華中農業大學

HUAZHONG AGRICULTURAL UNIVERSITY

web 课程设计

题　　目　模拟牛联网

姓　　名　付义　　学　　号　2016317200308

姓　　名　程时坤　学　　号　2016317200302

专业班级　计算机科学与技术 1603 班

指导老师　金星

中国・武汉

2019 年 1 月

目录

前言：项目地址

http://123.207.19.172:8081/

一、需求分析

模拟牛场使用物联网存储和分析牛的生理状态，通过抬头和低头次数分析牛的消化情况，通过牛站起和趴下的次数分析牛的关节状态，通过牛的呼吸频率和深度分析牛的呼吸系统健康状态。 这些数据通过芯片采集，芯片原理是测量三维坐标系中加速度，通过不同的公式计算牛不同动作状态。

首先需要模拟牛不同行为状态下的数据，每种类型数据均考虑到噪声的存在， 数据产生原理和方法在后面会专门阐述； 数据产生后存放到数据库中，这就要求后台对数据库进行操作； 数据产生并存放到数据库之后由后台分析数据的程序顺序读取牛的数据，然后通过算法分析，得出数据对应牛的状态， 得到牛的状态之后进行两步操作，第一，将分析结果存放到结果数据库表中以备后续查询；第二，将结果封装返回前台，由 js 处理之后显示到 html 界面，可视化结果。

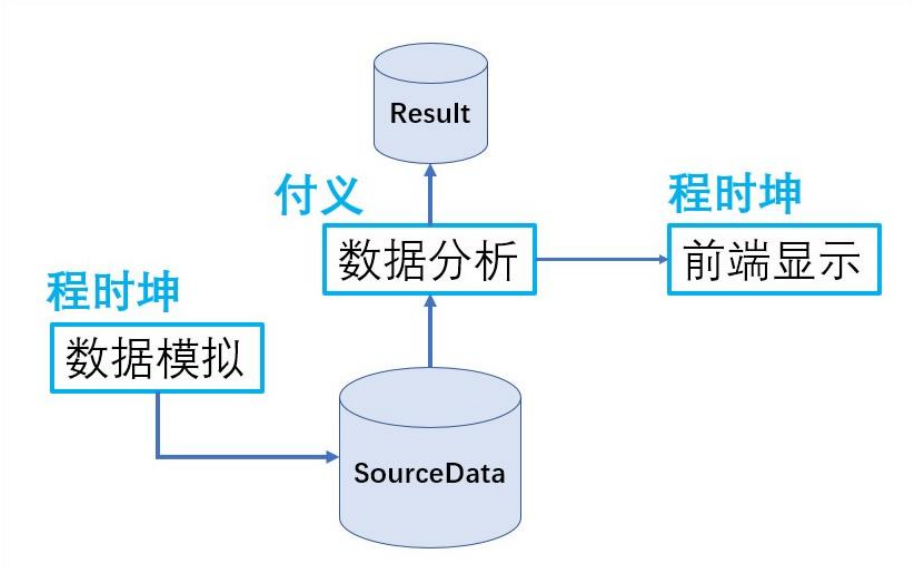
在牛的行为中，我们假定：抬头一次或者低头一次用时 1 秒钟左右（有噪声）， 1 分钟到 2 分钟抬头或低头一次； 站起或者趴下用时 3 秒钟左右， 大约 1 分钟到 5 分钟站起或者趴下一次， 站起或者趴下的位移为大约 1m； 呼吸是正弦周期性的，呼吸周期 1 秒钟到 2 秒钟，有噪声。 芯片采样频率 50HZ，即 1 秒钟采样 50 次，每次采样 1 分钟，采样间隔大约 30 秒钟。

二、小组分工

数据模拟：程时坤

数据分析：付义

前端显示：程时坤



三、实验

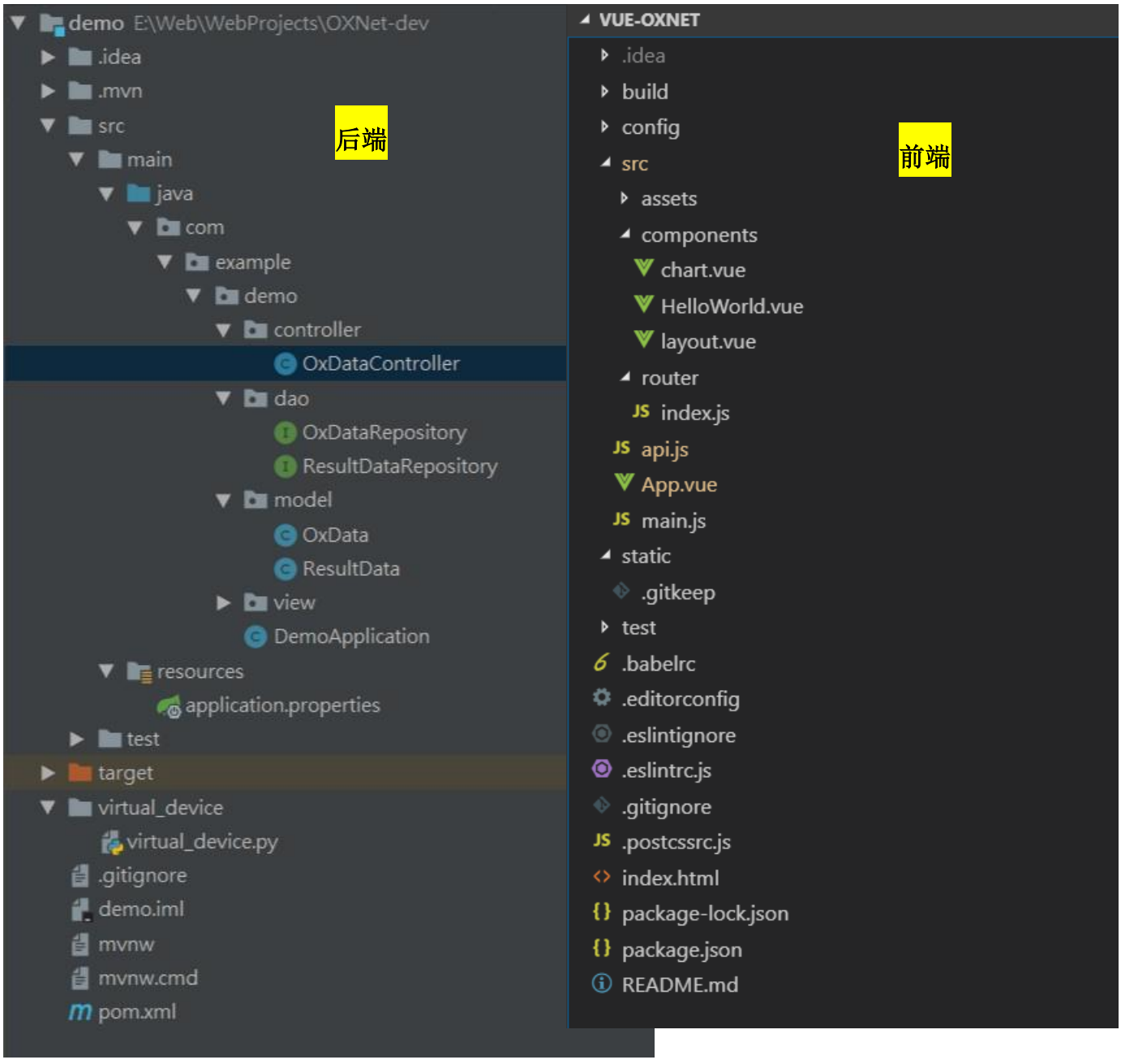
—）实验环境和框架

实验环境：IntelliJ IDEA 2018.3.2 x64 （集成 tomcat：apache-tomcat-8.5.37，python）， mysql：mysql-8.0.13-winx64

前端框架：vue　　**语言**：HTML，js（vue.js）

后端框架：springboot　　**语言**：java， sql， python

项目工程结构


demo E:\Web\WebProjects\OXNet-dev
.idea
.mvn
src
main
java
com
example
demo
controller
OxDataController
dao
OxDataRepository
ResultDataRepository
model
OxData
ResultData
view
DemoApplication
resources
application.properties
test
target
virtual_device
virtual_device.py
.gitignore
demo.iml
mvnw
mvnw.cmd
pom.xml
后端
VUE-OXNET
.idea
build
config
src
assets
components
chart.vue
HelloWorld.vue
layout.vue
router
index.js
api.js
App.vue
main.js
static
.gitkeep
test
.babelrc
.editorconfig
.eslintignore
.eslintrc.js
.gitignore
.postcssrc.js
index.html
package-lock.json
package.json
README.md
前端

二）实验步骤

（一）数据模拟 （程时坤）

1. 思路：

牛的身体、头部和呼吸活动是同步进行的，而物联网设备采样也是与它们并行的。所以就自然想到使用多线程的方法模拟数据采样。

牛的身体、头部和呼吸活动以及物联网设备采样过程分别占用一个线程，在采样线程中综合牛活动的三个线程的数据就可以得到采样的结果。

具体模拟方法：

2.1 牛呼吸的过程产生的加速度建模为一个正弦信号，在呼吸线程中随机隔一定时间改变正弦信号的振幅和频率，模仿呼吸速率的改变。

```
# 每隔一段时间改变一些呼吸的频率和深度
time_slot = random.randint(10, 60)
time.sleep(time_slot)
# a range 0.1 - 0.5
global a
global w
a = random.random() * 0.4 + 0.1 + noise()
# w range 2PI - 20PI
w = random.random() * 2 * math.pi + 2 * math.pi
```

2.2 牛身体的上下移动产生的加速度使用一个常数来模拟，移动持续的时间使用利用位移公式精确地计算出来。

```
time_slot = random.randint(60, 5 * 60)
time.sleep(time_slot)
global body_a
body_a = random.random() * 2 - 1 + noise()
time.sleep(math.sqrt(2.0 / math.fabs(body_a)))
body_a = 0 + noise()
```

2.3 头部的动作进行了简化，假设传感器水平放置，牛的抬头和低头动作也是垂直的，不会歪着头，于是头部的动作就会导致 z 信号的变化，是 z 信号不为 0。

```
time_slot = random.randint(60, 2 * 60)
time.sleep(time_slot)
global z_angle
z_angle = random.randint(-30, 60) + noise()
# x_angle = random.randint(-60, 60)
```

2.4 噪声的产生使用一个较小的随机数来模拟。

```
def noise():
    return random.random() * 0.1 - 0.05
```

2.5 采样过程将所有在牛运动的线程中的参数代入总的运动的方程中，计算出综合的 x, y, z 值并存入数据库中

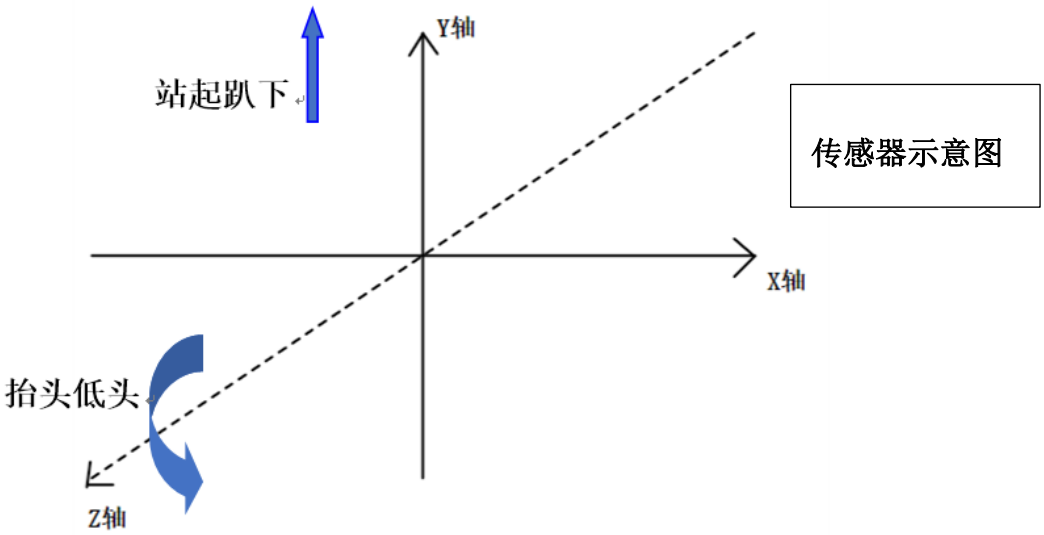
```
while True:
    time.sleep(0.02)
    t = time.clock()
    cnt += 1
    f = 9.8 + a * math.sin(w * t) + body_a
    z_ = f * math.cos(2 * math.pi * (90 + z_angle) / 360.)
    y_ = f * math.sin(2 * math.pi * (90 + z_angle) / 360.)
    x_ = 0
    # print("%d\tx:%.1f\ty:%.1f\tz:%.1f" % (cnt, x_, y_, z_))
    # print("a:%f\tw:%f\tbody_a:%f\tz_angle:%d\t" % (a, w, body_a,
z_angle))
    # print("analize:%f"%(math.sqrt(x_ ** 2 + y_ ** 2 + z_ ** 2)))
    sql = "insert  into ox_data(x,y,z) values(%f, %f, %f);" %(x_, y_, z_)
    print("sql:", sql)
    cursor.execute(sql)
    db.commit()
```

2. 模拟结果

模拟的过程对真实情景进行了大量的简化--规定传感器初始时是水平放置的，牛头部的动作也是垂直的。这样简化的结果就是：x 的值一直为 0，可以直接通过 z 的值的符号判断牛的抬头或着低头。

id	x	y	z
26921	0	8.34629	-5.01138
26922	0	8.34385	-5.00991
26923	0	8.34137	-5.00843
26924	0	8.3388	-5.00688
26925	0	8.33637	-5.00542
26926	0	8.33376	-5.00386
26927	0	8.33115	-5.00229
26928	0	8.3285	-5.0007
26929	0	8.32589	-4.99913
26930	0	8.32339	-4.99763
26931	0	8.32167	-4.9966

（二）数据分析 （付义）



读取数据库：采用 springboot 内置 mysql 数据库服务，连接我们服务器的存储。

读取方法是通过 springboot 框架，编写自定义 sql 语句（springboot 里有原声的 SQL 语句，例如 getall(), getBy*()…）

首先需要编写 model 层并申明为实体 entity， 编写需要的对象属性和 get， set 方法

```
@Entity
public class OxData implements Serializable {
    private static final long serialVersionUID = 1L;


    @Id
    @GeneratedValue
    private Long id;

    @Column(nullable = true)
    private Float x;

    @Column(nullable = true)
    private Float y;

    @Column(nullable = true)
    private Float z;

public Float getX() {
    return x;
}

public void setX(Float x) {
    this.x = x;
```

```
}

public Float getY() {
    return y;
}

public void setY(Float y) {
    this.y = y;
}

public Float getZ() {
    return z;
}

public void setZ(Float z) {
    this.z = z;
}
```

首先在 dao 层编写接口并声明为库

```
@Repository
@Transactional
public interface OxDataRepository extends JpaRepository<OxData, Long>
```

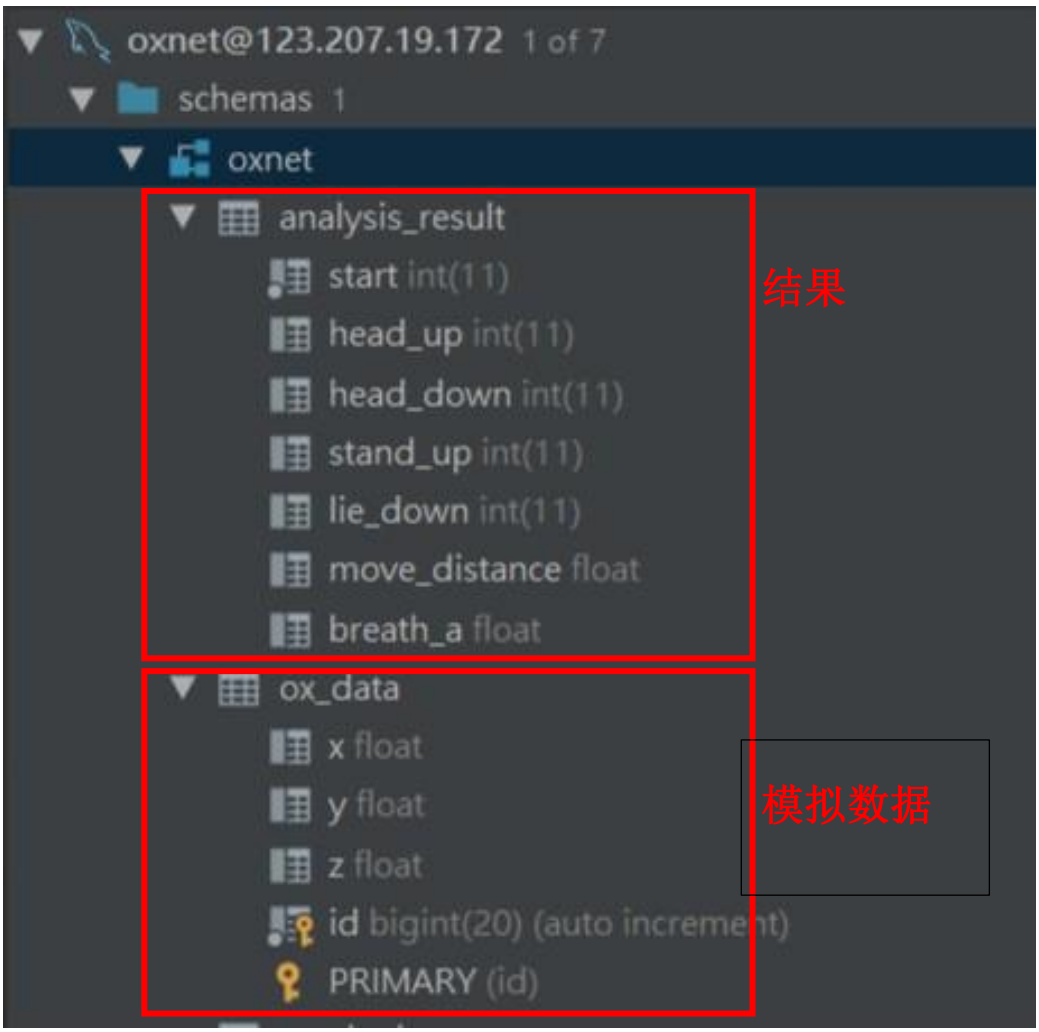
然后编写 SQL 语句

```
@Modifying
@Query(value = "select * from ox_data LIMIT ?1,?2", nativeQuery = true)
List<OxData> findNext(int start,int offset);
```

这样就可以通过 springboot 的框架查询数据库， 返回的是 List 类型， 通过 OxData 对象属性的 get 方法可以得到数据

```
for(int i = 0; i < 150; i++){
    a1 += firstSet.get(i).getY();
    a2 += SecondSet.get(i).getY();
    a3 += ThirdSet.get(i).getY();
}
```

数据库：一共两个数据库表格， 一个存放模拟数据，一个存放分析结果



模拟数据格式

1001-1500 of 1501+ Tx: Auto

<Filter criteria>

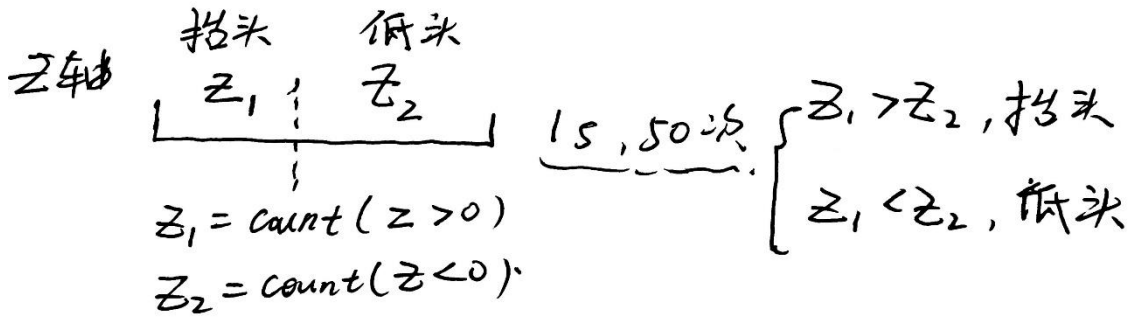
	x	y	z	id
1154	0	9.88252	0	26870
1155	0	9.87961	0	26871
1156	0	9.87754	0	26872
1157	0	9.87474		
1158	0	9.87163		
1159	0	9.86845		
1160	0	9.86517		
1161	0	9.86183		
1162	0	9.85907		
1163	0	9.85622		
1164	0	9.85344		
1165	0	9.85059		
1166	0	9.84829		
1167	0	9.845		
1168	0	9.84243		
1169	0	9.83954		
1170	0	9.83675		
1171	0	9.83395		
1172	0	8.42867		
1173	0	8.42698		
1174	0	8.42442		
1175	0	8.42188		
1176	0	8.41918		
1177	0	8.41658		
1178	0	8.41483	-5.05…	26894
1179	0	8.41227	-5.051	26895
1180	0	8.40977	-5.04…	26896
1181	0	8.4075	-5.04…	26897
1182	0	8.40486	-5.04…	26898

分析结果数据格式

	start	head_up	head_down	stand_up	lie_down	move_distance	breath_a
91	2250	0	1	1	0	0	-1.41971
92	2250	0	1	1	0	0	-1.41971
93	2300	0	1	1	0	0	-1.41971
94	2300	0	1	1	0	0	-1.39142
95	2350	0	1	1	0	0	-1.39142
96	2350	0	1	1	0	0	-1.36328
97	2400	0	1	1	0	0	-1.36328
98	2400	0	1	1	0	0	-1.33938
99	2400	0	1	1	0	0	-1.33938
100	2450	0	1	1	0	0	-1.33938
101	2450	0	1	1	0	0	-1.32137
102	2450	0	1	1	0	0	-1.32137
103	2500	0	1	0	1	0.981447	-1.52912
104	2500	1	0	0	1	0.981447	-1.52912
105	2500	1	0	0	1	0.981447	-1.52912
106	2500	1	0	0	1	0.981447	-1.52912
107	2600	1	0	0	1	0.981447	-0.26499
108	2600	1	0	0	1	0.981447	-0.26499
109	2650	1	0	0	1	0.855276	-0.316942
110	2650	1	0	0	1	0.855276	-0.316942
111	2700	1	0	0	1	1.15955	-0.384558
112	2700	1	0	0	1	0.855276	-0.376402
113	2700	1	0	0	1	0.855276	-0.376402
114	2700	1	0	0	1	0.855276	-0.376402
115	2700	1	0	0	1	0.855276	-0.376402
116	2800	1	0	0	1	1.15955	-0.458598
117	2800	1	0	0	1	1.15955	-0.458598
118	2750	1	0	0	1	1.15955	-0.458598
119	2850	1	0	0	1	1.05046	-0.398537

行为分析

抬头、低头：在产生抬头低头的模拟数据的时候我们采用了抬头低头角度，抬头 30 度， 低头 60 度，通过正弦函数计算之后，Z 轴的数据在抬头时大于 0，低头时小于 0， 那么可以通过这个特点很容易计算出当前是抬头还是低头。 我们前端每



1s 请求一次后端，那么数据分析就是 1s 分析一次，1s 采样 50 次，考虑到这一秒钟里面可能有部分在抬头，有部分在低头，那么通过统计每次采样 Z 轴正负，然后比较正负次数来判定这一秒内牛是处于抬头还是低头状态

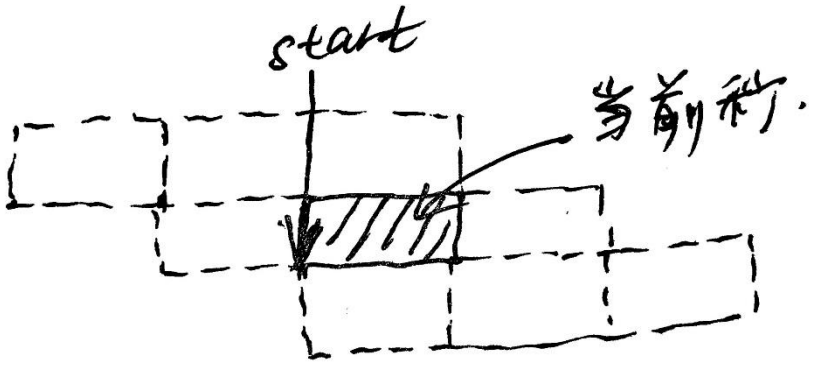
代码实现比较简单：

```
int z_count = 0;
if ((oneSet = oxDataRepository.findNext(start,offset)) != null){
    float ox_data[][] = new float[5][3];
    for (int i = 0; i < 50; i += 10 ){
        int index = i/10;
        System.out.println("index:"+index+"\n");
        ox_data[index][0] = oneSet.get(i).getX();
        ox_data[index][1] = oneSet.get(i).getY();
        ox_data[index][2] = oneSet.get(i).getZ();
    }
    flag.put("data",ox_data);
    System.out.println("oneSet:"+oneSet+"\n");
    for (OxData next: oneSet) {
        if(next.getZ() > 0 ){
            z_count ++;
        }else if(next.getZ() < 0){
            z_count--;
        }
    }
    if(z_count > 0){  //head_up
        flag.put("head_up","1");
        flag.put("head_down", "0");
```

```
        head_up = 1;    head_down = 0;
    }else if(z_count < 0){ //head_down
        flag.put("head_up", "0");
        flag.put("head_down", "1");
        head_up = 0;    head_down = 1;
    }else {  //not move head
        flag.put("head_up", "-1");
        flag.put("head_down", "-1");
        head_up = -1;   head_down = -1;
    }
```

站起、趴下：加速度传感器，只能探测到线性加速度，而且因为不知道牛在哪几秒钟发生了站起或趴下的动作，需要用一个 3 秒的观察窗口进行扫描。对每一次请求，内存中有个计数器计数上一次的数据访问下标 start， 下次请求数据库时从 start+offset 开始（offset 为每次请求数据条目数，设定为 50， 因为芯片频率 50HZ，每秒采样 50 次）， 对于一次请求，需要 3s 的数据来分析是站起还是趴下，并且考虑到站起和趴下的时间是 3s 左右， 那么当前秒可能处于 3s 的第一秒或者第二秒或者第三秒，那么就要进行三次 3s 的数据读取和分析，只要有一次当前秒处在站起或趴下状态，就认为当前站起或趴下

分析当前状态（1s），进行三次读取数据库，每次读取 3s （150 条）数据。



对每 3s 数据计算其平均加速度，将其站起趴下过程看做匀加速运动，计算运动位移，将计算位移和实际位移进行比较，如果和实际位移（0.8m ~ 1.2m， 参数可调）相差较小，就认为这 1s 在上升或下降，然后通过比较计算的平均加速度和静

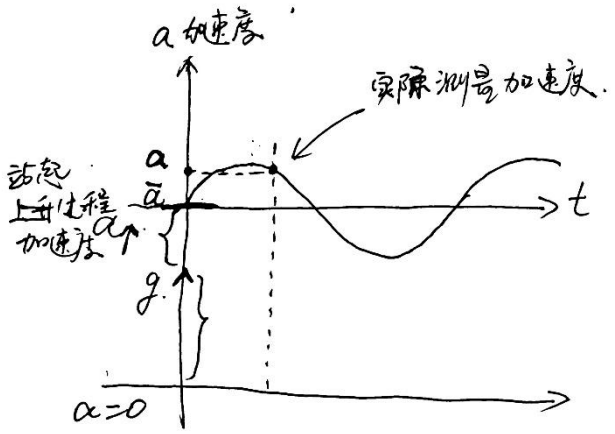
止时的重力加速度比较， 如果计算的平均加速度大于重力加速度，则认为牛在站起，反之在趴下； 如果计算的位移比较小，那么认为牛没有运动。

```
a1 = a1/150;    heigth1= Math.abs((float) (1.0/2*(a1-9.8)*9));
a2 = a2/150;    heigth2= Math.abs((float) (1.0/2*(a2-9.8)*9));
a3 = a3/150;    heigth3= Math.abs((float) (1.0/2*(a3-9.8)*9));

float body_a = 0;
List<Object> stand_lie ;
if(heigth1 > distance_lower && heigth1 < distance_upper){  //stand_up or
lie_down and move_distance

    stand_lie = position_judge(a1,heigth1,flag);
    body_a =Math.abs( (float) (a1 - 9.8));

}else if(heigth2 > distance_lower && heigth2 < distance_upper){
    stand_lie = position_judge(a2,heigth2,flag);
    body_a =Math.abs( (float) (a2 - 9.8));
}else if (heigth3 > distance_lower && heigth3 < distance_upper){
    stand_lie = position_judge(a3,heigth3,flag);
    body_a =Math.abs( (float) (a3 - 9.8));
}else {
    body_a = 0f;
    stand_lie = position_judge(9.8f,0,flag);
    System.out.println("ox don't move\n");
}
```

运动量：在计算牛站起和趴下的过程中可以得到当前 1s 的位移，如果牛在站起或趴下，则可将此位移当做牛的运动量。

```
a1 = a1/150;    heigth1= Math.abs((float) (1.0/2*(a1-9.8)*9));
a2 = a2/150;    heigth2= Math.abs((float) (1.0/2*(a2-9.8)*9));
a3 = a3/150;    heigth3= Math.abs((float) (1.0/2*(a3-9.8)*9));
```

呼吸频率和深度：在计算站起和趴下的过程中，可以计算得 3s 内 Y 轴数据的平均值 $\bar{a}$，因为呼吸是正弦，所以在一个周期里面呼吸的正半轴和负半轴相互抵消，最后就在 $\bar{a}$ 附近波动，那么 $\bar{a} - 9.8$ 就得到上升或者下降的近似加速度a^1，这样在 1s 采样中，我们就可以用实际值 a – 9.8 - a^1 并求和求平均得到 1s 内呼吸加速度的平均值，这样做是考虑到呼吸周期为 2s 左右，那么 1s 就只有半个周期，不会正负周期相互抵消； 呼吸深度可以通过呼吸平均加速度和呼吸频率求出。



这部分代码还有少许 bug，仍在调试中

```
/* analysis breath: breath_frequency and breath_depth */
if( start%3000 == 0){
    int start_tmp = start;
    //oneSet = oxDataRepository.findNext(start,3000);
    int j = 0;
    boolean end_flag = false;
    for (int i = 0; i < 60; i++, start_tmp+=50){  //get the first index
where ox only breath
        oneSet = oxDataRepository.findNext(start_tmp,offset);
        int z = -1, y=-1;
        for( ; j < 50; j+=5){
            OxData sample = oneSet.get(j);
            if(sample.getZ() == 0){ z = 0; }
            if(sample.getY() < 10 && sample.getY() > 9.5){  y = 0; }
            if(z == 0 && y ==0){
                end_flag = true;
                break;
            }
        }
        if(end_flag){ break; }
    }
```

```
    oneSet = oxDataRepository.findNext(j,3*offset);
    float a_sum = 0;
    for (int i =0; i < 50; i++){
        a_sum += Math.abs(oneSet.get(i).getY() - 9.8);
    }
    breath_a = a_sum/100;  //breath acceleration
    /* analysis breath frequency and depth */
    int y_count = 0;
    int t_count = 0;
    float y_start = oneSet.get(0).getY();
    float y_tmp = 0;
    for(int i = 0; i < 3*offset; i++){
        t_count++;
        if(Math.abs(oneSet.get(i).getY()-y_start) < 0.2 ){
            y_count ++;
            i += 10;
        }

    }
}
```

分析结果返回： 分析的结果保存为 **HashMap** 格式，使用简单而且功能强大，通过 **springboot** 框架返回到前端成为 **json** 格式，很方便前端解析；

```
@CrossOrigin("*")
@RequestMapping("/analysis_pers")
public HashMap<String, Object> analysis_pers(){
    HashMap<String,Object> flag = new HashMap<>();
```

```
flag.put("head_up","1");
flag.put("head_down", "0");
```

```
flag.put("standUp",standUp.toString());
flag.put("lieDown",lieDown.toString());
flag.put("headUp", headUp.toString());
flag.put("headDown", headDown.toString());
```

（三）前端显示 （程时坤）

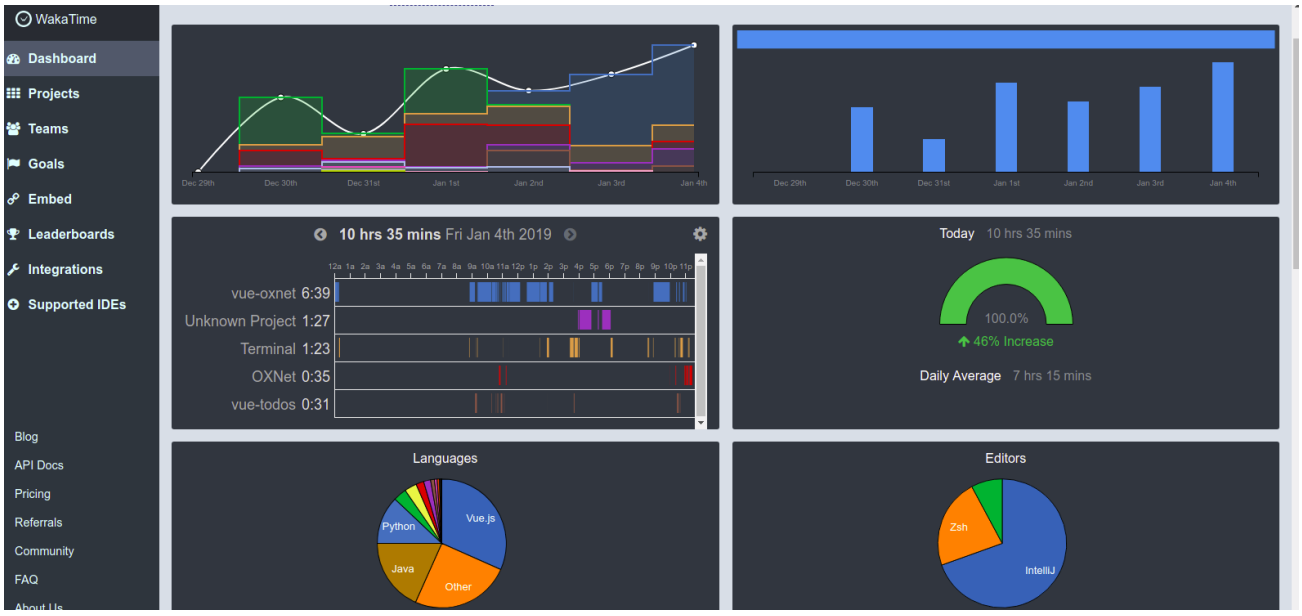
1. 框架

因为作为一个数据可视化的页面，必然要不断刷新页面的数据，为了保证刷新过程的流畅性和美观性，选用了最新的响应式框架 Vue.js

2. 界面设计

2.1 采用单页面无跳转的设计方法，便于实时地全面地观察数据的变化情况。

2.2 页面的布局参考了 Wakatime 网站（一个统计敲代码时间的网站）



2.3 它暗色调的风格十分有科技感，所以也被作为牛联网前端的配色风格。在选择暗色风格的配色时也参考了其他网站。如：阿里云，腾讯云，Bmob 后端云


阿里云
云AP
中国站
购物车
控制台
文档
备案
邮箱
登录
最新活动
产品
解决方案
定价
ET大脑
数据智能
安全
云市场
支持与服务
合作伙伴
免费注册
阿里云ocr全新功能上线
英文专项识别、文档结构化、自定义全新上线。单价低至0.005元/次起
免费体验
机器学习PAI
人人都用得起的机器学习平台
ocr自定义模版
满足各种个性化图片识别需求
数据智能体验馆
可视化界面，操作简单
工业设备预测性维护
设备状态信息透视全景在线
咨询·建议


Bmob
首页
产品
源码
价格
文档
下载
实战视频
登录
Bmob 后端云 全方位一体化的后端服务平台
提供可靠的 Serverless 云服务 轻松拥有开发中需要的各种后端能力

2.4 最终结果

具体实现

3.1 图表的绘制使用 v-charts 框架，页面布局组件使用 element 框架

3.2 主要的布局

```
<el-row :gutter="40">
 <el-col :span="12"  >
  <h1>呼吸</h1>
  <ve-line :extend="chartExtend" :data="breathData"></ve-line>
 </el-col>
 <el-col :span="12">
```

```
    <h1>头部姿势</h1>
    <el-row>
      <el-col :span="12" >
        <ve-pie  :extend="chartExtend" :data="headData"
style="/*border-top-right-radius: 0;border-bottom-right-radius:
0*/" ></ve-pie>
      </el-col>
      <el-col :span="12" >
        <!--<ve-waterfall  :extend="chartExtend" :data="headTimeData"
style="border-bottom-left-radius: 0;border-top-left-radius: 0"></ve-
waterfall>-->
        <h1>当前姿势</h1>
        <h1 style="font-size: 100px;">{{headData.pose}}</h1>
      </el-col>
    </el-row>
  </el-col>
 </el-row>
```

3.3 数据绑定，Vue 为响应式布局，这些绑定好的变量在被改变后页面就会被重新渲染。

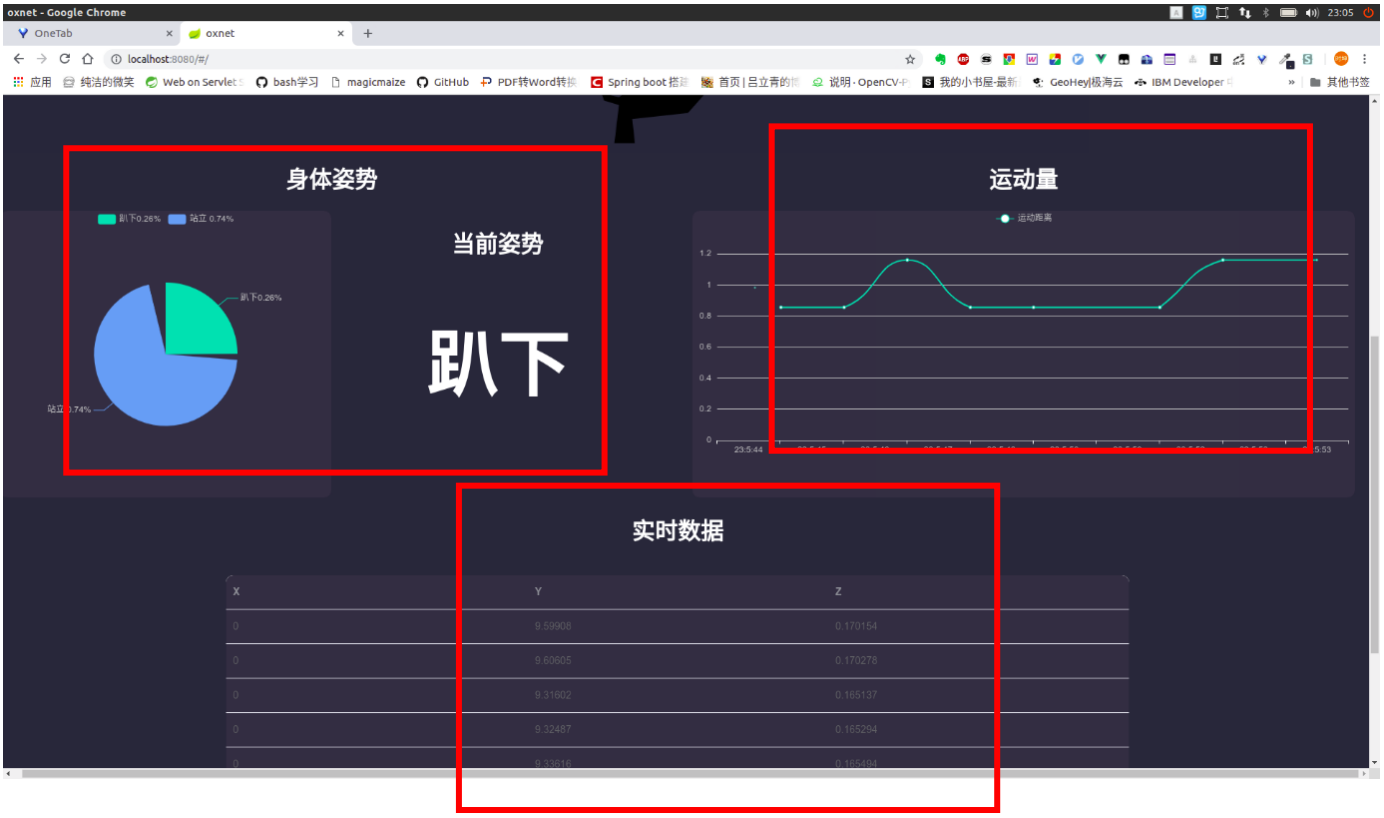
```
breathData: breathDataTemp,
headData: headDataTemp,
activityData: activityDataTemp,
bodyData: bodyDataTemp,
tableData: tableDataTemp,
```

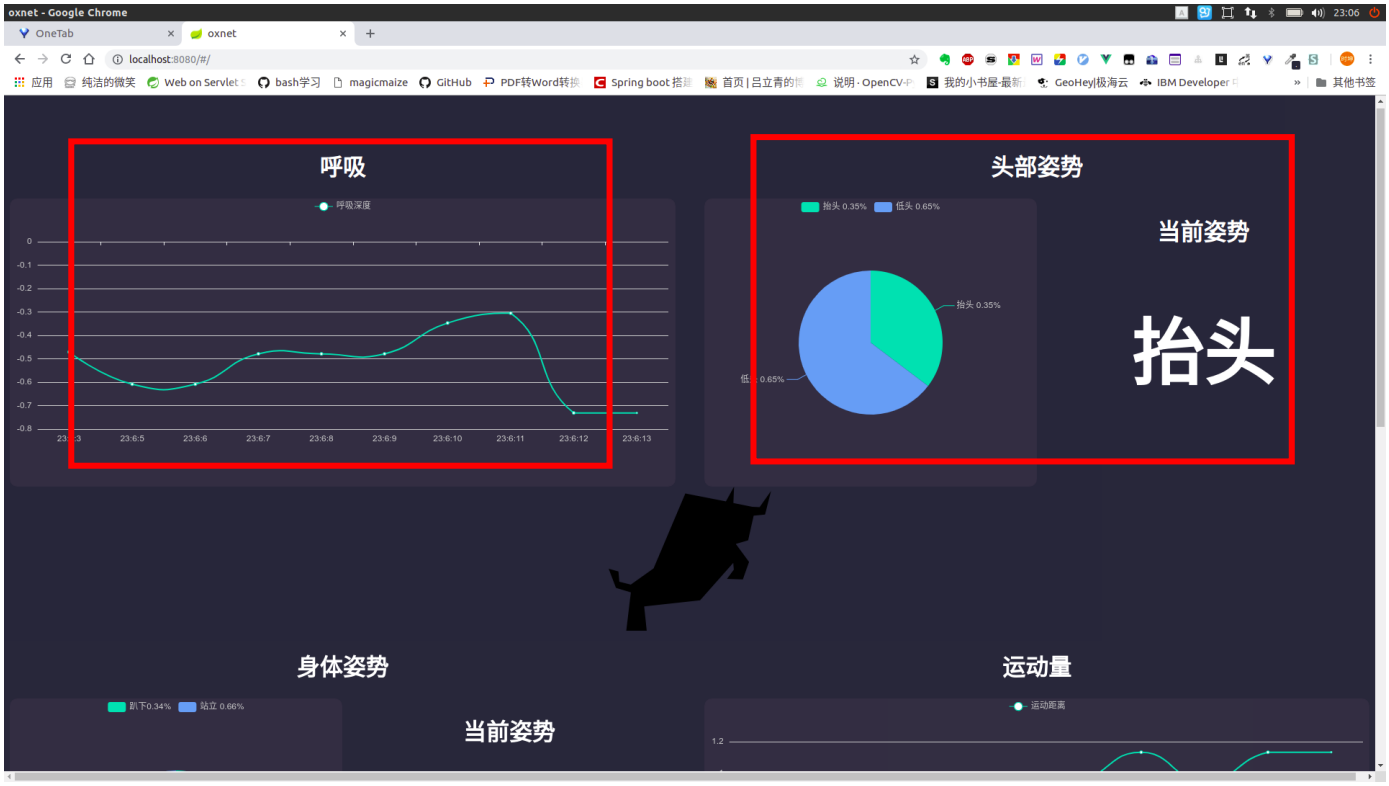
3.4 每隔一秒请求一次数据并更新绑定的数据

```
setInterval(function () {
 // 发送请求，获取数据
  window.fetch('http://172.16.124.157:8080/analysis_pers').then(res
```

```
=> res.json()).then(function (myJson) {
  // 收到响应后马上更新页面
   console.log(myJson)
  // 判断头部姿势
   if (myJson.head_down === '1') {
    headDataTemp.pose = '低头'
  } else if (myJson.head_down === '0') {
    headDataTemp.pose = '抬头'
  } else if (myJson.head_down === '-1') {
    headDataTemp.pose = '水平'
  } else {
    headDataTemp.pose = '未知'
  }
```

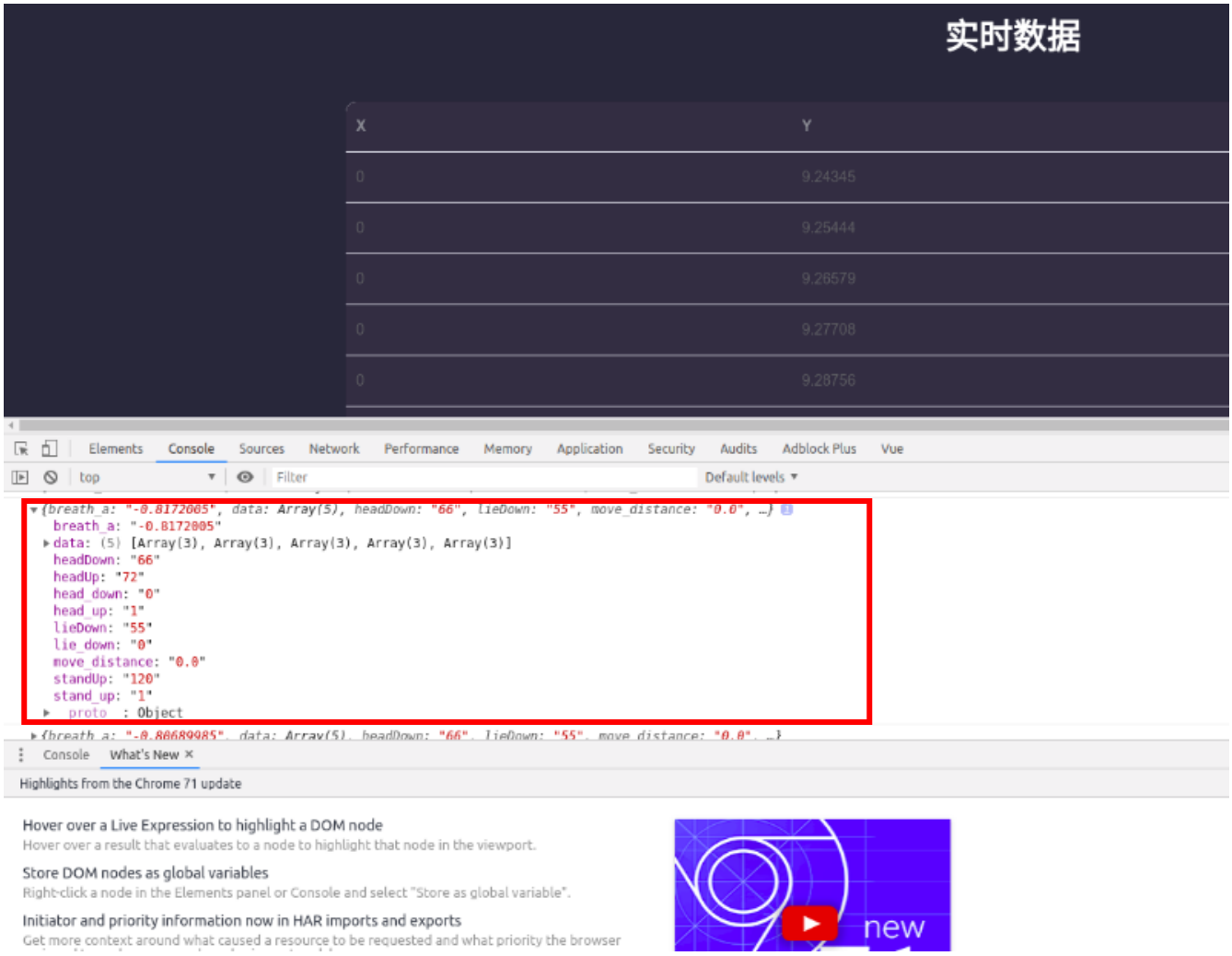
三）成果展示和分析


oxnet - Google Chrome
OneTab
oxnet
localhost:8080/#/
身体姿势
趴下0.26%
站立 0.74%
趴下0.26%
站立 0.74%
当前姿势
趴下
运动量
运动距离
实时数据
X
Y
Z
0
9.59908
0.170154
0
9.60605
0.170278
0
9.31602
0.165137
0
9.32487
0.165294
0
9.33616
0.165494


oxnet - Google Chrome
OneTab
oxnet
localhost:8080/#/
呼吸
呼吸深度
头部姿势
抬头 0.35%
低头 0.65%
抬头 0.35%
低头 0.65%
当前姿势
抬头
身体姿势
趴下0.34%
站立 0.66%
当前姿势
运动量
运动距离

分析：

前台请求，后台相应请求，通过 json 返回，前台解析后显示到 html， 通过浏览器控制台查看请求返回回来的数据



四）优缺点和进一步工作

在计算呼吸频率和呼吸深度的时候，因为呼吸是每时每刻都存在，并且加速度和周期都不是很大，很难计算，而且没有函数能根据输入数据模拟波形，很难计算出频率； 第二个是在计算运动量的时候，由于站起或趴下需要 3s，而我们设置请求是 1s， 所以只能计算 3s 运动的平均加速度，这样会导致通过平均加速度 计算出来的位移和实际位移偏差较大。

四、参考资料

1、Springboot 简单教程： http://www.ityouknow.com/spring-boot.html

2、springboot 官网：https://spring.io/projects/spring-boot

3、springboot 博客：https://blog.csdn.net/forezp/column/info/15397/2

4、vue.js：https://cn.vuejs.org/v2/guide/index.html#