# 2016.6.27

数据库操作：

1、导入库

2、在dlg.h创建指针对象

3、在dlg.cpp的initialdlg中创建连接，初始化表

4、进行数据操作。

# 2016.6.28

今日规划：

1、绘图

1）十字线；完成

2）坐标转换；未完成

3）画机器人；完成

4）画路径；未完成

# 2016.6.29

1、出现abort错误，调试，调用堆栈处找到自己熟悉最近的函数，错误就在附近，然后调试运行，

可能是数据库本身有问题。

2、完成了机器人绘制，路径的绘制（即坐标转换）Demo2.0

# 2016.6.30

1、打印了STR的论文，并阅读了

Tuning-free controller to accurately regulate flow rates in a microfluidic network的前半部分。

# 2016.7.1

1、在绘图区标出了target点。

2、忽略障碍情况下，实现跟踪目标点。

# 2016.7.19

1、阅读UAV文档（ok）

2、研究跟踪点算法

# 2016.9.6

今日目标：

将经纬度坐标转为平面xy坐标系。并在上位机显示出来。

# 2016.9.26

数据结构

时间复杂度，推导大O的方法

1. 常数阶：执行次数不随n的增大而增大，故单纯的分支结构（不包含在循环结构中），其时间复杂度为O(1)。
2. 线性阶：

# 2016.10.19

1. 线性表：零个或多个数据元素的有限序列；

2. 栈：限定仅在表尾进行插入和删除操作的线性表（后进先出）；

3. 队列：只允许一端插入，一端删除操作的线性表（先进先出，后进后出，排队）。

# 2016.11.20

1. 下一步工作规划：

2. 另一条小船功能测试和算法测试实验验证；

3. 确定一种合适的使用小IMU的方法（6/9轴？）；

4. 大船GPS的使用（使用方法，电源方案等）；

5. 大船无线电的配置和使用（主要为电源解决方案）；

6. 上位机软件（接收数据卡顿问题/多船协同软件融合）；

7. 上位机嵌入地图显示；

8. 无刷电机/有刷电机的选择等。

# 2016.11.21

大船GPS调试和使用。具体如下：

1. 熟悉GPS设备的性能（各种参数等）

2. 掌握GPS使用方法（流程、各种通信命令）

3. 选择与小GPS相同的通信格式

**4. 统一下位机通信协议**

5. 上位机软件对GPS数据解析

通信协议

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 内容 | 帧头 | | Len | ID | GPS | | | | IMU | 舵角 | 速度等级 | CRC | 帧尾 |
| A5 | 5A | 纬度 | 经度 | 时间 | 速度 |
| 长度  （byte） | 1 | 1 | 1 | 1 | 4 | 4 | 3 | 2 | 4 | 1 | 1 | 2 | 1 |

共26字节。参考gps出厂源码确定gps帧数据传输原格式或是处理后的数据。

# 2016.11.25

抢占优先级和响应优先级

例如，现在有三个中断向量：

中断向量 抢占优先级 响应优先级

A 0 0

B 1 0

C 1 1

若内核正在执行 C 的中断服务函数，则它能被抢占优先级更高的中断 A 打

断，由于 B 和 C 的抢占优先级相同，所以 C 不能被 B 打断。但如果 B 和 C 中断

是同时到达的，内核就会首先响应响应优先级别更高的 B 中断。

# 2016.12.3

完成了小船的六个实验：

1. 圆内逆时针跟随；

2. 圆内顺时针跟随；

3. 圆外逆时针跟随；

4. 圆外顺时针跟随；

5. 直线初始同向跟随；

6. 直线初始反响跟随。

# 2016.12.5

主任务：**大船上位机的编写及其实验**。

任务细化：

1. 数据接收的方式及其数据的处理；

2. 拟发送命令的组包处理及发送（考虑获取当前状态信息与控制的冲突问题）；

3. 实验；

4. 三船融合软件。

# 2016.12.6

1. 三船数据接收部分已完成；【完成】

2. 开环部分要加入命令ID选择；【完成】

3. 闭环部分改为三船轮巡控制（状态的获取和控制信息的发送）；【完成】

4. 开环时采用定时器获取状态数据，闭环时采用延时获取状态数据；【完成】

5. 坐标系的统一，调整和优化；【完成】

5. 界面的优化。【完成】

# 2016.12.8

1. 保存三船运行状态数据（经纬度，速度，航向角等）；【完成】

2. 解决三船数据接收混合问题（定时器内设标志？）；【定时器方式失败】

3. 采用结构体保存三船状态数据，及进行控制。【完成】

# 2016.12.21

1. **通信问题**

通过给三条船设置不同的请求命令包头，避免请求/相应包头相同造成通信混乱，具体如下：

若上位机请求命令包头为A5 5A，下位机响应包头也为A5 5A，则会造成：

Case 1：若1号机返回响应数据期间，上位机发出了请求2号机的状态数据的命令，则2号机也会返回响应数据，如果轮巡请求时间间隔过短，则由于无线通信会出现将一组数据分两段发送的现象而导致两组或三组数据的相互混合，从而使得上位机无法接收到每条船的完整正确的响应数据。

Case 2： 1号及接收到请求命令后返回A5 5A开头的相应报文，2号和3号也会同时收到包头，进入命令接收中断（由于协议相同，会根据数据长度接收数据，接完之后才判断是否属于本机），而此时上位机发给2号和3号机的命令无法被其接收，然后造成无法获取2号机和3号机的响应数据。

1. **解决方案及测试**

为了避免上面出现的通信问题，提出了两种解决方案，分别进行了测试，具体如下：

**方案2**：

A. **解决方案：**

上位机每次请求下位机状态参数之前设置独立的靶船请求标识（如1号），用以区分1号船，2号船和3号船。上位机接收到响应数据之后先进行靶船标识判断，将数据存入属于不同靶船的数据存储区。一组数据接收完毕之后，将靶船请求标识更换位下一条轮询靶船标识（如2号）。然后再请求2号靶船状态参数，如此循环直至停止。

**B. 测试结果**：失败。

**C. 原因分析**：

采用请求之前设置靶船响应标识的方法虽然能判断出接收到的数据属于哪个靶船的信息，但依然没有解决1号船返回数据时，2号和3号也会同时接收到1号船的响应数据，阻塞了2号、3号靶船与上位机的通信通道，使得上位机请求命令无法下达，无法解决上述通信问题。

**方案2：**

A. **解决方案：**

上位机请求报文和下位机响应报文设置不同的通信格式协议。具体如下：

上位机请求报文对于三条不同靶船设置三种不同的报头。如1号船报文包头设置为A1 1A，2号船设置为A2 2A，三号船设置为A3 3A。下位机响应报文包头依然为A5 5A不变。

**B. 测试结果**：成功。

**C. 原因分析**：

上位机请求1号靶船信息时发送以A1 1A开头的请求报文，则只有1号靶船会进入通信中断进行命令接收，由于A1 1A不是2号靶船和3号靶船请求报文，则2号和3号不会进入接收命令中断。同理，当1号靶船返回响应数据时，返回报文为A5 5A开头，2号和3号靶船同样判断出不属于本机命令从而不进入接收状态。当上位机接收完1号靶船的信息后，再开始请求2号靶船信息，发送A2 2A开头的请求报文，同理，只有2号靶船能进入命令接收处理过程，其他两条靶船不会进入，也不会阻塞其它两条船通信通道。3号船同理。如此循环下去直至停止。

# 2016.12.27

采用ZigBee传输数据开环控制时产生卡顿现象，但采用有线和无线数传模块则不会卡顿。不是数据卡顿，而是点击控制键时，有时下位机状态没有改变。有可能下位机没接到，或者上位机没发送出去。

# 2017.1.5

1. 拟采用网络化系统对无人艇进行监控。即采用客户机-服务器模式。客户机分为现场客户机和远程客户机。对于无人艇的状态信息的显示拟加入3D显示。

2. 对当前上位机加入网络通信和服务器进行通信。

获取服务器信息采用Get方法；上传采用Post方法，将数据直接填入表单即可。

# 2017.1.6

委托

1.为什么要使用委托

将一个方法作为参数传递给另一个方法

2.委托概念

声明一个委托类型

委托所指向的函数必须跟委托具有相同的签名

3.匿名函数

4.练习：使用委托求数组最大值

5.联系：使用委托求任意数组最大值

6.泛型委托

7.多播委托

8.lamda表达式

9.使用委托实现窗体传值

# 2017.1.10

完成GoogleMapDemo(离线谷歌/必应地图)，读取Excel经纬度直接将路径标在地图上。

1. 各大地图均由各自的经纬度系统，并且和GPS硬件设备经纬度系统不同。

2. 谷歌地图和谷歌地球采用不同的经纬度坐标系统。

3. 硬件GPS设备接收的数据和谷歌地球的坐标系统是统一的。

4. 在地图上使用硬件GPS坐标数据的方法：

1）先将硬件GPS经纬度数据转换为对应拟使用的坐标系统数据（如将数据转为谷歌地图上的经纬度）；

2）使用对应转换的地图进行数据显示（在谷歌地图上进行显示）。

# 2017.1.11

1月8日实验数据处理。

## 一、单船实验

### A. 直线跟踪实验：

1. 2号船航向角和航迹角跟踪效果对比

1）轨迹对比2）跟随误差对比3）航迹角和航向角对比





2. 3号船航向角和航迹角跟踪效果对比

1）轨迹对比

2）跟随误差对比

3）航迹角和航向角对比

### B. 圆轨迹跟踪实验

3. 2号船航向角和航迹角跟踪效果对比

1）轨迹对比2）跟随误差对比3）航迹角和航向角对比





4. 3号船航向角和航迹角跟踪效果对比

1）轨迹对比

2）跟随误差对比

3）航迹角和航向角对比

## 二、多船实验（均采用航迹角控制）

### A. 直线跟踪实验

1. 2号船和3号船跟踪效果对比（包含速度控制）

1）轨迹对比2）跟随误差对比3）航迹角和航向角对比4）速度控制





### B. 圆轨迹跟踪实验

2. 2号船和3号船跟踪效果对比（包含速度控制-跟踪同一圆）

1）轨迹对比、误差对比、航迹角对比





3. 2号船和3号船跟踪效果对比（包含速度控制-跟踪同心圆）

半径R1=4米，R2=7米

分别为轨迹对比、误差对比、航迹角对比、速度对比





4. 2号船和3号船跟踪效果对比（不包含速度控制）

1）轨迹对比

2）跟随误差对比

3）航迹角和航向角对比

# 2017.2.9

大船右螺旋桨卡顿问题排查：

现象1：螺旋桨反转时，逐级缓慢增加时，左动右不动，快速增加时右动，闭环时右动。正转时，满增和徒增都动。

解决：通过首先左右调换螺旋桨，发现依然左动右不动，然后调换驱动器，依然左动右不动，得出结论问题在驱动板。测试**电位器电压**发现两者启动电压存在较大差异，调整后问题解决。

现象2：调整电位器后发现反转正常，但左螺旋桨无法正转，发转正常。通过测量驱动板发现驱动换向端电压不改变（棕色线），最后排查发现为**插头松动接触不良**。之前左螺旋桨抖动可能就与插头松动有关。

需要和刘工沟通更换新的插头。

# 2017.2.10

下一步工作计划：

1. 代码重构（基本完成）

2. 通信冲突稳定性问题（待测试）

3. 控制代码移植到下位机问题（进行中）

4. 网络接口（已完成）

5. 谷歌地图清晰化问题（进行中）

# 2017.2.14

**A. 接口功能设计**

1. 控制参数接口（可暂时只设置修改接口，修改成功后返回修改成功标志即可）

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | | 长度(8 bytes) | 命令标识 | Kp | Ki | Kd | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x04 | 1byte | 1byte | 1byte | 0xAA |

2. 跟随目标设置接口（点、直线、圆轨迹）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 包头 | | 长度(12 bytes) | 命令  标识 | 目标选择 | Para1 | Para2 | Para3 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x01 | 1byte | 2byte | 2byte | 2byte | 0xAA |
| 点：0x01 | X | Y | Null |
| 线：0x02 | Null | Y | Null |
| 圆：0x03 | X | Y | R |

3. 原点设置接口（经度，纬度）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 包头 | | 长度(13 bytes) | 命令  标识 | 纬度 | 经度 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x02 | 4byte | 4byte | 0xAA |

4. 跟随方式选择接口（航向角/航迹角）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 包头 | | 长度(6 bytes) | 命令  标识 | 选择值 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x03 | 0x01：航向角  0x02：航迹角 | 0xAA |

5. 航向初始化接口（使得当前惯导值的相反数为补偿值）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | | 长度(5 bytes) | 命令  标识 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x05 | 0xAA |

6. 控制模式选择接口（开环和闭环，为了应急措施应设计开环优先级高于闭环，即开环可打断或终止闭环）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 包头 | | 长度(6 bytes) | 命令  标识 | 选择值 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x06 | 0x01：开环  0x02：闭环 | 0xAA |

7. 恢复初始设置（即复位功能，所有参数设置为初始状态）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 包头 | | 长度(5 bytes) | 命令  标识 | 包尾 |
| 0xA1 | 0x1A | 1byte | 0x00 | 0xAA |

**B. 接口内容设计**

接口内容应包括：

1. 船号标识（和当前保持一致，a1,1a等，使得下位机知道当前控制的为哪条船）

2. 命令长度（每个接口实现不同的功能，故命令长度也会不同设置变长度的命令更灵活，用命令长度进行控制接收。和当前接收保持一致）

3. 命令标识（标识接口功能，即告诉下位机执行何种操作）

4. 命令参数值（即对应命令的值）

5. 结束标识（标识一条完整命令的结束）。

# 2017.2.17

下一步工作：

1. 数据记录中加入控制输入量 r。

2. 三船速度控制问题解决。

4. 大船的大GPS敏感度太高问题。

5. 无线电台的使用（电源，板线等东西）。

# 2017.11.27（近期工作方案）

## 一、云服务器

1. Tornado框架的学习（**4-5天**）——官方文档和视频、示例等
2. 云服务器框架和功能模块的了解和学习（**2天**）——本服务器源代码
3. MongoDB数据库和Tornado联合操作的学习（**4天**）——相关文档视频和本服务器源代码
4. 视频能否一起也存储在服务器MongoDB数据库中（**3天**）？

**当时没有和当前实例绑定和没有存储在数据库中的难度和原因在哪里？是否能解决？或者采用其他方式？（前提是已经完全理解本服务器的框架和代码，再与何一闻沟通）**

1）视频数据（byte）无法与当前实例（instance绑定）——无法传入instanceID

2）byte数据无法存入MongoDB数据库

1. 解决PH值bug问题（**1天左右**）

## 二、unity客户端

1. 如果云服务器中的问题4能解决，客户端是否可以也将视频信息从云端获取在界面显示？

——**时间待定**

1. 与云服务器中的问题5结合问题，PH值显示bug原因探究：
2. 服务器问题？客户端问题？
3. 如何解决？

附加问题：

1. 控制和显示能否改为多选？

答：软件整体架构采用**单例模式**实现，其控制、**参数显示**、以及场景视角转换均与当前选定船绑定，且toggle组件是在toggle group中从属关系，保证单例选定状态和模式，避免参数显示、场景转换和控制冲突。

画线是从独立模块，可实现同时绘制三条船的轨迹。（2017.12.5完成）

1. 能否导入其他模型驱动？——模型是什么模型？

## 三、本地监控站

挑战杯国赛版本当前问题汇总：

1. 与下位机通信频繁中断
2. 运行不稳定，偶尔出现卡死现象
3. 调试模式下无法接收数据，导致无法定位问题所在
4. 多段直线编队跟随最后一个点程序卡死问题
5. 虚拟仿真模式下，航向角不更新问题。

解决方案：

本地监控站退回至省赛版本，重新对以上所有功能重新编写（**3-4天左右**），包括：

1. 新通信协议模块修改
2. 新传感器模块增加
3. 多段直线编队重新编写或者其他方案
4. 卡死和稳定性问题

## 四、下位机

当前存在问题：

1. 与上位机通信频繁中断，原因？是上位机还是下位机问题？
2. 通信模式是否还可优化？

解决方案（**2天左右**）：

与甘帅奇、雒宗同一起讨论解决方案

# 2017.12.5-云服务器环境搭建

## 1.准备一台服务器

我用的是腾讯云的学生机，装的Ubuntu Server 14.04.1 LTS 64位的系统。

## 2.配置python运行环境。

cd /usr/bin

ls

发现系统自带了python2,2.7,3,3.4各种版本。

不过输入python后发现版本是2.7的。

/usr/bin目录下，使用命令sudo mv python3.4 python，直接覆盖，再运行pyhton -V,发现版本成功变为3.4.0了，搞定。

PS：可使用alias py=python方便下次直接输入py即可运行python。

## 3.安装Tornado

输入命令：pip install tornado

发现运行报错，原因是使用了新版本的python导致pip不兼容，干脆直接去官网安装一个最新的pip。

链接地址：https://bootstrap.pypa.io/get-pip.py

get-pip.py文件下载至服务器，运行python get-pip.py，等待安装成功。

再次输入命令：pip install tornado，等待安装成功。

tornado的安装目录：/usr/local/lib/python3.4/dist-packages/tornado

## 4.必不可少的hello，world

随便找个文件夹/home/ubuntu，新建一个hello.py的文件。

命令：vim hello.py

#! /usr/bin/python

# -\*- coding:utf-8 -\*-

import tornado.ioloop

import tornado.web

class MainHandler(tornado.web.RequestHandler):

def get(self):

self.write("Hello, world")

def make\_app():

return tornado.web.Application([

(r"/", MainHandler),

])

if \_\_name\_\_ == "\_\_main\_\_":

app = make\_app()

app.listen(8888)

tornado.ioloop.IOLoop.current().start()

执行命令:python hello.py

然后在浏览器中输入xxx.xxx.xxx.xxx:8888即可get hello world！

## 5.后台挂起程序

虽然我们成功跑出了hello world，但是终端一关程序就跑不了了，这怎么能忍！

执行命令：nohup python hello.py &

这样我们就算关闭终端后，程序也仍在后台不断地运行了！

PS：jobs可观察本终端执行挂载后台的进程，用fg %N 可将后台进程提至前台。

在其他的终端无法通过上述方法结束后台运行的进程，不过可用ps -aux寻找后台挂起的进程，或直接用ps -aux | grep ‘hello.py’ 来直接寻找，最后用kill #pid杀死进程。

## 6.hello.py分析

4~5:引入tornado的部分模块。

7~9：这是一个页面请求的Handler。get方法表示Http的Get请求。write是一个内建的方法，它的目的是将其中的字符串写在页面中。

11~14：建立一个tornado应用。Application()内是一个由一系列二元组组成的列表，其中每个二元组的第一个元素是一个正则表达式，表示网址路由，如果其中包含捕获分组，则将匹配的内容送往第二个元素处理，如例子中的MainHandler。

16~19：给定app，设置app的监听端口，启动服务器。

## 7.Linux安装MongoDB数据库

方法一：

//配置公钥

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb\_release -sc)"/mongodb-org/3.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list

sudo apt-get update

sudo apt-get install -y mongodb-org

//完成上面的安装步骤配置mongodb的数据库的位置

sudo mongod --dbpath /data/db

//启动mongod

sudo service mongod start

方法二：

离线安装，先从官网下好安装包。具体参照http://blog.csdn.net/flyfish111222/article/details/51891130

PS：安装完成后别忘了把mongodb的安装目录加到环境变量中，访问更便捷

2.远端连接数据库

下载软件RoboMongo，支持windows,mac。

3.安装pymongo模块

//配置公钥

sudo pip install pymongo

4.使用pymongo操作MongoDB

#连接MongoDB服务器

client = MongoClient('localhost', 27017)

#进入一个数据库，若不存在则新建

db = client.example

db = client['example']

#获得数据库中的集合列表

db.collection\_names()

#进入或新建集合（文档）

widgets = db.widgets

widgets = db['widgets']

#向集合中插入一项数据，将返回一个ObjectId('4eada0b5136fc4aa41000000')

widgets.insert({"name":"coolyiwen", "sex":"man", "old":12})

#查找集合中的一项数据,返回的是一个字典数据

doc = widgets.find\_one({"name":"coolyiwen"})

#注意，数据中包含一个自动生成的识别id

>>>doc

{"name":"coolyiwen", "sex":"man", "old":12, "\_id":ObjectId('4eada0b5136fc4aa41000000')}

#修改并保存数据

doc['old'] = 20

widgets.save(doc)

#添加更多的文档

widgets.insert({"name":"coolyiwen", "content":"hello"})

#find\_one()只能查询到符合要求的最前面的一个数据，查找多项数据用find(),遍历输出

for doc in widgets.find({"name":"coolyiwen"}):

print(doc)

#删除指定的符合要求的所有文档

widgets.remove({"content": "hello"})

#因为网络传输常用到JSON格式，所以了解下用自带的JSON库

import json

#JSON转换时不识别\_id内的ObjectId项，需在转换前删除

del doc['\_id']

json.dumps(doc)

#更复杂的方法是使用PyMongo的json\_util库,以完成序列化其他MongoDB特定数据类型到JSON

5.以下是一个pymongo添加到Web服务里的例子

#从数据库中读取数据和保存数据

import tornado.httpserver

import tornado.ioloop

import tornado.options

import tornado.web

import pymongo

import json

from tornado.options import define, options

define("port", default=8000, help="run on the given port", type=int)

class Application(tornado.web.Application):

def \_\_init\_\_(self):

handlers = [(r"/(\w+)", WordHandler)]

#链接数据库服务器

client = pymongo.MongoClient("localhost", 27017)

#再app中添加对数据库的索引

self.db = client["example"]

#self.db2 = client.example2

tornado.web.Application.\_\_init\_\_(self, handlers, debug=True)

class WordHandler(tornado.web.RequestHandler):

def get(self, word):

#打开数据库中的words集合

coll = self.application.db.words

#再集合中查找接受到的word关键词

word\_doc = coll.find\_one({"word": word})

#如果存在，返回字典的JSON数据

if word\_doc:

del word\_doc["\_id"]

self.write(json.dumps(word\_doc))

else:

self.set\_status(404)

self.write({"error": "word not found"})

def post(self, word):

des = self.get\_argument("description")

coll = self.application.db.words

word\_doc = coll.find\_one({"word": word})

#如果存在，修改该word的描述

if word\_doc:

word\_doc['description'] = des

coll.save(word\_doc)

#如果不存在，新添加该word及其描述

else:

word\_doc = {"word":word, "description":des}

coll.insert(word\_doc)

del word\_doc['\_id']

self.write(json.dumps(word\_doc))

if \_\_name\_\_ == "\_\_main\_\_":

tornado.options.parse\_command\_line()

http\_server = tornado.httpserver.HTTPServer(Application())

http\_server.listen(options.port)

tornado.ioloop.IOLoop.instance().start()

## 8. Windows安装MongoDB

<http://blog.csdn.net/mzbonnt/article/details/51461331>

1. 创建数据库路径（data目录）、日志路径（logs目录）和日志文件（mongo.log文件）
2. 创建配置文件mongo.conf

dbpath=D:\Program Files\MongoDB\Server\3.2\data #数据库路径

logpath=D:\Program Files\MongoDB\Server\3.2\logs\mongo.log #日志输出文件路径

logappend=true #错误日志采用追加模式

journal=true #启用日志文件，默认启用

quiet=true #这个选项可以过滤掉一些无用的日志信息，若需要调试使用请设置为false

port=27017 #端口号 默认为27017

1. 启动MongoDB

mongod --config "D:\Program Files\MongoDB\Server\3.2\mongo.conf"

mongod --config "d:\mongodb\data\log\mongo.conf"

1. 创建MongoDB服务

(1) mongod --config "D:\Program Files\MongoDB\Server\3.2\mongo.conf" --install --serviceName "MongoDB"

(2) net start MongoDB

# 2017.12.7—工作记录（从11.23开始）

1. 服务器完成了将上传视频数据和当前实例绑定（首字节位实例号）

2. 服务器完成了服务器将视频数据存入MongoDB数据库（和实例ID统一命名）

3. 服务器完成了通过实例ID获取视频数据的接口

4. Unity 3D客户端完成了同时绘制三条船轨迹：

1）脚本中定义三个shiprender

2）Draw控件中创建三个shipRender控件（emptyobject-reder-材质）

3）Manager中将脚本的三个shiprender和Draw中的三个render控件绑定