浙江大学
Zhejiang University

OneLink Tutorial

董 玮

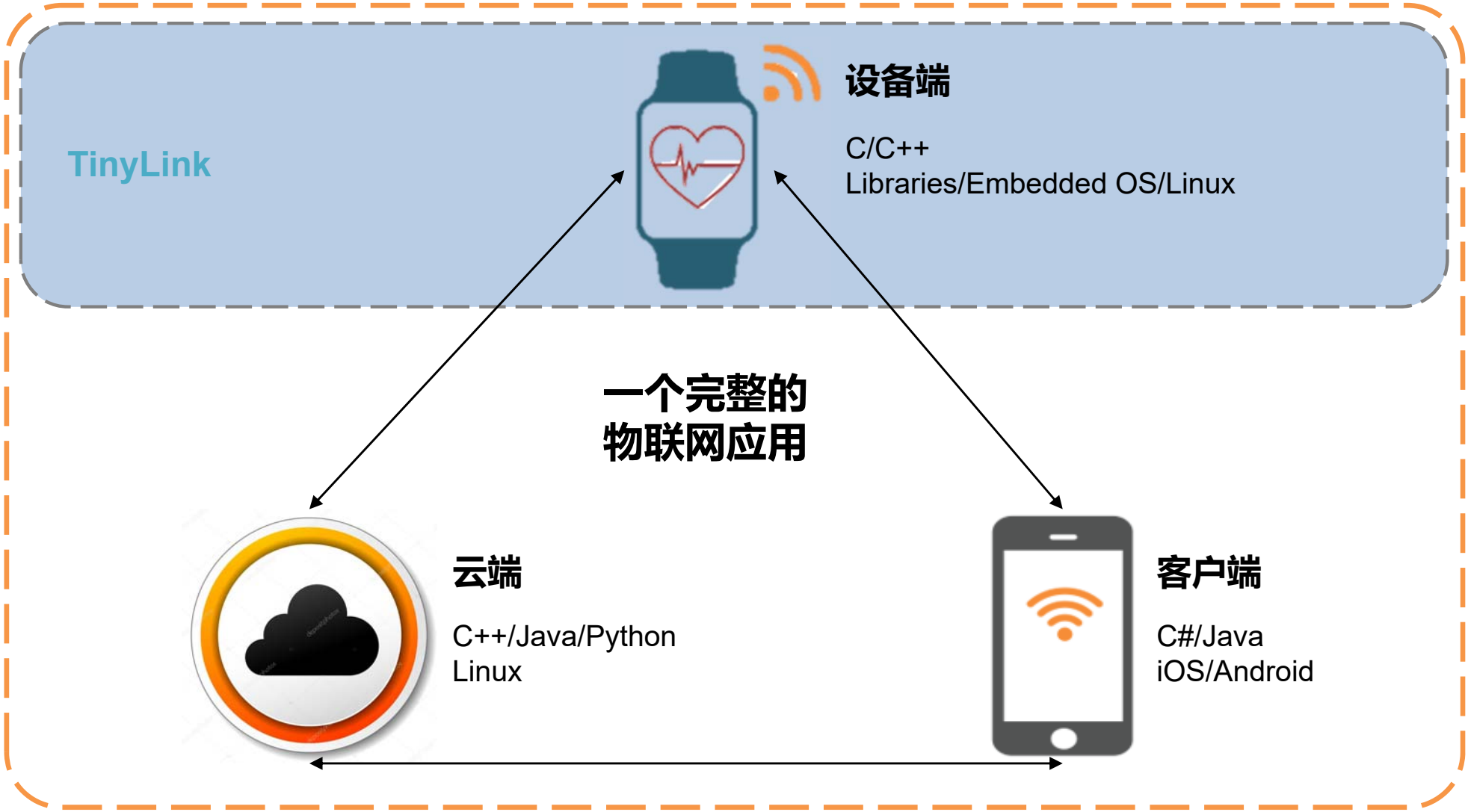
浙江大学

目录

- OneLink概括
  - TinyLink的限制
  - OneLink的特点
- OneLink使用
  - OneLink使用案例
  - OneLink语法和API

TinyLink的限制

- TinyLink缺少完整IoT应用所包含的云端和客户端代码
- TinyLink缺少多设备应用之间的交互，如数据传递和控制



OneLink智能花盆应用

```
TinyApp SmartPlant {
Interface:
  TL_Data SOIL_H, LIGHT, TEMP;
Program:
  void setup() {
    TL_Connector.bind(TL_WiFi);
    TL_WiFi.join("SSID", "PASSWD");
    LIGHT.bind(TL_Light);
    TEMP.bind(TL_Temperature);
    SOIL_H.bind(TL_Soil_Humidity);
  }
} SP;
```

设备端

```
TinyApp VPanel@Mobile {
Program:
  UI_Chart C1, C2, C3;
  UI_Text T;
  void setup() {
    C1.bind(SP::LIGHT);
    C2.bind(SP::TEMP);
    C3.bind(SP::SOIL_H);
    T.bind(PH::Status);
    TL_UI.append({C1,C2,C3});
  }
} VP;
```

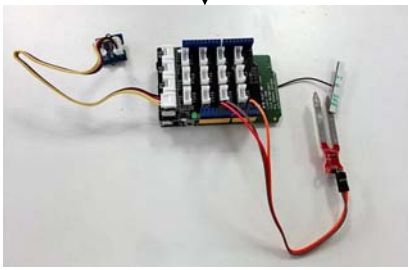
客户端

多设备交互

```
Policy PlantHumidity {
Interface:
  TL_Event Status("Dry", "Wet");
Rule:
  If (SP::SOIL_H.last() < 100) {
    Status.triggerOnce("Dry");
  } Else {
    Status.triggerOnce("Wet");
  }
} PH;
```

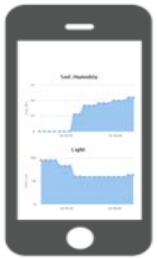
云端

TinyLink



TinyLink 应用

OneLink



WebView 应用

OneLink

规则配置

OneLink特点

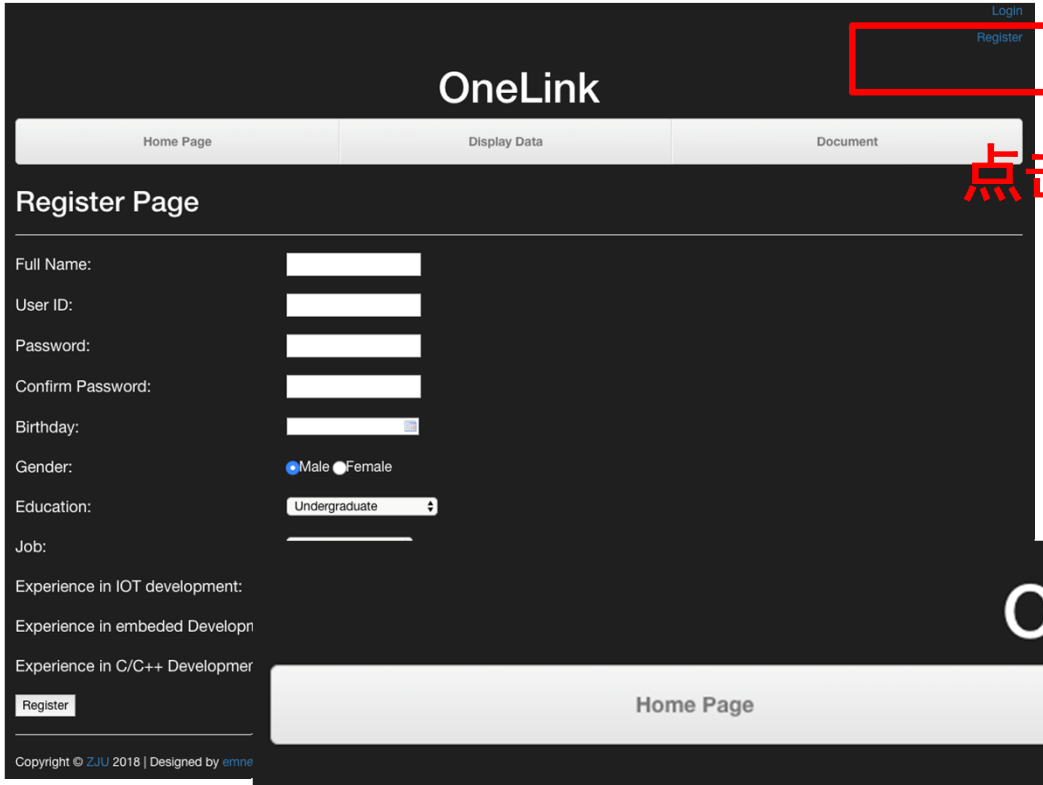
- 向下兼容：兼容TinyLink代码
- 简单易用
  - 编程模型和API简单易懂
  - 多端交互代码编写简单
- 表达力强
  - 可在一份代码中编写三端代码
  - 可以表达复杂IFTTT语法
- 扩展性好
  - 允许用户添加开发板、外设及其库代码
  - 为用户提供客户端代码模板，允许用户修改上传并使用

目录

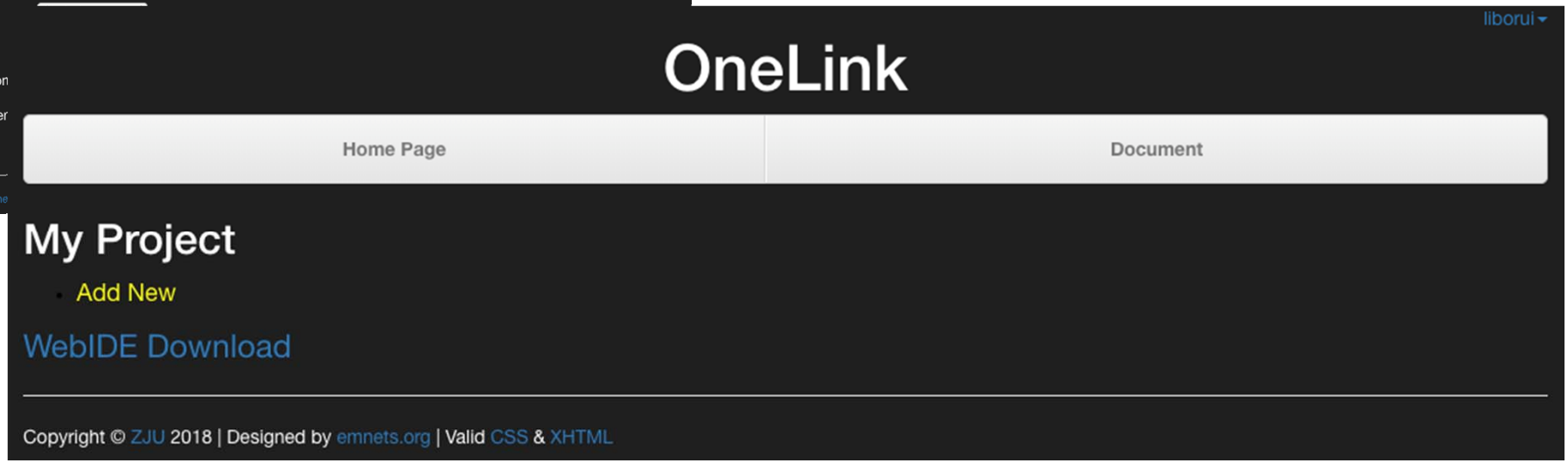
- OneLink概括
  - TinyLink的限制
  - OneLink的特点
- OneLink使用
  - OneLink使用案例
  - OneLink语法和API

准备1：注册并熟悉OneLink网页平台

- http://10.214.147.171/onelink/



点击注册



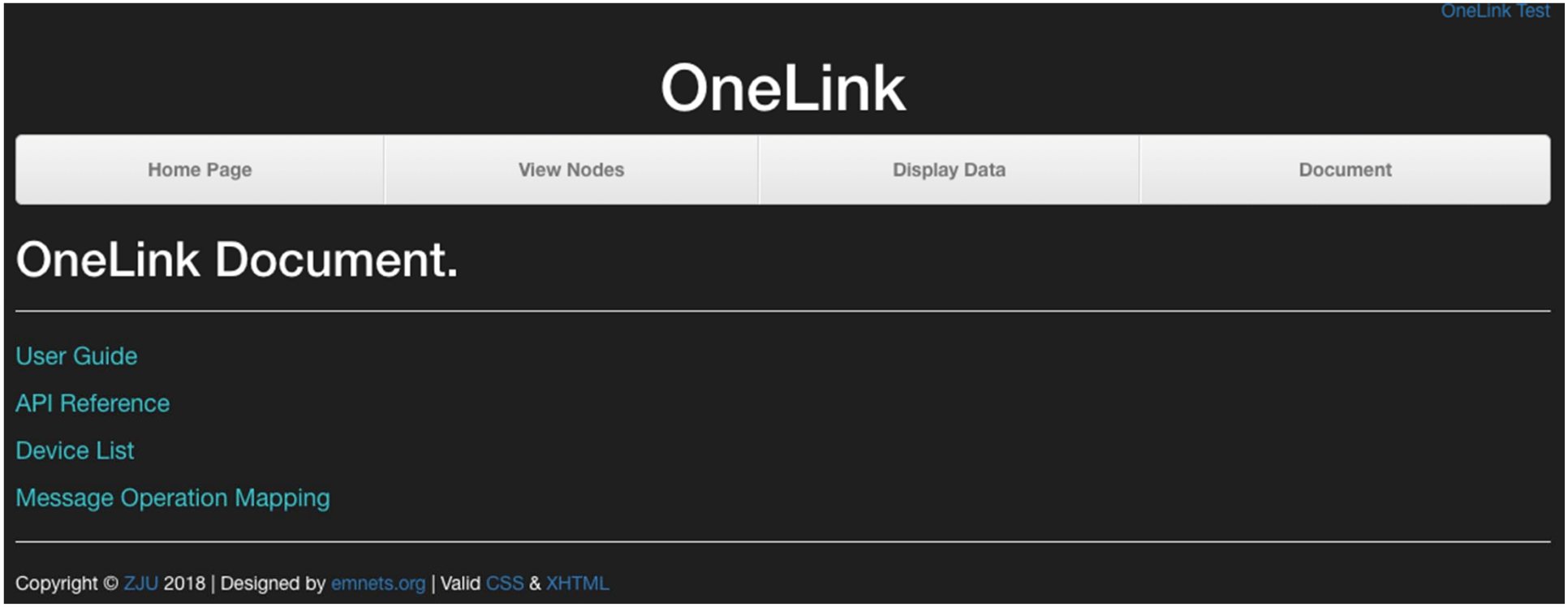
注册完成后进入主页面

准备2：下载、安装一键烧写客户端

- 在个人主页（Home Page）出下载客户端并完成安装。
- 客户端中集成了所需的驱动和一键烧写的客户端

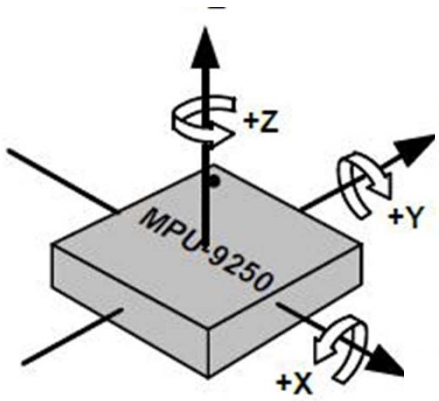
准备3：阅读OneLink文档

- 通过阅读本Tutorial与OneLink网站中Document/API Reference页面，用户可完整的掌握使用OneLink开发的方法



说明用例：智能感应灯

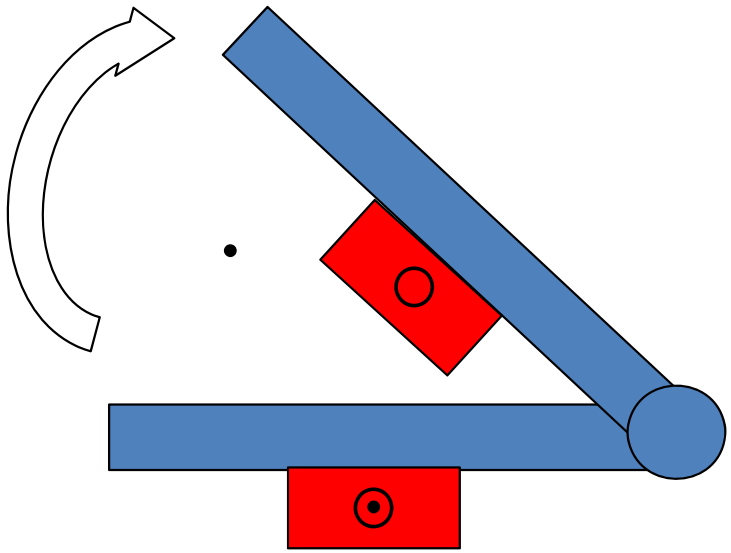
- **适用场景：**
  - 用户进入家门的时候将家中灯打开
    - 判断条件：门打开且家中光强很小
  - 用户离开家的时候将家中灯打开
    - 判断条件：门关闭且周围光强较大
  - 用户可手动关灯
- **开门/关门检测：使用陀螺仪**



陀螺仪三轴正方向图

开门（顺时针）：
Gyro_z < 0

关门（逆时针）：
Gyro_z > 0



俯视图

步骤①：编写程序代码

❑ 本地文本编辑器编写智能门灯程序 - 1

```
TinyApp SmartDoor{   //TinyApp名为SmartDoor
Interface:
    TL_Event Ev({“Open”, “Close”}); // 声明Event
Program:
    double gyro_z[5] = {0.0}; //一些全局变量
    int i = 0;
    void setup() {
        //设置连接类型与参数
        TL_Connector.bind(TL_WiFi);
        TL_WiFi.join("EmNets-301", "eagle402");
        TEMP.bind(TL_Temperature);
}
    void loop(){
        TL_Gyro.read(); //读取Gyro读数
        gyro_z[i] = TL_Gyro.data_z();
        i++;
        if(i == 5){
            i = 0;
        }
        double sum = 0.0;
        double average = 0.0;
        for(int j=0;j<5;j++){
            sum += gyro_z[j];
        }
        average = sum/5;
        if(average < -10 && sum < -150){
            Ev.trigger("Open");
            TL_Time.delayMillis(1000);
            TurnOnLED();
            for(int j=0; j<5; j++){gyro_z[j] = 0;}
        } else if (average > 10 && sum > 150){
            Ev.trigger("Close");
            TurnOffLED();
            TL_Time.delayMillis(1000);
            for(int j=0; j<5; j++){gyro_z[j] = 0;}
        }
        TL_Time.delayMillis(100);
    }
}SD; //此TinyApp的实例为SD
```

步骤①：编写程序代码

❑ 本地文本编辑器编写智能门灯程序 - 2

```
TinyApp SmartLED{  //TinyApp名为SmartLED
Interface:
    TL_Data LIGHT; //声明TL_Data
    bool TurnOnLED(); //声明TL_Service
    bool TurnOffLED();
Program:
    void setup() {
        TL_Connector.bind(TL_WiFi);
        TL_WiFi.join("EmNets-301", "eagle402");
        //将TL_Data绑定至对应的TinyLink API
        LIGHT.bind(TL_Light);
    }
    bool TurnOnLED() {  //实现TL_Service逻辑
        TL_Bulb.turnOn();
        return true;
    }
    bool TurnOffLED() {
        TL_Bulb.turnOff();
        return true;
    }
}SL1, SL2; //此TinyApp有两个实例：SL1和SL2
```

```
TinyApp SmartDoorPanel@Mobile{
Program:
    TL_Button B1, B2; //声明TL_Button
    TL_Chart C1; //声明TL_Chart
    TL_Text T1; //声明TL_Text
    // 调整具体显示元素的参数
    void setup(){
        B1.setText("SL1 board LED Turn On!");
        B2.setText("SL1 board Turn Off!");
        T1.bindData(SD::Ev);
        C1.bindData(SL1::LIGHT);
        TL_UI.append({B1, B2, T1, C1});
    }
    // 实例化Button按下的动作
    void B1.isPressed(){
        SL1::TurnOnLED();
    }
    void B2.isPressed(){
        SL1::TurnOffLED();
    }
}CP; //此TinyApp实例名为CP
```

步骤①：编写程序代码

❑ 本地文本编辑器编写智能门灯程序 - 3

```
Policy IfNeedLED{  //定义云端规则
Rule:
    If (SD::Ev.isTriggered("Open") ||    SL1::LIGHT.last() < 100){
        For (L In {SL1, SL2}){
            L::TurnOnLED();
        }
    }
    If (SD::Ev.isTriggered("Close") || SL1::LIGHT.last() > 100){
        For (L In {SL1, SL2}){
            L::TurnOffLED();
        }
    }
}INL;
```

总计：

TinyApp： SmartDoor(SD)，SmartLED(SL1, SL2)

数据展示页面： ControlPanel@Mobile(CP)

Policy: IfNeedLED(INL)

步骤②：建立项目、上传代码并编译

- 在主界面新建项目，并填写项目信息、上传代码

步骤③：查看编译结果

- 编译结果包括设备端代码、移动端显示和云端规则三部分

设备端（下载cpp代码、一键烧写）

云端规则
（右侧的复选框可启停规则）

移动端（代码中定义的TinyApp@Mobile）

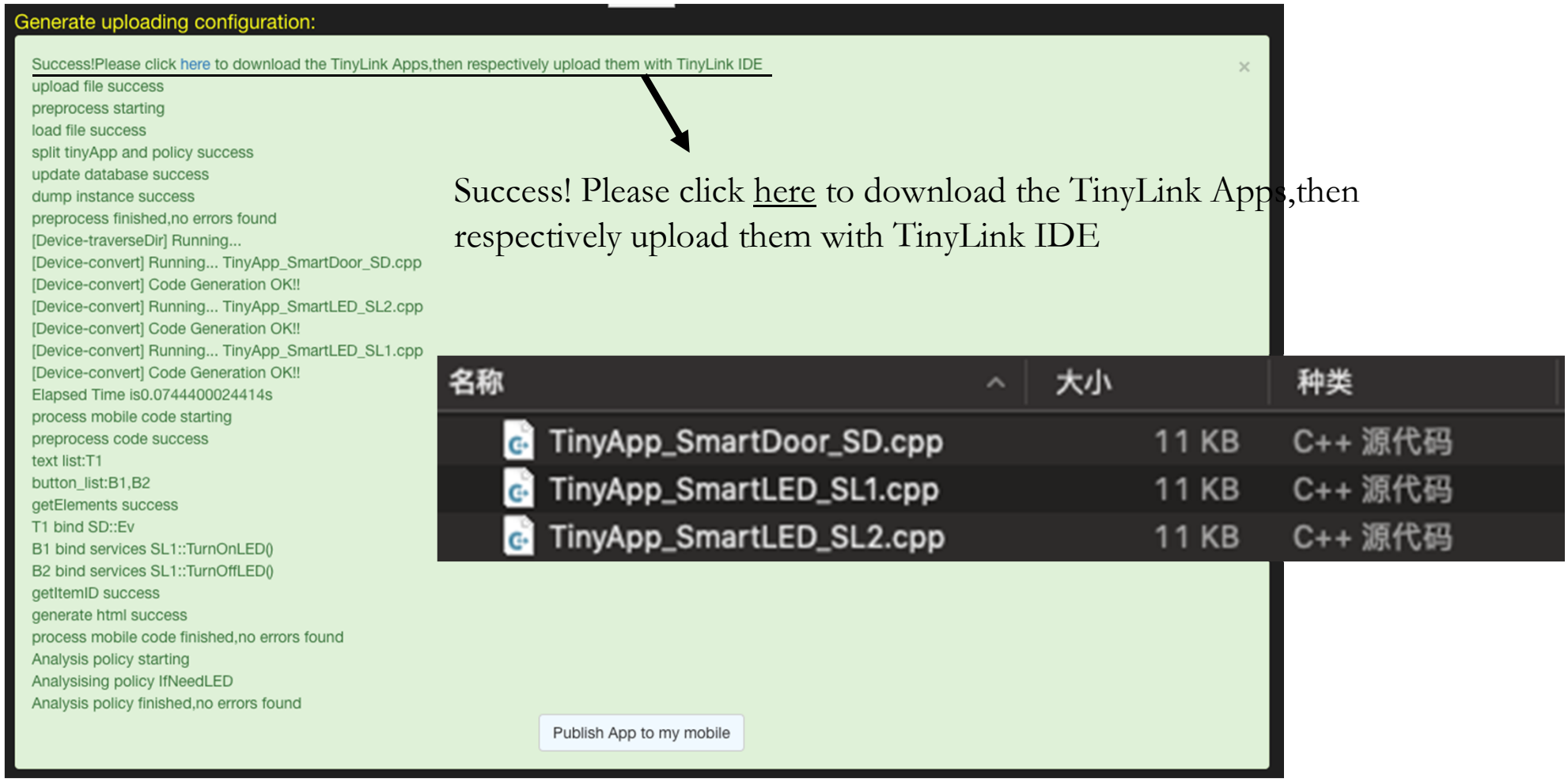
显示Success即为编译成功

步骤③：查看编译结果

- 编译结果包括设备端代码、移动端显示和云端规则三部分

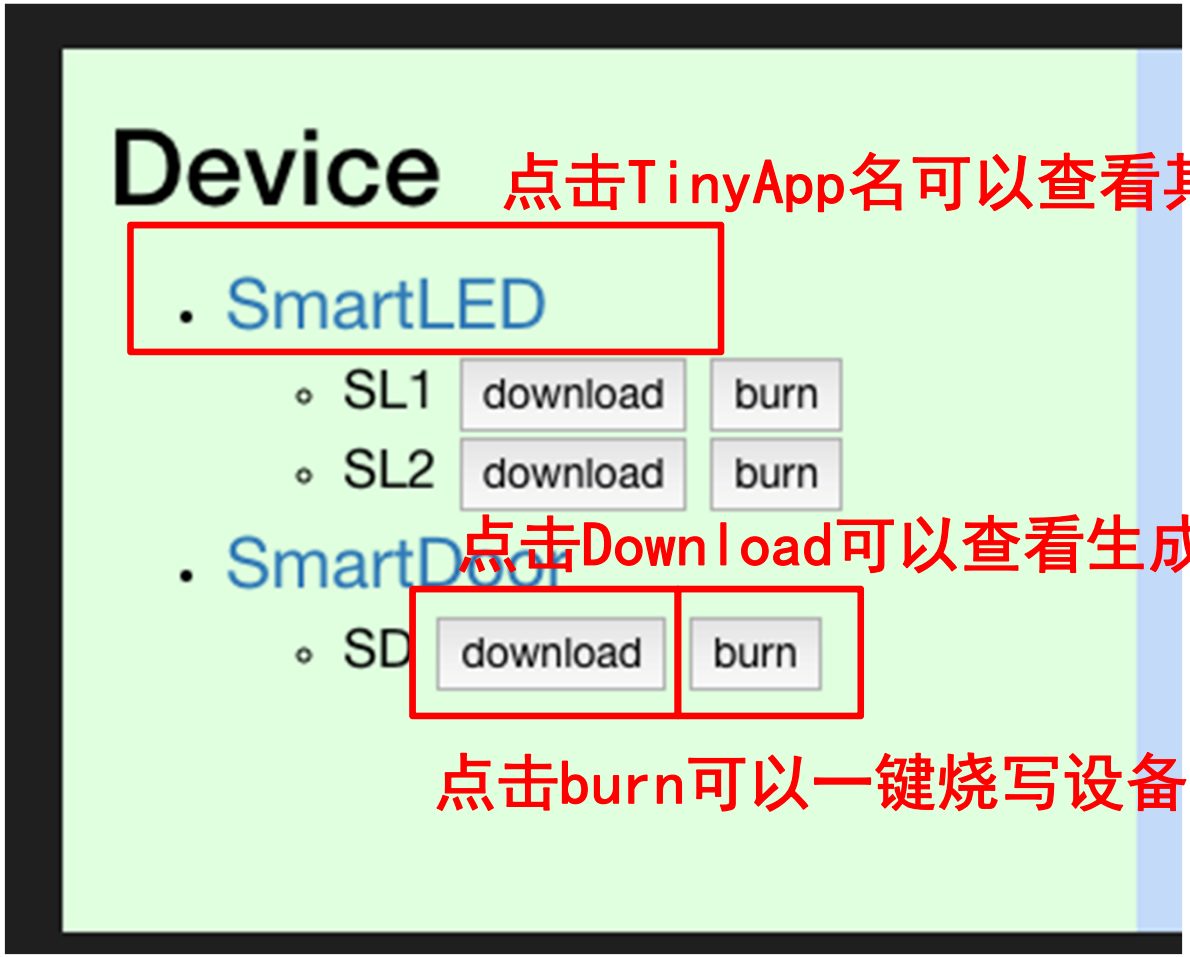
步骤④：下载、烧录生成的设备端代码

- 点击click下载生成的设备端代码压缩包，压缩包内文件以“TinyApp名_实例名.cpp”命名。
- 组装硬件，并用TinyLink分别烧写至对应的节点。



步骤④：查看连接图、烧录设备端代码

- 点击TinyApp名可查看对应的硬件连接图
- 组装好硬件之后，连接电脑，点击burn可以一键烧写二进制进入开发板（请提前安装一键烧写应用程序，可在项目主页点击链接下载）

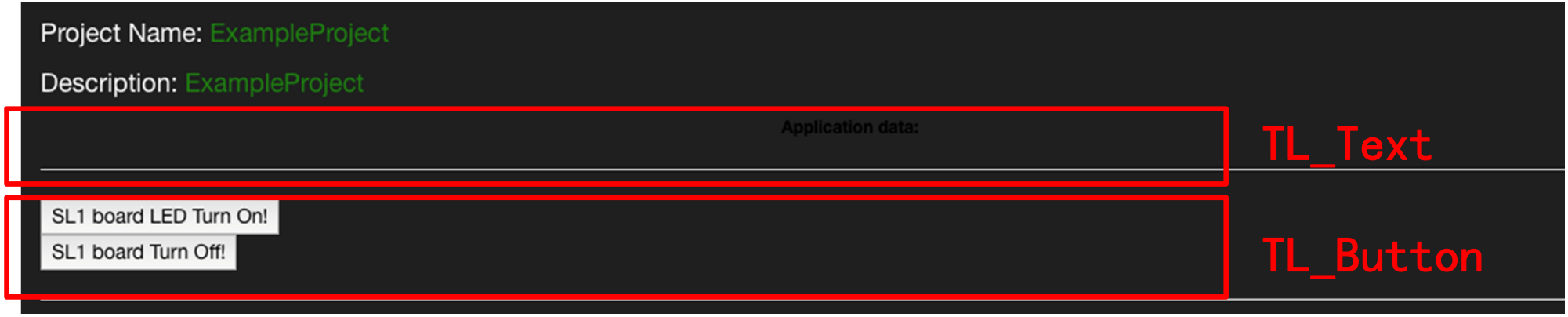


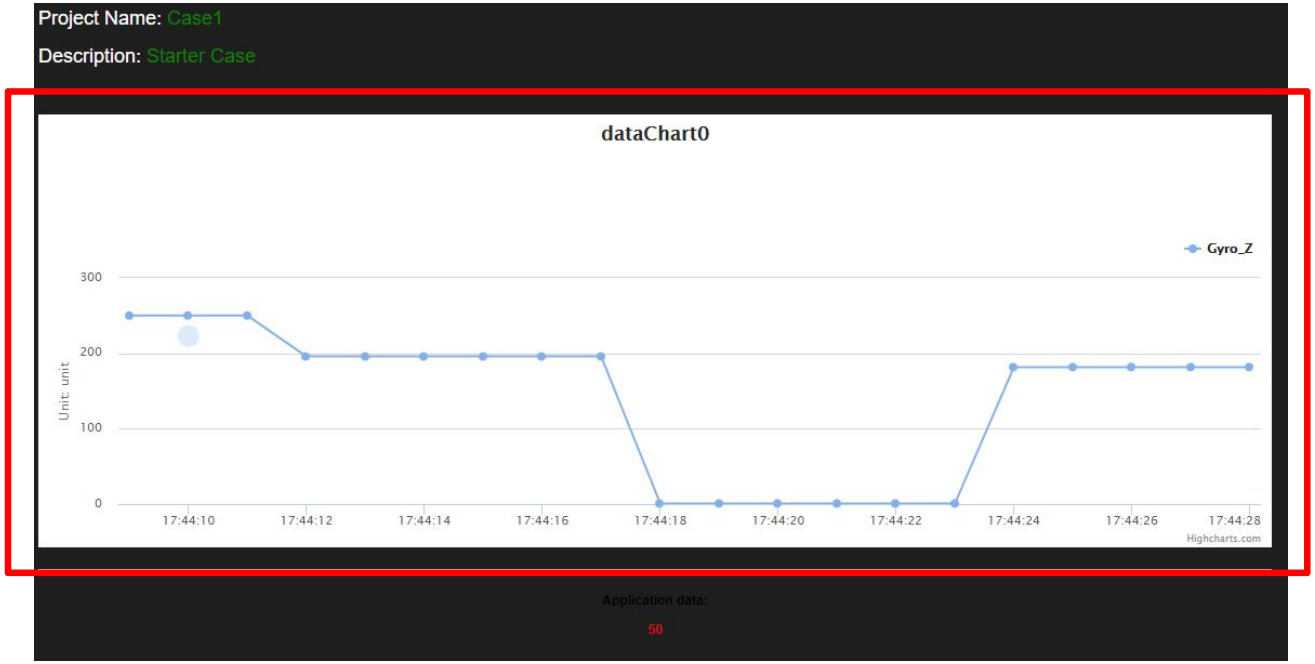
步骤⑤：查看数据显示页面

- 从项目主界面“Display Data”连接进入数据显示界面

步骤⑤：查看数据显示页面

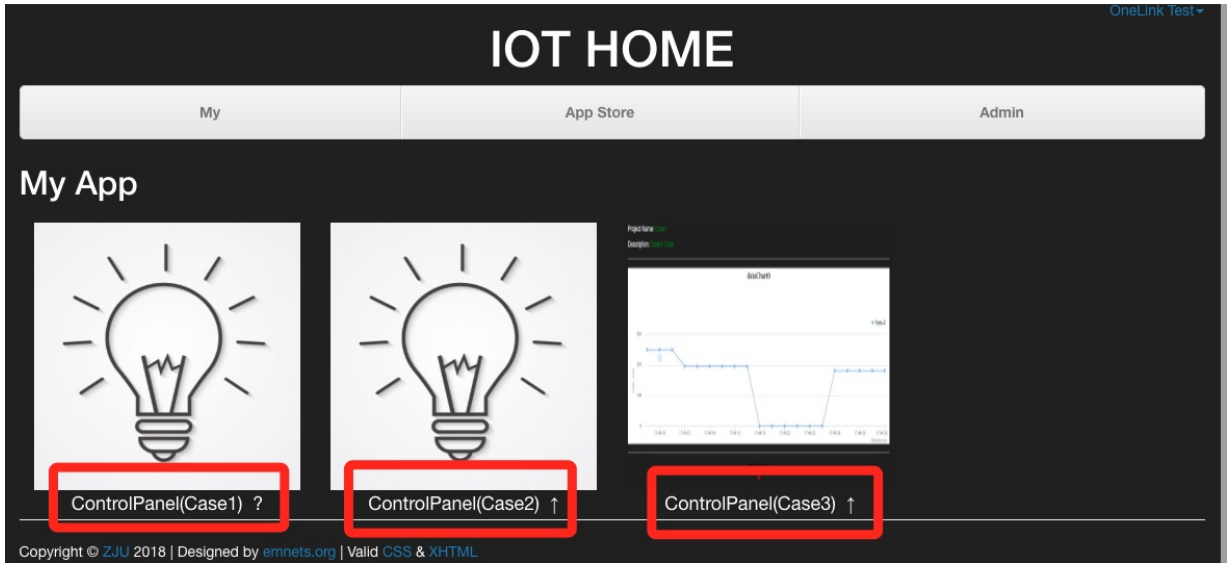
- 数据显示界面将显示用户在代码中定义的元素。包括按钮、文本和图表，对应TL_Button 、TL_Text和TL_Chart

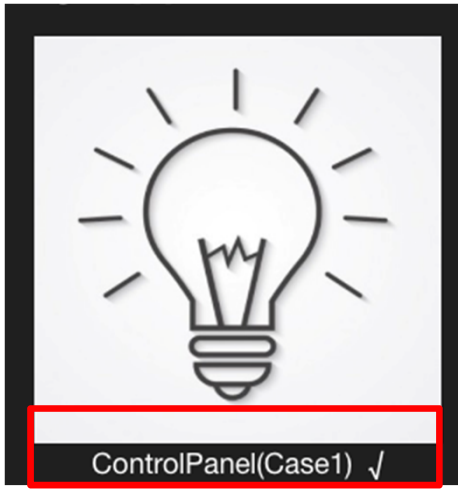




步骤⑥：提交审核数据显示页面

- 进入IoT Home：http://10.214.147.171/onelink/iot_home，登陆后显示本用户已有的应用
  - 未提交审核的app名右边会显示“↑”，
  - 已提交未审核的显示“？”，
  - 提交审核成功的显示“√”，
- 如图：Case1为已publish未审核，case2和case3为“未publish”，右侧图Case1为审核通过



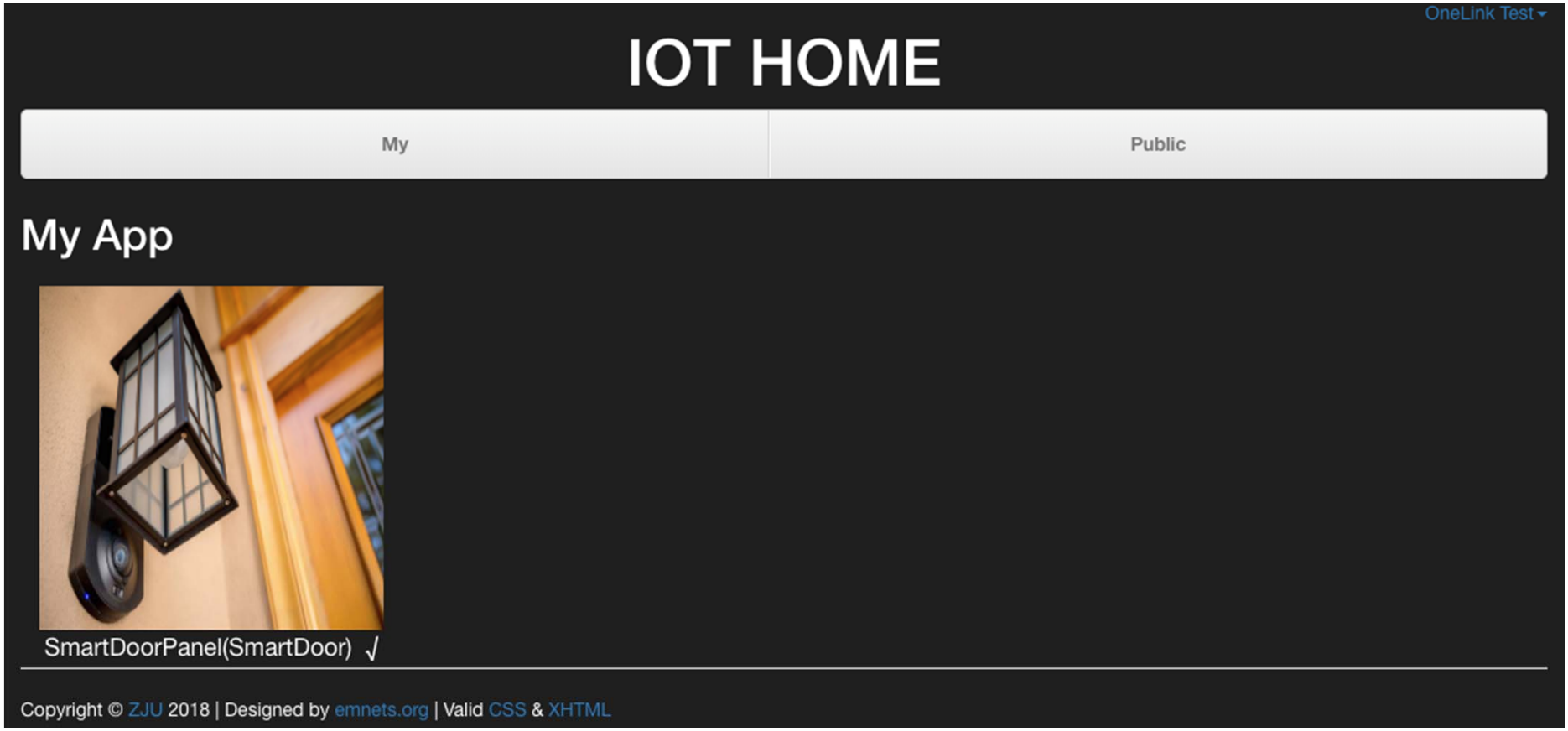


步骤⑥：提交审核数据显示页面

- 进入任意一个应用，会显示定义好的显示空间，最下面有“publish”按钮，点击即可提交管理员审核。

步骤⑦：查看审核通过的公开应用

- 待管理员审核通过后，进入IoT Home/Public页面即可查看到目前OneLink系统中公开的应用。

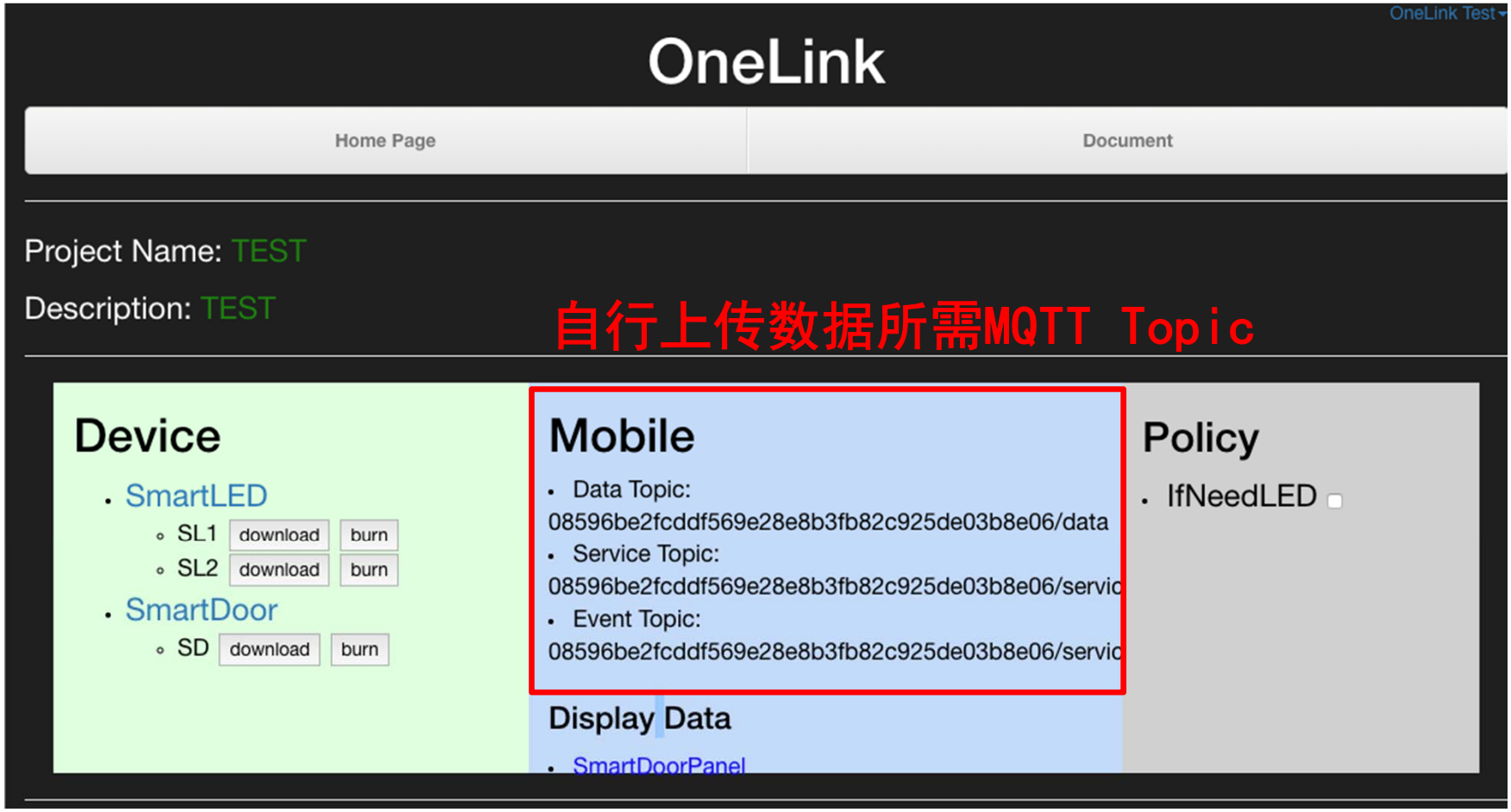


进阶①：调整数据显示页面

- 用户通过“Origin Template File”下载到的是HTML源代码文件，其中已经使用注释标注好了具体的数据获取、样式等API，开发者可通过修改此模板自定义数据显示方式。

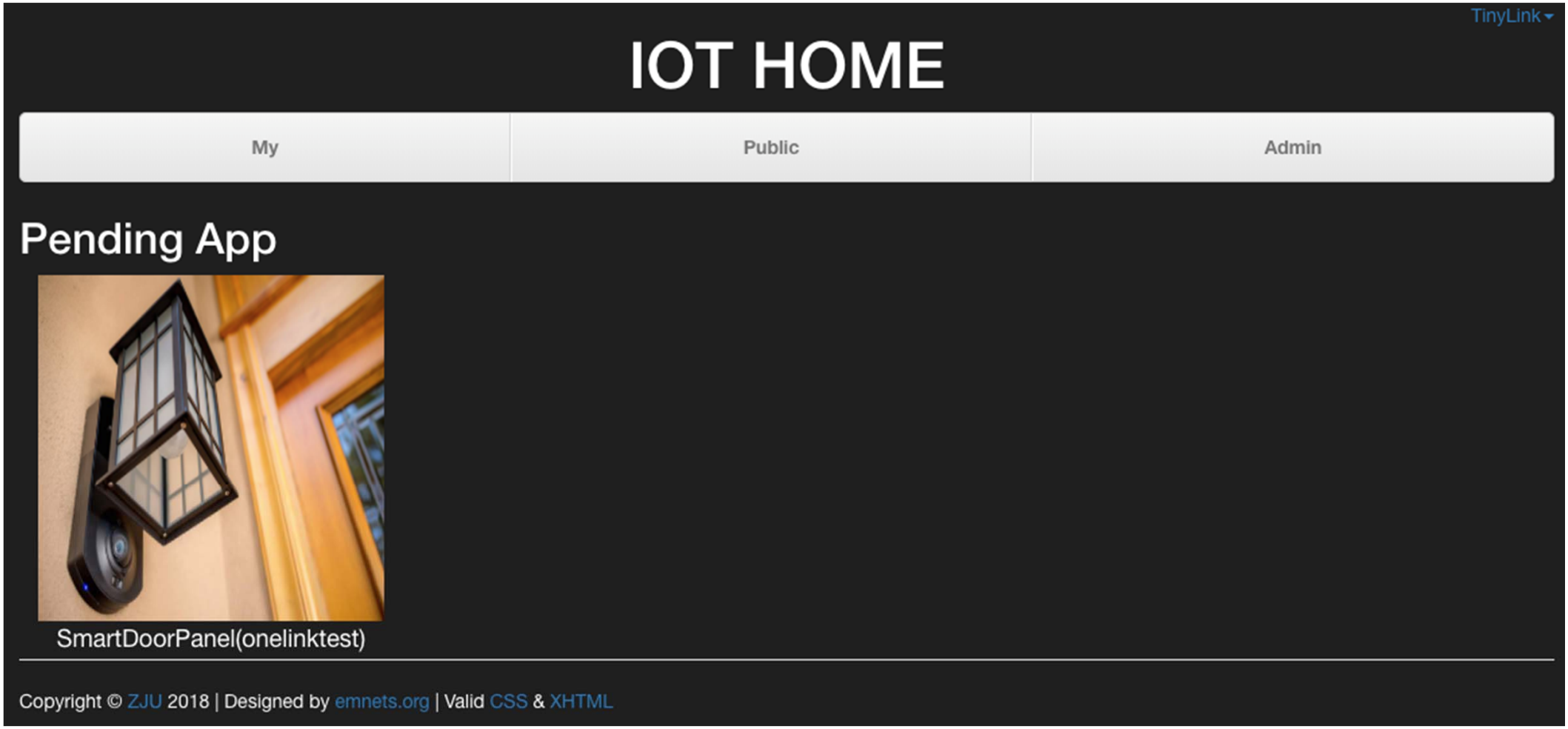
进阶②：用户自行上传数据至OneLink

- 若用户使用自定义硬件构建用户逻辑，OneLink提供MQTT接收Topic。用户自行将数据或事件经过此Topic上传，则数据可以从数据显示页面显示。



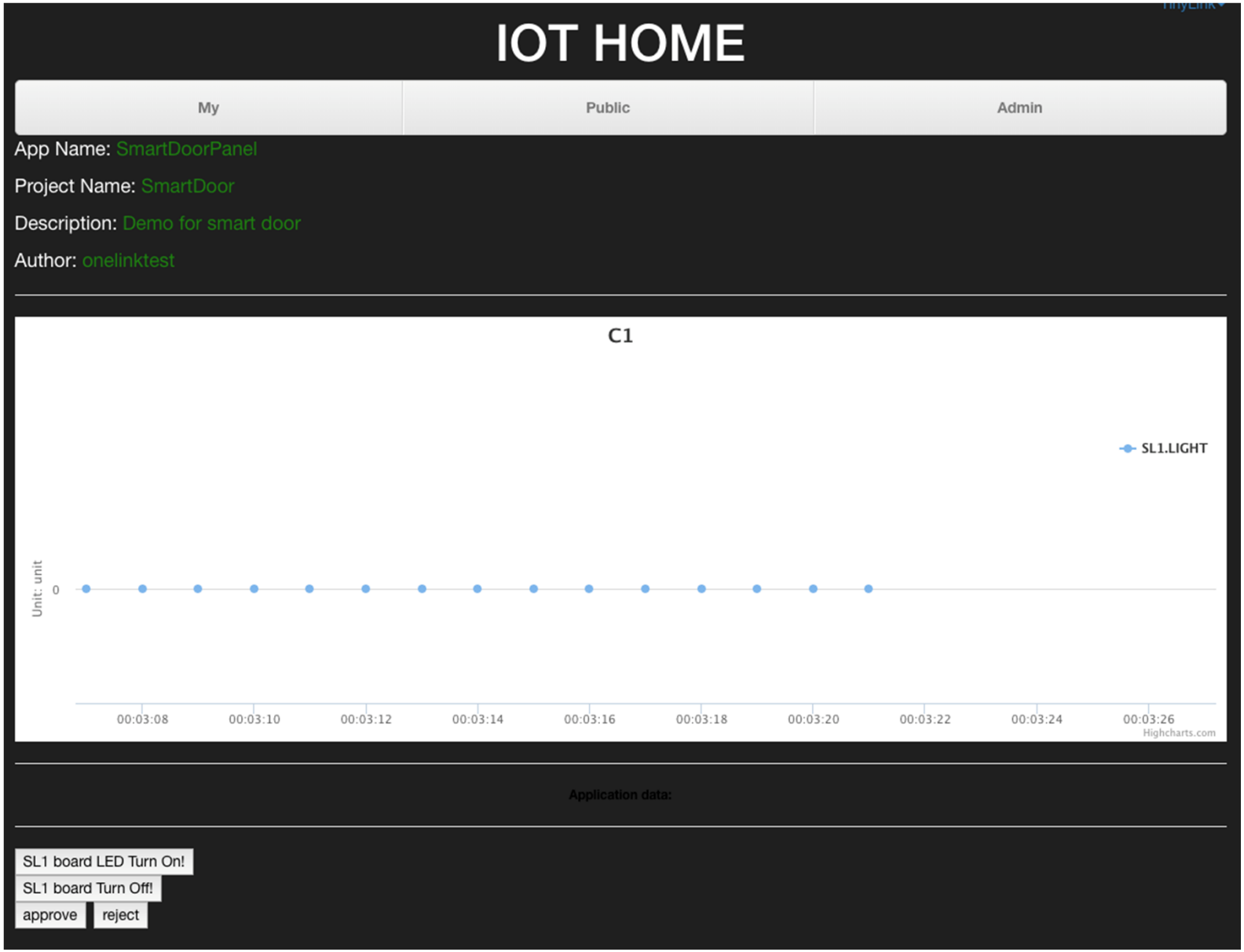
[注]管理员审核

- 使用管理员账号登陆IOT HOME，在Admin下可看到目前等待审核的App



[注]管理员审核

- 点击Approve即可审核通过



浙江大学
Zhejiang University

谢谢！

Email: dongw@zju.edu.cn