Implementation of Forward & Inverse Geometric Models with Wireless Client/Server Environment on Stargate Board

Course Project: COE 584 Robotics Fall 2006

Course Instructor: Dr. Mayez Al-Mouhamed

Student: Umair Farooq Siddiqi

Student ID: 240304

Contents

Introduction to Stargate Board

The Stargate is a high performance processing platform designed for sensor, signal processing, control and wireless sensor networking applications. The Stargate is preloaded with Linux and basic device drivers. One Stargate board has the following components, Fig. 1 shows the Stargate board:

1. 32-bit 400 MHz, Intel PXA 255 Intel Xscale RISC Processor
2. 32 MB Flash and 64 MB SDRAM
3. RS 232 serial port that by default displays screen output to the terminal program at the other side.
4. 10/100 Ethernet that can provide wired LAN connectivity
5. USB host that provides connections to USB devices when their drivers are installed.
6. Wireless LAN Ambicom 802.11b CF card that provides wireless connectivity
7. JTAG that provides additional connectivity and data transfer.

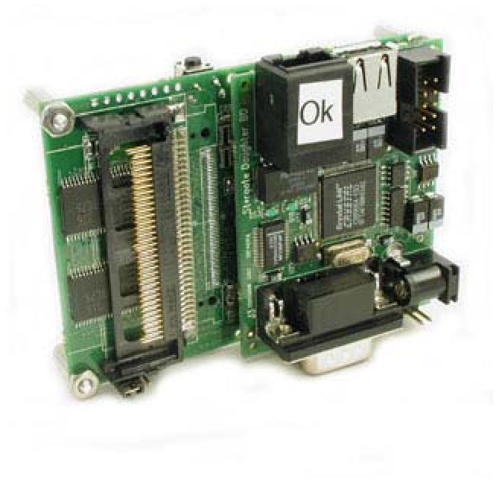


Fig. 1: The Stargate board

Literature Survey

Paper: *R. Rajaravivarma, & L. Cetinski, "A Development Platform for Wireless Internet Connected Robotic Devices" Consortium for Computing Sciences in Colleges, 2005.*

In this paper a model car containing the Stargate computer is build. The car is connected to the Host PC through the Stargate wireless communication. The Stargate also takes instructions from the Host PC & control the driving mechanics of the car. A webcam camera is attached in the car and to the Stargate that relays the video of the front screen to the Host PC. A user at the Host can see the video and issue commands to the driving mechanics of the car.

The software architecture consists of a web-server running on the Stargate & a web browser on the Host PC. The Apache is used for the web-server that runs on top of Linux operating system. Similarly, the Modzilla used for the web browser at the client (Host PC), which also runs on top of Linux OS. The server contains two CGI scripts written in C-language. One script continuously send the webcam data to the client & the second script control the driving mechanics of the car & take commands from the web-server. Image conversion is also performed in the server using additional libraries.

Design of the Wireless Client/Server Network

Manual Configuration:

The Host PC and the Stargate boards are installed with Linux Operating System (OS). The Host PC have Ethernet connection and all Stargate boards have both Ethernet and Wireless cards. In the first step we connect one Stargate board that should not be a mobile robot using the RJ-45 cross-over cable. After making the connection we run the following commands on the Host PC:

```
# su
# ifconfig eth0 10.0.0.1
```

We also run the following commands on the Stargate board:

```
# su
# ifconfig eth0 10.0.0.2
```

The configuration of the rest of the network in described in the following text.

First we check the installation of AmbiCom card on each Stargate board (including the board we configured above). The following commands are run on each board:

```
# su
# cardctl Ident
```

If the AmbiCom card is properly installed then the card name and model is displayed. Next we configure wireless link on Stargate boards. This is accomplished by issuing the following commands on each Stargate board:

```
# iwconfig wlan0 essid robotics
# iwconfig wlan0 mode Ad-Hoc
# iwconfig wlan0 key s:jerry
# ifconfig wlan0 11.0.0.x
```

Where x = 1 if the current Stargate is connected to the Host PC through RJ-45 cross-over cable otherwise x can any number from 2 to 254. A unique value of x should be chosen for each board. Next we run the following command on each Stargate:

```
# route add –net 10.0.0.0 netmask 255.0.0.0 gw 11.0.0.1 wlan0
```

The configuration of Stargate boards is complete and following additional commands run only on the Host PC and the Stargate connected to it through the RJ-45 cross-over cable. The command for Stargate is:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

The above command enables IP forwarding on the Stargate board and makes it act as a router for the Host PC and other wired network. The command for the Host PC is:

```
# route add –net 11.0.0.0 netmask 255.0.0.0 gw 10.0.0.2 eth0
```

The configuration of the complete network is complete at this point. The illustration in Fig.2 shows the above network configuration.

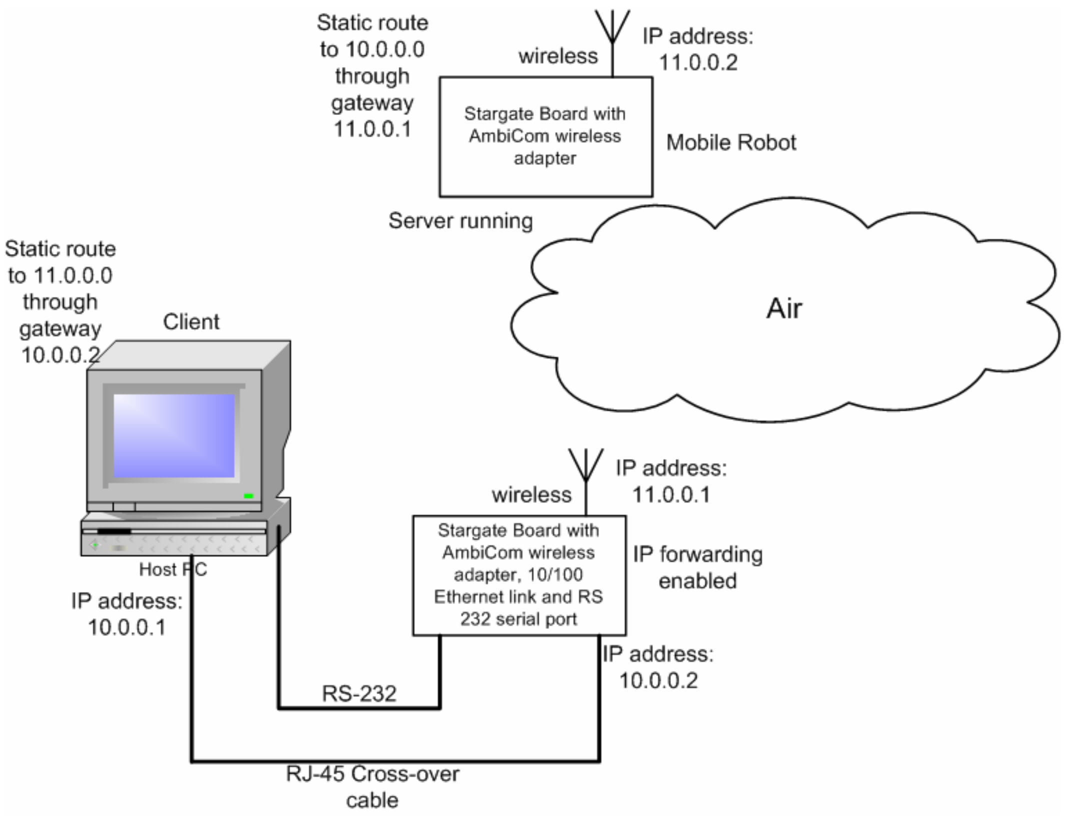

Static route to 10.0.0.0 through gateway 11.0.0.1
wireless
IP address: 11.0.0.2
Stargate Board with AmbiCom wireless adapter
Mobile Robot
Server running
Air
Static route to 11.0.0.0 through gateway 10.0.0.2
Client
Host PC
IP address: 10.0.0.1
wireless
IP address: 11.0.0.1
Stargate Board with AmbiCom wireless adapter, 10/100 Ethernet link and RS 232 serial port
IP forwarding enabled
IP address: 10.0.0.2
RS-232
RJ-45 Cross-over cable

Fig.2: Illustration of the Client/Server Network

Automatic Configuration:

Each Stargate should be booted and configured manually for the very first time it is connected to the proposed wireless network but from next time and onwards we can automatic the network setup using the method described in this section. The Stargate boards can be configured for automatic network configuration through editing in two configuration files /etc/pcmcia/wireless.opts and /etc/opts/network.opts present in the Stagate. The wireless.opts includes configuration for Ambicom wireless card and also specify the name of wireless network in the essid field and the type of the wireless network in the mode field. At present we set essid=robotics and mode=Ad-Hoc to indicate an ad-hoc wireless network having name robotics, the remaining items are fixed for Ambicom wireless cards. Same wireless.opts should be present on all Stargate boards present in the network. The network.opts sets the network parameters that are specific to each Stargate board and need to be configured for each Stargate board separately on the host PC. The network.opts sets DHCP=0 because there is no DHCP in our network, it also sets default gateway to 11.0.0.1 i.e. the stargate board acting as router. The field IPAddr= 11.0.0.x where x should be distinct for each board. Both files can be edited on the host PC and send to the Stargate boards through the following commands:

scp wireless.opts root@11.0.0.x:/etc/pcmcia and

scp network.opts root@11.0.0.x:/etc/pcmcia

The Java Run Time (JRE) for arm is installed in the directory /mnt/cf1 that resides in the Flash card. A short-cut should be added to the /usr/sbin directory to avoid export path problems. The method of this is described in the Stargate developer's manual and also written below:

ln –s /mnt/cf1/jre /usr/sbin/jre

The three files wireless.opts, network.opts and jre are included with the source code to help in any future installation. The JRE for arm is also provided.

Software Architecture of the Server in C++

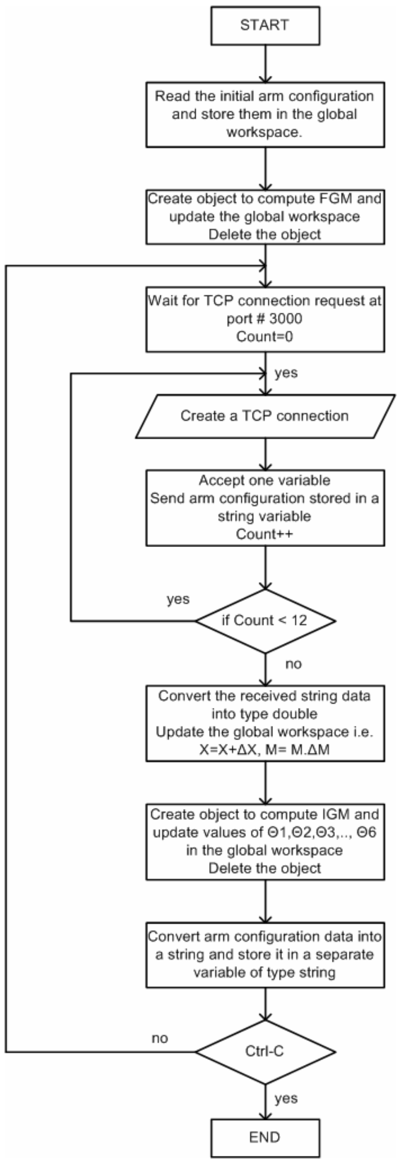

START
Read the initial arm configuration and store them in the global workspace.
Create object to compute FGM and update the global workspace Delete the object
Wait for TCP connection request at port # 3000 Count=0
yes
Create a TCP connection
Accept one variable Send arm configuration stored in a string variable Count++
yes
if Count < 12
no
Convert the received string data into type double Update the global workspace i.e. X=X+ΔX, M= M.ΔM
Create object to compute IGM and update values of Θ1,Θ2,Θ3,.., Θ6 in the global workspace Delete the object
Convert arm configuration data into a string and store it in a separate variable of type string
no
Ctrl-C
yes
END

Fig. 3: The Flow for Server Program running on Stargate boards

The Stargate boards are configured as running servers that establish TCP socket connection at port # 3000 upon request. The server program at the server has the flow shown in Fig. 3 and also described in the following text:

The server program start and load the initial arm configuration in the global work space variables. The variables in the global work space are θ1, θ2, θ3, θ4, θ5, θ6, Xx, Xy, Xz, Yx, Yy, Yz, Zx, Zy, Zz, X, Y, and Z. Next the server program creates an object FGM to compute the Forward Geometric Model (FGM). The object update the global work space after computation. The server program then deletes the object and waits for TCP connection request at Port # 3000. When it receives a connection request it makes a TCP connection to the client and receives variables ΔX= (Δx Δy Δz), and ΔM=(Xx, …, Zx). The server then computes $X^+ = X + \Delta X$ and $M^+ = M.\ \Delta M$ and update the global workspace. The server program next creates an object to compute Inverse Geometric Model (IGM) & update values of θ1, θ2, θ3, θ4, θ5, & θ6 in the global workspace. The server also computes FGM and IGM. The flow for computation of FGM is shown in Fig. 4 and the flow for computation of IGM is shown in Fig. 5.

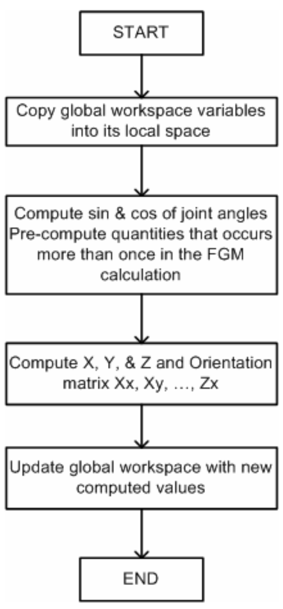

START
Copy global workspace variables into its local space
Compute sin & cos of joint angles Pre-compute quantities that occurs more than once in the FGM calculation
Compute X, Y, & Z and Orientation matrix Xx, Xy, …, Zx
Update global workspace with new computed values
END

Fig. 4: Flow for Computation of Forward Geometric Model (FGM)

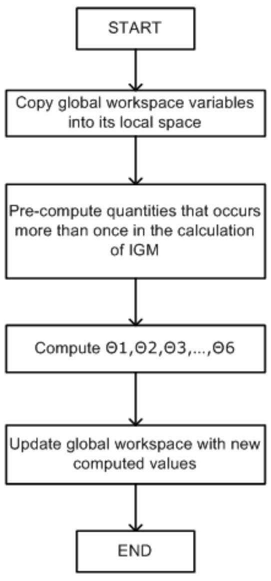

START
Copy global workspace variables into its local space
Pre-compute quantities that occurs more than once in the calculation of IGM
Compute Θ1,Θ2,Θ3,...,Θ6
Update global workspace with new computed values
END

Fig. 5: Flow for Computation of Inverse Geometric Model (IGM)

Software Architecture of the Client in C++

The client program flow is shown in Fig. 6 and described in the following text:

The client program runs at the Host PC and starts by taking the IP address of the server from the user. The IP address of the server is the IP address of the wireless link of the Stargate. The wireless links of Stargates are assigned IP address 11.0.0.x. Then the client program takes input for (Δx Δy Δz) and (φ_1, φ_2, φ_3). It then computes $\Delta M= (Ry(\varphi_1). Rx(\varphi_2). Rz(\varphi_3))$. The client program then sends a TCP connection request for Port # 3000 at the IP address of the server. After establishment of TCP connection the client sends the change ΔX & ΔM to the server and receives the current arm configuration.

Time Measurement and Results

The time of computation of FGM and IGM is measured in microseconds using time measurement function gettimeofday() defined in the Linux library sys/time.h.

The time of computing FGM and IGM is measured on Stargate boards and is shown in the Table I below:

Table 1: Time of Computing FGM & IGM on Stargate

FGM	**2029 µsec**	**2019 µsec**	**2023 µsec**	**2028 µsec**	**2024 µsec**
IGM	**1899 µsec**	**1757 µsec**	**1756 µsec**	**1633 µsec**	**1798 µsec**

Software Architecture of the Server in Java

The client/server programming and the computation of FGM and IGM are also programmed in Java 2 programming language. The client uses the J2RE version 1.6 SE while the server uses the J2RE 1.3 for arm processors. The server JRE for arm was downloaded from http://www.blacksdown.org and installed to /mnt/cf1/jre folder on the Stargate. The Java Client/Server programming is much simplified and is error free than the C++ version. There is no need for data type conversion before sending and receiving data as required in C++ and all data send at once instead of in a loop. The flow for the server is shown in the Fig. 6 below:

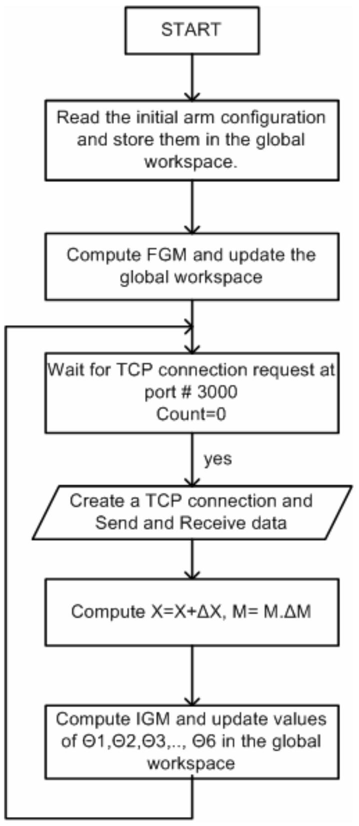
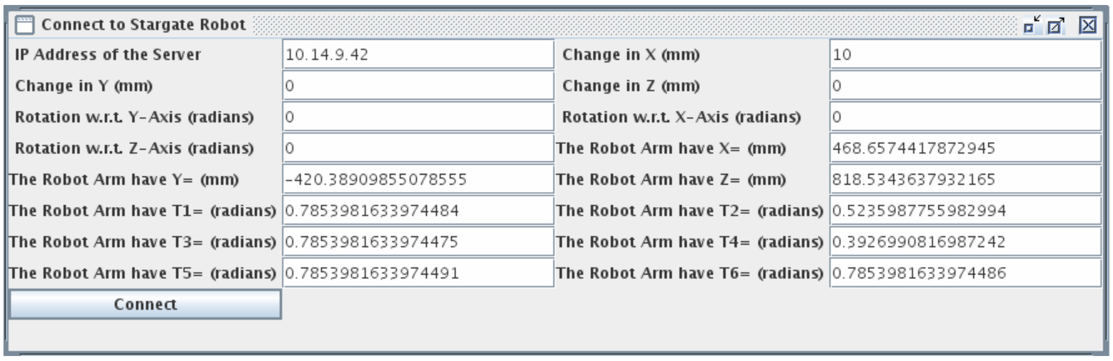

START
Read the initial arm configuration and store them in the global workspace.
Compute FGM and update the global workspace
Wait for TCP connection request at port # 3000 Count=0
yes
Create a TCP connection and Send and Receive data
Compute X=X+ΔX, M= M.ΔM
Compute IGM and update values of Θ1,Θ2,Θ3,.., Θ6 in the global workspace

Fig. 6: Flow of the Java Server

Software Architecture of the Client in Java

The client in written with a GUI that takes input from the user as well as show the current robotic arm configuration. The GUI is shown in Fig. 7 and the flow the client program in shown in Fig. 8.


Connect to Stargate Robot

IP Address of the Server	10.14.9.42	Change in X (mm)	10
Change in Y (mm)	0	Change in Z (mm)	0
Rotation w.r.t. Y-Axis (radians)	0	Rotation w.r.t. X-Axis (radians)	0
Rotation w.r.t. Z-Axis (radians)	0	The Robot Arm have X= (mm)	468.6574417872945
The Robot Arm have Y= (mm)	-420.38909855078555	The Robot Arm have Z= (mm)	818.5343637932165
The Robot Arm have T1= (radians)	0.7853981633974484	The Robot Arm have T2= (radians)	0.5235987755982994
The Robot Arm have T3= (radians)	0.7853981633974475	The Robot Arm have T4= (radians)	0.3926990816987242
The Robot Arm have T5= (radians)	0.7853981633974491	The Robot Arm have T6= (radians)	0.7853981633974486

Fig**. 7: Client GUI for Inputs and display of the Current Robotic Arm Configuration**

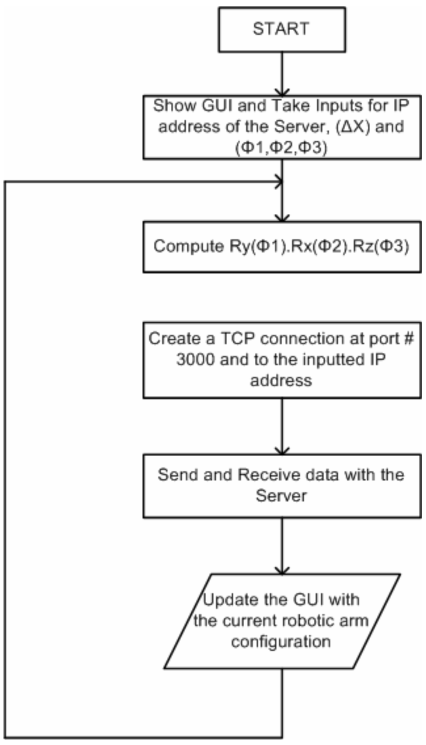

START
Show GUI and Take Inputs for IP address of the Server, (ΔX) and (Φ1,Φ2,Φ3)
Compute Ry(Φ1).Rx(Φ2).Rz(Φ3)
Create a TCP connection at port # 3000 and to the inputted IP address
Send and Receive data with the Server
Update the GUI with the current robotic arm configuration

Fig. 8: The Flow of the Client in Java

Conclusion

The FGM and IGM are successfully implemented in the Stargate computers with minimum computation time in both C++ and java. Both C++ and Java code have the same architecture.

The wireless communication between Stargate computers and with the Host PC is also successfully performed using TCP socket programming in Java and C++. The Java outperformed C++ in terms of its ease of use and reliable data transmission.

C++ Source Code

Server Side Software

```
#include <iostream>
using namespace std;
#include "ServerSocket.h"
#include "SocketException.h"
#include <string>
#include <stdio.h>
#include <sstream>
#include <stdexcept>
#include <cmath>
#include <sys/time.h>
```

```
#define PI 3.14159265

double T1,T2,T3,T4,T5,T6;
double L1,L2,L3,L4,L5,L6,L56,L34;
double X,Y,Z;
double Xx,Xy,Xz,Yx,Yy,Yz,Zx,Zy,Zz;
class FGM {
private:
double t1,t2,t3,t4,t5,t6;
double l1,l2,l3,l4,l5,l6,l34,l56;
double x,y,z;
double x1,x2,x3,y1,y2,y3,z1,z2,z3;
public:
FGM() { //This constructor copies global variables into its local space
t1=::T1;
t2=::T2;
t3=::T3;
t4=::T4;
t5=::T5;
t6=::T6;
l1=::L1;
l2=::L2;
l3=::L3;
l4=::L4;
l5=::L5;
l6=::L6;
l34=::L34;
l56=::L56;
};
//This function performs precomputation of all arithmetic operations that appear more then once in the
FGM calculation
void compute() {
//Precomputation
double s1,s2,s3,s4,s5,s6,c1,c2,c3,c4,c5,c6,s23,c23,p1,p2,c4c6,c23s4c6,s4c5s6;
double c23c4c5c6,s23s5s6,c5c6,c23c5,c23c4s5,s23c4,s23c5,l56,x4,y4,z4,s4s5,c4s5;
s1=sin(t1);
s2=sin(t2);
s3=sin(t3);
s4=sin(t4);
s5=sin(t5);
s6=sin(t6);
c1=cos(t1);
c2=cos(t2);
c3=cos(t3);
c4=cos(t4);
c5=cos(t5);
c6=cos(t6);
s23=s2*c3+c2*s3;
c23=c2*c3-s2*s3;
p1=s2*l2+s23*(l34);
p2=c2*l2 + c23*(l34);

c4c6=c4*c6;
c23s4c6=c23*s4*c6;
s4c5s6=s4*s5*s6;
c23c4c5c6=c23*c4*c5*c6;
s23s5s6=s23*s5*s6;
c5c6=c5*c6;
c23c5=c23*c5;
c23c4s5=c23*c4*s5;
s23c4=s23*c4;
```

```
s23c5=s23*c5;
s4s5=s4*s5;
c4s5=c4*s5;
//Calculation
x1=c1*c4c6 - s1*c23s4c6 - c1*s4c5s6 - s1*c23c4c5c6 + s1*s23s5s6;
x2=s1*c4c6 + c1*c23s4c6 - s1*s4c5s6 + c1*c23c4c5c6 - c1*s23s5s6;
x3=s23*s4*c6 + s23*c4*c5*c6 + c23*s5*s6;
y1=c1*c4*c6 + s1*c23*s4*s6 - c1*s4*c5c6 - s1*c23*c4*c5c6 + s1*s23*s5*c6;
y2=-s1*c4*s6 - c1*c23*s4*s6 - s1*s4*c5*c6 + c1*c23*c4*c5c6 - c1*s23*s5*c6;
y3=s23*s4*s6 + s23c4*c5c6 + c23*s5*s6;
z1=c1*s4s5 + s1*c23*c4s5 + s1*s23c5;
z2=s1*s4s5 - c1*c23c4s5 - c1*s23c5;
z3=-s23*c4s5 + c23c5;
x4=s1*p1;
y4=-c1*p1;
z4=l1+p2;
l56=l5+l6;
x=x4 + z1*(l56);
y=y4 + z2*(l56);
z=z4 + z3*(l56);

//Update the global variables

::X=x;
::Y=y;
::Z=z;
::Xx=x1;
::Xy=x2;
::Xz=x3;
::Yx=y1;
::Yy=y2;
::Yz=y3;
::Zx=z1;
::Zy=z2;
::Zz=z3;

//finish
};
~FGM() {};
};

class IGM {
private:
double t1,t2,t3,t4,t5,t6;
double l1,l2,l3,l4,l5,l6,l56,l34;
double x,y,z;
double x1,x2,x3,y1,y2,y3,z1,z2,z3;
double x4,y4,z4;
public:
IGM() { //The constructor copies the global variables into its local space
x=::X;
y=::Y;
z=::Z;
l1=::L1;
l2=::L2;
l3=::L3;
l4=::L4;
l5=::L5;
l6=::L6;
x1=::Xx;
x2=::Xy;
```

```
x3=::Xz;
y1=::Yx;
y2=::Yy;
y3=::Yz;
z1=::Zx;
z2=::Zy;
z3=::Zz;
l34=::L34;
l56=::L56;
};

void compute() {
double x4,y4,z4,p1,p2,p3,p4,p5,p6,p7,p8,p9,s55;
double c1,s1,c2,s2,c3,s3,c4,s4,c5,s5,c6,s6,c23,s23,s4s5,c4s5;
x4=x - l56*z1;
y4=y - l56*z2;
z4=z - l56*z3;

//precomputations
p1=((x4*x4) + (y4*y4));
p2=sqrt(p1);
p3=(z4-l1);
s1=x4/p2;
c1=-y4/p2;
//if ((c1<=1) && (c1>=-1))
    t1=acos(c1);
//else t1=PI/2;
p7=2*l2*l34;
p8=l34*l34;
p9=l2*l2;
if (s1<0) t1=-t1;
p5=(x4*s1)-(y4*c1);
c3=(p1+(p3*p3)-(p9)-(p8))/(p7);
s3=sqrt(1-(c3*c3));
//if ((c3<=1) && (c3>=-1))
    t3=acos(c3);
//else t3=PI/2;


//s3=sin(t3);
p4=(p9)+(p8)+(p7*c3);
s2=(p5*(l2+c3*l34)-p3*(s3*l34))/p4;
c2=(p5*(s3*l34)+p3*(l2+(c3*l34)))/p4;
//if ((c2<=1) && (c2>=-1))
    t2=acos(c2);
//else t2=PI/2;
if (s2<0) t2=-t2;
c23=c2*c3-s2*s3;
s23=s2*c3+s3*c2;
s4s5=c1*z1+s1*z2;
p6=(s1*z1-c1*z2);
c4s5=c23*(p6)-s23*(z3);
c5=s23*(p6)+c23*z3;
//if ((c5<=1) && (c5>=-1))
    t5=acos(c5);
//else t5=PI/2;
s5=sqrt((s4s5)*(s4s5)+(c4s5)*(c4s5));
if (s5<0)
s55=-s5;
else s55=s5;
```

```
if (s55>0.0001) {
  s4=s4s5/s5;
  c4=c4s5/s5;
//if ((c4<=1) && (c4>=-1))
     t4=acos(c4);
//else t4=PI/2;

  if (s4<0)
   t4=-t4;
} else
 t4=::T4;
p2=s1*c23;
p3=c1*c23;
c6=c4*(c1*x1+s1*x2)+s4*(-p2*x1 + p3*x2 + s23*x3);
s6=-c4*(c1*y1+s1*y2)-s4*(-p2*y1+p3*y2+s23*y3);
//if ((c6<=1) && (c6>=-1))
     t6=acos(c6);
//else t6=PI/1;
if (s6<0)
   t6=-t6;

//Update the global variables

::T1=t1;
::T2=t2;
::T3=t3;
::T4=t4;
::T5=t5;
::T6=t6;
//printf("\n %f %f %f %f %f %f ",t1,t2,t3,t4,t5,t6);

};
~IGM() { };
};

class BadConversion : public std::runtime_error {
 public:
   BadConversion(const std::string& s)
     : std::runtime_error(s)
     { }
 };

inline double convertToDouble(const std::string& s)
 {
   std::istringstream i(s);
   double x;
   if (!(i >> x))
     throw BadConversion("convertToDouble(\"" + s + "\")");
   return x;
 }
 inline std::string stringify(double x)
 {
   std::ostringstream o;
   if (!(o << x))
     throw BadConversion("stringify(double)");
   return o.str();
 }

int main ( int argc, int argv[] )
{
double n[15];
```

```
std::string ns[12];
double x1,x2,x3,y1,y2,y3,z1,z2,z3;
std::string dp,cp,d1,d2,d3,d4,d5,d6,d7,d8,d9,;
::T1=PI/4;
::T2=PI/6;
::T3=PI/4;
::T4=PI/8;
::T5=PI/4;
::T6=-PI/3;
::L1=500;
::L2=300;
::L3=300;
::L4=100;
::L5=50;
::L6=50;
::L34=::L3+::L4;
::L56=::L5+::L6;

FGM *f;  //An Object for FGM is created
struct timeval tv1,tv2;
struct timezone tz1,tz2;
gettimeofday(&tv1, &tz1);
f= new FGM();
f->compute(); //The Global variables are updated
delete f; //The Object for FGM is destroyed
gettimeofday(&tv2, &tz2);
printf("\nTime Elapsd is FGM is seconds= %d and microseconds= %d",(tv2.tv_sec-
tv1.tv_sec),(tv2.tv_usec-tv1.tv_usec));
cp=" Please pass one program iteration to see the robot hand configuration";
//int c=0;

  std::cout << "\nStargate server is running....\n";
while(true) {
  try
    {
     // Create the socket
     ServerSocket server ( 3000 );
     //c=0;
     for (int i=0;i<12;i++)
            {

             ServerSocket new_sock;
             server.accept ( new_sock );

             try
               {
                 while ( true )
            {

              new_sock>>dp;
              ns[i]=dp;
              //c++;

              new_sock << cp;


                     }

               }
             catch ( SocketException& ) {}
```

```
               //printf("\n LOOP IS OVER AT SERVER");
             }

//UPDATE GLOBAL WORKSPACE
               for (int gc=0;gc<12;gc++)
                 n[gc]=convertToDouble(ns[gc]);
               ::X=::X+n[0];
               ::Y=::Y+n[1];
               ::Z=::Z+n[2];
               x1=::Xx*n[3] + ::Yx*n[4] + ::Zx*n[5];
               x2=::Xy*n[3] + ::Yy*n[4] + ::Zy*n[5];
               x3=::Xz*n[3] + ::Yz*n[4] + ::Zz*n[5];
               y1=::Xx*n[6] + ::Yx*n[7] + ::Zx*n[8];
               y2=::Xy*n[6] + ::Yy*n[7] + ::Zy*n[8];
               y3=::Xz*n[6] + ::Yz*n[7] + ::Zz*n[8];
               z1=::Xx*n[9] + ::Yx*n[10] + ::Zx*n[11];
               z2=::Xy*n[9] + ::Yy*n[10] + ::Zy*n[11];
               z3=::Xz*n[9] + ::Yz*n[10] + ::Zz*n[11];

               ::Xx=x1;
               ::Xy=x2;
               ::Xz=x3;
               ::Yx=y1;
               ::Yy=y2;
               ::Yz=y3;
               ::Zx=z1;
               ::Zy=z2;
               ::Zz=z3;
               gettimeofday(&tv1, &tz1);
               IGM *I;
                         I=new IGM();
                         I->compute();
                         delete I;
               gettimeofday(&tv2, &tz2);
               /*FGM *f2;  //An Object for FGM is created
               f2= new FGM();
               f2->compute(); //The Global variables are updated
               delete f2; //The Object for FGM is destroyed
               */


printf("\nTime    Elapsd    is    IGM    is    seconds=    %d    and    microseconds=    %d",(tv2.tv_sec-
tv1.tv_sec),(tv2.tv_usec-tv1.tv_usec));
               printf("\n %f %f %f %f %f %f %f %f %f ",::X,::Y,::Z,::T1,::T2,::T3,::T4,::T5,::T6);
               d1=stringify(::X);
               d2=stringify(::Y);
               d3=stringify(::Z);
                         d4=stringify(::T1);
               d5=stringify(::T2);
               d6=stringify(::T3);
               d7=stringify(::T4);
               d8=stringify(::T5);
               d9=stringify(::T6);



          cp="X= " +d1 + " Y= " + d2 + " Z= " +d3 + " T1=" +d4+ " T2= " +d5+ " T3= " +d6+ " T4= "
+d7+ " T5= " +d8+ " T6= "+d9;

   }
```

```
  catch ( SocketException& e )
    {
      std::cout << "Exception was caught:" << e.description() << "\nExiting.\n";
    }


}
 return 0;

}
```

Client Side Software

```
#include "ClientSocket.h"
#include "SocketException.h"
#include <iostream>
using namespace std;
#include <string>
#include <cmath>
#include <sstream>
#include <stdexcept>


class BadConversion : public std::runtime_error {
 public:
   BadConversion(const std::string& s)
     : std::runtime_error(s)
     { }
 };

 inline std::string stringify(double x)
 {
   std::ostringstream o;
   if (!(o << x))
     throw BadConversion("stringify(double)");
   return o.str();
 }


int main ( int argc, int argv[] )
{ double n1[12]={0.0,0.0,0.0,1,0,0,0,1,0,0,0,1};
  char ch1;
  char ipaddr[11];
  std::string s1[12];
  int sd1=0;
  int c=0;

  double r1,r2,r3,S1,S2,S3,C1,C2,C3;
printf("\nClient Side Software Connected to the server on Stargate Computer to compute FGM and IGM:");
printf("\nThissoftware is written by Umair Farooq Siddiqi, 240304 MS(COE) in course COE: 584 Robotics under supervision of course teacher Dr. Mayez Al-Mouhamed");
printf("\n Please press control-c to quit the program at any time");
printf("\n Please Enter the IP address of the server (Stargate):  ");
cin>>ipaddr;

while (true) {

  if (sd1==1 ) {
    n1[0]=0;
    n1[1]=0;
    n1[2]=0;
    n1[3]=1;
    n1[4]=0;
    n1[5]=0;
    n1[6]=0;
    n1[7]=1;
    n1[8]=0;
    n1[9]=0;
    n1[10]=0;
    n1[11]=1;
    sd1=0;
    sd1=0;
    int gh=0;
    for(int h=0;h<50000;h++)
      gh=gh+1;
    }
else {
  printf("\n Do you want to enter the change in the Position (X, Y, Z) or in Orientation (Ry, Rx, Rz)
(y=yes, others=No: ");
  cin>>ch1;
  if (ch1=='y' || ch1=='Y') {
  cout<<"\n Please enter the change in X you want (mm): ";
  cin>>n1[0];
  printf("\n Please enter the change in Y you want (mm): ");
  cin>>n1[1];
  printf("\n Please enter the change in Z you want (mm): ");
  cin>>n1[2];
  sd1=1;
  }
  else {
  n1[0]=0;
  n1[1]=0;
  n1[2]=0;
  sd1=0;
  int wc;
  for (int w=0;w<50000;w++)
     wc++;
  };


   printf("\n Do you want to enter the change in the arm orientation  (y=yes, others=No: ");
   cin>>ch1;
   if (ch1=='Y' || ch1=='y') {
    printf("\n Please enter the change in radians with respect to y-axis : ");
    cin>>r1;
    printf("\n Please enter the change in radians with respect to x-axis : ");
    cin>>r2;
    printf("\n Please enter the change in radians with respect to z-axis : ");
    cin>>r3;
    S1=sin(r1);
    S2=sin(r2);
    S3=sin(r3);
    C1=cos(r1);
    C2=cos(r2);
    C3=cos(r3);

    n1[3]=(C1*C3) + (S1*S2*S3);
    n1[4]=C2*S3;
    n1[5]=(-S1*C3)+(C1*S2*C3);
    n1[6]=(-C1*S3)+(S1*S2*C3);
    n1[7]=(C2*C3);
    n1[8]=(S1*S3)+(C1*S2*C3);
     n1[9]=S1*C2;
     n1[10]=-S2;
     n1[11]=C1*C2;

     }
    else {
    n1[3]=1;
    n1[4]=0;
    n1[5]=0;
    n1[6]=0;
    n1[7]=1;
    n1[8]=0;
    n1[9]=0;
    n1[10]=0;
    n1[11]=1;
    int wc=0;
    for (int w=0;w<50000;w++)
      wc++;

    };
};

//};
//c=1;



for (int i=0;i<12;i++)
   s1[i]=stringify(n1[i]);


for (int i=0;i<12;i++) {
  try
    {

      ClientSocket client_socket ( ipaddr, 3000 );

      std::string reply1;
      //std::string reply2;

      try
            {
              client_socket <<s1[i];

          client_socket >> reply1;
            }
      catch ( SocketException& ) {}
      if (i==11)
       std::cout << "\n The current arm configuration is: "<< reply1 ;


    }
  catch ( SocketException& e )
    {
      std::cout << "Exception was caught:" << e.description() << "\n";
    }
//---------------
}
};
  return 0;
```

```
}
```

Java Source Code

Server Side Software

```
public class global {

public static double T1,T2,T3,T4,T5,T6;
public static double L1,L2,L3,L4,L5,L6,L56,L34;
public static double X,Y,Z;
public static double Xx,Xy,Xz,Yx,Yy,Yz,Zx,Zy,Zz;
public global() { };
}

import java.net.*;
import java.math.*;



public class IGM
{

double t1,t2,t3,t4,t5,t6;
double l1,l2,l3,l4,l5,l6,l56,l34;
double x,y,z;
double x1,x2,x3,y1,y2,y3,z1,z2,z3;
double x4,y4,z4;

global g;

public IGM() {

x=g.X;
y=g.Y;
z=g.Z;
l1=g.L1;
l2=g.L2;
l3=g.L3;
l4=g.L4;
l5=g.L5;
l6=g.L6;
x1=g.Xx;
x2=g.Xy;
x3=g.Xz;
y1=g.Yx;
y2=g.Yy;
y3=g.Yz;
z1=g.Zx;
z2=g.Zy;
z3=g.Zz;
l34=g.L34;
l56=g.L56;

compute();

g.T1=t1;
g.T2=t2;
g.T3=t3;
```

```
g.T4=t4;
g.T5=t5;
g.T6=t6;

}

void compute() {
double x4=0,y4=0,z4=0,p1=0,p2=0,p3=0,p4=0,p5=0,p6=0,p7=0,p8=0,p9=0,s55=0;
double c1=0,s1=0,c2=0,s2=0,c3=0,s3=0,c4=0,s4=0,c5=0,s5=0,c6=0,s6=0,c23=0,s23=0,s4s5=0,c4s5=0;
x4=x - l56*z1;
y4=y - l56*z2;
z4=z - l56*z3;

//precomputations
p1=((x4*x4) + (y4*y4));
p2=Math.sqrt(p1);
p3=(z4-l1);
s1=x4/p2;
c1=-y4/p2;
t1=Math.acos(c1);
p7=2*l2*l34;
p8=l34*l34;
p9=l2*l2;
if (s1<0) t1=-t1;
p5=(x4*s1)-(y4*c1);
c3=(p1+(p3*p3)-(p9)-(p8))/(p7);
s3=Math.sqrt(1-(c3*c3));
t3=Math.acos(c3);
//s3=sin(t3);
p4=(p9)+(p8)+(p7*c3);
s2=(p5*(l2+c3*l34)-p3*(s3*l34))/p4;
c2=(p5*(s3*l34)+p3*(l2+(c3*l34)))/p4;
t2=Math.acos(c2);
if (s2<0) t2=-t2;
c23=c2*c3-s2*s3;
s23=s2*c3+s3*c2;
s4s5=c1*z1+s1*z2;
p6=(s1*z1-c1*z2);
c4s5=c23*(p6)-s23*(z3);
c5=s23*(p6)+c23*z3;
t5=Math.acos(c5);
s5=Math.sqrt((s4s5)*(s4s5)+(c4s5)*(c4s5));
if (s5<0)
s55=-s5;
else s55=s5;

if (s55>0.0001) {
  s4= s4s5/s5;
  c4= c4s5/s5;
  t4= Math.acos(c4);
  if (s4<0)
   t4=-t4;
} else
 t4=g.T4;


p2= s1*c23;
p3= c1*c23;
c6= c4*(c1*x1+s1*x2)+s4*(-p2*x1 + p3*x2 + s23*x3);
s6= c4*(c1*y1+s1*y2)-s4*(p2*y1-p3*y2-s23*y3);
t6= Math.acos(c6);
```

```
if (s6<0)
   t6=-t6;

}

};


import java.net.*;
import java.math.*;



public class FGM
{

private double t1,t2,t3,t4,t5,t6;
private double l1,l2,l3,l4,l5,l6;
private double x,y,z;
private double x1,x2,x3,y1,y2,y3,z1,z2,z3;

global g;

public FGM() {

t1=g.T1;
t2=g.T2;
t3=g.T3;
t4=g.T4;
t5=g.T5;
t6=g.T6;
l1=g.L1;
l2=g.L2;
l3=g.L3;
l4=g.L4;
l5=g.L5;
l6=g.L6;

compute();

g.X=x;
g.Y=y;
g.Z=z;
g.Xx=x1;
g.Xy=x2;
g.Xz=x3;
g.Yx=y1;
g.Yy=y2;
g.Yz=y3;
g.Zx=z1;
g.Zy=z2;
g.Zz=z3;

}

void compute() {

double s1,s2,s3,s4,s5,s6,c1,c2,c3,c4,c5,c6,s23,c23,p1,p2,c4c6,c23s4c6,s4c5s6;
double c23c4c5c6,s23s5s6,c5c6,c23c5,c23c4s5,s23c4,s23c5,l56,x4,y4,z4,s4s5,c4s5;
s1= Math.sin(t1);
s2= Math.sin(t2);
```

```
s3= Math.sin(t3);
s4= Math.sin(t4);
s5= Math.sin(t5);
s6= Math.sin(t6);
c1= Math.cos(t1);
c2= Math.cos(t2);
c3= Math.cos(t3);
c4= Math.cos(t4);
c5= Math.cos(t5);
c6= Math.cos(t6);
s23= s2*c3+c2*s3;
c23= c2*c3-s2*s3;
p1= s2*l2+s23*(l3+l4);
p2= c2*l2 + c23*(l3+l4);

c4c6=c4*c6;
c23s4c6=c23*s4*c6;
s4c5s6=s4*s5*s6;
c23c4c5c6=c23*c4*c5*c6;
s23s5s6=s23*s5*s6;
c5c6=c5*c6;
c23c5=c23*c5;
c23c4s5=c23*c4*s5;
s23c4=s23*c4;
s23c5=s23*c5;
s4s5=s4*s5;
c4s5=c4*s5;
//Calculation
x1=c1*c4c6 - s1*c23s4c6 - c1*s4c5s6 - s1*c23c4c5c6 + s1*s23s5s6;
x2=s1*c4c6 + c1*c23s4c6 - s1*s4c5s6 + c1*c23c4c5c6 - c1*s23s5s6;
x3=s23*s4*c6 + s23*c4*c5*c6 + c23*s5*s6;
y1=c1*c4*c6 + s1*c23*s4*s6 - c1*s4*c5c6 - s1*c23*c4*c5c6 + s1*s23*s5*c6;
y2=-s1*c4*s6 - c1*c23*s4*s6 - s1*s4*c5*c6 + c1*c23*c4*c5c6 - c1*s23*s5*c6;
y3=s23*s4*s6 + s23c4*c5c6 + c23*s5*s6;
z1=c1*s4s5 + s1*c23*c4s5 + s1*s23c5;
z2=s1*s4s5 - c1*c23c4s5 - c1*s23c5;
z3=-s23*c4s5 + c23c5;
x4=s1*p1;
y4=-c1*p1;
z4=l1+p2;
l56=l5+l6;
x=x4 + z1*(l56);
y=y4 + z2*(l56);
z=z4 + z3*(l56);
}

};

import java.io.*;
import java.net.*;
import java.math.*;

public class SerializeServer {
  public static void main(String args[]) throws Exception {
    double d[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    long t1,t2;
    global g=new global();
    g.T1= Math.PI/4;
    g.T2= Math.PI/6;
    g.T3= Math.PI/4;
    g.T4= Math.PI/8;
```

```
g.T5= Math.PI/4;
g.T6= Math.PI/4;
g.L1=500;
g.L2=300;
g.L3=300;
g.L4=100;
g.L5=50;
g.L6=50;
g.L34=g.L3+g.L4;
g.L56=g.L5+g.L6;
t1=System.nanoTime();
FGM f=new FGM();
t2=System.nanoTime();
System.out.println(t2-t1);
ServerSocket connection = new ServerSocket( 3000 );
while (true)  {
Socket s = connection.accept();
ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream( ));
ObjectInputStream  in  = new ObjectInputStream(s.getInputStream( ) );
 double sum=0.0;
 double x1,x2,x3,y1,y2,y3,z1,z2,z3;
 double e[]= (double []) in.readObject();

      //out.writeObject(new Double(d));
   g.X=g.X+e[0];
   g.Y=g.Y+e[1];
   g.Z=g.Z+e[2];
   x1=g.Xx*e[3] + g.Yx*e[4] + g.Zx*e[5];
   x2=g.Xy*e[3] + g.Yy*e[4] + g.Zy*e[5];
   x3=g.Xz*e[3] + g.Yz*e[4] + g.Zz*e[5];
   y1=g.Xx*e[6] + g.Yx*e[7] + g.Zx*e[8];
   y2=g.Xy*e[6] + g.Yy*e[7] + g.Zy*e[8];
   y3=g.Xz*e[6] + g.Yz*e[7] + g.Zz*e[8];
   z1=g.Xx*e[9] + g.Yx*e[1] + g.Zx*e[11];
   z2=g.Xy*e[9] + g.Yy*e[10] + g.Zy*e[11];
   z3=g.Xz*e[9] + g.Yz*e[10] + g.Zz*e[11];
   g.Xx=x1;
   g.Xy=x2;
   g.Xz=x3;
   g.Yx=y1;
   g.Yy=y2;
   g.Yz=y3;
   g.Zx=z1;
   g.Zy=z2;
   g.Zz=z3;

 d[0]=g.T1;
 d[1]=g.T2;
 d[2]=g.T3;
 d[3]=g.T4;
 d[4]=g.T5;
 d[5]=g.T6;
 d[6]=g.X;
 d[7]=g.Y;
 d[8]=g.Z;
 d[9]=g.Xx;
 d[10]=g.Xy;
 d[11]=g.Xz;
 d[12]=g.Xz;
 d[13]=g.Yx;
 d[14]=g.Yy;
```

```
     d[15]=g.Yz;
     d[16]=g.Zx;
     d[17]=g.Zy;
     d[18]=g.Zz;

     IGM i=new IGM();
     out.writeObject(d);

          }
  }
}
```

Client Side Software

```
import java.io.*;
import java.net.*;
import java.math.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.IOException;

public class TCPClientGUI implements ActionListener {
   static String IP;
   double u[]={0,0,0,0,0,0};
   double d[] = { 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1 };
   double e2[]= {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
   JFrame converterFrame;
   JPanel converterPanel;
   JTextField text1=null;
   JTextField text2=null;
   JTextField text3=null;
   JTextField text4=null;
   JTextField text5=null;
   JTextField text6=null;
   JTextField text7=null;
   JTextField at1=null;
   JTextField at2=null;
   JTextField at3=null;
   JTextField at4=null;
   JTextField at5=null;
   JTextField at6=null;
   JTextField at7=null;
   JTextField at8=null;
   JTextField at9=null;
   JTextField at10=null;
   JTextField at11=null;
   JTextField at12=null;
   JTextField at13=null;
   JTextField at14=null;
   JTextField at15=null;
   JTextField at16=null;
   JTextField at17=null;
   JTextField at18=null;



   JLabel label1,label2,label3,label4,label5,label6,label7;
   JLabel
arm1,arm2,arm3,arm4,arm5,arm6,arm7,arm8,arm9,arm10,arm11,arm12,arm13,arm14,arm15,arm16,ar
m17,arm18;
```

```
JButton connectServer;

public TCPClientGUI() {
   //Create and set up the window.
   converterFrame = new JFrame("Connect to Stargate Robot");
   converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   converterFrame.setSize(new Dimension(1000, 1000));

   //Create and set up the panel.
   converterPanel = new JPanel(new GridLayout(10, 10));

   //Add the widgets.
   addWidgets();

   //Set the default button. //extra
   converterFrame.getRootPane().setDefaultButton(connectServer);

   //Add the panel to the window.
   converterFrame.getContentPane().add(converterPanel, BorderLayout.CENTER);

   //Display the window.
   converterFrame.pack();
   converterFrame.setVisible(true);
}

/**
 * Create and add the widgets.
 */
private void addWidgets() {
   //Create widgets.

   label1 = new JLabel("IP Address of the Server ", SwingConstants.LEFT);
   text1 = new JTextField(10);
   connectServer = new JButton("Connect");
   text2 = new JTextField(10);
   label2 = new JLabel("Change in X (mm) ", SwingConstants.LEFT);
   text3 = new JTextField(10);
   label3 = new JLabel("Change in Y (mm) ", SwingConstants.LEFT);
   text4 = new JTextField(10);
   label4 = new JLabel("Change in Z (mm) ", SwingConstants.LEFT);
   text5 = new JTextField(10);
   label5 = new JLabel("Rotation w.r.t. Y-Axis (radians) ", SwingConstants.LEFT);
   text6 = new JTextField(10);
   label6 = new JLabel("Rotation w.r.t. X-Axis (radians) ", SwingConstants.LEFT);
   text7 = new JTextField(10);
   label7 = new JLabel("Rotation w.r.t. Z-Axis (radians) ", SwingConstants.LEFT);
   arm1 = new JLabel("The Robot Arm have X= (mm) "  , SwingConstants.LEFT);
   at1= new JTextField(5);
   arm2 = new JLabel("The Robot Arm have Y= (mm) ", SwingConstants.LEFT);
   at1= new JTextField(5);
       arm3 = new JLabel("The Robot Arm have Z= (mm) ", SwingConstants.LEFT);
   at2= new JTextField(5);
       arm4 = new JLabel("The Robot Arm have T1= (radians) ", SwingConstants.LEFT);
   at3= new JTextField(5);
       arm5 = new JLabel("The Robot Arm have T2= (radians) ", SwingConstants.LEFT);
   at4= new JTextField(5);
       arm6 = new JLabel("The Robot Arm have T3= (radians) ", SwingConstants.LEFT);
   at5= new JTextField(5);
       arm7 = new JLabel("The Robot Arm have T4= (radians) ", SwingConstants.LEFT);
   at6= new JTextField(5);
       arm8 = new JLabel("The Robot Arm have T5= (radians) ", SwingConstants.LEFT);
```

```
at7= new JTextField(5);
    arm9 = new JLabel("The Robot Arm have T6= (radians) ", SwingConstants.LEFT);
at8= new JTextField(5);
    arm10 = new JLabel("The Robot Arm have Xx= ", SwingConstants.LEFT);
at9= new JTextField(5);
    arm11 = new JLabel("The Robot Arm have Xy= ", SwingConstants.LEFT);
at10= new JTextField(5);
    arm12 = new JLabel("The Robot Arm have Xz= ", SwingConstants.LEFT);
at11= new JTextField(5);
    arm13 = new JLabel("The Robot Arm have Yx= ", SwingConstants.LEFT);
at13= new JTextField(5);
    arm14 = new JLabel("The Robot Arm have Yy= ", SwingConstants.LEFT);
at14= new JTextField(5);
    arm15 = new JLabel("The Robot Arm have Yz= ", SwingConstants.LEFT);
at15= new JTextField(5);
    arm16 = new JLabel("The Robot Arm have Zx= ", SwingConstants.LEFT);
at16= new JTextField(5);
    arm17 = new JLabel("The Robot Arm have Zy= ", SwingConstants.LEFT);
at17= new JTextField(5);
    arm18 = new JLabel("The Robot Arm have Zz= ", SwingConstants.LEFT);
at18= new JTextField(5);

//Listen to events from the Convert button.
connectServer.addActionListener(this);


//Add the widgets to the container.
converterPanel.add(label1);
converterPanel.add(text1);
converterPanel.add(label2);
converterPanel.add(text2);
converterPanel.add(label3);
converterPanel.add(text3);
converterPanel.add(label4);
converterPanel.add(text4);
converterPanel.add(label5);
converterPanel.add(text5);
converterPanel.add(label6);
converterPanel.add(text6);
converterPanel.add(label7);
converterPanel.add(text7);

converterPanel.add(arm1);
converterPanel.add(at1);
converterPanel.add(arm2);
converterPanel.add(at2);
converterPanel.add(arm3);
converterPanel.add(at3);
converterPanel.add(arm4);
converterPanel.add(at4);
converterPanel.add(arm5);
converterPanel.add(at5);
converterPanel.add(arm6);
converterPanel.add(at6);
converterPanel.add(arm7);
converterPanel.add(at7);
converterPanel.add(arm8);
converterPanel.add(at8);
converterPanel.add(arm9);
converterPanel.add(at9);
/*converterPanel.add(arm10);
```

```
        converterPanel.add(at10);
        converterPanel.add(arm11);
        converterPanel.add(at11);
        converterPanel.add(arm12);
        converterPanel.add(at12);
        converterPanel.add(arm13);
        converterPanel.add(at13);
        converterPanel.add(arm14);
        converterPanel.add(at14);

        converterPanel.add(arm15);
        converterPanel.add(at15);
        converterPanel.add(arm16);
        converterPanel.add(at16);
        converterPanel.add(arm17);
        converterPanel.add(at17);
        converterPanel.add(arm18);
        converterPanel.add(at18);
*/

        converterPanel.add(connectServer);


        label1.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label2.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label3.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label4.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label5.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label6.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        label7.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        /*at1.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at2.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at3.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at4.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at5.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at6.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at7.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at8.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at9.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at10.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at11.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at12.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at13.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at14.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at15.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at16.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at17.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
        at18.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
*/

    }

    public void actionPerformed(ActionEvent event) {
        double S1=0,S2=0,S3=0,C1=0,C2=0,C3=0;

        IP = text1.getText();

        u[0] = (double)(Double.parseDouble(text2.getText()));
        u[1] = (double)(Double.parseDouble(text3.getText()));
        u[2] = (double)(Double.parseDouble(text4.getText()));
        u[3] = (double)(Double.parseDouble(text5.getText()));
    u[4] = (double)(Double.parseDouble(text6.getText()));
    u[5] = (double)(Double.parseDouble(text7.getText()));
    S1=Math.sin(u[3]);
    S2=Math.sin(u[4]);
    S3=Math.sin(u[5]);
    C1=Math.cos(u[3]);
    C2=Math.cos(u[4]);
    C3=Math.cos(u[5]);
    d[3]=(C1*C3) + (S1*S2*S3);
    d[4]=C2*S3;
    d[5]=(-S1*C3)+(C1*S2*C3);
    d[6]=(-C1*S3)+(S1*S2*C3);
    d[7]=(C2*C3);
    d[8]=(S1*S3)+(C1*S2*C3);
    d[9]=S1*C2;
    d[10]=-S2;
    d[11]=C1*C2;
    d[0]=u[0];
    d[1]=u[1];
    d[2]=u[2];

    try {
     connect();
    }
    catch (Exception e) { }

 }


 private static void createAndShowGUI()    {
    //Make sure we have nice window decorations.
    JFrame.setDefaultLookAndFeelDecorated(true);
    TCPClientGUI ClientGUI = new TCPClientGUI();

  }
private  void connect() throws Exception{
  Socket s = new Socket(IP, 3000 );
    ObjectOutputStream out = new ObjectOutputStream(s.getOutputStream( ));
    ObjectInputStream  in  = new ObjectInputStream(s.getInputStream( ) );
    out.writeObject( d );
     e2= (double []) in.readObject();
     for (int h=0;h<e2.length;h++)
        System.out.println(e2[h]);
     String s1=Double.toString(e2[0]);
     String s2=Double.toString(e2[1]);
     String s3=Double.toString(e2[2]);
     String s4=Double.toString(e2[3]);
     String s5=Double.toString(e2[4]);
     String s6=Double.toString(e2[5]);
     String s7=Double.toString(e2[6]);
     String s8=Double.toString(e2[7]);
     String s9=Double.toString(e2[8]);
     String s10=Double.toString(e2[9]);
     String s11=Double.toString(e2[10]);
     String s12=Double.toString(e2[11]);
     String s13=Double.toString(e2[12]);
     String s14=Double.toString(e2[13]);
     String s15=Double.toString(e2[14]);
     String s16=Double.toString(e2[15]);
     String s17=Double.toString(e2[16]);
     String s18=Double.toString(e2[17]);
```

```
        String s19=s10+" "+s11+" "+s13+" "+s14+" "+s15+" "+s16+" "+s17+" "+s18;
        at1.setText(s7);
        at2.setText(s8);
        at3.setText(s9);
        at4.setText(s1);
        at5.setText(s2);
        at6.setText(s3);
        at7.setText(s4);
        at8.setText(s5);
        at9.setText(s6);
        /*at10.setText(s10);
        at11.setText(s11);
        at12.setText(s12);
        at13.setText(s13);
        at14.setText(s14);
        at15.setText(s15);
        at16.setText(s16);
        at17.setText(s17);
        */


  }

   public static void main(String[] args) {
      //Schedule a job for the event-dispatching thread:
      //creating and showing this application's GUI.

    javax.swing.SwingUtilities.invokeLater(new Runnable() {
          public void run() {
            createAndShowGUI();
          }
      });

   }
}
```