**京东平台与broadlink模块**

1. **需求目标**

复杂的嵌入式WIFI的开发，云平台开发，复杂的网络交互，将大多数的传统硬件厂商挡住物联网智能硬件门外。

京东平台和broadlink模块的组合就是为了解决这个问题。

京东云服务器平台与broadlink的模块组合之后，传统厂商无需了解上述复杂的开发工作，即可简单的实现产品的开发。

1. broadlink模块提供的功能



Broadlink模块的功能：

1. 直接支持京东云的接口定义，感知数据状态上传，
2. 与云服务器的长连接，接收并响应控制命令。
3. 提供标准的模块“设备发现与检测”功能

通过标准的mDNS接口，实现设备的发现检测功能

mDNS返回的内容： VENDOR-PRODUCTNAME-MAC地址

1. 提供HTTP REST API接口，当用户在京东云上面重新create新的产品，将获取到的产品ID和产品秘钥保存在设备上，以改产品形式运行。
2. 提供API接口，对产品名称，厂家，型号等上传配置，上传完成之后，支持LOCK锁定功能。
3. 提供smartconfig设备上网配置功能
4. 提供HTTP REST API， 实现对模块的硬件功能配置，局域网直接控制
5. 提供多种预定义的硬件功能，GPIO UART ADC PWM等。

用户在使用过程中，无需对模块进行编程，只需要在京东云上注册产品，将注册信息上传模块，同时选择预定义的逻辑，如GPIO控制，UART接口，ADC，即可完成整个智能控制的所有网络交互，嵌入式WIFI开发等功能。

1. UART透传解决方案

上图为使用UART功能，实现透传的方案。

原有厂商，只需要将UART连接到broadlink的WIFI模块，注册并上传相应的产品信息，同时使用京东云和broadlink本地控制的REST API接口，就可以实现通过APP上层到其原有设备UART口直接的透传控制，状态获取等功能。

1. GPIO ADC PWM等逻辑控制解决方案



如开关控制，PWM，ADC采样逻辑控制，只需要将外围硬件（如继电器，ADC温敏电阻，PWM电机控制口）与broadlink模块相连，只需要在HTTP REST API上面调用预定义的格式，即可完成现有状态的获取，报告功能。

1. 模块上面的配置存储定义

厂家预定义的配置、用户可修改的配置。

Broadlink提供2类配置定义，厂家预定义配置，这是厂家使用broadlink的模块之后，可以动态配置的区域，当厂家完成配置之后，可以启动lock之后，配置信息不能再被修改。

用户可修改的配置，这是另一块配置区域，这块配置区域，运行用户在使用过程中修改，但是RESET之后，会还原成厂家预定义信息。

1、厂家预定义的配置

product\_id: 京东云上面申请的产品ID；

product\_secret：获取到的产品secret

cloud\_addr: 云端域名

cloud\_port: 端口

vendor：厂家名称

product\_name:产品名称

GPIO 默认状态

UART默认波特率配置

……

2、用户自定义的配置

Feed\_id : 这是京东云根据产品ID和设备的MAC地址，返回给设备端的ID

Access\_key: 京东云返回的access key

Local\_lock: 锁定，当请求用户配置时，不返回access\_key，返回空

GPIO修改的配置

UART 修改配置信息

UART命令超时时间，即从发送命令，到被控设备返回的时间，单位ms

UART 超时，重试次数。

Network：网络部分的配置

……

1. HTTP REST API控制
2. 厂家配置区域信息修改

|  |  |
| --- | --- |
| 请求方法 | POST |
| 请求URL | /vendor\_config |
| 请求头 |  |
| 请求参数  (JSON) | {  " product\_id " : "京东云上面申请的产品ID  " product\_secret " : "获取到的产品secret ", //(必填)  " vendor " : "厂家名称",  " product\_name " : "产品名称",  " GPIO0 " : "on/off", //默认的多个GPIO配置  ……  " UART " : "115200/n/8/1", //配置串口波特率115200 8数据位 1停止位  " UART\_TIMEOUT " : "100", 超时 100ms  " UART\_REPEAT " : "3", 超时重试次数 3次  } |
| 返回状态码 | 200 |
| 返回JSON | {  "code" : "200",//状态码  "error\_msg" : "OK或错误信息",  } |

1. 获取厂家配置信息

|  |  |
| --- | --- |
| 请求方法 | GET |
| 请求URL | /vendor\_config |
| 请求头 |  |
| 请求参数  (JSON) | 无 |
| 返回状态码 | 200 |
| 返回JSON | {  " lock " : "on/off" //厂家配置信息是否上锁  " product\_id " : "京东云上面申请的产品ID  " product\_secret " : "获取到的产品secret ", //(必填)  " vendor " : "厂家名称",  " product\_name " : "产品名称",  " GPIO0 " : "on/off", //默认的多个GPIO配置  ……  " UART " : "115200/n/8/1", //配置串口波特率115200 8数据位 1停止位  " UART\_TIMEOUT " : "100", 超时 100ms  " UART\_REPEAT " : "3", 超时重试次数 3次  } |

1. 锁定厂家配置信息

|  |  |
| --- | --- |
| 请求方法 | GET |
| 请求URL | /vendor\_lock |
| 请求头 |  |
| 请求参数  (JSON) | 无 |
| 返回状态码 | 200 |
| 返回JSON | {  "code" : "200",  "error\_msg" : "错误信息",//code=200为OK,code!=200时为错误信息  } |

1. 配置用户配置信息

|  |  |
| --- | --- |
| 请求方法 | POST |
| 请求URL | /user\_config |
| 请求头 | GPIO修改的配置  UART 修改配置信息 |
| 请求参数  (JSON) | {  " GPIO0 " : "on/off", //默认的多个GPIO配置  ……  " UART " : "115200/n/8/1", //配置串口波特率115200 8数据位 1停止位  " UART\_TIMEOUT " : "100", 超时 100ms  " UART\_REPEAT " : "3", 超时重试次数 3次  "network" :  {  “ssid”: “SSID”,  “psk”: “PSK”,  “ipaddr”: “192.168.1.21”, //ip地址为空，表示DHCP方式  “mask”: “255.255.255.0”,  “gateway”:”192.168.1.1”  }  } |
| 返回状态码 | 200 |
| 返回JSON | {  "code" : "200",//状态码  "error\_msg" : "OK或错误信息",  } |

1. 获取用户配置信息

|  |  |
| --- | --- |
| 请求方法 | GET |
| 请求URL | / user\_config |
| 请求头 |  |
| 请求参数  (JSON) | 无 |
| 返回状态码 | 200 |
| 返回JSON | {  " feed\_id " : "这是京东云根据产品ID和设备的MAC地址，返回给设备端的ID "  " Access\_key" : "京东云返回的access key  " Local\_lock" : "定，当请求用户配置时，不返回access\_key，返回空"  " GPIO0 " : "on/off", //默认的多个GPIO配置  ……  " UART " : "115200/n/8/1", //配置串口波特率115200 8数据位 1停止位  " UART\_TIMEOUT " : "100", 超时 100ms  " UART\_REPEAT " : "3", 超时重试次数 3次  "network" :  {  “ssid”: “SSID”,  “psk”: “PSK”,  “ipaddr”: “192.168.1.21”, //ip地址为空，表示DHCP方式  “mask”: “255.255.255.0”,  “gateway”:”192.168.1.1”  }  } |

1. 设置GPIO的状态

|  |  |
| --- | --- |
| 请求方法 | POST |
| 请求URL | /gpio |
| 请求头 | access\_key=ACCESS\_KEY |
| 请求参数  (JSON) | {  " 0 " : "on/off " //设置GPIO0 on/off    ……  } |
| 返回状态码 | 200 |
| 返回JSON | {  " 0 " : "on/off " //设置GPIO0 on/off    } |

1. 串口发送命令

想串口发送命令，发送命令之后，等待100ms获取串口的响应内容，并返回。

|  |  |
| --- | --- |
| 请求方法 | POST |
| 请求URL | /uart |
| 请求头 | access\_key=ACCESS\_KEY |
| 请求参数  (JSON) | {  " data " : "发送UART数据内容”  } |
| 返回状态码 | 200 |
| 返回JSON | {  "data " : "返回的数据内容"  } |

1. 模块版本号

|  |  |
| --- | --- |
| 请求方法 | GET |
| 请求URL | /version |
| 请求头 | access\_key=ACCESS\_KEY |
| 请求参数  (JSON) | 无 |
| 返回状态码 | 200 |
| 返回JSON | {  "version " : "1120"  } |

1. 升级到最新版本

将模块版本升级到最新

|  |  |
| --- | --- |
| 请求方法 | GET |
| 请求URL | /update |
| 请求头 | access\_key=ACCESS\_KEY |
| 请求参数  (JSON) | 无 |
| 返回状态码 | 200 |
| 返回JSON | {  "version " : "1120"  } |

1. UART数据交互的Json定义

模块 -> 其他被控设备 控制命令：

**{**

**"controll": "xxxxxxxxx" //透传的控制命令**

**}**

其他被控设备-> 模块 响应控制命令：

**{**

**"response": "xxxxxxxxx" //返回响应的结果**

**}**

模块主动上报目前传感设备的状态

{

"post" : [ {

"stream\_id" : "example",

"datapoints":[

{"at":"2013-04-22T00:35:43+08:00","value":"xx"},

{"at":"2013-04-22T00:55:43+08:00","value":"xx"},

{"at":"2013-04-22T01:15:43+08:00","value":"xx"},

{"at":"2013-04-22T01:35:43+08:00","value":"xx"}

],

},

{

"stream\_id" : "key",

"datapoints":[

{"at":"2013-04-22T00:35:43+08:00","value":"xx"},

{"at":"2013-04-22T00:55:43+08:00","value":"xx"},

{"at":"2013-04-22T01:15:43+08:00","value":"xx"},

{"at":"2013-04-22T01:35:43+08:00","value":"xx"}

],

},

{

"stream\_id" : "datastream",

"datapoints":[

{"at":"2013-04-22T00:35:43+08:00","value":"xx"},

{"at":"2013-04-22T00:55:43+08:00","value":"xx"},

{"at":"2013-04-22T01:15:43+08:00","value":"xx"},

{"at":"2013-04-22T01:35:43+08:00","value":"xx"}

],

}

]  
}

模块接收到需要上报的传感器之后，发送到京东云服务器，并返回结果。

{

"code" : "200",//状态码

"error\_msg" : "OK或错误信息"

}

被控设备可以查询京东云上面的数据：

{

"get" : " FEED\_ID ",//获取stream ID的数据

"stream\_id" : " STREAM\_ID ",//获取stream ID的数据

}

返回：

{

"code" : "200",//状态码

"error\_msg" : "OK或错误信息"

data:{

"stream\_id":"example",

"current\_value":"500",

"at":"2013-05-06T00:30:45+08:00"

}

}

发送复位命令给模块，进行网络配置，用户配置复位

{

"command" : " reset ",//获取stream ID的数据

}

返回：

{

"code" : "200",//状态码

"error\_msg" : "OK或错误信息"

}

1. 底层软件系统设计

主要的业务处理逻辑：HTTP REST SERVER， SSL SOCKET，UART interface



主要的模块与实现

1. UART模块处理

控制命令串行化，发送命令之前，加锁，等待返回结果之后，解锁。

（第二阶段，完成数据上报过程，上报过程，使用线程接收UART数据，判别为上传数据之后，上报给京东云。）

1. HTTP server线程
2. SSL socket长连接，

创建线程一直接收socket数据，并响应。

同时提供提供接口，可以通过SSL socket上传数据，heatbeat等，注意加锁。