

当 安 卓 遇 到 物 联 网

——程磊



CMU Coke Machine

1999!

**“THE INTERNET OF THINGS IS
ABOUT EMPOWERING COMPUTERS
...SO THEY CAN SEE, HEAR
AND SMELL THE WORLD FOR
THEMSELVES”**

**KEVIN ASHTON
INVENTOR OF THE TERM
“INTERNET OF THINGS”**

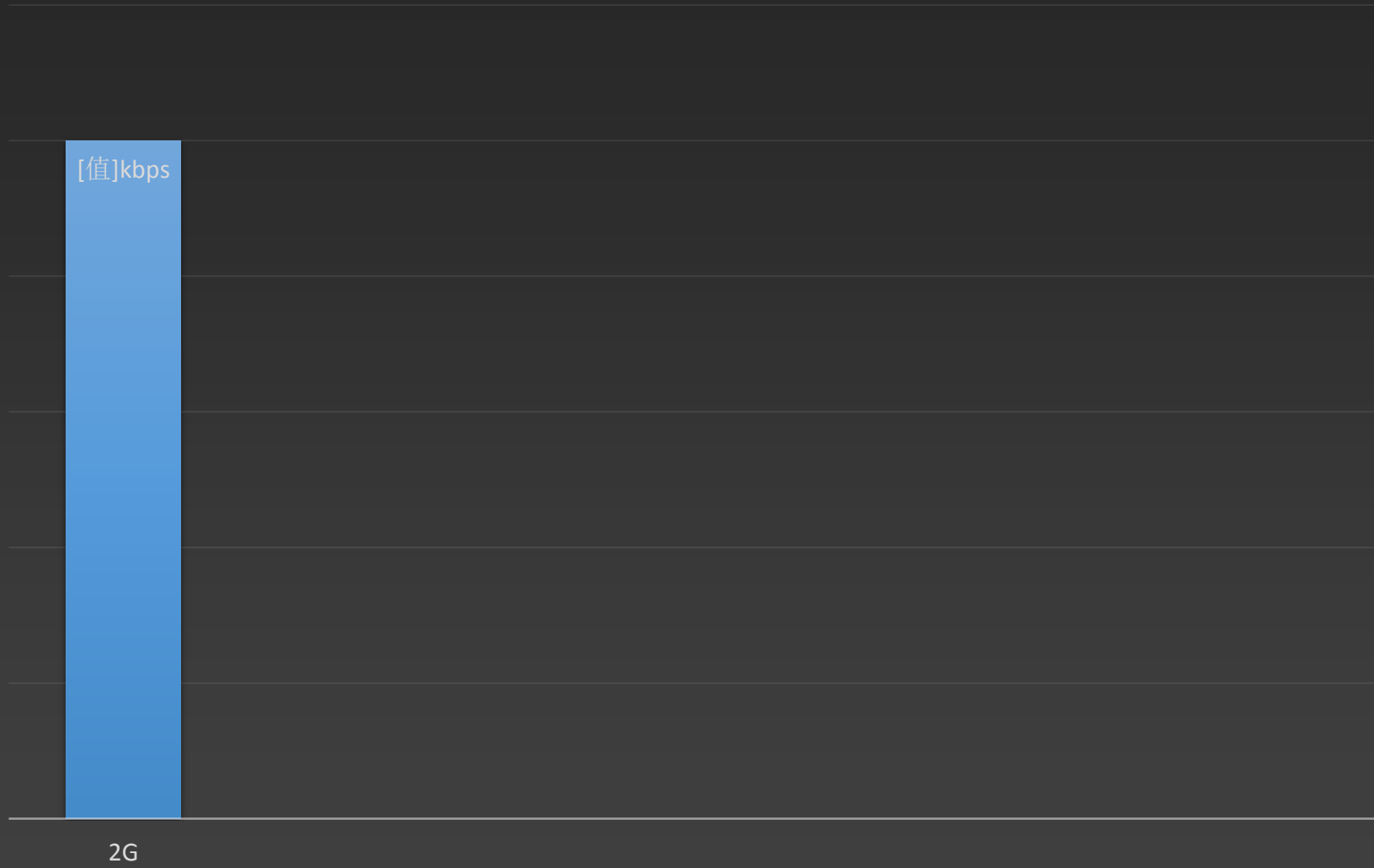


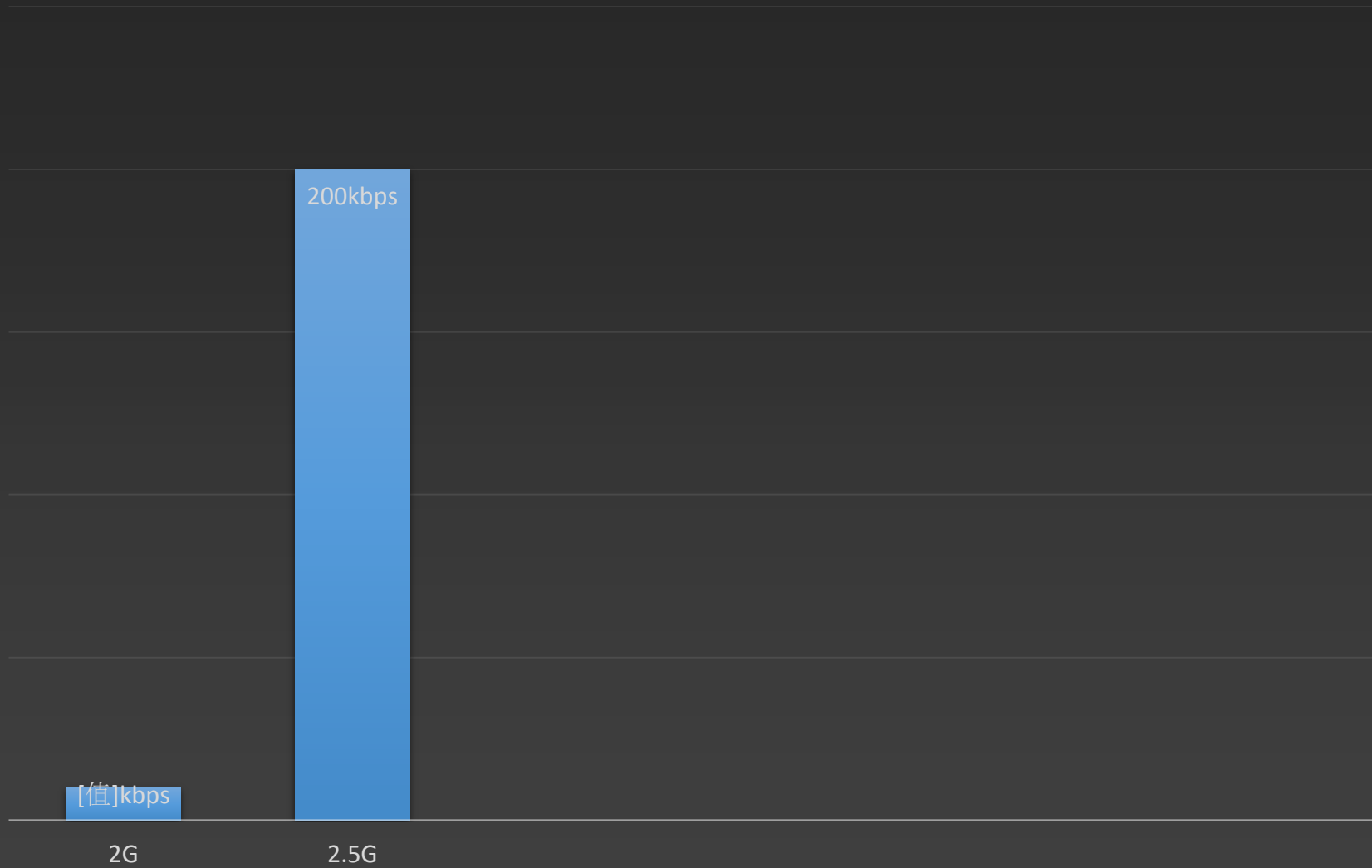


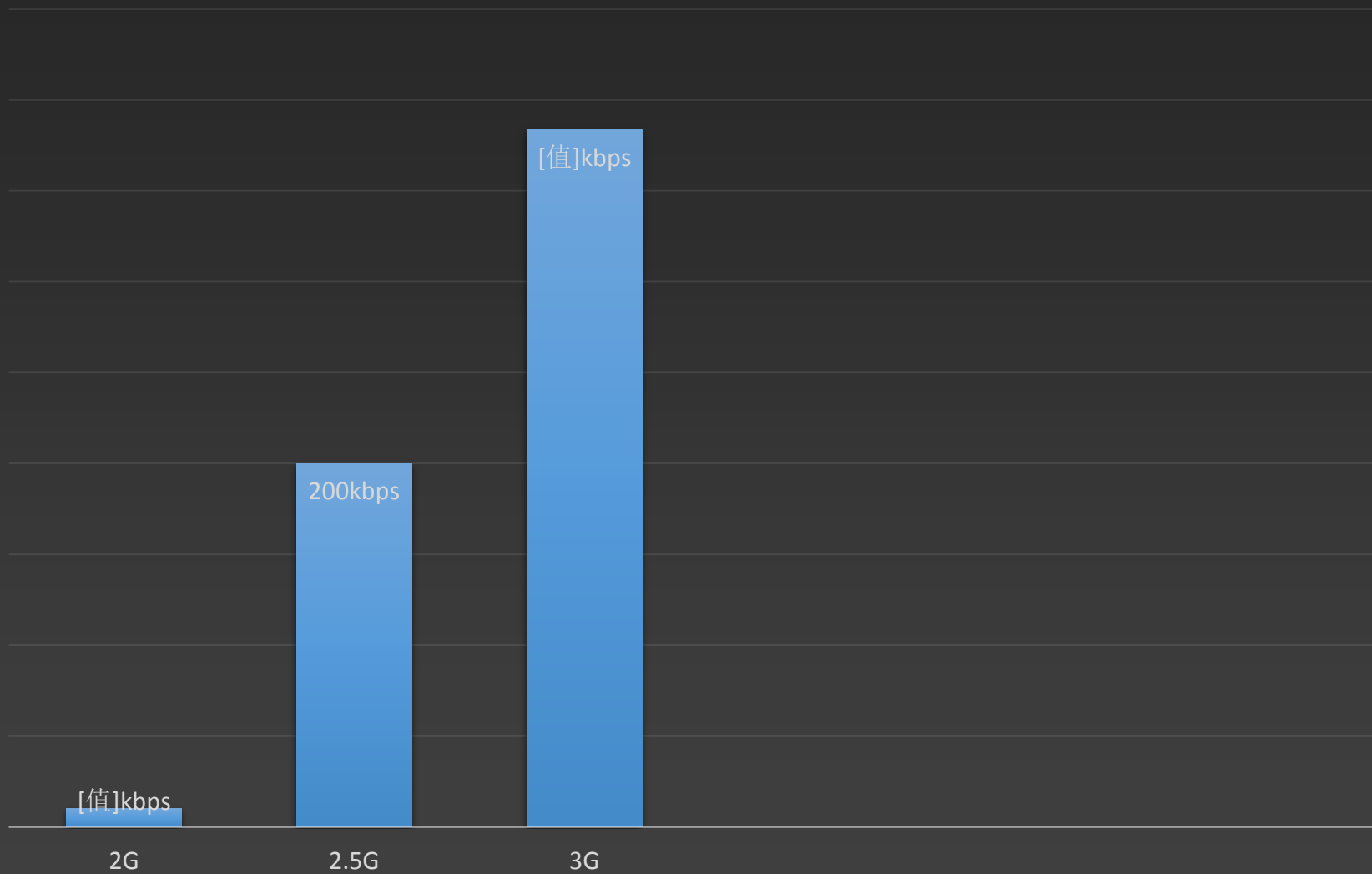
十年过去了，我们虽然取得了长足的进步，但同时我们要意识到我们的技术所能做的事情非常重要，并一直支持它。我们不能让我们的愿景局限在商品上打上条形码和提高公路收费站的通行效率上。物联网能改变世界，一如过去互联网那样，甚至超过互联网。

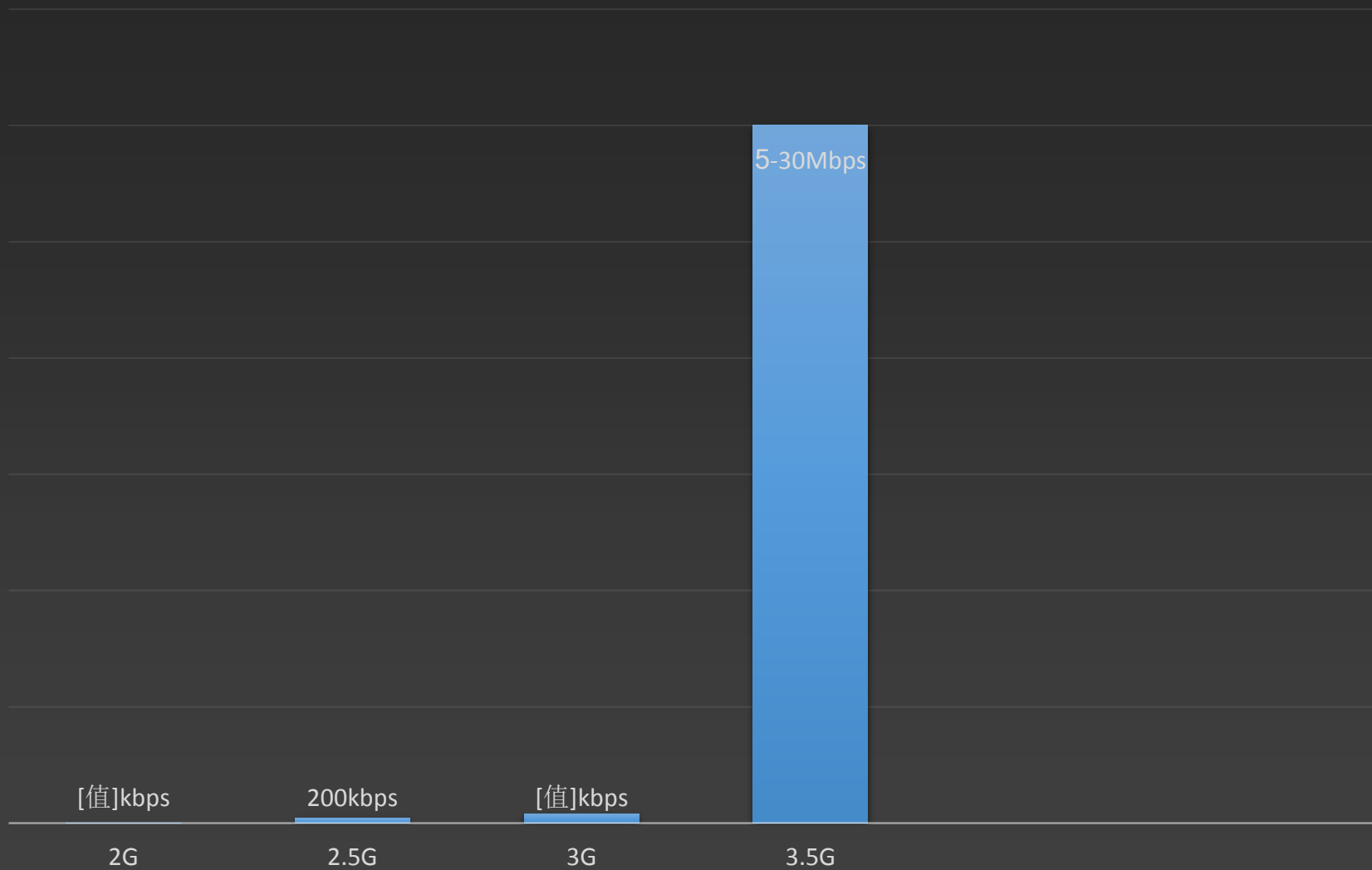
—— Kevin Ashton

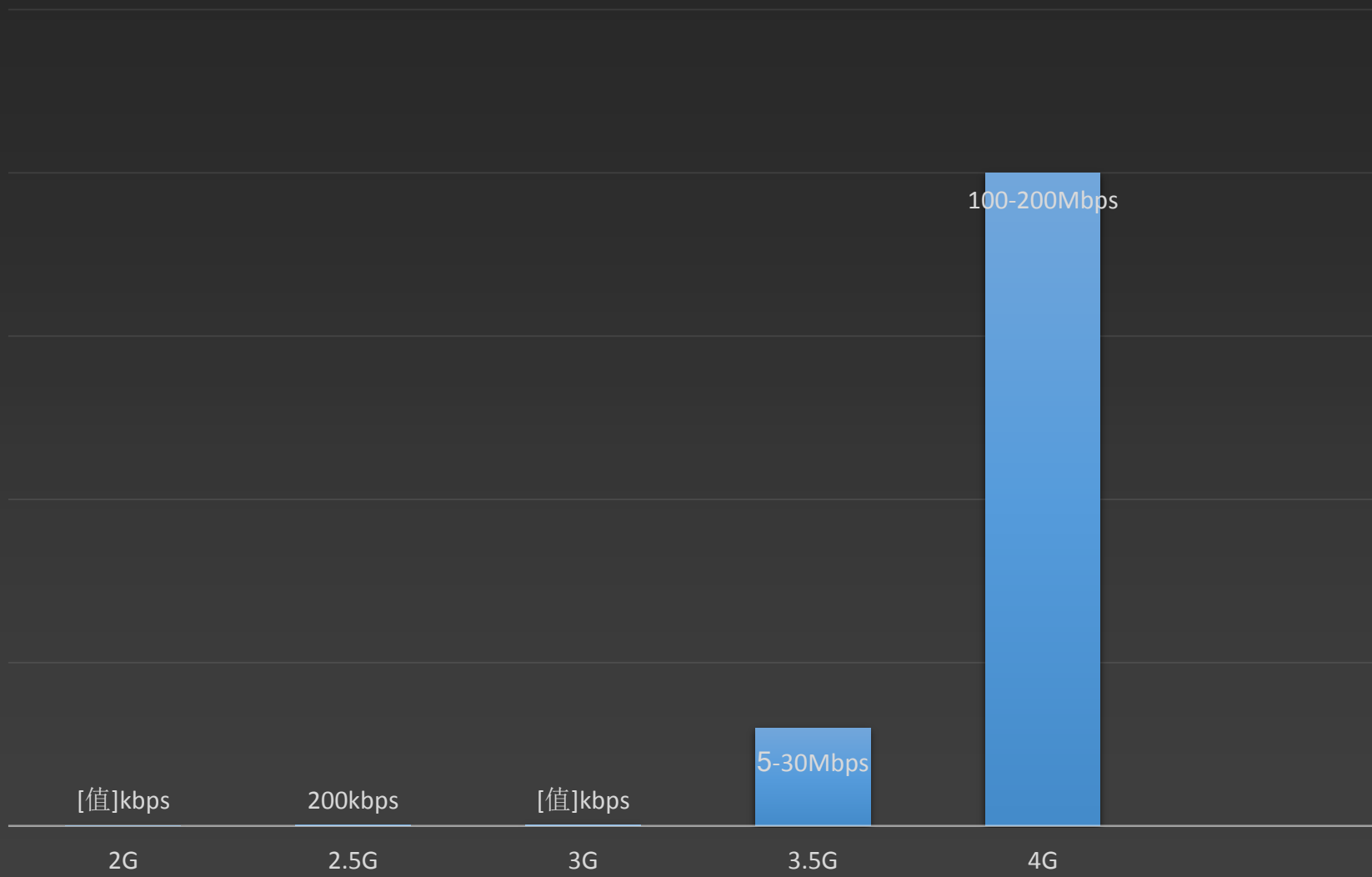


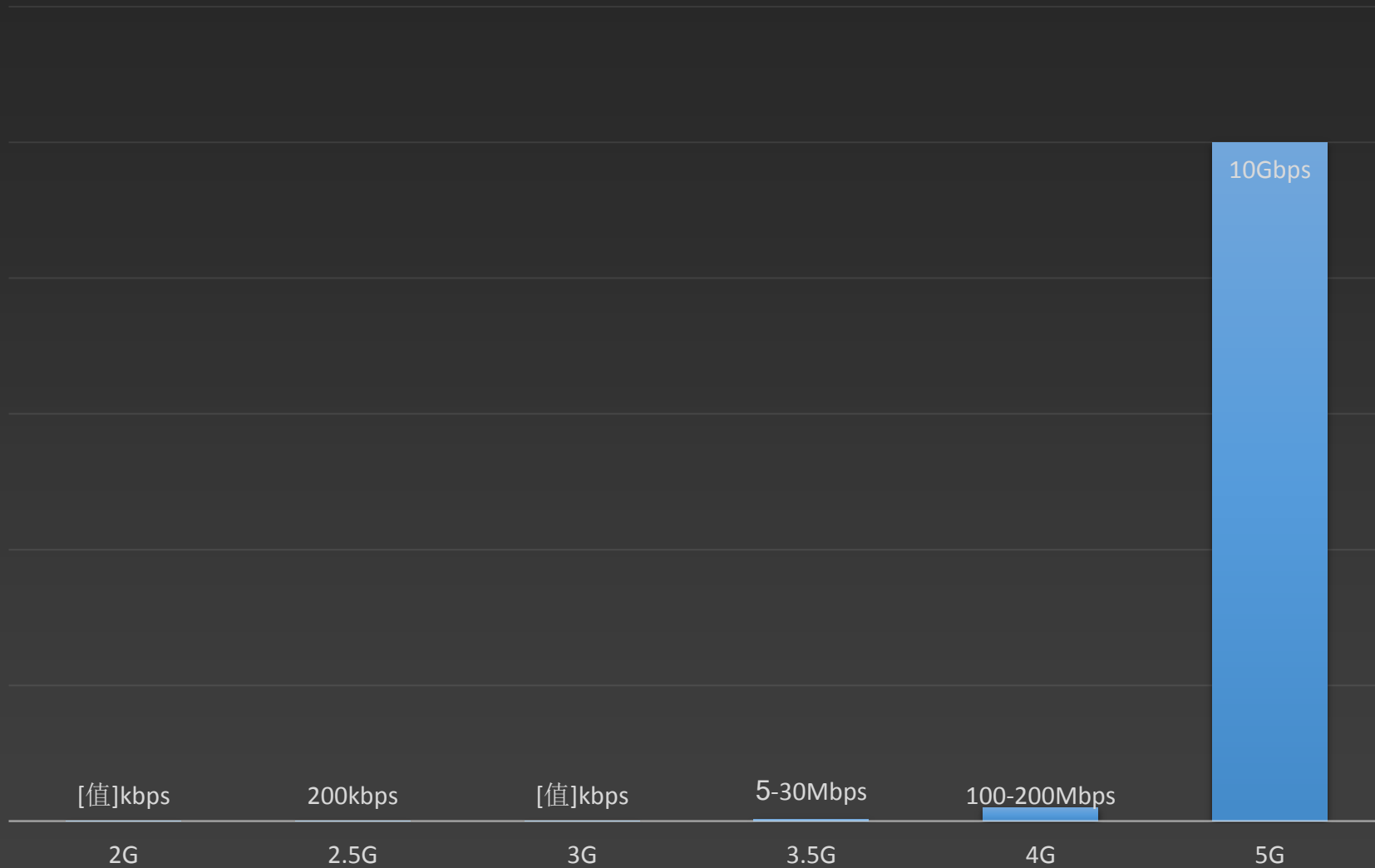


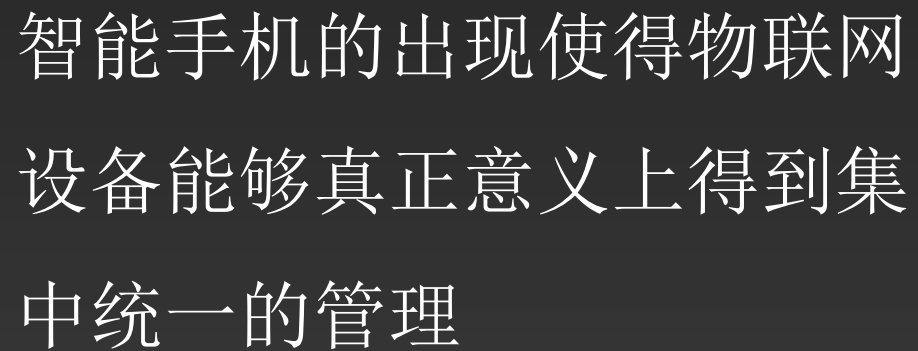












IoT	Consumer applications	Smart Home
	Enterprise applications	
	Infrastructure applications	Manufacturing
		Agriculture
		Energy management
		Environmental monitoring
		Building and home automation
		Metropolitan scale deployments
	Other fields of application	Medical and healthcare
		Elder Care
		Transportation

open.iot.10086.cn

中国移动物联网开放平台OneNET 从连接开始



- 多协议接入支持
- 完善的设备端和应用端SDK
- 应用定制化开发
- 数据展现和数据分析服务



智慧停车解决方案



共享经济解决方案



城市消防监测解决方案



畜牧物联网解决方案



二次供水联网方案



智能水表抄表系统
解决方案



智慧燃气解决方案



智慧能源节能服务
平台





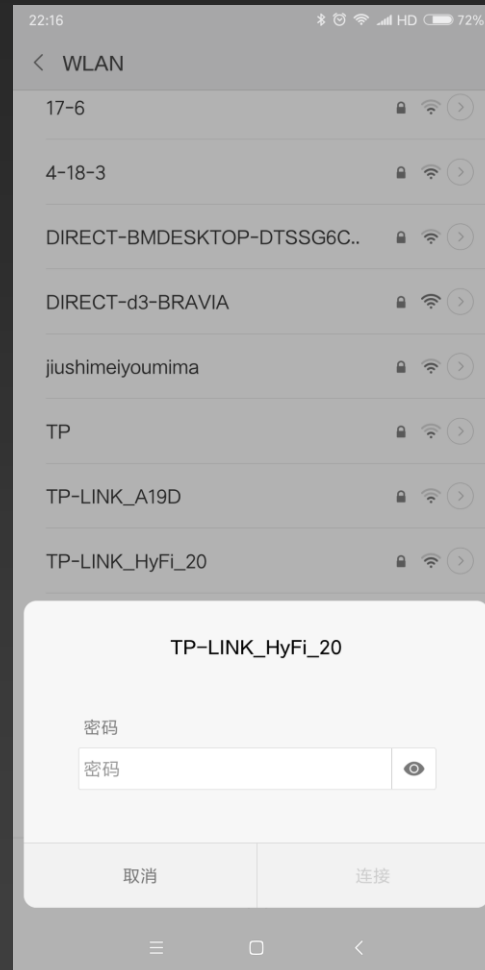
中移和物™
ZHONG YI HE WU

- 设备接入 + 设备管理后台开发 + APP后台开发 + APP开发
- 设备接入 + 应用定制化开发
- 设备接入 + APP自动生成

做出和物这款产品要解决的两个问题：

1、设备如何联网

2、APP控制界面如何展示



• 运行状态

• 设置向导

- 网络参数

• WAN口设置

• LAN口设置

• WAN口速率/模式

• MAC地址克隆

• 无线频段设置

+ 无线设置 2.4GHz

+ 无线设置 5GHz

+ DHCP服务器

+ 转发规则

+ 安全功能

• 家长控制

+ 上网控制

+ 路由功能

+ IP带宽控制

+ IP与MAC绑定

WAN口设置

WAN口连接类型:

PPPoE ▼

自动检测

PPPoE连接:

上网账号:

02367084236

上网口令:

确认口令:

特殊拨号:

自动选择拨号模式 ▼

根据您的需要, 请选择对应的连接模式:



按需连接, 在有访问时自动连接

自动断线等待时间: 15 分 (0 表示不自动断线)



自动连接, 在开机和断线后自动连接



定时连接, 在指定的时间段自动连接

注意: 只有当您到“系统工具”菜单的“时间设置”项设置了当前时间后, “定时连接”功能才能生效。

连接时段: 从 0 时 0 分到 23 时 59 分



手动连接, 由用户手动连接

自动断线等待时间: 15 分 (0 表示不自动断线)

连接

断线

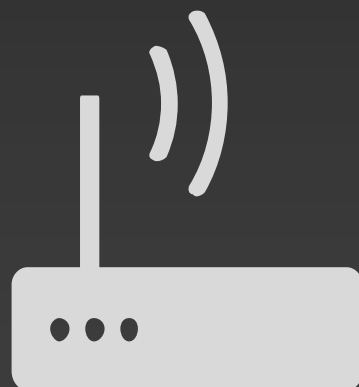
已连接

高级设置

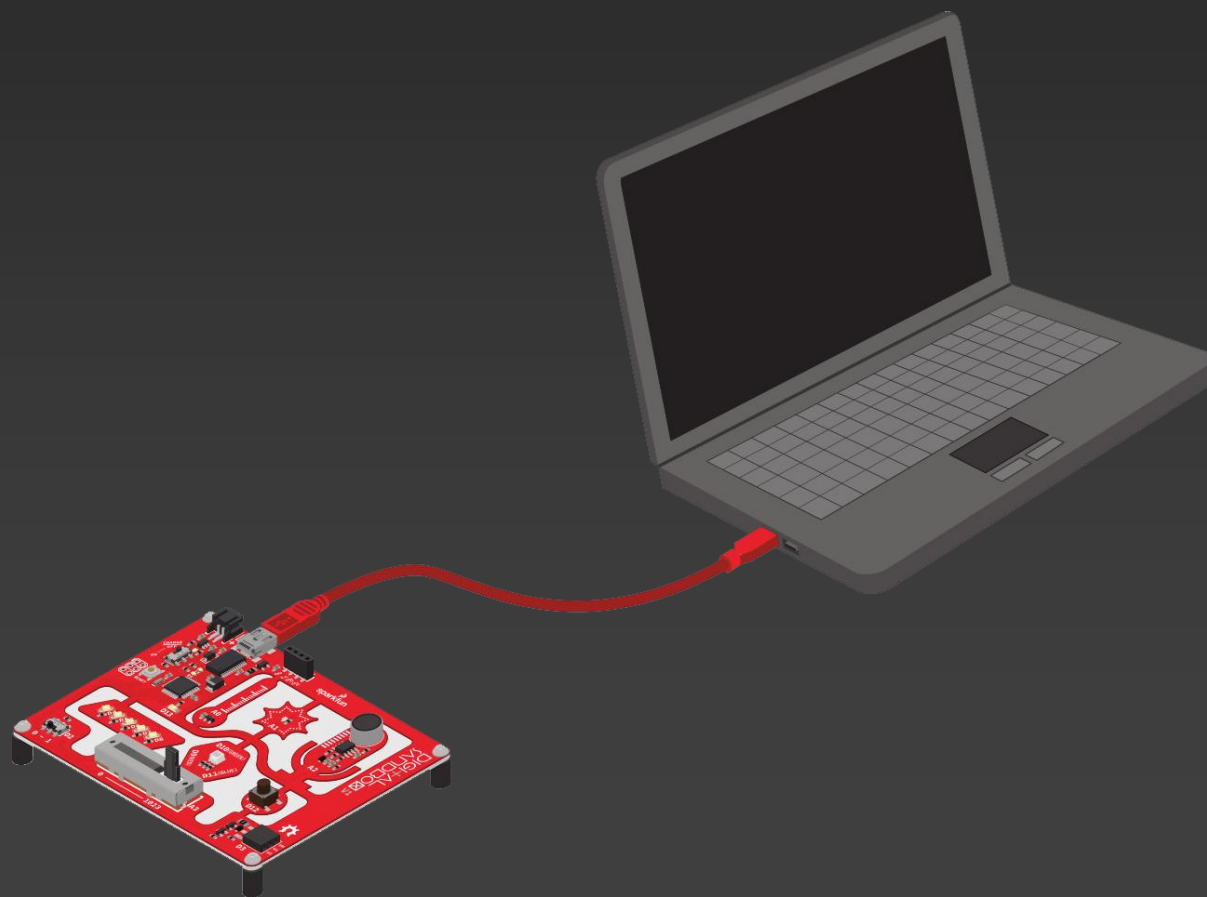
保存

帮助

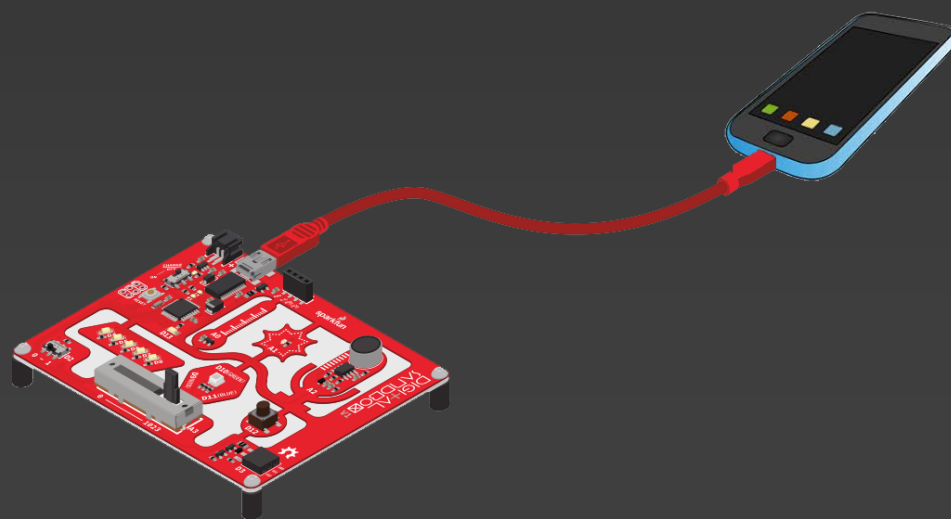
配网方式一：设备开启 WEB 服务器



配网方式二：串口 AT 指令



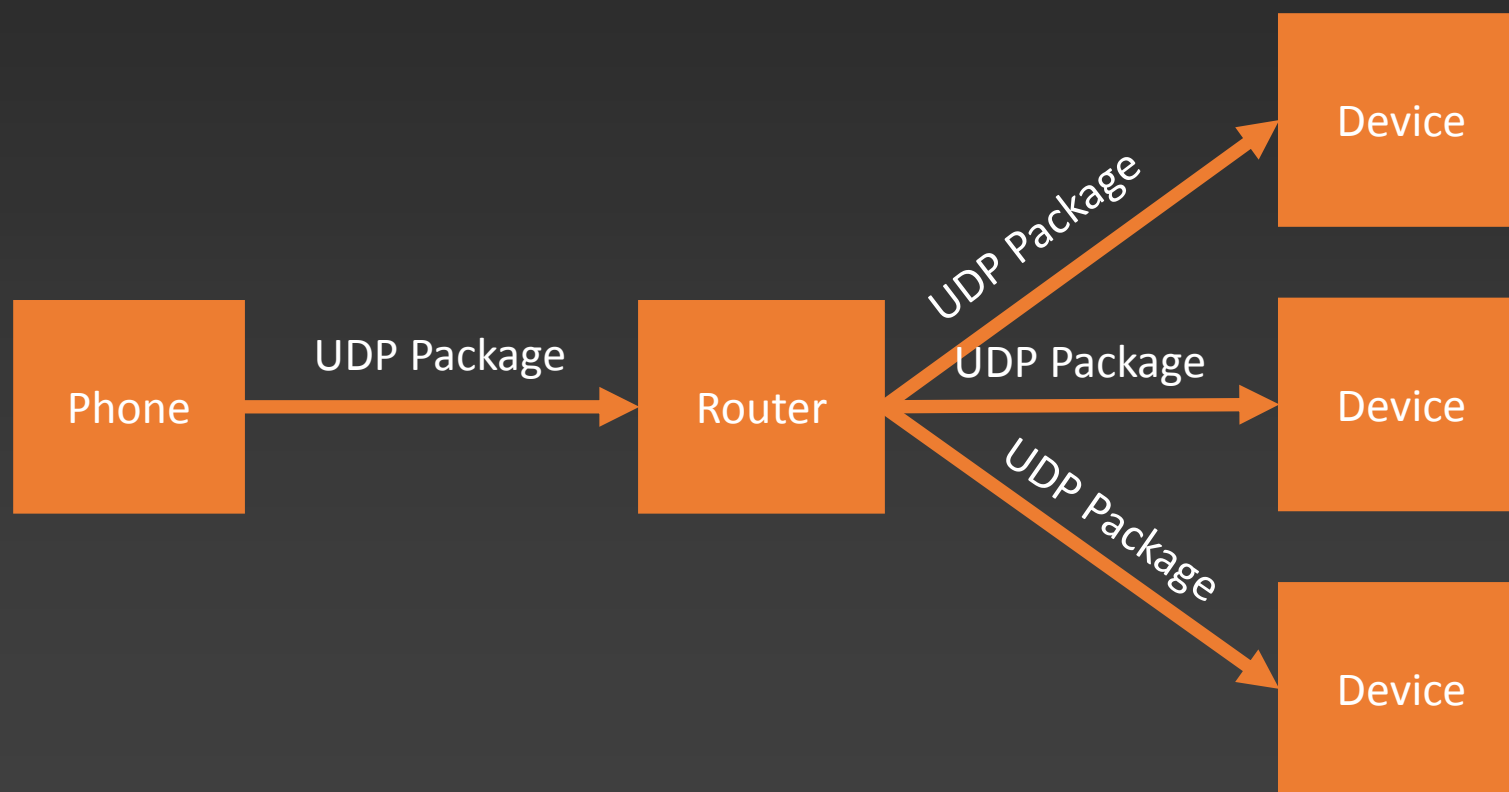
配网方式二：串口 AT 指令



配网方式二：串口 AT 指令



配网方式三： Smart Config

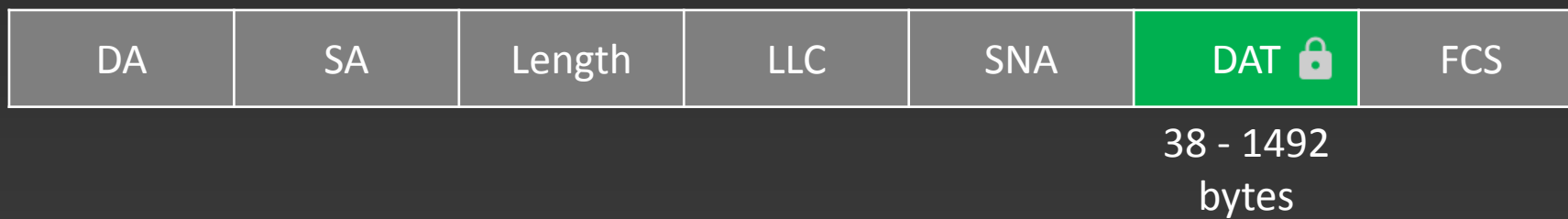


配网方式三： Smart Config

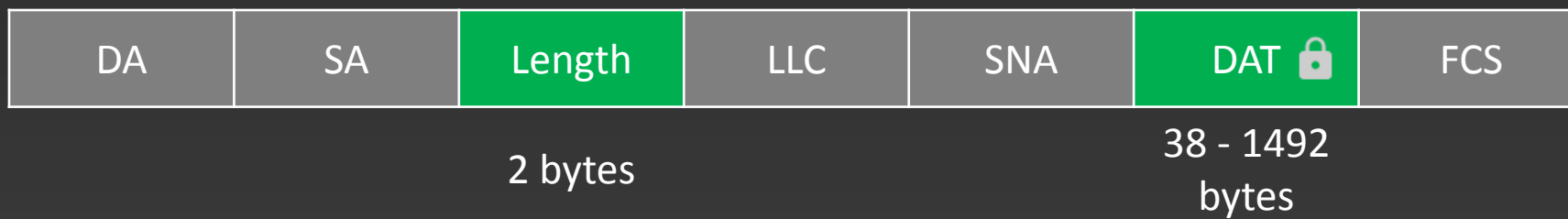
DA	SA	Length	LLC	SNA	DAT	FCS
6 bytes	6 bytes	2 bytes	3 bytes	5 bytes	38 - 1492 bytes	4 bytes

802.2 SNAP 格式数据包

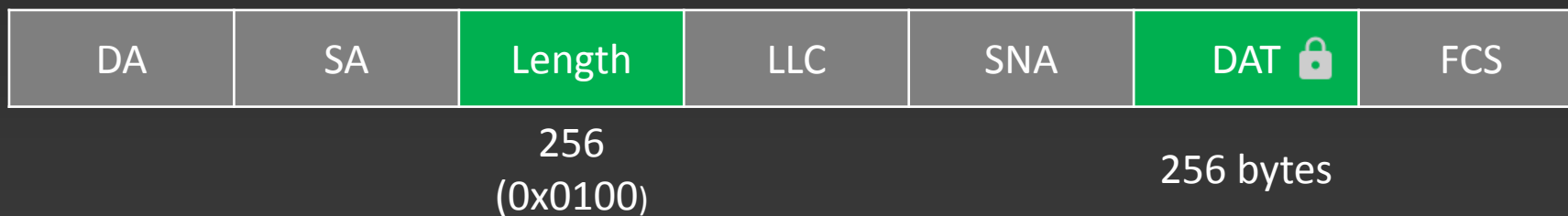
配网方式三： Smart Config



配网方式三： Smart Config



配网方式三： Smart Config



```
byte[] data = new byte[256];  
for (int i = 0; i < data.length; i++) {  
    data[i] = '1';  
}
```

配网方式三： Smart Config

255.255.255.255	UDP	225	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	171	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	348	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	346	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	347	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	348	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	232	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	172	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	410	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	417	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	467	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	349	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	172	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	173	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	352	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	346	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	415	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	413	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	247	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	174	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	396	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	417	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	403	Source port: 63413	Destination port: 10000
255.255.255.255	UDP	400	Source port: 63413	Destination port: 10000

配网方式三：Smart Config

组播地址：224.0.0.0 ~ 239.255.255.255

配网方式三： Smart Config

239.36.176.205	UDP	78	Source port: 8863	Destination port: 18864
239.37.128.252	UDP	79	Source port: 8863	Destination port: 18864
239.38.36.87	UDP	80	Source port: 8863	Destination port: 18864
239.39.173.205	UDP	81	Source port: 8863	Destination port: 18864
239.40.6.115	UDP	82	Source port: 8863	Destination port: 18864
239.41.9.111	UDP	83	Source port: 8863	Destination port: 18864
239.42.147.228	UDP	84	Source port: 8863	Destination port: 18864
239.43.230.144	UDP	85	Source port: 8863	Destination port: 18864
239.44.5.112	UDP	86	Source port: 8863	Destination port: 18864
239.45.41.75	UDP	87	Source port: 8863	Destination port: 18864
239.46.22.93	UDP	88	Source port: 8863	Destination port: 18864
239.47.11.103	UDP	89	Source port: 8863	Destination port: 18864
239.48.141.228	UDP	90	Source port: 8863	Destination port: 18864
239.0.161.0	UDP	42	Source port: 8863	Destination port: 18864
239.1.160.0	UDP	43	Source port: 8863	Destination port: 18864
239.2.157.2	UDP	44	Source port: 8863	Destination port: 18864
239.3.109.49	UDP	45	Source port: 8863	Destination port: 18864
239.4.202.211	UDP	46	Source port: 8863	Destination port: 18864
239.5.200.212	UDP	47	Source port: 8863	Destination port: 18864
239.6.199.212	UDP	48	Source port: 8863	Destination port: 18864
239.7.199.211	UDP	49	Source port: 8863	Destination port: 18864
239.8.199.210	UDP	50	Source port: 8863	Destination port: 18864
239.9.252.156	UDP	51	Source port: 8863	Destination port: 18864
239.10.103.48	UDP	52	Source port: 8863	Destination port: 18864

配网方式四： 声波通信



小米快传



支付宝
ALIPAY

配网方式四： 声波通信

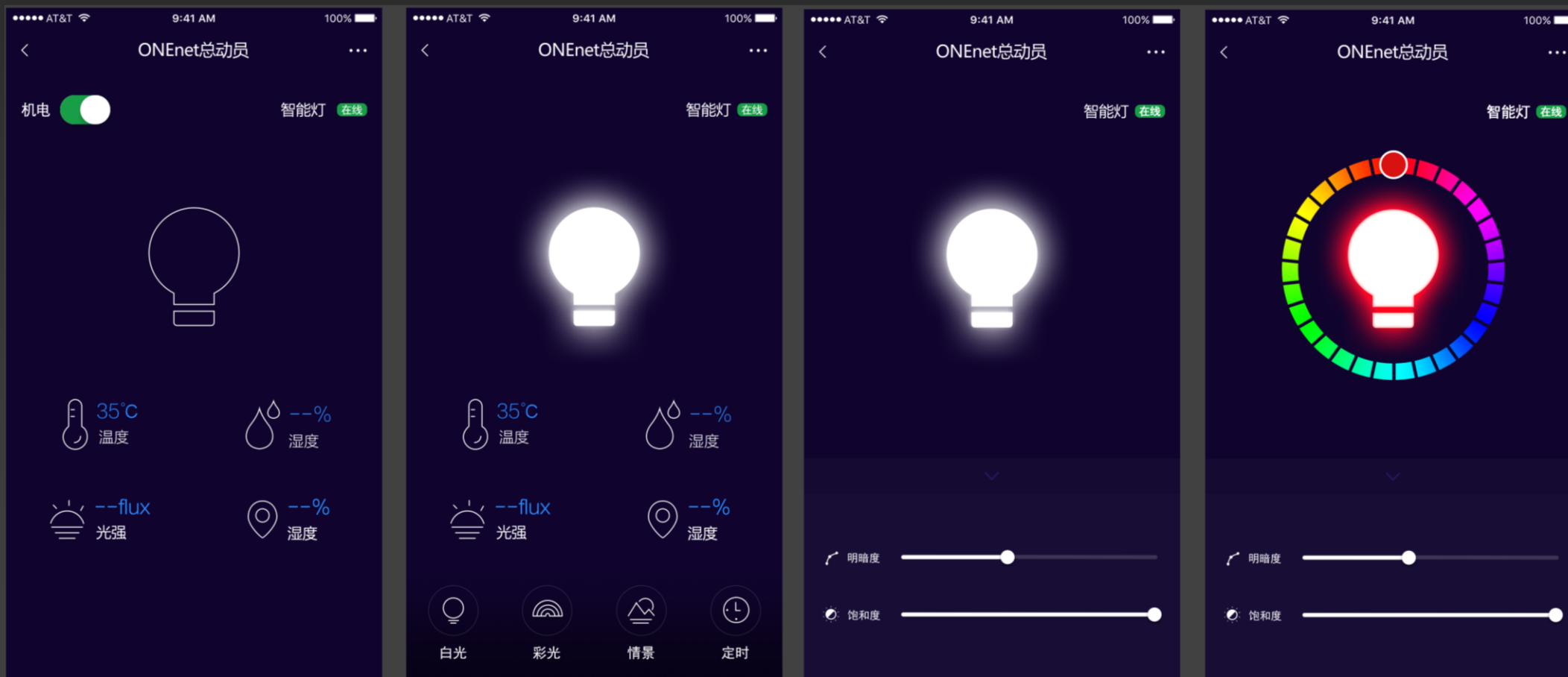
用单频率声音信号对数据进行编码，然后播放这些单频率声音，接收方在收到声音后，识别出频率，然后根据频率解码出数据。

比如：我们可以将 1500Hz 的正弦波对应数字 1，1600Hz的正弦波对应数字 2，1700Hz 的正弦波对应数字 3。那么数字串 3123 就对应 4 段正弦波，规定每段正弦波持续 100ms，则 3123 对应 400 毫秒的声音段。接收方录制声音，对收到的声音进行解析，识别出 1700Hz，1500Hz，1600Hz，1700Hz 四段正弦波频率，然后查找码本，解码出的数字 就是 3123。

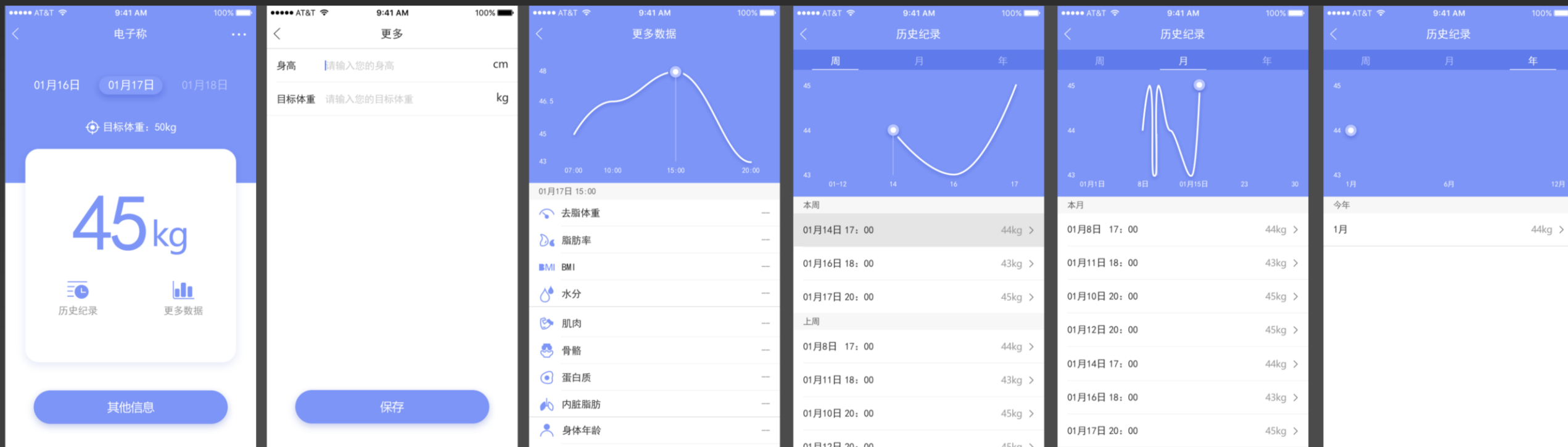
设备控制界面的定制



设备控制界面的定制



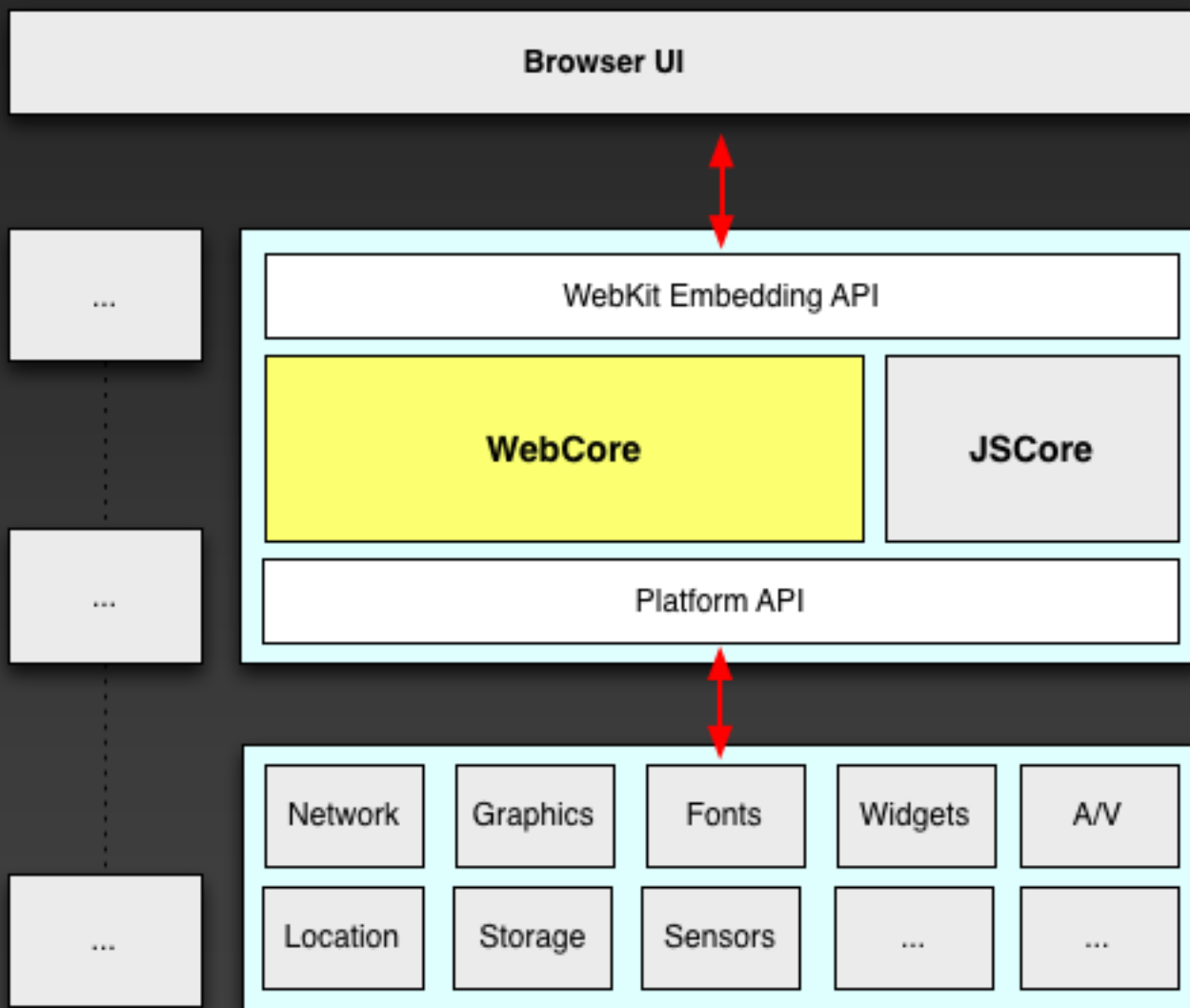
设备控制界面的定制



技术方案

1、混合开发 (Html5 + WebView)

2、React Native / Weex



背景选择



开关功能选定

开关

非必须，如设置，开关按钮将设置屏幕底部

功能点样式设置和排序（可拖动功能进行排序）

工作模式	小模块	选择图标	(+)		
目标温度	大模块	选择图标	(+)	选择背景	自定义背景尺寸：建议4:2.5比例
当前温度	小模块	选择图标	(+)		
开关	大模块	选择图标	(+)	选择背景	自定义背景尺寸：建议4:2.5比例
风速	小模块	选择图标	(+)		
工作状态	小模块	选择图标	(+)		
自动模式	小模块	选择图标	(+)		
ECO模式	小模块	选择图标	(+)		
设置湿度	小模块	选择图标	(+)		
故障告警	不显示	选择图标	(+)		
蜂鸣	小模块	选择图标	(+)		
风摆	小模块	选择图标	(+)		

保存

预览

背景选择



开关功能选定

开关

非必须，如设置，开关按钮将设置屏幕底部

功能点样式设置和排序（可拖动功能进行排序）

工作模式	小模块	选择图标	(+)
目标温度	大模块	选择图标	(+)
当前温度	小模块	选择图标	(+)
开关	大模块	选择图标	(+)
风速	小模块	选择图标	(+)
工作状态	小模块	选择图标	(+)
自动模式	小模块	选择图标	(+)
ECO模式	小模块	选择图标	(+)
设置湿度	小模块	选择图标	(+)
故障告警	不显示	选择图标	(+)
蜂鸣	小模块	选择图标	(+)
风摆	小模块	选择图标	(+)



保存

预览



✔ 产品已上架

< 返回 | 虚拟设备调试

模拟设备存在于和物OneNET云端，用于模拟真实硬件设备。可以像真实设备一样接收手机客户端下发的指令，也可以模拟真实设备上报数据。即使您的硬件设备未完成开发，也可完成 APP 界面调试。

虚拟设备

(手动操作)

我是设备名称	
开关	请选择
目标温度	^ v
当前温度	^ v
工作模式	请选择
风速	请选择
上报	

通讯日志

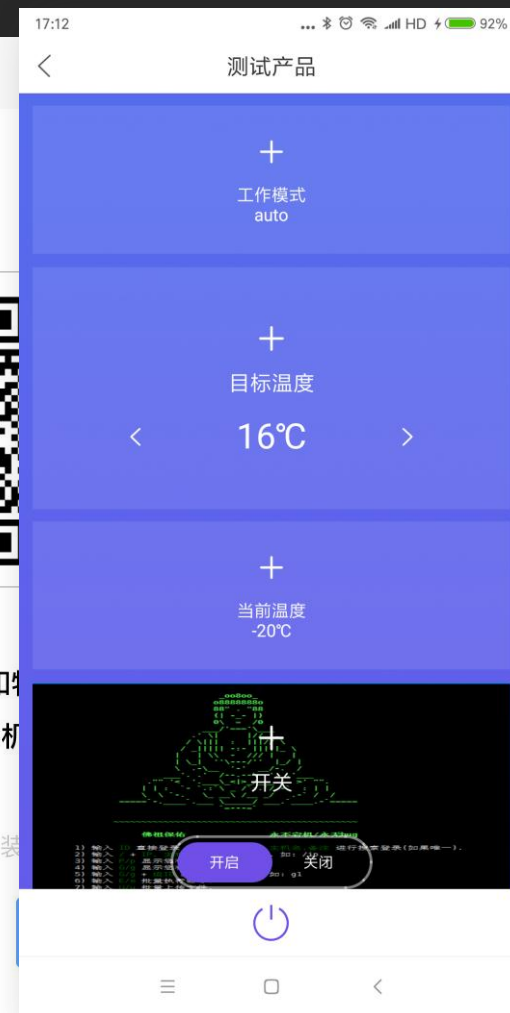
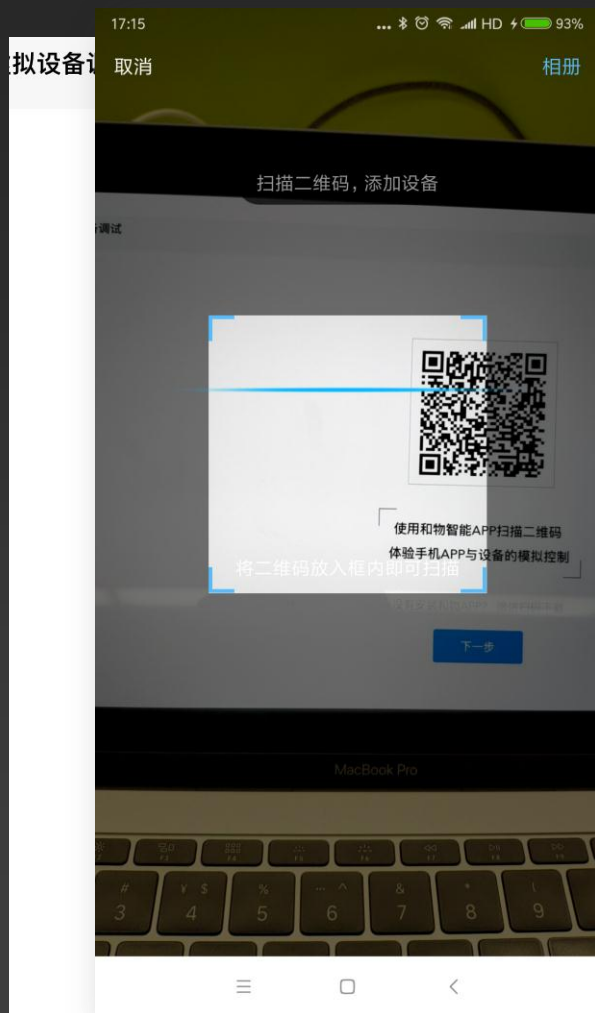
虚拟设备-平台-H5通讯日志

17:08:00 设备上线

和物APP



请在和物APP上查看界面并操作调试



展 望



Brillo (2015)



android things (2016)

	开发语言	Android Studio	用户界面	Android SDK	Google Service
Brillo	C/C++	-	-	-	-
Android things	Java/Kotlin/C/C++	支持	支持	支持	支持

如何构建一个 android things app?

1 新建一个 Android Studio 工程

SDK API 27 or higher

SDK tools 25.0.3 or higher

```
dependencies {
```

```
    ...
```

```
    compileOnly 'com.google.android.things:androidthings:+'
```

```
}
```

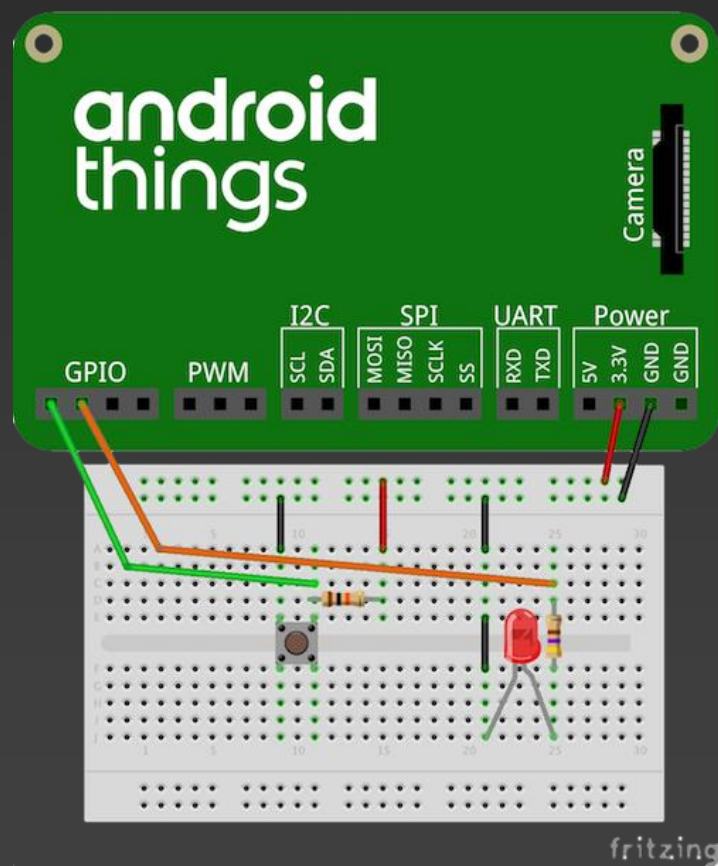
1 新建一个 Android Studio 工程

```
<application>
  <uses-library android:name="com.google.android.things" />
  <activity android:name=".HomeActivity">
    <!-- Launch activity as default from Android Studio -->
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>

    <!-- Launch activity automatically on boot, and re-launch if the app terminates. -->
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.HOME" />
      <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
  </activity>
</application>
```

2 连接硬件

GPIO (Genral Purpose Input Output)



3 与设备交互

```
import com.google.android.things.pio.PeripheralManager;
...

public class HomeActivity extends Activity {
    private static final String TAG = "HomeActivity";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        PeripheralManager manager = PeripheralManager.getInstance();
        Log.d(TAG, "Available GPIO: " + manager.getGpioList());
    }
}
```

监听按钮事件

```
PeripheralManager manager = PeripheralManager.getInstance();
try {
    // Step 1. Create GPIO connection.
    mButtonGpio = manager.openGpio(BUTTON_PIN_NAME);
    // Step 2. Configure as an input.
    mButtonGpio.setDirection(Gpio.DIRECTION_IN);
    // Step 3. Enable edge trigger events.
    mButtonGpio.setEdgeTriggerType(Gpio.EDGE_FALLING);
    // Step 4. Register an event callback.
    mButtonGpio.registerGpioCallback(mCallback);
} catch (IOException e) {
    Log.e(TAG, "Error on PeripheralIO API", e);
}
```

监听按钮事件

```
// Step 4. Register an event callback.  
private GpioCallback mCallback = new GpioCallback() {  
    @Override  
    public boolean onGpioEdge(Gpio gpio) {  
        Log.i(TAG, "GPIO changed, button pressed");  
  
        // Step 5. Return true to keep callback active.  
        return true;  
    }  
};
```

监听按钮事件

```
@Override
protected void onDestroy() {
    super.onDestroy();

    // Step 6. Close the resource
    if (mButtonGpio != null) {
        mButtonGpio.unregisterGpioCallback(mCallback);
        try {
            mButtonGpio.close();
        } catch (IOException e) {
            Log.e(TAG, "Error on PeripheralIO API", e);
        }
    }
}
```

控制 LED

```
// Step 1. Create GPIO connection.
PeripheralManager manager = PeripheralManager.getInstance();
try {
    mLedGpio = manager.openGpio(LED_PIN_NAME);
    // Step 2. Configure as an output.
    mLedGpio.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);

    // Step 4. Repeat using a handler.
    mHandler.post(mBlinkRunnable);
} catch (IOException e) {
    Log.e(TAG, "Error on PeripheralIO API", e);
}
```

控制 LED

```
private Runnable mBlinkRunnable = new Runnable() {
    @Override
    public void run() {
        // Exit if the GPIO is already closed
        if (mLedGpio == null) {
            return;
        }

        try {
            // Step 3. Toggle the LED state
            mLedGpio.setValue(!mLedGpio.getValue());

            // Step 4. Schedule another event after delay.
            mHandler.postDelayed(mBlinkRunnable, INTERVAL_BETWEEN_BLINKS_MS);
        } catch (IOException e) {
            Log.e(TAG, "Error on PeripheralIO API", e);
        }
    }
};
```

控制 LED

```
@Override
protected void onDestroy() {
    super.onDestroy();

    // Step 4. Remove handler events on close.
    mHandler.removeCallbacks(mBlinkRunnable);

    // Step 5. Close the resource.
    if (mLedGpio != null) {
        try {
            mLedGpio.close();
        } catch (IOException e) {
            Log.e(TAG, "Error on PeripheralIO API", e);
        }
    }
}
```

Phone + Android / iOS = Smart Phone

Device + Android Things = Smart Device

Smart $\neq$ Intelligent

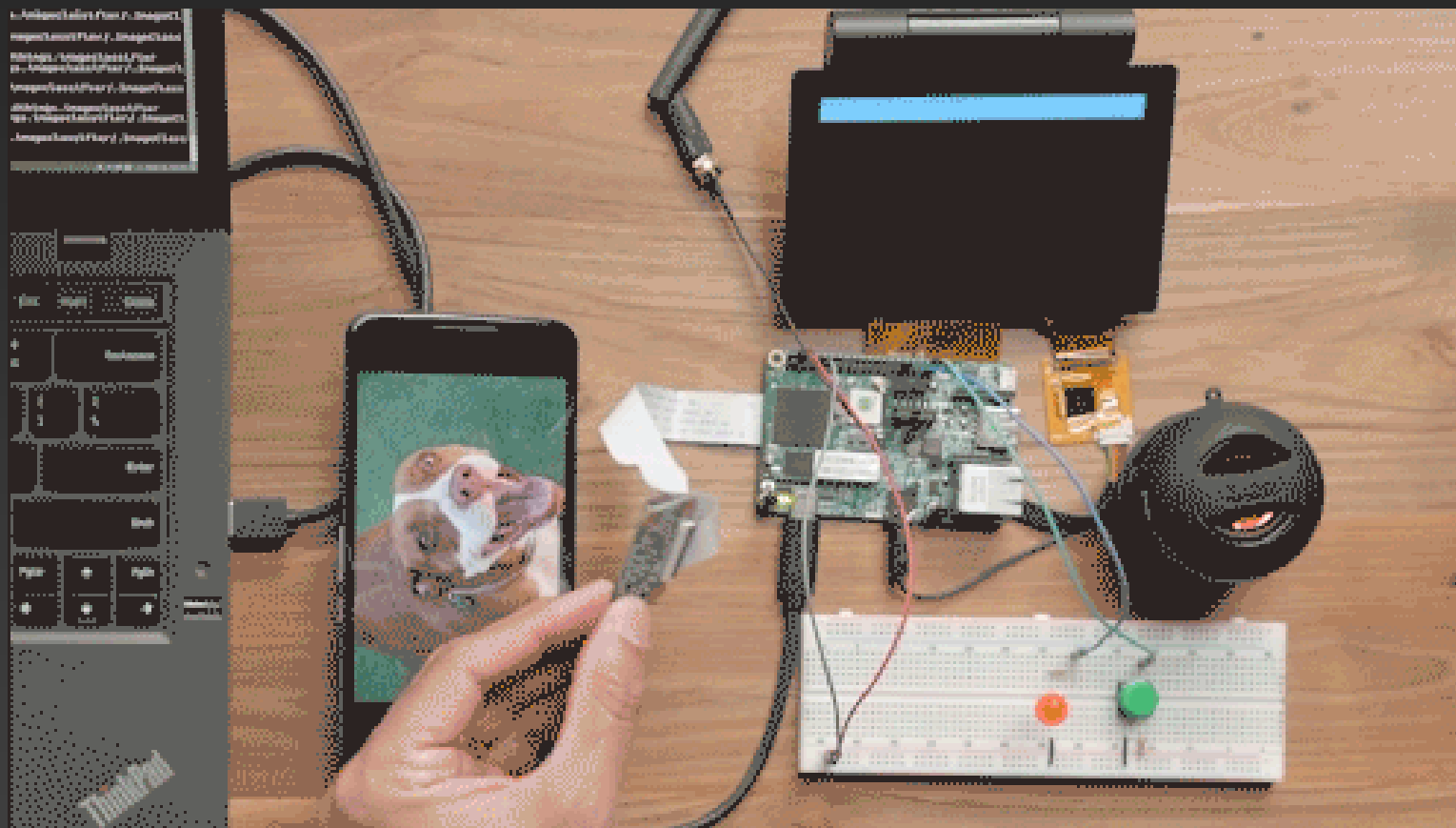
android
things

+


TensorFlow

TensorFlow 简介

TensorFlow™ 是一个开放源代码软件库，用于进行高性能数值计算。借助其灵活的架构，用户可以轻松地将计算工作部署到多种平台（CPU、GPU、TPU）和设备（桌面设备、服务器集群、移动设备、边缘设备等）。TensorFlow™ 最初是由 Google Brain 团队（隶属于 Google 的 AI 部门）中的研究人员和工程师开发的，可为机器学习和深度学习提供强力支持，并且其灵活的数值计算核心广泛应用于许多其他科学领域。



<https://github.com/androidthings/sample-tensorflow-imageclassifier>

```
249     final Collection<Recognition> results = mTensorFlowClassifier.doRecognize(bitmap);
250
251     Log.d(TAG, "Got the following results from Tensorflow: " + results);
252
253     runOnUiThread(new Runnable() {
254         @Override
255         public void run() {
256             if (results == null || results.isEmpty()) {
257                 mResultText.setText("I don't understand what I see");
258             } else {
259                 StringBuilder sb = new StringBuilder();
260                 Iterator<Recognition> it = results.iterator();
261                 int counter = 0;
262                 while (it.hasNext()) {
263                     Recognition r = it.next();
264                     sb.append(r.getTitle());
265                     counter++;
266                     if (counter < results.size() - 1 ) {
267                         sb.append(", ");
268                     } else if (counter == results.size() - 1) {
269                         sb.append(" or ");
270                     }
271                 }
272                 mResultText.setText(sb.toString());
273             }
274         }
275     });
```

Thanks