编队实验

我主要负责编队机协作时的组内信息交换问题的讨论、选择和验证。

编队过程的信号流是主机PC进行信息收集（队内不同个体之间需要把自身位姿、速度等信息上传主机PC），然后规划路径点，再分发姿态调整指令给各个从机。所以需要选择一种合适的通信方式来完成多机通信。且尽量保持实时性。

通信方式

两台设备之间的通信方式有许多种，最常见的就是互联网，互联网将接入互联网的设别互联起来，可以相互交换信息。或者以太网构建的局域网，也是可以在局域内交换信息的。但是在工控方面，因为实时性要求较高，以太网和互联网有非确定性延迟，所以不是很常用于实时信息交换。

不局限于有线无线通信方式地我们讨论目前在竞赛或者工业中常用的通信方式（无线只是接收端和发送端之间是射频传输，本质上收发端和控制器之间还是需要一种有线通信方式的）。

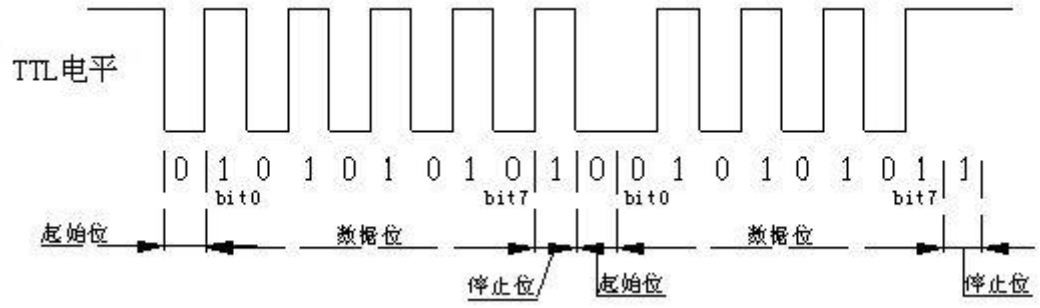
在进入不同通信方式介绍之前，我们介绍一下同步异步、串行并行意思。

- 同步通信：通信过程有时钟信号同步进行同步，控制和推进通信过程（clockout）
- 异步通信：无时钟同步线，通信过程按规定的比特率（或间接bitrate）进行控制。由于没有同步信号，时序不同步有出错可能，所以通常配备校验位。
- 串行：数据由数据线在时间线上依次传出/入。
- 并行：多跟数据线同时传出数据。

UART/USART

USART为通用同步/异步串行接收/发送器，而UART为纯异步方式。TTL电平，可以实现全双工至单工的（双线或单线）的通信过程。数据线TX，RX,时钟线CLK(UASRT)，地线。

一下图为异步串口通信，通常数据段结尾有奇偶校验位。



RS485/232

通常485/232是总线指电平协议。为什么需要不同的电平协议？因为信号在长距离传输中，会遇到更多的干扰，或者传输距离太远导致电(压)信号的衰减。这也是为什么电信号出了电压信号，还有电流信号。通常传感器模拟输出标准接口为5-20mA电流。电流信号相对于电压信号有有坑强的抗干扰能力，因为空间中的电磁噪声功率小，由于W=UI，可见能引起的电流变化极小。

回到电压信号，在远距离传输中TTL已经不能满足传输的信噪要去了。使用其他电平协议来抑制噪声或提高信噪比。方法有2种：一种是提高有用信号（数据电平）的幅度，这样相对于同样强度的噪声信号，更能分辨出有用信号。另一种方法是使用差分电平，差分走线，双绞屏蔽线传输的方式，差分电平由一对差分线的电平构成，差分信号线通常紧密耦合，这样共模噪声引起的噪声将同时作用在两个信号线上，引起同步变化。差分后的信号近似不受噪声干扰。

RS232是第一种方法的实现，规定逻辑“1”的电平为-5V~-15 V，逻辑“0”的电平为+5 V ~ +15 V，普通情况下可以满足30m传输距离。RS485是第二种方法的实现，差分正电平在+2V ~ +6V之间，表示一个逻辑状态；差分负电平在-2V ~ -6V之间，实际上可以满足1Km的传输距离。

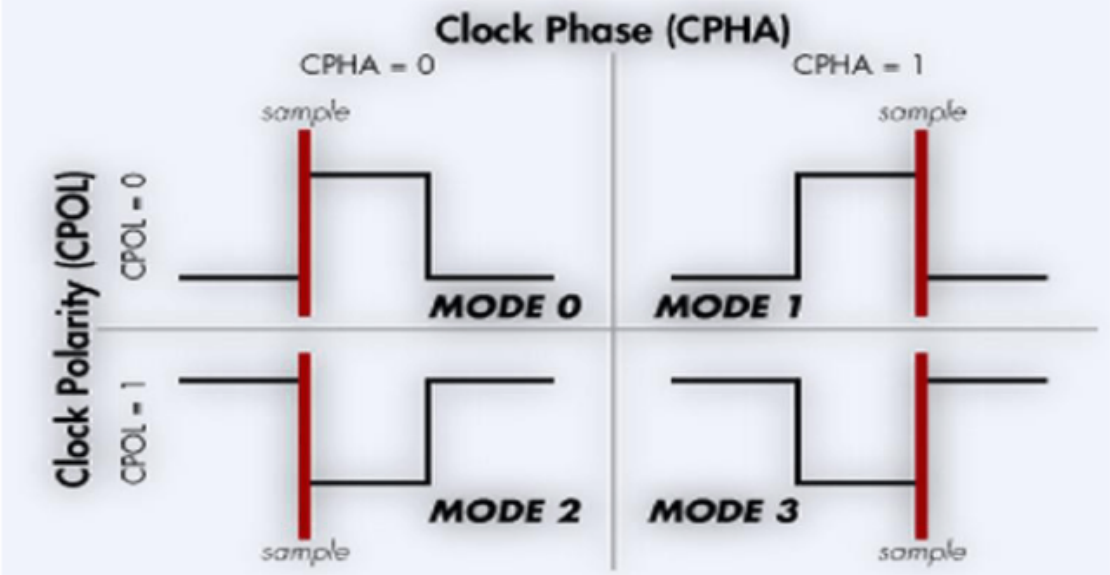
除了RX,TX，有多个流控线，但也可以使用3线通信。总线可以实现多机在同一数据线上并入，减少布线压力。通常可以使用MAX485或MAX232芯片进行TTL微机和485/232设备的电平转换。

SPI

SPI全称串行外设接口，一主多从，任意时刻通信都是由主机发起。是一种高速（0~100Mbps），双全工通信总线接口。四线或两线制配合片选线（MISO,MOSI,SCLK,GND,NSS）。常用在芯片或芯片与外设之间的快速信息交换。

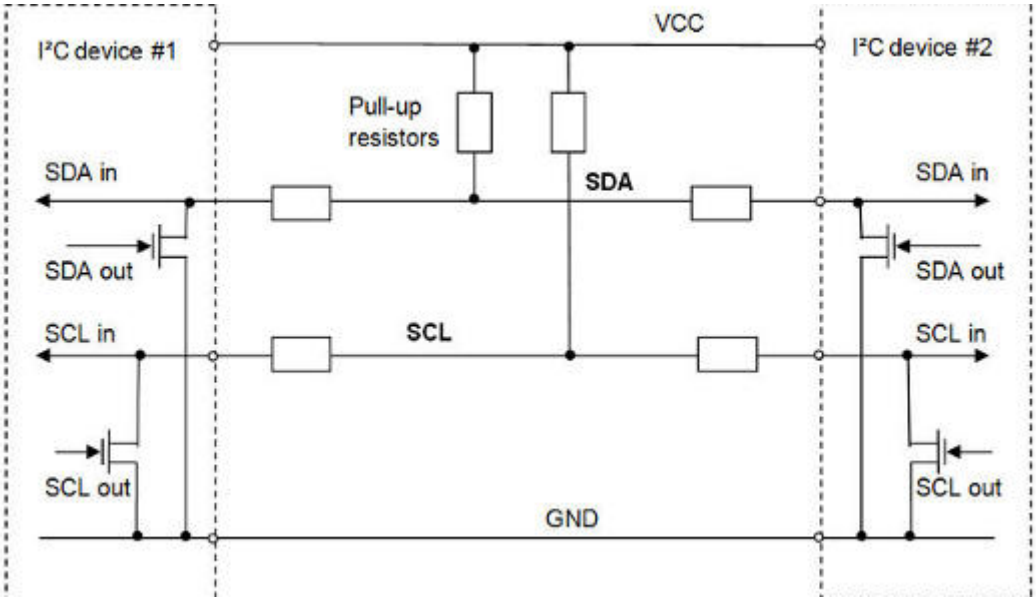
需要确定两边的时钟的采样点和极性。于是衍生出4种工作模式。

SPI需要约定主从时序一致，一般需要规定极性CPOL(clock-polarity)和相位CPHA(clock-phase)



I2C

I2C全称集成电路总线，是一种两线总线，SDA,SCLK，由于芯片间的通信。通信速度在100k-3MHz。

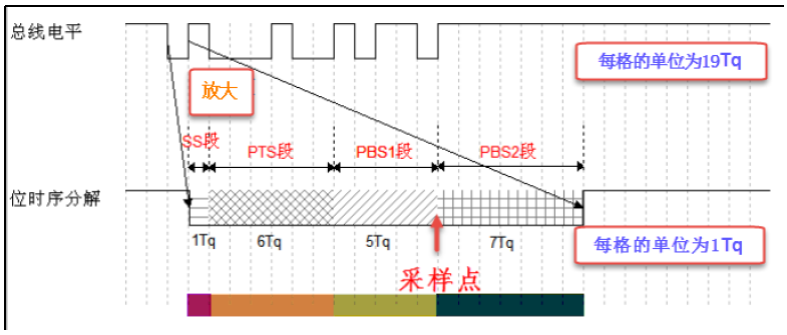


多主多从，但是同一时间只能有一个主机，每个设备有自己的通信地址，所以总线有总裁机制。每个设备通过开漏输出的方式来控制总线输出，两个I2C设备之间连接出了GND和VCC外，需要将SDA和SCL通过上拉电阻上拉到VCC。如上图，所以线路空闲的时候位1，当把信号接地的时候信号位0。基于IIC总线的设计，线路上不可能出现电平冲突现象。如果一支设备发送逻辑0，其它发送逻辑1，那么线路看到的只有逻辑0。也就是说，如果出现电平冲突，发送逻辑0的始终是“赢家”。 总线的物理结构亦允许主设备在往总线写数据的同时读取数据。这样，任何设备都可以检测冲突的发生。当两支主设备竞争总线的时候，“赢家”并不知道竞争的发生，只有“输家”发现了冲突——当它写一个逻辑1，却读到0时——而退出竞争。

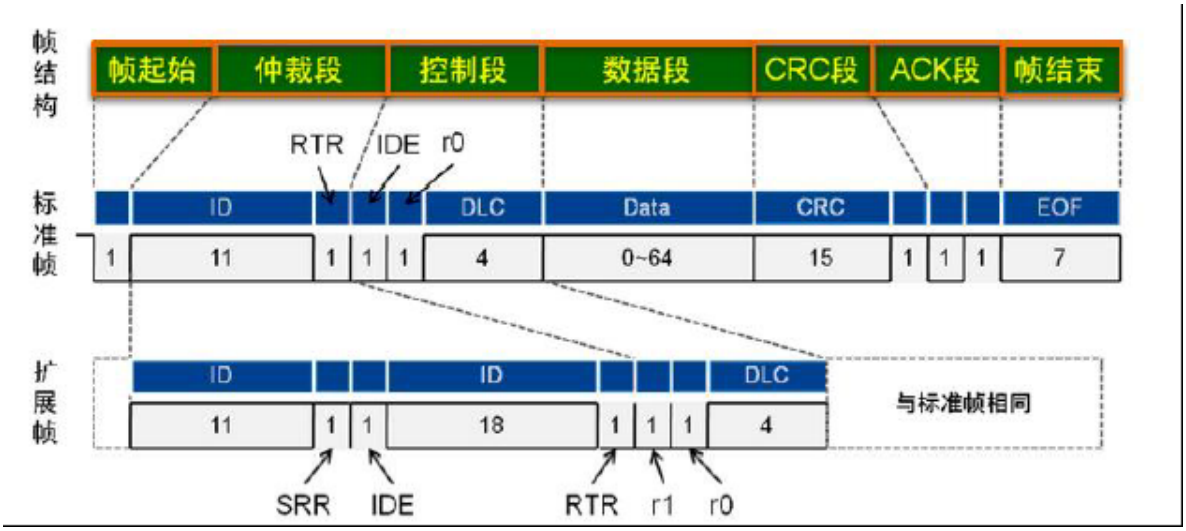
CAN

can总线全程控制局域网，成为汽车计算机控制系统和嵌入式工业控制局域网的标准总线。CAN是一种异步总线通信，所以需要规定波特率（在can种除了波特率，还要规定一个bit构成时间内各段的时长，传播时间段，相位缓冲段1、2，同步段），且有强大的重同步、总裁、错误检测机制。只有CAN_H,CAN_L两条线。以 差分信号的形式进行通讯。提高压差和使用差分方式抑制共模干扰。

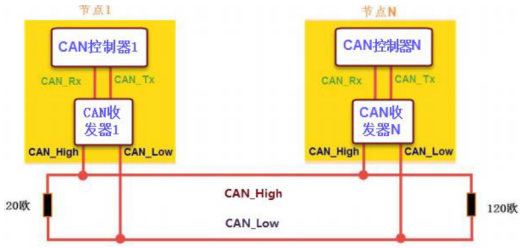
位时序：



数据帧：



总线两端各有一个120欧姆的匹配电阻，传输距离最远40m-1Km（与并入设备个数有关），典型can最高速度1M。can通信物理层有3部分：总线与匹配电阻，CAN收发器(电平转换），CAN控制器（时序控制）。



无线方案

无线方案有许多种，可以使用WIFI，蓝牙，Zigbee，无线串口，或各个频段的射频芯片。

以最常用的NRF24L01的射频芯片是一款2.4GHz附近免费频段的射频芯片，配合PA与LNA可以实现1Km以上空旷场合的传输。此芯片是SPI接口与控制器之间通信，依靠内部基带、控制器、寄存器可以实现控制器通过SPI配置，信息交换。填充和写入内部FIFO和配置寄存器，实现Enhance ShockBurst自动应答协议，依靠内部GFSK(高斯频移键控,监控频移就是0信号时高于对应频段的频率，1信号的时候低于对应频段频率)进行信号调制，将信号叠加到高频2.4GHz频率的射频信号后进行空种传播。

使用这类点对点设备需要自己边写底层驱动、组网协议等，是很麻烦的，除非是工程需要低价格、低功耗。且这类设备，如果周围同类设备过多，可能导致信号空总碰撞，增加误码率。

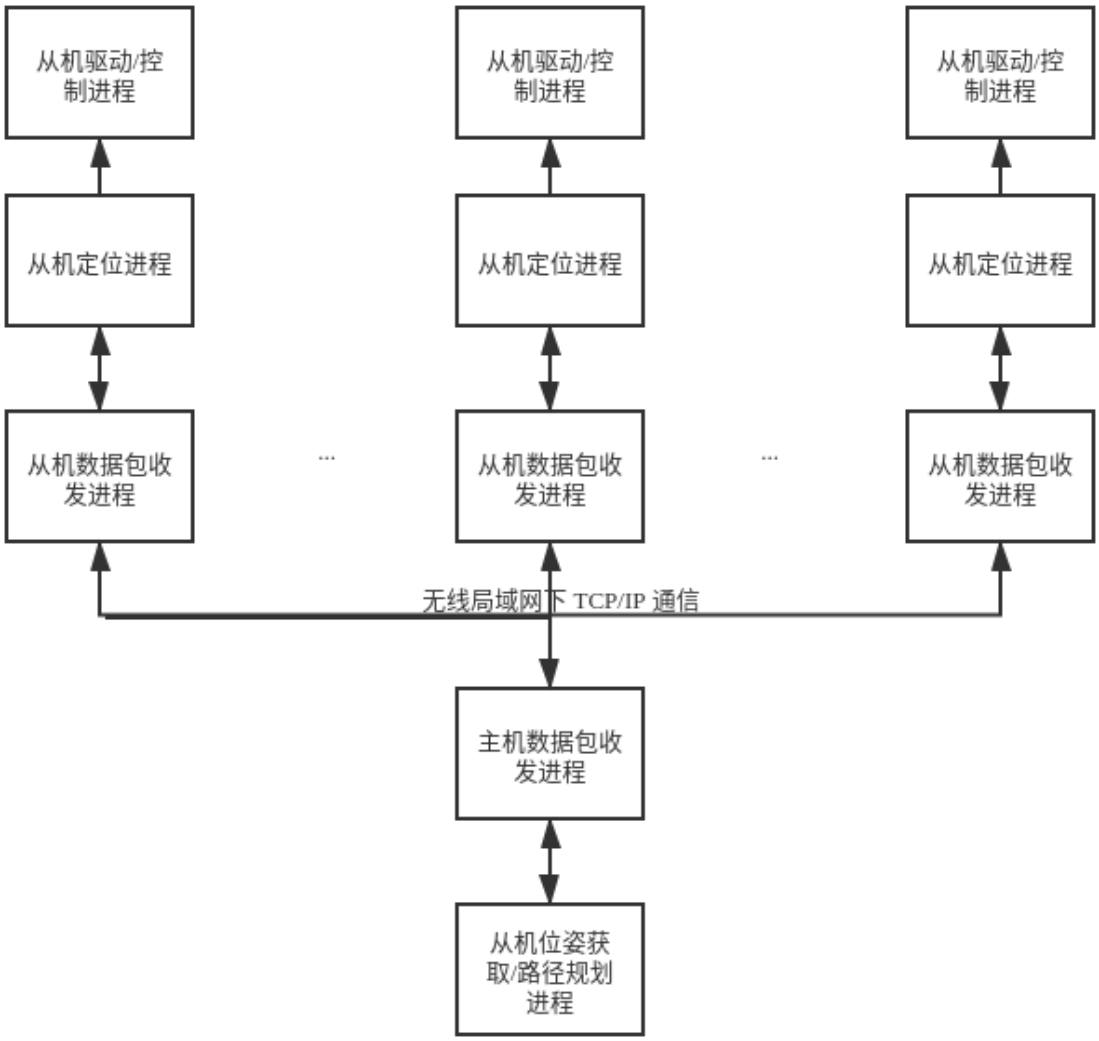
由于传输的信息是定位信息，无线多机定位方案还有一种叫UWB（超带宽）的方案，是利用类似TOF，不过把光变成了射频信号（都是电磁波啦），计算接收和发送时间差，即可得到距离，在通过三点定位可以得到空间位置。由于时钟晶振有频率误差，所以时间误差会带来10cm左右的定位误差。

方案确定

由于是编队实验，多机之间只能使用总线，又因为编队不可能使用有线传输，所以选择了无线传输。以开始是使用无线串口的。但是这样每个从机都需要一个无线串口，是比较麻烦且费用高的。

利用一个路由器生成一个局域网WIFI，或者直接利用学校的公用WIFI。通过分布在队内个体的PC机作为接收载体，使用TCP/IP协议进行多机通信。而对于各操作系统，都有一套编写好了的库可以使用，直接调用socket库即可完成简单的网络编程，实现多机之间无线传输部分。

编队的第一个目标就是：上位机发送一个定位(x,y)指令，从机接收，然后执行指令到对应位置待命。然后从机上传位置，再利用主机PC进行路径规划、调度算法、然后下达指令即可实现编队。



代码

编写2个ROS节点。

- 一个TCP/IP客户端：tcp_client.cpp 发布，监听/master_topic话题，通过无线通信转发。并附带简易终端GUI
- 一个TCP/IP服务器：server_node.cpp，接收无线通信数据，发布/WTR/target_pose话题。
- TCP/IP发送的数据包：geometry_msgs::Pose的二进制内容。

tcp_client

从机ros节点进程通过监听tcp_master节点发布的/master

```
#include <ros/ros.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <netdb.h>
#include "std_msgs/String.h"
#include "geometry_msgs/Pose.h"




#define ADD0 "192.168.0.133"  //dell-4  OK
#define ADD1 "192.168.0.134"  //dell-5  OK
#define ADD2 "192.168.0.165"  //dell-7  OK
#define ADD3 "192.168.0.104"  //dell-10 OK
#define ADD4 "192.168.0.177"  //dell-11 OK
#define ADD5 "192.168.0.129"  //dell-13 OK
#define ADD6 "192.168.0.135"  //dell-14 OK


#define PORT "1234"
#define NUMBER_ADD 7


#define ADD_WIDTH 20
#define PORT_WIDTH 20
#define MESSAGE_FREQ 100


#define NONE "\033[m"
#define RED "\033[0;32;31m"
#define LIGHT_RED "\033[1;31m"
#define GREEN "\033[0;32;32m"
#define LIGHT_GREEN "\033[1;32m"
#define BLUE "\033[0;32;34m"
#define LIGHT_BLUE "\033[1;34m"
#define DARY_GRAY "\033[1;30m"
#define CYAN "\033[0;36m"
#define LIGHT_CYAN "\033[1;36m"
#define PURPLE "\033[0;35m"
#define LIGHT_PURPLE "\033[1;35m"
#define BROWN "\033[0;33m"
#define YELLOW "\033[1;33m"
#define LIGHT_GRAY "\033[0;37m"
#define WHITE "\033[1;37m"


using namespace std;

class Listener {
private:
    char topic_message[NUMBER_ADD][256] ;
public:
    int count=0;
    char *address[NUMBER_ADD];
    char *port;
    bool isconnect[7]={false};

    char* getMessageValue(int num){return topic_message[num];}
    void callback(const geometry_msgs::Pose::ConstPtr& msg)
```

```
    {
//        cout<<*msg<<endl;
        int whitch=msg->position.z;
        if(whitch != 99){bzero(topic_message[whitch],256);
            geometry_msgs::Pose pose=*msg;
            char * ptr=(char *)&pose;
            for (int i = 0; i < sizeof(geometry_msgs::Pose); ++i) {
                topic_message[whitch][i]=ptr[i];
            }
            topic_message[whitch][sizeof(geometry_msgs::Pose)]='\0';}
//            return;

    }
    Listener()
    {
        for (int i = 0; i <NUMBER_ADD ; ++i) {
            address[i]=new char[ADD_WIDTH];
            bzero((char *) &(topic_message[i]), sizeof(topic_message[i]));
        }
        port=new char[PORT_WIDTH];
        port="1234";
    }
    ~Listener()
    {
        cout<<"Say you~"<<endl;
////        for (int i = 0; i <NUMBER_ADD ; ++i) {
////            delete[] address[i];
//        }delete port;
    }
};


int main(int argc, char *argv[]) {
    ros::init(argc, argv, "tcp_client");
    ros::NodeHandle nh;
    ros::Rate loop_rate(MESSAGE_FREQ); // set the rate as defined in the macro
MESSAGE_FREQ

    Listener listener;

    ros::Subscriber client_sub = nh.subscribe("/master_topic", 20,
&Listener::callback, &listener);
    geometry_msgs::Pose pose;

    int sockfd, portno, n;
    struct sockaddr_in serv_addr[NUMBER_ADD];
    struct hostent *server[NUMBER_ADD];


    while(ros::ok()) {
        listener.count++;
        ros::spinOnce();
    printf("\33[2J  \33[0;0H \033[1;32;40m ");
        printf(" \033[1;31;40m  master:%d \033[0m\n",listener.count);
        for (int i = 0; i <NUMBER_ADD ; ++i) {

            listener.address[0]=ADD0;
            listener.address[1]=ADD1;
```

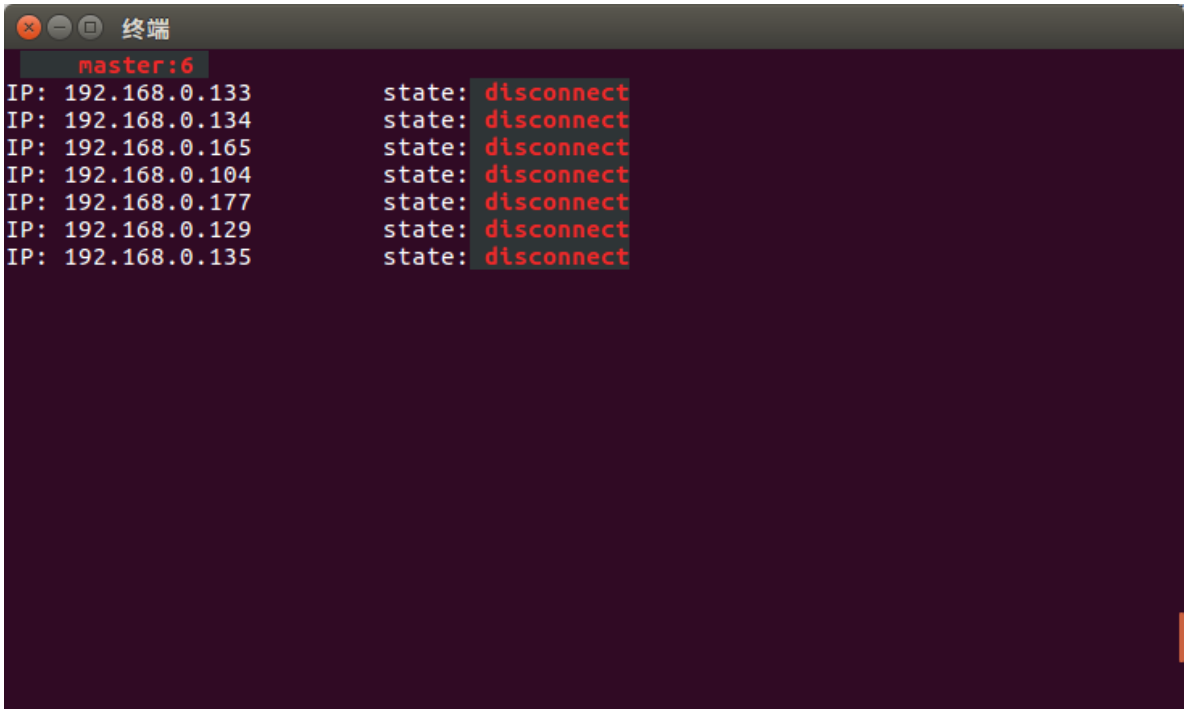
```
            listener.address[2]=ADD2;
            listener.address[3]=ADD3;
            listener.address[4]=ADD4;
            listener.address[5]=ADD5;
            listener.address[6]=ADD6;

            listener.port = "1234";
            portno = atoi(listener.port);
            sockfd = socket(AF_INET, SOCK_STREAM, 0);
            if (sockfd< 0)
            {
                listener.isconnect[i] = false;
                ROS_INFO("ERROR socket open : %d", i);
            } else
            {
                server[i] = gethostbyname(listener.address[i]);
                bzero((char *) &(serv_addr[i]), sizeof(serv_addr[i]));
                serv_addr[i].sin_family = AF_INET;
                bcopy((char *)server[i]->h_addr,(char
*)&serv_addr[i].sin_addr.s_addr, server[i]->h_length);
                serv_addr[i].sin_port = htons(portno);

                if (connect(sockfd,(struct sockaddr *) &
(serv_addr[i]),sizeof(serv_addr[i])) < 0)
                {
                   // ROS_INFO("ERROR connect : %d", i);
                    listener.isconnect[i] = false;
                }
                else
                    listener.isconnect[i]= true;
            }

            if(listener.isconnect[i] )
            {
                n = write(sockfd,listener.getMessageValue(i),56);//后面这个n取错了
不行。是0会导致那边不受
                //if (n < 0)
                  //  ROS_INFO("ERROR write : %d", i);
            }

            loop_rate.sleep();
            close(sockfd);
        }
        for(int i=0;i<NUMBER_ADD;++i)
        {
            printf("IP: %-20s
state:%s\n",listener.address[i],listener.isconnect[i]?"\033[1;32;40m
connect\33[0m":"\033[1;31;40m disconnect\33[0m");
        }
    }
    return 0;
}
```



server_node

```
#include <ros/ros.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include "std_msgs/String.h"
#include "geometry_msgs/Pose.h"

using namespace std;
geometry_msgs::Pose pose_get;

void error(const char *msg) {
    perror(msg);
    exit(1);
}


//template <typename T>
//void decode_2_T(const unsigned char *input,T& output)
//{
//    output=*(T *)(input);
//}
//
//template <typename T>
//void code_2_uc(T& input, unsigned char * output)
//{
//    unsigned char *ptr=(unsigned char *)&input;
//    for (int i = 0; i < sizeof(T) ; ++i) {
//        output[i]=ptr[i];
//    }
//}
```

```
void decode(char *buf) {
    char temp[sizeof(geometry_msgs::Pose)];
    for (int i = 0; i < sizeof(geometry_msgs::Pose); ++i) {
        temp[i] = buf[i];
    }
    pose_get = *(geometry_msgs::Pose *) temp;
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "server_node");
    ros::NodeHandle nh;

    ros::Publisher server_pub = nh.advertise<geometry_msgs::Pose>
("/WTR/target_pose", 1000);
    //Testing package is working fine
    int sockfd, newsockfd, portno; //Socket file descriptors and port number
    socklen_t clilen; //object clilen of type socklen_t
    char buffer[256]; //buffer array of size 256
    struct sockaddr_in serv_addr, cli_addr; ///two objects to store client and
server address
    std_msgs::String message;
    std::stringstream ss;
    int n;
    ros::Duration d(0.01); // 100Hz
//    if (argc < 2) {
//        fprintf(stderr, "ERROR, no port provided\n");
//        exit(1);
//    }
    portno = atoi("1234");
//    cout << "Hello there! This node is listening on port " << portno << " for
incoming connections" << endl;

    while (ros::ok()) {
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0)
            error("ERROR opening socket");
        int enable = 1;
        if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &enable, sizeof(int)) <
0)
            error("setsockopt(SO_REUSEADDR) failed");
        bzero((char *) &serv_addr, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = INADDR_ANY;
        serv_addr.sin_port = htons(portno);
        if (bind(sockfd, (struct sockaddr *) &serv_addr,
                 sizeof(serv_addr)) < 0)
            error("ERROR on binding");
        listen(sockfd, 5);
        clilen = sizeof(cli_addr);
        newsockfd = accept(sockfd,
                           (struct sockaddr *) &cli_addr,
                           &clilen);
        if (newsockfd < 0)
            error("ERROR on accept");

        ss.str(std::string()); //Clear contents of string stream
        bzero(buffer, 256);
```

```
        n = read(newsockfd, buffer, 255);
        decode(buffer);
        cout<<n<<endl;
        cout << pose_get << endl;
        server_pub.publish(pose_get);
        if (n < 0) error("ERROR reading from socket");
        close(sockfd);
        close(newsockfd);
//      ss << buffer;
//      message.data = ss.str();
//      ROS_INFO("%s", message.data.c_str());
//      server_pub.publish(message);
//      n = write(newsockfd,"I got your message",18);
        //if (n < 0) error("ERROR writing to socket");
    }
    return 0;
}
```

代码使用

只需要终端输入ifconfig查询本机ip。输入到tcp_client对应宏下即可。