Node.js 开发物联网应用

张子发 (穆客) 2017/7/16

关于我

Node.js 内核

曾经的 IoT 开发者

议程

关于物联网

为什么是 Node.js

DEMO

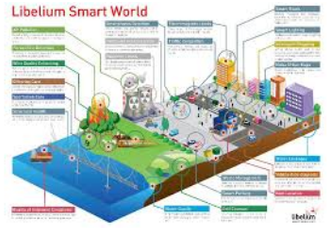
Tips

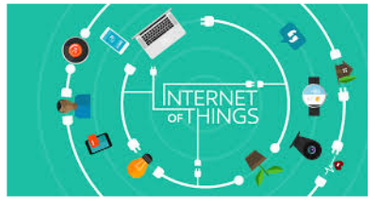
物联网

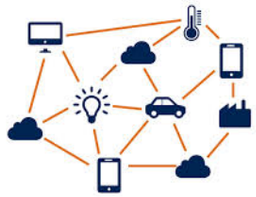

THE INTERNET of THINGS

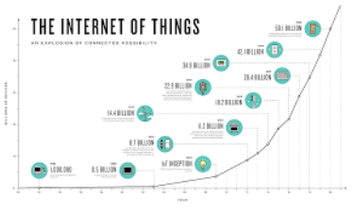

INTERNET of THINGS

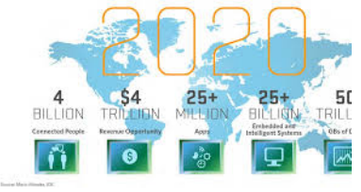

THE INTERNET of THINGS

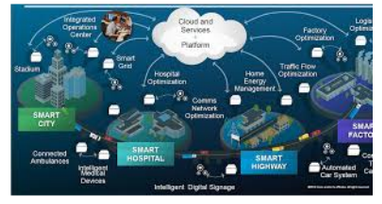

Libelium Smart World


INTERNET of THINGS



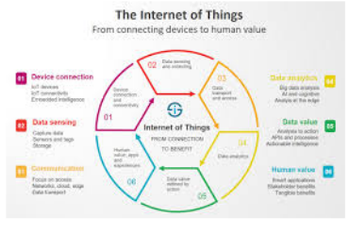

THE INTERNET OF THINGS


2020
4
BILLION
$4
TRILLION
25+
MILLION
25+
BILLION


Cloud and Services Platform
SMART CITY
SMART HOSPITAL
SMART HIGHWAY
Intelligent Digital Signage

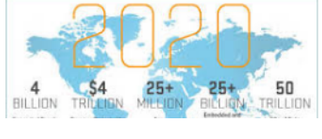

IoT


Internet of Things


The Internet of Things
From connecting devices to human value
Internet of Things
FROM CONNECTION TO BENEFIT
Device connection
Data sensing
Communication
Data analytics
Data value
Human value


THE INTERNET


Internet of THINGS


2020
4
BILLION
$4
TRILLION
25+
MILLION
25+
BILLION
50
TRILLION


THE INTERNET OF

物联网

移动支付 / 智能交通 / 物流 / 环境检测 / 安防

智慧建筑 / 智能家居 / 智慧医疗 / 食品溯源 / 电网

公共安全 / 教育 / 智能制造 /...

物联网

数据产生 - 传感器

数据收集 - 网络传输

数据分析 -(云) 服务器

执行分析结果 - 执行机构 / 推送

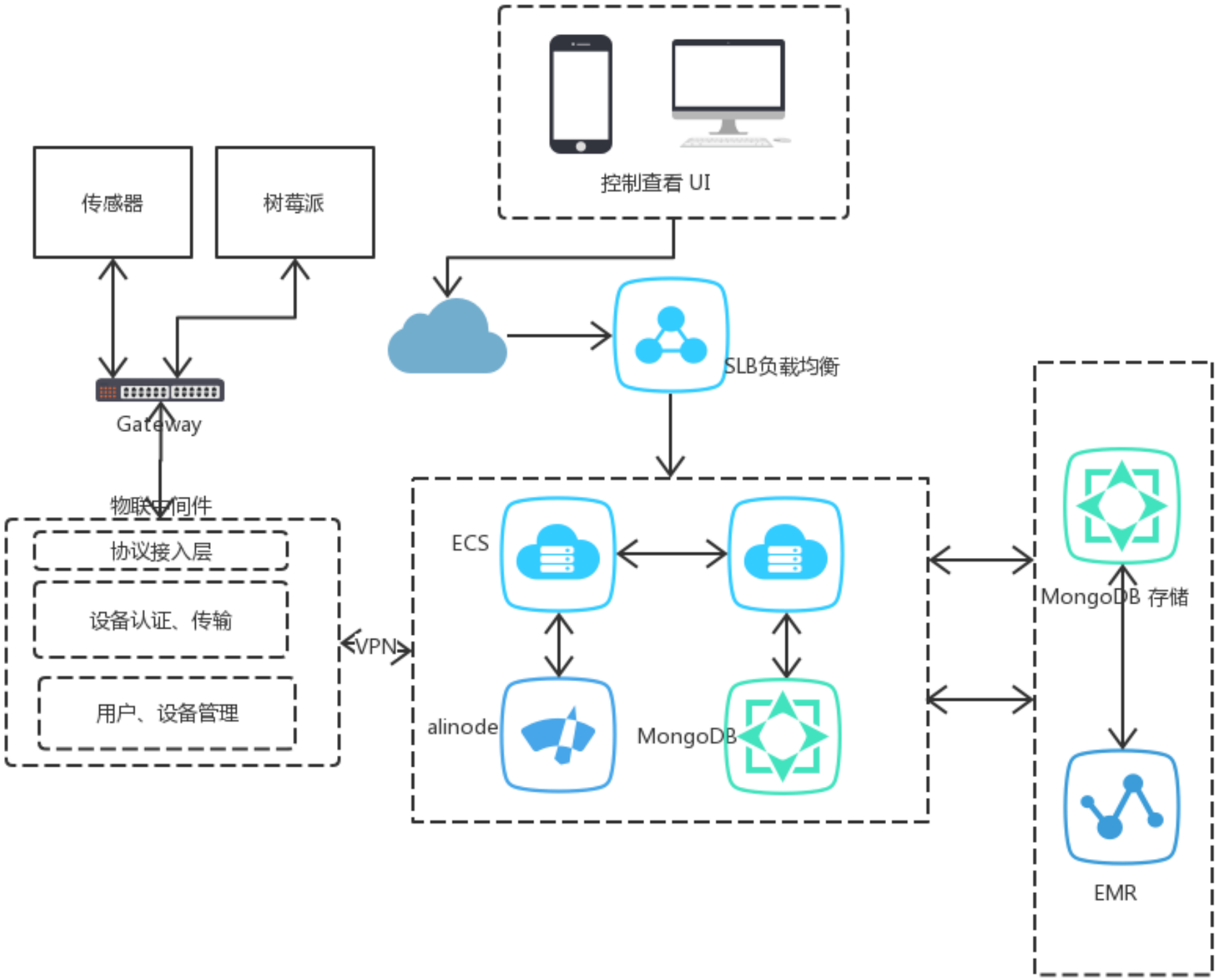
物联网开发

硬件的发展

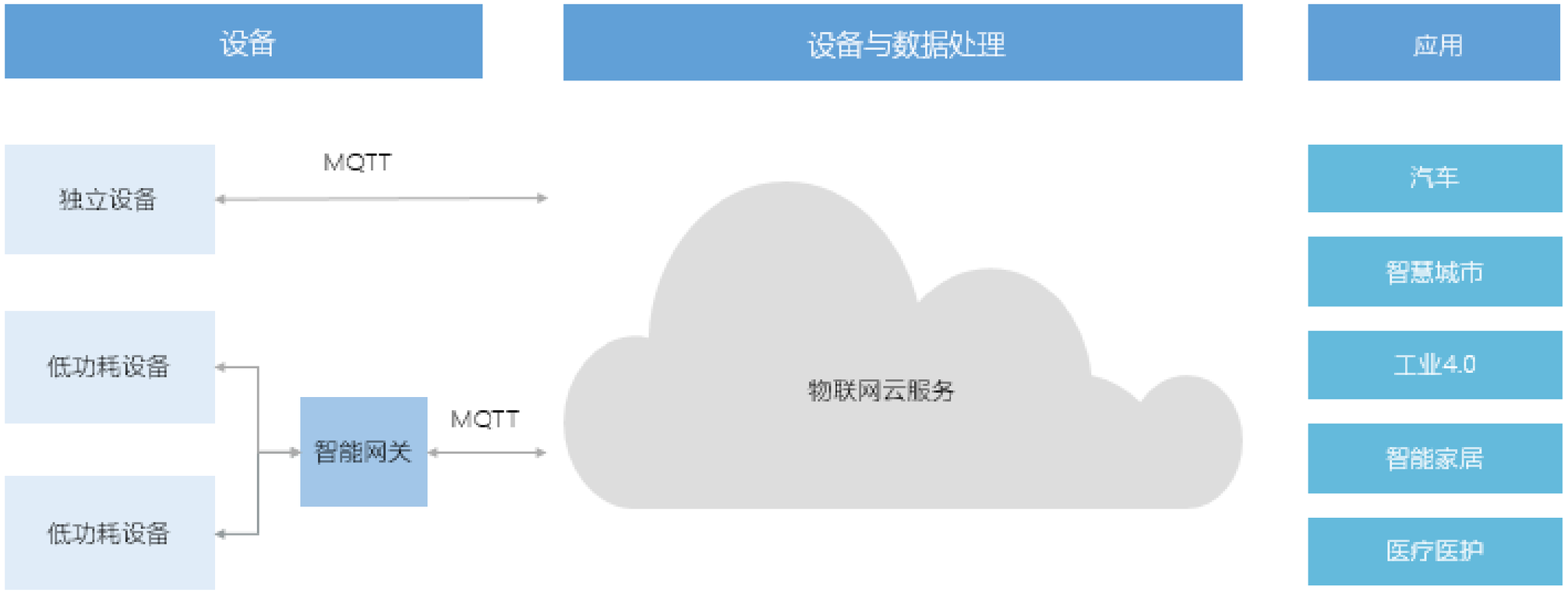
软件的进步

云计算 / 大数据

物联网开发


传感器
树莓派
控制查看 UI
Gateway
物联网间件
协议接入层
设备认证、传输
用户、设备管理
VPN
SLB负载均衡
ECS
alinode
MongoDB
MongoDB 存储
EMR

物联网开发


设备
设备与数据处理
应用
独立设备
MQTT
低功耗设备
低功耗设备
智能网关
MQTT
物联网云服务
汽车
智慧城市
工业4.0
智能家居
医疗医护

为什么是 Node.js

生态

高并发

易扩展

学习曲线

开发效率

前后端沟通

DEMO

DEMO

树莓派

通过串口接收 pm2.5 传感器数据

通过 IO 控制蜂鸣器

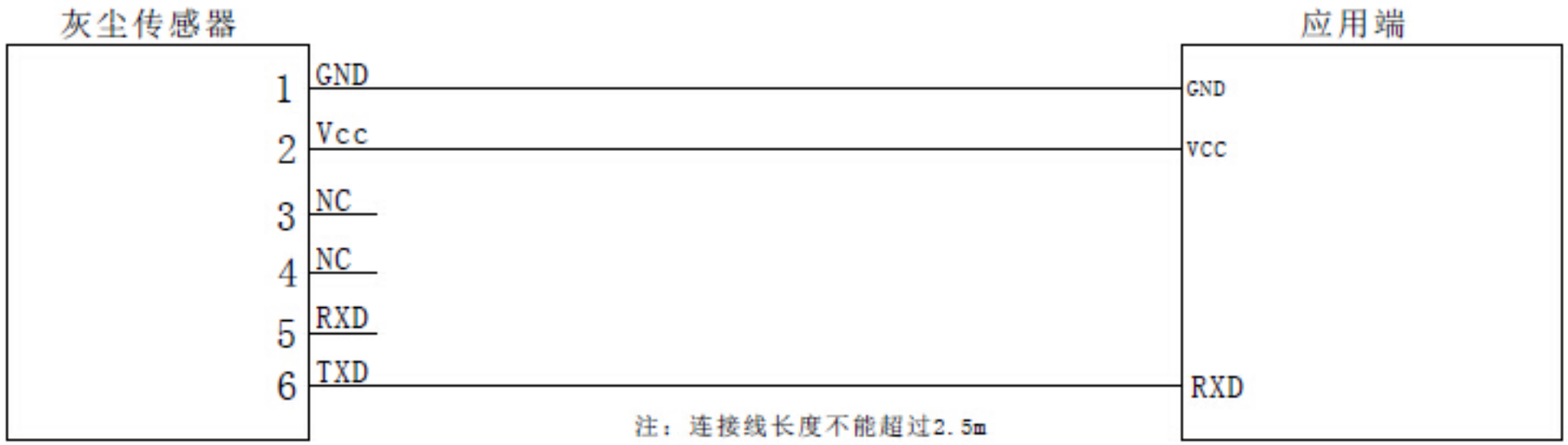
通过 IO 控制电机

通过 mqtt 与云服务器交互

依赖：

wiring-pi　serialport　mqtt

DEMO — UART



串口输出参数

- i. 波特率:2400 bit/s;
- ii. 每10ms发送一个字节,共7个字节,其中校验位=Vout(H)+ Vout(L)+Vref(H)+ Vref(L);
- iii. 数据发送格式:

起始位	Vout(H)	Vout(L)	Vref(H)	Vref(L)	校验位	结束位
0xaa	如:0x01	如:0x3a	如:0x00	如:0x7a	如:0xd0	0xff

- 4)数据处理: 接收到的数据按公式计算后得到Vout的值: Vout=(Vout(H)*256+Vout(L))/1024 * 5

DEMO

```
const COM = require('serialport');
const port = new COM('/dev/ttyAMA0', { baudRate: 9600});
port.on(‘data’, function(data) {
  dataHandler(data);
});
```

DEMO

```
const wpi = require('wiring-pi');
const BUZZER = 7;

wpi.setup(‘wpi’);

wpi.pinMode(BUZZER, wpi.OUTPUT);

wpi.digitalWrite(BUZZER, wpi.HIGH);

wpi.digitalWrite(BUZZER, wpi.LOW);

wpi.digitalRead(BUZZER);
```

DEMO

```
const wpi = require('wiring-pi');
const [M1, M2, M3, M4] = [1, 4, 5, 6];
wpi.setup(‘wpi’);
wpi.pinMode(M1, wpi.OUTPUT);
wpi.pinMode(M2, wpi.OUTPUT);
wpi.pinMode(M3, wpi.OUTPUT);
wpi.pinMode(M4, wpi.OUTPUT);

const run = function() {
  wpi.digitalWrite(M1, wpi.HIGH);
  wpi.digitalWrite(M2, wpi.HIGH);
  wpi.digitalWrite(M3, wpi.HIGH);
  wpi.digitalWrite(M4, wpi.HIGH);
};

.....
```

DEMO

```
const mqtt = require('mqtt');
const client = mqtt.connect('mqtt://<ip:port>');

client.subscibe(‘/car’);
```

```
client.on(‘message’, function(topic, message) {
  If (topic === ‘/car’) {
    MessageHandler(message.toString());
  }
});
```

DEMO

云服务器 : ECS

数据传输 : MQTT

数据存储： mysql/mongoDB/redis

前端展现： express

依赖：

mosca/mysql/ mongo/redis/express

DEMO

```
const mosca = require('mosca');

const brokerSettings = {
    port: 1881,
    ttl: {
        subscriptions: 1000 * 60 * 10,
        packets: 1000 * 60 * 10
    }
}
var broker = new mosca.Server(brokerSettings);
broker.on('ready', brokerReady);
broker.on('clientConnected', clientConnected);
broker.on('clientDisconnected', clientDisconnected);
```

DEMO

```
broker.on('published', function(packet, client){
    switch(packet.topic) {
    }
});

exports.carCtrl = function(action){
  var packet = {
    topic: '/car',
    payload: action,
    qos:1,
    retain: false
  } ;

  broker.publish(packet, function() {  });
};
```

Tips

硬件 / 开发周期 / 生产 / 采购 / 评测 / 认证检测

消费电子 / 安全 / WiFi / 成本 /control4

支撑 Node.js 运行 / Ruff

通用云服务 / 大数据 / 机器学习 / 算法

行业 SaaS 服务 / 为服务付费 /Node.js 工具链

传统垂直行业，生产制作行业，汇编 /C/PLC

开发效率！ Node.js? Js? 有人给你验证？

玩票 / STEAM / 前端 / 分工合作 / 木匠

强需求，用户 / 行业痛点

用户体验 / MagicLight / 单火线开关

谢 谢