物联网智能家居项目 V1.0.0

1 项目背景

1.1 项目设计背景

物联网将是计算机发展的下一阶段，这个时间主要集中在“我们收集的数据类型”。

-----ARM 白皮书

英国半导体公司最近以 320 亿美元收购了 SoftBank，估计将在 2017 年至 2035 年之间建造 1 万亿 IoT 设备，为全球 GDP 增加 5 万亿美元。

------SoftBank

ARM 在所有业务领域都看到了 IoT 设备的优势：收入，利润和生产力。在本文中，食品生产和分销，制造，批发和零售以及医疗保健和社会援助的潜在产量增长了 5%。

-----ARM 白皮书

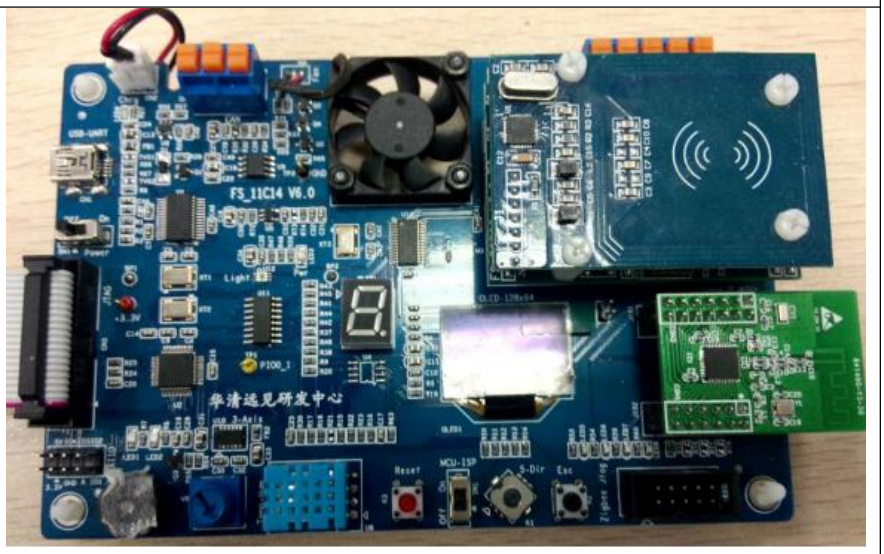
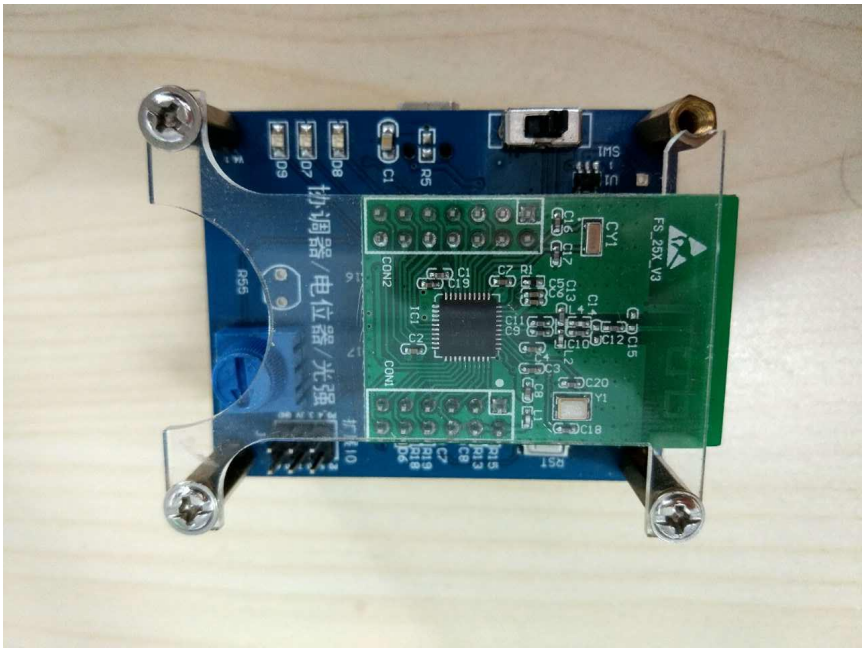
近年来，在国家政策的大力扶持和业内企业的不断努力下，中国物联网产业持续良好发展势头。技术研发取得重大进展，市场化应用稳步推进。

下图是 2017 年 5 月份物联网协会对物联网设备做出的统计表，可以看出物联网设备正在呈井喷式增长。而智能家居是物联网中一个比较典型的场景，也是学生比较感兴趣的方向。本项目即以智能家居为根本，以原“物联仓储”项目做参考，并针对山东企业技术需求，为学生呈现物联网整体开发流程，并掌握关键开发技术。

2 开发环境搭建

2.1 硬件清单

名称	描述

FS4412	 作为项目服务器终端，负责采集来自传感器采集终端的数据，处理后发送给各个客户端。
FS11C14	 模拟智能家居终端，使用 cortex-M0 处理器，带有重力、光照、温度等多种传感器，负责家庭环境信息采集。
Zigbee 协调器	

	Zigbee 协调器，负责采集终端和 FS4412 数据透传，协调器采用 USB 延长线连接到 FS4412，最终数据通过 zigbee 转串方式透传给 FS4412。
摄像头	 采集家庭视频数据，并上传给客户端
USB 数据线	若干

2.2 硬件连接

- 将 zigbee 协调器通过一根 USB 数据线连接到 FS4412 开发板上，FS4412 可任意选择 USB 口。启动开发板，系统启动完毕后打开 zigbee 协调器的开关，检查在/dev/目录下是否有 ttyUSB0 结点。如果没有，请尝试重启 zigbee 协调器。
- FS11C14 包括其上的 zigbee 等模块烧写好单片机程序（这部分可选做，默认已经烧写好可用采集程序，可直接使用），启动 FS11C14，OLED 屏幕会显示开机 logo，然后进入采集状态。屏幕会显示当前环境信息。
- 等待 zigbee 连接，协调器启动后默认三个灯亮，等待几秒后会熄灭一个指示灯，此时协调器进入等待连接状态。然后开启 FS11C14，屏幕下方也会有三个指示灯，等待几秒后会熄灭一个指示灯，如果此时协调器正常，那么会连接成功。此时协调器和 FS11C14 指示灯都会变为 1 个。此时代表连接成功。注意：一定先开启协调器，然后再开启 FS11C14，如果连接不成功请重启尝试。

2.3 软件环境说明

- **单片机**：使用 IAR、keil 等 IDE 进行开发，（这部分可选做，默认已经烧写好可用采集程序，可直接使用），开发烧写详见《单片机文档》。
- **FS4412**：使用虚拟机或服务器进行开发，u-boot、内核、文件系统可选做，直接烧写即可，烧写请查看《FS4412 智能家居烧写文档》。重点开发服务器端程序。
- **PC 客户端**：使用 QT 开发，可到 QT 官网下载开发 IDE，默认使用 5.2.1 版本进行开发。

3 整体框架设计

3.1 代码维护

- **版本控制**：组内使用 svn 或者 github 进行项目维护

- **代码规范：**组内统一一套规范，强制规范：linux 开发命名使用_分割单词；QT/c++使用首字母大写来规范；
- 文件头要写文件头注释，函数要写函数注释

如下提供一个模板：

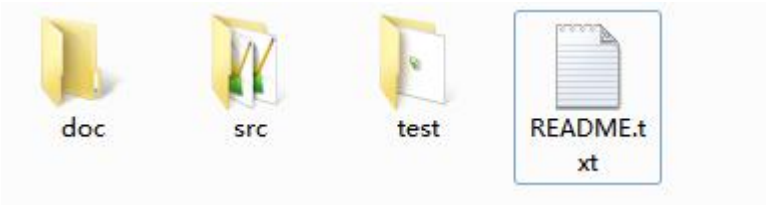
```
/**
 ** @author:        浓咖啡
 ** @date:      2017.7.7
 ** @brief:       对4412硬件进行管理操作
 ** 对于4412控制led使用了sysfs子系统，beep使用了输入子系统
 ** 这部分可以根据自己板子实际情况修改，比如自己写字符设备驱动
 */

#include "fs4412dev_manage.h"

/********************led部分**************************/
#define DEV_LED "/sys/class/leds/led2/brightness"
static int led_dev;

/**
 * @brief led 初始化
 * @return 成功true,失败false
 */
static bool led_init()
{
    led_dev = open(DEV_LED, O_RDWR);
    if(led_dev < 0){
```

- 项目源码和文档注意文件夹分布：比如 smart-home-server 端的源码存放如下：



doc 放服务器相关设计文档，src 放所有源码，test 放测试程序，README 写一些说明。

3.2 软件框架

- 为了软件可维护性及组内分工方便，必须**分文件、分模块**。尽量做到每个模块功能集中在一个.c 和.h 文件中。
- 全局变量和宏定义该如何处理？
- 每个模块设计要注意什么？
  - **模块文件命名原则：根据模块功能来决定**。比如数据库处理线程使用“pthread_database.c”来命名
  - **对于对外提供接口或者模块对外提供信息定义的模块，使用.h 来单独封装**。比如环境信息处理模块单独封装 env.h，存放环境信息结构体的定义，这种属于需要对外导出自定义结构的。再比如 4412 板子硬件操作，单独封装文件尾 fs4412dev_manage.c 和 fs4412dev_manage.h，.h 文件中主要写对外提供的硬件操作接口，例如：void beep_control(int on);
  - 注意模块变量或者函数区域限定，比如只在本文件中使用的全局变量或者函数，加入 static 变量修饰，只在本文件中使用的宏定义应该放到.c 文件中而不应该放到.h 中。
  - 使用宏编译开关来控制一些模块的开启和关闭，比如调试模块

```
#define __LOG__
```

```
#ifdef __LOG__
#define LOG(format,...) printf(__FILE__ ":%d: " format "\n", __LINE__, ##__VA_ARGS__)
#else
#define LOG(format,...)
#endif

#define err_log(format,...) printf(__FILE__ ":%d: " format "\n",      __LINE__, ##__VA_ARGS__)
```

思考上面宏定义的文件位置？
如何使用上面的宏？

3.3 通信协议

注：以下统一对 M0 端称为下位机、FS4412 称为服务器、QT 上位机软件称为客户端。

凡是涉及到通信必然要规定通信协议，项目中现涉及通信的有两处：下位机采集的数据传递给服务器、服务器处理后的数据发送给客户端。这属于内部通信协议，通常由公司或者项目组内自己规定。

- **下位机和服务器**

下位机负责采集传感器数据，采集数据后通过 zigbee 透传给服务器。两者交互使用如下结构体：

```
struct env_info {
    uint8_t head[3];    //标识位 st:
    uint8_t type;               //数据类型
    uint8_t snum;               //房间编号
    uint8_t temp[2];    //温度
    uint8_t hum[2];             //湿度
    uint8_t x;                  //三轴信息
    uint8_t y;
    uint8_t z;
    uint32_t ill;               //光照
    uint32_t bet;               //电池电量
    uint32_t adc;               //电位器信息
};
```

对结构体部分成员进行说明：

head：信息头标示，可以作为数据报之间区分的标识。下位机默认发送”st:”三个字符。
type：数据类型，用来区分不同的信息，例如环境信息、刷卡信息等。
snum：房间编号，代表不同的房间信息。
temp[0]：温度整数部分
temp[1]：温度小数部分
hum[0]：湿度的整数部分
hum[1]：湿度的小数部分
其它信息请参考注释。

- **服务器和客户端**

服务器拿到数据后需要进行处理，比如把温湿度转换为真实数据，方便客户端收到数据后直接进行显示等处理。两者交互使用如下结构体（这部分仅做参考，可以组内自己重新定义结构）：

```
struct conver_env_info {
    int snum;           //仓库编号
    float temperature;  //温度
    float humidity;             //湿度
    float ill;          //光照
    float bet;          //电池电量
    float adc;          //电位器信息

    signed char x;                          //三轴信息
    signed char y;
    signed char z;
};
```

如需考虑其它数据，如刷卡信息等需要继续完善结构体。

- **服务器对下位机的命令控制**

客户端部分操作需要控制下位机，采用命令格式如下：

7	6	5	4	3	2	1	0
房间号		设备编号		操作设备			

房间编号	
0x40	XXX
0x80	客厅
0xc0	XXX

设备编号		操作掩码	
0x00	风扇	0x00	关闭
		0x01	1 档
		0x02	2 档
		0x03	3 档
0x10	蜂鸣器	0x00	关闭
		0x01	打开
		0x02	自动报警关闭
		0x03	自动报警打开
0x20	LED	0x00	关闭
		0x01	打开
0x30	数码管	0x00~0x09	显示 0~9 数字
		0x0f	关闭数码管

仓库编号 + 设备编号 + 操作掩码 = 命令

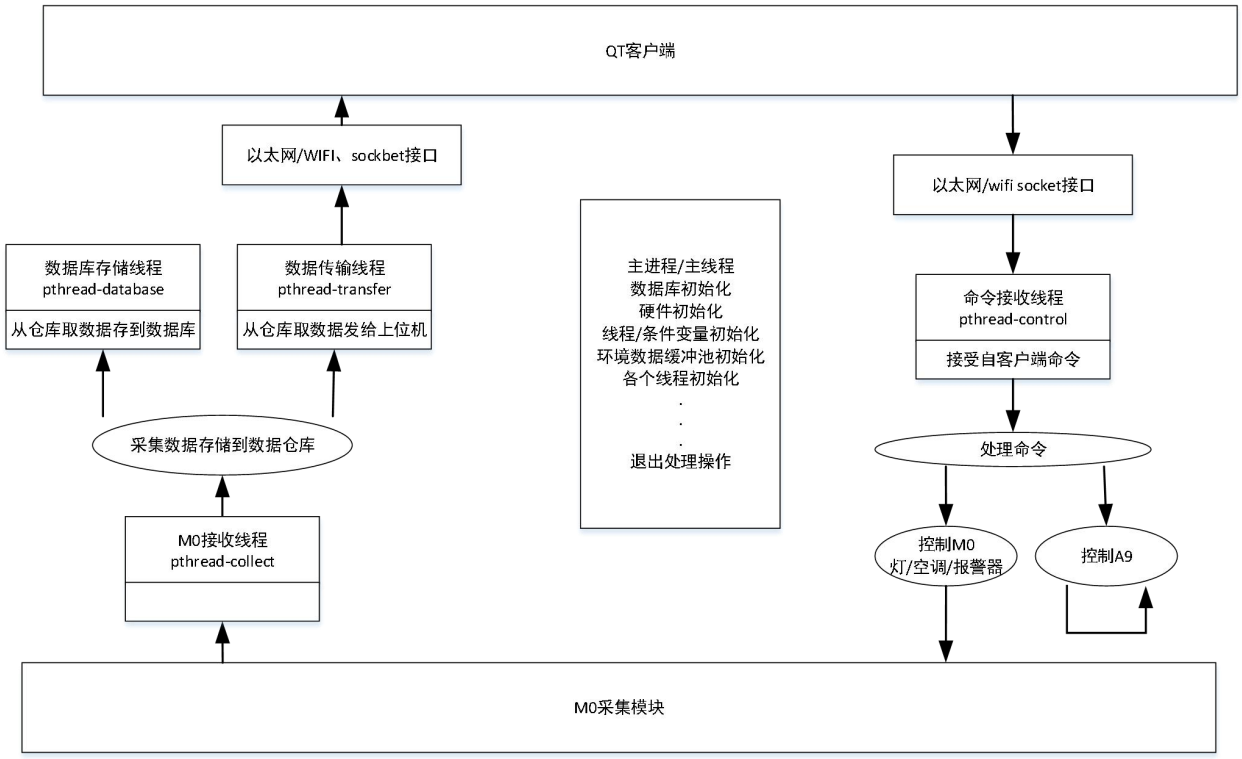
注：下位机默认代表客厅（0x80）

例如：

0x80 + 0x00 + 0x01 = 0x81　客厅空调一档

0x80 + 0x20 + 0x01 = 0xA1　客厅灯打开

3.4 整体框架图



4 数据上行

4.1 整体思路

下位机采集数据，通过 zigbee 透传给服务器，服务器收到数据进行相应处理，处理完成后通过网线或者无线使用 socket 接口发送给客户端。

4.2 搭建基本服务

- 假如通信数据是下图结构体，建立基本 socket 通信，服务器端**模拟产生结构体对应的数据**，写一个测试客户端来测试。

```
struct conver_env_info {
    int snum;          //仓库编号
    float temperature;    //温度
    float humidity;       //湿度
    float ill;         //光照
    float bet;         //电池电量
    float adc;         //电位器信息

    signed char x;              //三轴信息
    signed char y;
    signed char z;
};
```

- 思考如何进行数据模拟？模拟功能实际上是在模拟什么硬件？
- 写一个 QT 客户端，接收服务器消息并做简单显示。

4.3 设计思路

- 服务器不断接收模拟数据包，一方面要把最新数据包发给客户端显示，一方面要把数据存储到数据库，怎么实时达到两个需求或者以后添加更多需求？
- 假设下位机每 100ms 产生一次数据，客户端界面每 1s 产生一次数据。
- 为了协调各个线程的速率，设计一个数据仓库。
- 数据库存储：只要仓库有数据就进行存储。
- 如何做到给客户端的数据是最新的？
- 如何做到把所有数据都保留并存储到数据库？
- 对于数据仓库，选择什么样的数据结构达到上述两点要求？
- 如何防止数据仓库过大？
- 怎么防止多线程同时操作仓库时的竞争问题？
- 数据存储线程如何及时获知数据到来？
- 数据存储速度比上传快，怎么保证上传每次都有数据？
- 为防止数据库数据量过大，数据存储每隔 60s 存储一次，其它数据包丢掉。如何操作？
- 如何支持多个客户端？当没有客户端时候上行线程该如何处理？当第一个客户端到达时候服务器模型线程和数据上行线程应该做怎样的同步？
- 把模拟数据替换成真实设备，完成整个上行过程。

5 数据下行

5.1 整体思路

用户操作客户端，执行远程控制。客户端封装命令，下发给服务器。服务器收到命令后进行处理，控制下位机进行相应的响应。

5.2 搭建基本服务

FS4412 除了作为服务器功能，还当做一个终端（模拟卧室），通信协议部分的定义只是针对 M0 控制。故还需要统一通信结构同时兼容 FS4412 和 M0 控制。

- 思考如何定义数据结构来实现上述需求？

5.3 设计思路

- FS4412 需要对哪些硬件进行初始化？如何封装？
- 服务器如何通过 zigbee 模块把命令发送给下位机？
- 写一个服务器模拟程序，发送命令给下位机测试下位机是否工作正常
- 写一个服务器模拟程序，接收来自上位机的命令并打印
- 整合代码，封装到下发线程中
- 尝试重新编写 FS4412 板端驱动

5.4 智能控制

可以模拟智能家居应用场景，优化产品体验。例如当检测到温度过高时，会在界面显示异常并自动打开空调进行降温，当达到预先设定正常值后又可以自动关闭空调。诸如此类优化还有很多，可自行发挥。所以需要提供一个界面进行设置限制值并能保存。

	最新数值	正常范围	状态
环境温度	27.00	18-30	正常
环境湿度	14.00	10-20	正常
环境光照	63.00	100-1000	异常
电池电量	3.08	3-4	正常
模拟电压	1.70	1-100	正常

6 摄像头监控

6.1 库移植

根据《FS4412 mjpg-streame 移植.pdf》完成 jpeg 和 mjpg-streamer 库的移植，移植过程中需要根据自己实际情况修改配置，不能照抄文档。

6.2 库测试

终端输入：

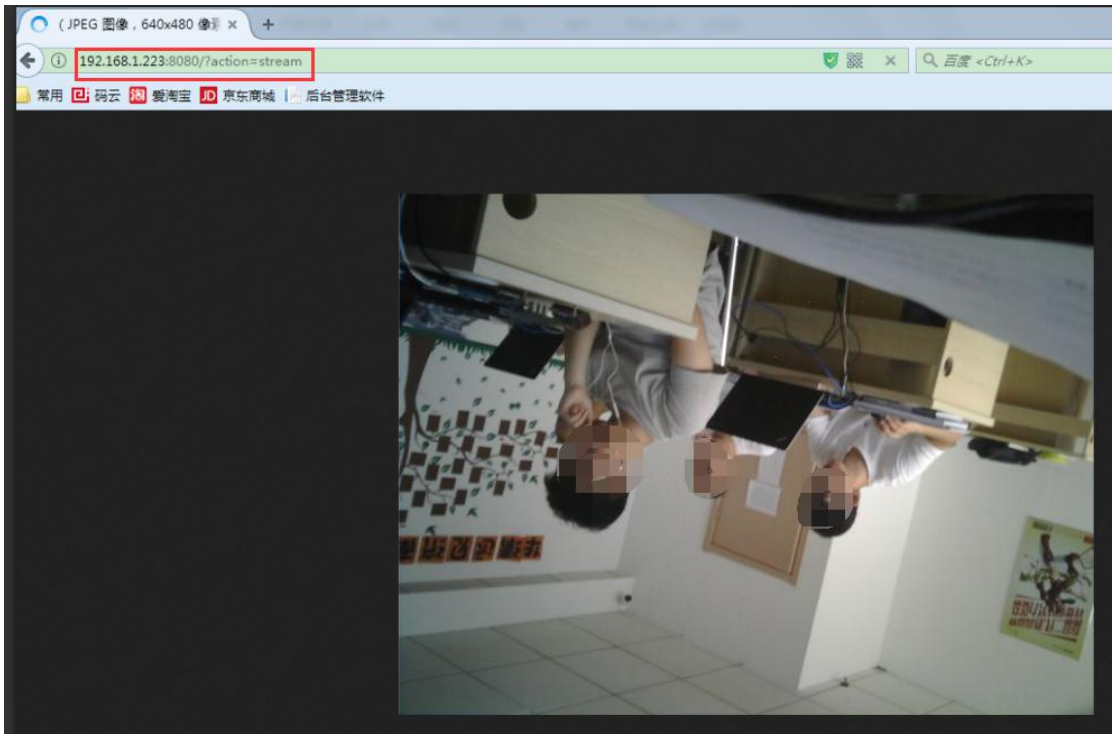
```
mjpg_streamer -i "/mjpg/input_uvc.so -y -d /dev/video0" -o "/mjpg/output_http.so -w /www"
```

如果程序正常运行，未提示错误，打开浏览器（最好使用火狐浏览器，部分浏览器可能不支持），输入：

http://192.168.1.223:8080/?action=stream

IP 根据自己开发板自行修改。

如果正确移植，此时在浏览器能看到视频图像。



6.3 客户端集成

完成浏览器显示后，再集成到 QT 客户端。并增加拍照、图片浏览等功能。

7 持续优化

7.1 程序自启动

产品开发交给用户后，使用肯定越简单越好，开发板需要的服务需要通过配置文件配置自启动操作。

可以修改**/etc/profile** 或**/etc/init.d/rcS** 文件，把需要自启动的服务配置进去。例如摄像头服务，可以做对**/etc/profile** 做如下修改：

```
#!/bin/sh
export HOSTNAME=farsight
export USER=root
export HOME=root
export PS1="[$USER@$HOSTNAME \W]\# "
PATH=/bin:/sbin:/usr/bin:/usr/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH

export LD_LIBRARY_PATH=/mjpg:$LD_LIBRARY_PATH

PATH=$PATH

mjpg_streamer -i "/mjpg/input_uvc.so -y -d /dev/video0" -o "/mjpg/output_http.so -w /www" &
~
~
```

7.2 服务器保活机制（下一版本更新）

如果一个客户端连接后，长期不进行数据交换，那么可以认为此客户端为死连接，那么需要检测这种客户端并处理。参考心跳包、keepalive 机制完成这部分功能。

7.3 动态变化图（下一版本更新）

客户端以动态曲线图形式直观显示温湿度变化趋势，也可选择某一时段进行查询显示。qcustomplot 开源库使用。

7.4 远程升级（暂未实现）

可以通过客户端进行远程升级服务器主程序，涉及多进程编程、进程通信等知识。

7.5 人脸识别（下一版本更新）

移植 opencv 实现。