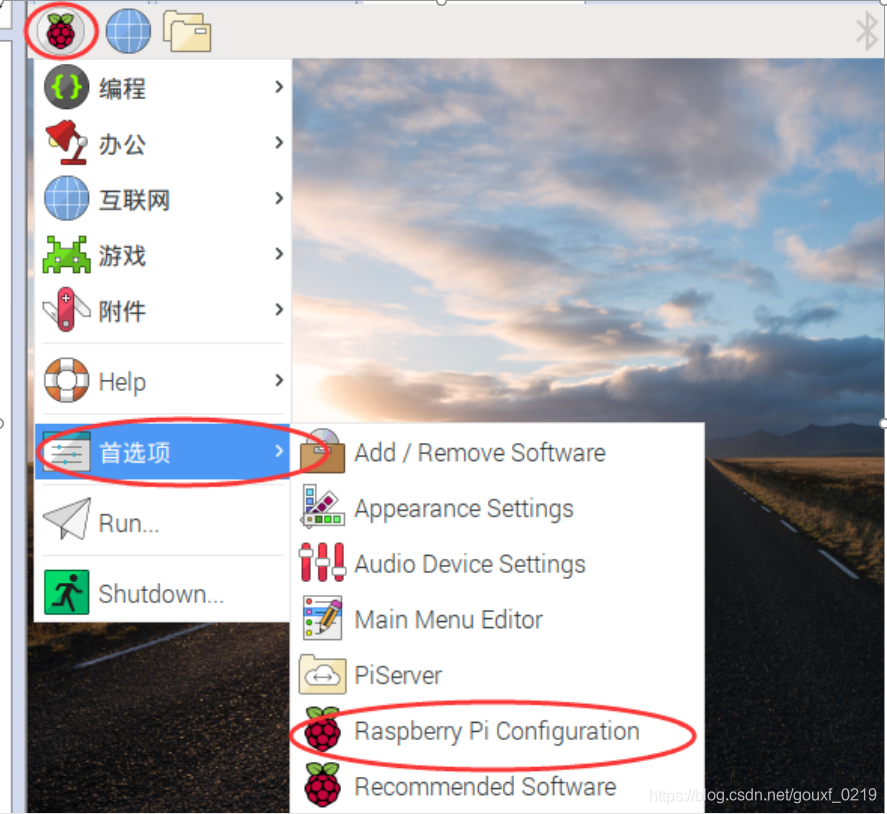
下载Raspberry Pi OS镜像文件。

使用工具（如Balena Etcher）将镜像烧录到microSD卡。

插入microSD卡并启动树莓派。

将树莓派连接HDMI采集器并连接电脑，树莓派接入鼠标键盘

连接手机热点并打开树莓派ssh



选择Interfaces将ssh服务enable

环境准备:

更新系统:

bash

sudo apt update

sudo apt upgrade

安装Python和必要的库:

bash

sudo apt install python3 python3-pip

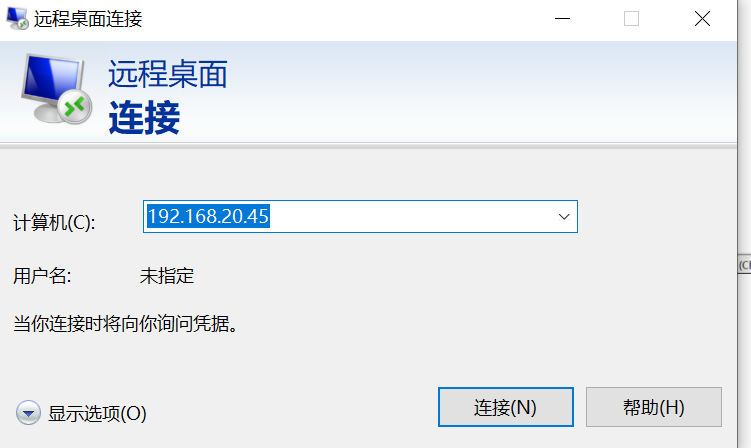
安装远程桌面

sudo apt-get install xrdp

在电脑和树莓派连接同一个热点

打开终端  
输入：ifconfig查看wla0 ip地址

win+r输入mstsc将查看到的ip输入即可连接树莓派



**安装WiringPi库**

sudo apt-get update

#如果之前没有安装过 git，执行下面的命令安装一下

sudo apt-get install git-core

#下载 wiringPi

git clone https://github.com/WiringPi/WiringPi

#编译 wiringPi

cd WiringPi

sudo ./build

**添加库**

访问github下载地址：[https://github.com/eclipse/paho.mqtt.c](https://github.com/eclipse/paho.mqtt.c" \t "https://support.huaweicloud.com/devg-iothub/_blank)，下载 paho.mqtt.c源码。解压后上传到linux编译机。

执行清空命令

make clean

执行编译命令

make

复制paho库文件。当前SDK仅用到了libpaho-mqtt3as，将“libpaho-mqtt3as.so” 和“libpaho-mqtt3as.so.1”文件复制到demo的lib文件夹下。（同时回到paho 源代码路径，进入src目录，将MQTTAsync.h、MQTTClient.h、 MQTTClientPersistence.h、MQTTProperties.h、MQTTReasonCodes.h、 MQTTSubscribeOpts.h复制到demo的include/base文件夹下）。

在Makefile的第15行最后添加 -lm ，执行make进行编译。如果是32位的操 作系统，请删除Makefile中的"-m64"。

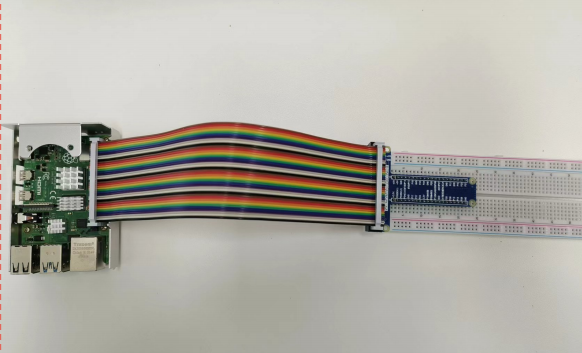
执行export LD\_LIBRARY\_PATH=./lib/加载库文件。

运行./MQTT\_Demo.o。

#### 硬件配置

**接线安装：**

先把T型版安装到面包板上，接着用彩板线把树莓派和T型版连接起来



**然后用跳线连接3v3、5v、GND到两侧**

将传感器借上树莓派

MQ2烟雾传感器：GPIO4

DHT11温湿度传感器：GPIO17

火焰检测传感器：GPIO18

光敏电阻：GPIO27

蜂鸣器：GPIO22

LED灯1：GPIO23

LED灯2：GPIO24

LED灯3：GPIO25

雨滴检测传感器：GPIO5

#### 五、部署服务器

**MQTT通信:**

**搭建MQTT服务器：**

docker创建：

docker run -d --name emqx -p 1883:1883 -p 8083:8083 -p 8084:8084 -p 18083:18083

emqx/emqx

**使用免费MQTT服务器：**

broker.emqx.io

**电脑部分：**

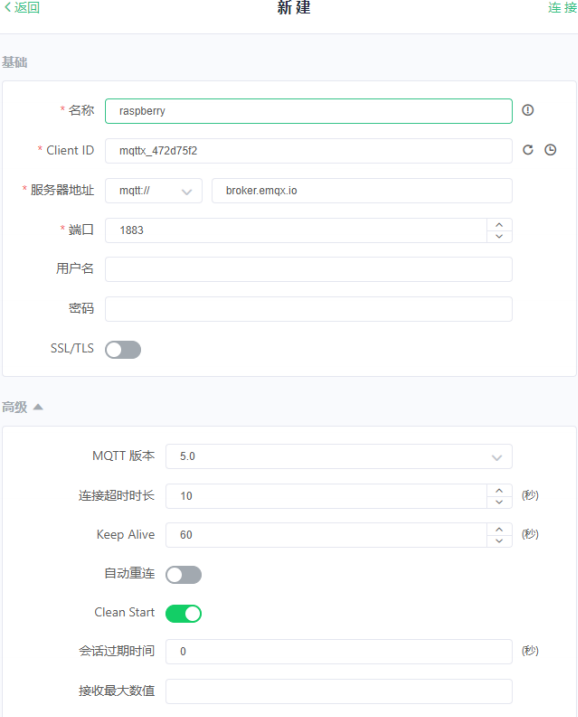
**软件：**

使用MQTTX 进行 windows电脑和树莓派的mqtt通信

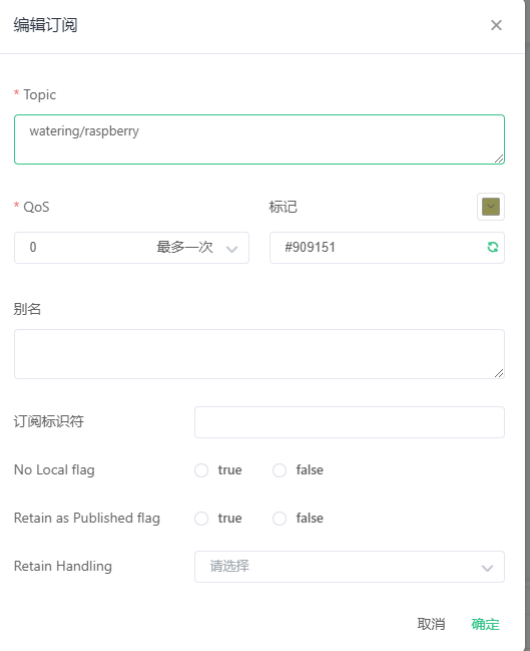
**下载地址：**

https://mqttx.app/zh

**新建连接：（名称可以自己取，其他的默认就可以）**



**添加订阅**



**（树莓派发送的消息往这个topic来发送，大家也可以起其他的名字，但是需要和下面的代码对应）**

六、编写代码

解析DHT11数据

    uint8 readSensorData(void)

    {

        uint8 crc;

        uint8 i;

        pinMode(pinNumber, OUTPUT); // set mode to output

        digitalWrite(pinNumber, 0); // output a high level

        delay(25);

        digitalWrite(pinNumber, 1); // output a low level

        pinMode(pinNumber, INPUT); // set mode to input

        pullUpDnControl(pinNumber, PUD\_UP);

        delayMicroseconds(27);

        if (digitalRead(pinNumber) == 0) //SENSOR ANS

        {

            while (!digitalRead(pinNumber))

                ; //wait to high

            for (i = 0; i < 32; i++)

            {

                while (digitalRead(pinNumber))

                    ; //data clock start

                while (!digitalRead(pinNumber))

                    ; //data start

                delayMicroseconds(HIGH\_TIME);

                databuf \*= 2;

                if (digitalRead(pinNumber) == 1) //1

                {

                    databuf++;

                }

            }

            for (i = 0; i < 8; i++)

            {

                while (digitalRead(pinNumber))

                    ; //data clock start

                while (!digitalRead(pinNumber))

                    ; //data start

                delayMicroseconds(HIGH\_TIME);

                crc \*= 2;

                if (digitalRead(pinNumber) == 1) //1

                {

                    crc++;

                }

            }

            return 1;

        }

        else

        {return 0;

        }

    }

**初始化GPIO**

wiringPiSetupGpio();

        pinMode(23,OUTPUT);

        pinMode(24,OUTPUT);

        pinMode(25,OUTPUT);

        pinMode(6,OUTPUT);

        pinMode(22,OUTPUT);

        pinMode(4,INPUT);

        pinMode(18,INPUT);

        pinMode(27,INPUT);

        pinMode(5,INPUT);

**读取传感器数据**

flame=digitalRead (18);

light=digitalRead (27);

water=digitalRead (5);

LED1=digitalRead (23);

LED2=digitalRead (24);

LED3=digitalRead (25);

if (readSensorData())

{

DHT11\_H=((databuf >> 24) & 0xff);

DHT11\_T=((databuf >> 8) & 0xff);

databuf = 0;

}

**创建MQTT实例设置相关参数并将编写回调函数做退避重连**

        conn\_opts.cleansession = 1;

            conn\_opts.keepAliveInterval = keepAliveInterval;

            conn\_opts.connectTimeout = connectTimeout;

            conn\_opts.retryInterval = retryInterval;

            conn\_opts.onSuccess = mqtt\_connect\_success;

            conn\_opts.onFailure = mqtt\_connect\_failure;

int createRet = MQTTAsync\_create(&client, uri, "deployment-jeff0591", MQTTCLIENT\_PERSISTENCE\_NONE, NULL);

MQTTAsync\_setCallbacks(client, NULL, mqtt\_connection\_lost, mqtt\_message\_arrive, NULL);

**编写发布和订阅函数，对应回调函数做日志输出**

int mqtt\_publish(const char \*topic, char \*payload) {

        MQTTAsync\_responseOptions opts = MQTTAsync\_responseOptions\_initializer;

        MQTTAsync\_message pubmsg = MQTTAsync\_message\_initializer;

        opts.onSuccess = publish\_success;

        opts.onFailure = publish\_failure;

        pubmsg.payload = payload;

        pubmsg.payloadlen = (int) strlen(payload);

        pubmsg.qos = gQOS;

        pubmsg.retained = 0;

        int ret = MQTTAsync\_sendMessage(client, topic, &pubmsg, &opts);

        if (ret != 0) {

            printf( "mqtt\_publish() error, publish result %d\n", ret);

            return -1;

        }

        printf("mqtt\_publish(), the payload is %s, the topic is %s \n", payload, topic);

        return opts.token;

    }

int mqtt\_subscribe(const char \*topic) {

        MQTTAsync\_responseOptions opts = MQTTAsync\_responseOptions\_initializer;

        opts.onSuccess = subscribe\_success;

        opts.onFailure = subscribe\_failure;

        int qos = 1;

        int ret = MQTTAsync\_subscribe(client, topic, qos, &opts); //this qos must be 1, otherwise if subscribe failed, the downlink message cannot arrive.

        if (MQTTASYNC\_SUCCESS != ret) {

            printf("mqtt\_subscribe() error, subscribe failed, ret code %d, topic %s\n", ret, topic);

            return -1;

        }

        printf("mqtt\_subscribe(), topic %s, messageId %d\n", topic, opts.token);

        return opts.token;

}

**编写订阅数据解析函数将订阅和发布分为两个线程**

    void \*pth\_work\_func(void \*arg){

        char buff[1024];

        char buff2[1024];

        int size=0;

        int i=0;

        while(1)

        {

            int ret = mqtt\_subscribe("data/rec");

            if (ret < 0) {

                printf("subscribe topic error, result %d\n", ret);

            }

                strcpy(buff, g\_message);

                strcpy(buff2, g\_topic);

                size= strlen(g\_message);

                memset(g\_message, 0, sizeof(g\_message));

                memset(g\_topic, 0, sizeof(g\_message));

                printf("size=%d\r\n",size);

                if(size<0){

                    sleep(3);

                    continue;

                }

                if(size>0)

                {

                    mqtt\_publish("data/rec","Accepted successfully");

                    printf("TOPIC IS:%s\r\n",buff2);

                    printf("DATA IS:%s\r\n",buff);

                    if(strstr((char\*)&buff[0],"\"LED1\":1"))

                    {

                        digitalWrite(23,HIGH);

                    }

                    if(strstr((char\*)&buff[0],"\"LED1\":0"))

                    {

                        digitalWrite(23,LOW);

                    }

                    if(strstr((char\*)&buff[0],"\"LED2\":1"))

                    {

                        digitalWrite(24,HIGH);

                    }

                    if(strstr((char\*)&buff[0],"\"LED2\":0"))

                    {

                        digitalWrite(24,LOW);

                    }

                    if(strstr((char\*)&buff[0],"\"LED3\":1"))

                    {

                        digitalWrite(25,HIGH);

                    }

                    if(strstr((char\*)&buff[0],"\"LED3\":0"))

                    {

                        digitalWrite(25,LOW);

                    }

                }

                sleep(2);

        }

            printf("\r\n");

        }

        pthread\_t id;

        pthread\_create(&id, NULL,pth\_work\_func,NULL);

        pthread\_detach(id);

编写好程序后

执行**make**进行编译

执行**export LD\_LIBRARY\_PATH=./lib/**加载库文件。

运行**./MQT\_Demo.o**。

**电脑上使用vscode编写web端代码读取数据画图**

    <!-- Chart row containers -->

    <div class="chart-row">

        <div class="chart-container" id="chartTemp"></div>

        <div class="chart-container" id="chartHumidity"></div>

    </div>

    <div class="chart-row">

        <div class="chart-container" id="chartLight"></div>

        <div class="chart-container" id="chartWater"></div>

    </div>

    <!-- Include the ApexCharts library -->

    <script src="https://cdn.jsdelivr.net/npm/apexcharts"></script>

    <!-- Include the MQTT.js library -->

    <script src="https://cdn.jsdelivr.net/npm/mqtt/dist/mqtt.min.js"></script>

    <script>

        const tempDataPoints = [];

        const humidityDataPoints = [];

        const lightDataPoints = [];

        const waterDataPoints = [];

        const maxPoints = 50;

        // Chart options for each data type

        const tempOptions = {

            chart: {

                type: 'line',

                height: 400,

            },

            series: [{

                name: 'DHT11\_T',

                data: tempDataPoints

            }],

            xaxis: {

                type: 'datetime',

                title: {

                    text: 'Time'

                }

            },

            yaxis: {

                title: {

                    text: 'DHT11\_T'

                }

            }

        };

        const humidityOptions = {

            chart: {

                type: 'line',

                height: 400,

            },

            series: [{

                name: 'DHT11\_H',

                data: humidityDataPoints

            }],

            xaxis: {

                type: 'datetime',

                title: {

                    text: 'Time'

                }

            },

            yaxis: {

                title: {

                    text: 'DHT11\_H'

                }

            }

        };

        const lightOptions = {

            chart: {

                type: 'line',

                height: 400,

            },

            series: [{

                name: 'Light',

                data: lightDataPoints

            }],

            xaxis: {

                type: 'datetime',

                title: {

                    text: 'Time'

                }

            },

            yaxis: {

                title: {

                    text: 'Light'

                }

            }

        };

        const waterOptions = {

            chart: {

                type: 'line',

                height: 400,

            },

            series: [{

                name: 'Water',

                data: waterDataPoints

            }],

            xaxis: {

                type: 'datetime',

                title: {

                    text: 'Time'

                }

            },

            yaxis: {

                title: {

                    text: 'Water'

                }

            }

        };

        // Initialize charts

        const tempChart = new ApexCharts(document.querySelector("#chartTemp"), tempOptions);

        const humidityChart = new ApexCharts(document.querySelector("#chartHumidity"), humidityOptions);

        const lightChart = new ApexCharts(document.querySelector("#chartLight"), lightOptions);

        const waterChart = new ApexCharts(document.querySelector("#chartWater"), waterOptions);

        tempChart.render();

        humidityChart.render();

        lightChart.render();

        waterChart.render();

        // Connect to the MQTT broker

        const client = mqtt.connect('wss://broker.emqx.io:8084/mqtt');

        client.on('connect', () => {

            console.log('Connected to MQTT broker');

            client.subscribe('date/pub', (err) => {

                if (err) {

                    console.error('Subscription error:', err);

                }

            });

        });

        client.on('message', (topic, message) => {

            const messageString = message.toString().trim();

            console.log('Received message:', messageString);

            try {

                // Parse JSON data

                const data = JSON.parse(messageString);

                // Update the data display area

                const dataDisplay = document.getElementById('dataDisplay');

                dataDisplay.innerHTML = `

                    <div><strong>DHT11\_T:</strong> ${data.DHT11\_T}</div>

                    <div><strong>DHT11\_H:</strong> ${data.DHT11\_H}</div>

                    <div><strong>Light:</strong> ${data.light}</div>

                    <div><strong>Water:</strong> ${data.water}</div>

                `;

                // Add new data points and update the charts

                const timestamp = new Date().getTime();

                tempDataPoints.push({ x: timestamp, y: data.DHT11\_T });

                if (tempDataPoints.length > maxPoints) {

                    tempDataPoints.shift();

                }

                tempChart.updateSeries([{ data: tempDataPoints }]);

                humidityDataPoints.push({ x: timestamp, y: data.DHT11\_H });

                if (humidityDataPoints.length > maxPoints) {

                    humidityDataPoints.shift();

                }

                humidityChart.updateSeries([{ data: humidityDataPoints }]);

                lightDataPoints.push({ x: timestamp, y: data.light });

                if (lightDataPoints.length > maxPoints) {

                    lightDataPoints.shift();

                }

                lightChart.updateSeries([{ data: lightDataPoints }]);

                waterDataPoints.push({ x: timestamp, y: data.water });

                if (waterDataPoints.length > maxPoints) {

                    waterDataPoints.shift();

                }

                waterChart.updateSeries([{ data: waterDataPoints }]);

            } catch (error) {

                console.error('Failed to parse message:', error);

                document.getElementById('dataDisplay').textContent = '数据解析失败';

            }

        });

        client.on('error', (err) => {

            console.error('MQTT client error:', err);

        });

    </script>