

- Motion Control of Industrial Robots by 3D Mouse.

Examensarbete 15 högskolepoäng C-nivå

MOTION CONTROL OF INDUSTRIAL ROBOTS BY 3D MOUSE

Reg.kod: Oru-Te-ET3003-D111/08

Marcos Pascual Prieto

Elektroteknik C, Examensarbete, 15 högskolepoäng

Örebro 2008

Supervisor: Ivan Kalaykov
Examiner: Sune Bergelin

Örebro universitet
Institutionen för teknik
701 82 Örebro



Örebro University
Department of Technology
SE-701 82 Örebro, Sweden

- Motion Control of Industrial Robots by 3D Mouse.

Abstract:

Desired motions of industrial robots may be programmed by using 3D force/torque sensors included in an external position control loop. This, however, is a very expensive solution, as 3D force/torque sensors have high prices. Therefore, the goal of the project is to develop a low cost system for driving directly the motion of an industrial robot based on a commercial 3D mouse manufactured by 3DConnexion.

The mouse will be attached to the robot end effector such that a human operator can push/pull/rotate the mouse handle in the desired direction of motion. The objective is to develop an application that simulates the motion of one joint of the robot. This joint will be attached to one fixed point and because of that the translation and rotation movement will be considered the same.

The final application is a Microsoft Visual Studio project. In this application we have 6 different 2D charts. Three of them represent the current position, that is, the position that the robot has at every moment, or the physical location of the 3D mouse. The last three charts represent the vector state, that is, the values of the data that the 3D mouse received at any moment.

- Motion Control of Industrial Robots by 3D Mouse.

Table of contents:

1. Introduction.

1.1 Background.....	5
1.1.1. Lead-Through Programming.....	5
1.1.2. Walk-Through Programming.....	6
1.1.3. Off-Line Programming.....	6
1.1.4. Comparison between methods.....	6
1.2. Goal of the project.....	8
1.3. General requirements and limitations.....	8

2. Feasibility study.

2.1 Goal of the study.....	11
2.2. Technical details of 3D mouse.	
2.2.1. Kinematics.....	13
2.2.2. Electronic circuit.....	15
2.2.3. Communication protocol.....	17
2.2.3.1. Preliminary study with 2D mouse.....	17
2.2.3.2. Conclusion of the preliminary study.....	19
2.2.3.3. Communication Protocol of the 3D mouse.....	20

3. Technical description of the developed system.

3.1. Software.....	24
3.2. Programming language.....	25
3.3 General structure.....	25
3.3.1 Specifications of the project.....	26
3.3.2 Reading the 3D mouse commands.....	27
3.3.3 Representation of the data.....	32
3.3.4 Technical aspects of the representation.....	35
3.3.4.1. Translation vs. Rotation.....	35
3.3.4.2. Handle Vibrations.....	35

- Motion Control of Industrial Robots by 3D Mouse.

4. Experiment with the system.

4.1 Current position.....	36
4.2. Vector state.....	38
4.3. Test scenario.....	40
4.3.1. Purpose of the test.....	40
4.3.2. Hardware requirements.....	40
4.3.3. Software requirements.....	40
4.3.4. Description of how to perform the test.....	40
4.3.5. Expected results or succeed criteria for the test.....	41

5. Conclusions.

5.1. Project evaluation.....	43
5.2. Options for future development.....	44

6. References.....45

- Motion Control of Industrial Robots by 3D Mouse.

1. Introduction.

1.1 Background.

To program the motions of an industrial robot I can choose basically between 3 different techniques: lead-through method, walk through method and off-line programming. All of these three programming techniques are teaching method techniques. To choose the most suitable in our case, it is necessary to get some knowledge about these techniques and also to analyse their respective advantages and disadvantages to decide, then, which one is the best for this project.

Programming by teaching methods:

A program consists in individual command steps which state either the position or function to be performed, along with other data such as speed, delay times, sample input device, activate output device, execute, etc.

When making a robot program, it is necessary to establish a physical or geometrical relationship between the robot and other equipment to be served by the robot. To establish those coordinate points precisely within the robot's working envelope, it is necessary to control the robot manually and to teach physically the coordinate points. To do this and determine other functional programming information, three different teaching or programming techniques are used: lead-through, walk-through and off-line.

1.1.1. Lead-Through Programming:

This is performed manually, using a teaching pendant that will guide the robot in performing a prescribed task.

This method of teaching uses a proprietary teach pendant (the robot's control is placed in a "teach" mode), which allows trained personnel to lead physically the robot through the desired sequence of events by activating the appropriate pendant button or switch. Position data and functional information are taught to the robot, and a new program is written. The teach pendant can be the sole source by which a program is established, or it may be used in conjunction with an additional programming console and the robot's controller. When using this technique of teaching or programming, the person performing the teach function can be within the robot's working envelope, with operational safeguarding devices deactivated or inoperative. [1]

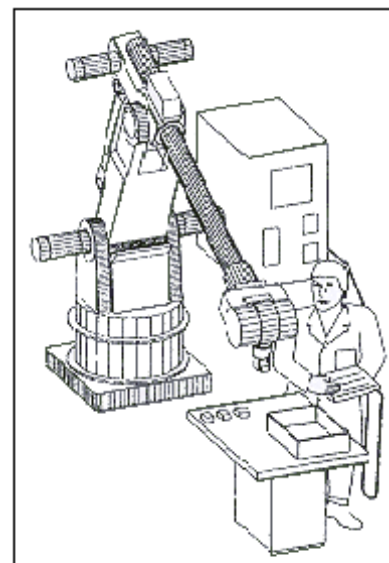


Figure 1

- Motion Control of Industrial Robots by 3D Mouse.

1.1.2. Walk-Through Programming:

This is performed manually when the user actually moves the robot through physical contact.

A person doing the teaching has physical contact with the robot arm and actually gains control and walk the robot's arm through the desired positions within the working envelope.

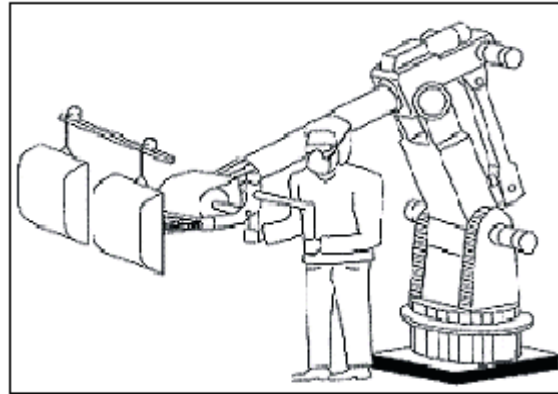


Figure 2

During this time, the robot's controller is scanning and storing coordinate values on a fixed time basis. When the robot is later placed in the automatic mode of operation, these values and other functional information are replayed and the program runs as it was taught. With the walk-through method of programming, the person doing the teaching is in a potentially hazardous position because the operational safeguarding devices are deactivated or inoperative. [1]

1.1.3. Off-Line Programming:

The robot program is generated using a remote computer console and is later transferred to the actual robot controller.

The programming establishing the required sequence of functional and required positional steps is written on a remote computer console. Since the console is distant from the robot and its controller, the written program has to be transferred to the robot's controller and precise positional data established to achieve the actual coordinate information for the robot and other equipment. The program can be transferred directly or by external memory. After the program has been completely transferred to the robot's controller, either the lead-through or walk-through technique can be used to obtain the current positional coordinate information for the robot's axes.

When programming robots with any of the three techniques discussed above, the program is generally required to be verified and slight modifications in positional information to be made. This procedure is called program touch-up and is normally carried out in the teach mode of operation. The teacher manually leads or walks the robot through the programmed steps. Again, there are potential hazards if safeguarding devices are deactivated or inoperative. [1]

- Motion Control of Industrial Robots by 3D Mouse.

1.1.4. Comparison between methods:

Now, we can see a comparison of the programming methods summarized in the table below:

	Lead-Through	Walk-Through	Off-line Programming
Robot Type	Real	Real	Virtual
Work Cell	Real	Real	Virtual
Evaluation	Inflexible	Inflexible	Flexible
Intuitiveness	Low	High	Low
Safety Issue	Yes	Yes	No

As we can see in the table and the description of the programming methods, there are several methods with their respective advantages and disadvantages.

In this project, the method used will be Walk-through. Indeed, in the specification of the project, it was said that the mouse should be attached to the robot arm, and with this specification, the programming method must be Walk-through.

The programming method Lead-through cannot be used to develop this project because in this method, the mouse should not be attached to the robot arm, it cannot be moved with every movement of the robot. Moreover, to handle the robot with this purpose, it already exists a tool called IRC5 Control System.



Figure 3: IRC5 Control System

In conclusion, the Walk-through programming method will be the method used to carry out this project because it is the most suitable option to meet with the specification of the project.

- Motion Control of Industrial Robots by 3D Mouse.

1.2 Goal of the project.

Now that it is clear that the walk-through programming technique is going to be used, I can talk about the project as I understand it.

The idea of the project is to develop a low cost system to drive the robot ABB IRB140, using the commercial 3D mouse SpaceNavigator manufactured by Logitech.

The device, in this case the 3D mouse, has to be attached to the robot, because the programming method used is walk-through.

It means that the human operator who is going to handle the robot has to follow the movement at the same time as he is handling/touching the handle of the mouse.

The simple fact that the mouse is attached to the robot and that it moves with each of its movement, changes everything when you have to programme the movements of the robot arm.

The point is that we have to know the current position of all the 6 joints of the robot. This data is totally necessary because otherwise, if you make a movement and the mouse makes this movement as well, the coordinates of the position of the mouse will change, the following movement will have the wrong coordinates and the movement will not reach the desired position.

For this reason that it is totally necessary to know the current position of the robot and, according to this data, decide which part of the robot has to be moved and in which direction to get the same movement with the mouse's handle.

1.3. General requirements and limitations.

■ The available robot is: ABB IRB-140

This robot is a compact, 6-axes multi-purpose robot. The IRB 140 can carry a maximum of 5kg and a reach of 810mm with 360 degrees rotational, fast acceleration, and a large working envelope. It can be attached to the floor, suspended or mounted on the wall at any angle to enable flexible automation.

In our case, the robot will be attached to the floor as we can see on the figure 4:

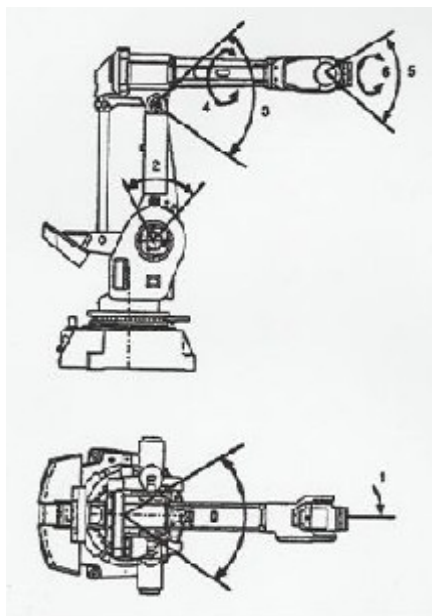


Figure 4: ABB IRB-140 Robot

- Motion Control of Industrial Robots by 3D Mouse.

The 6-axes anthropomorphic robot's arms are significantly more flexible, adaptable and standardised than others. Anthropomorphism comes from the Greek "anthropos" (human) and "morphe" (shape); it represents the fact to attribute human qualities to inanimate or non-human things. As suggested by their name, the arms mimic the movements of a human arm.

We can see on the diagram below that axis 1 and 2 represent the human shoulder, axis 3 and 4 the elbow and forearm, and axis 5 and 6 the wrist. This figure represents pretty well the arm of the IRB-140 robot:



Axis	Function:
Axis 1:	Rotation of the complete robot
Axis 2:	Forward and reverse movement of the lower arm.
Axis 3:	Vertical movement of the upper arm
Axis 4:	Rotation of the complete wrist centre
Axis 5:	Bending of wrist around the wrist centre
Axis 6:	Rotation of mounting flange (turn disc)

Figure 5: Motions of the robot

■ The available mouse: SpaceNavigator.

The mouse chosen to be used in our case is the SpaceNavigator manufactured by Logitech. This is a commercial device that everybody can buy. As it is a commercial device, we can access to the drivers, but the problem is that we cannot edit them.

At this point, the problem is that as the drivers are not editable, we have to develop an application which allows us to do what we need to handle the robot through the 3D mouse.

- Motion Control of Industrial Robots by 3D Mouse.

This is the device that is going to be used, the SpaceNavigator:



Figure 6: SpaceNavigator

The most important of 3D mouse devices is the controller cap. Push, tilt, pull or twist the cap a fraction of millimetre to simultaneously pan, rotate and zoom 3D imagery. Increase the pressure in order to go fast or decrease the pressure to make the appropriate adjustments.

On the picture below, we can see the kinds of movements we can do with the SpaceNavigator:

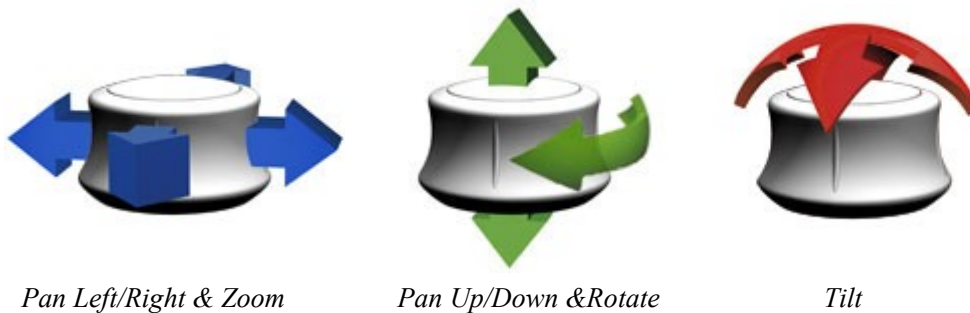


Figure 7: Different movements of the handle.

- Motion Control of Industrial Robots by 3D Mouse.

2. Feasibility study

2.1. Goal of the study.

The 3D mouse has a USB port connection. With this connection, the 3D mouse and the computer communicate with each other. Looking at this, at the end we have to connect directly the 3D mouse with the robot arm and we need to know which codes the mouse sends through the USB port with each movement of the mouse's joystick.

At the very beginning, we had two possible investigation lines. One of them was to disassemble the device and try to understand how it works. The other possibility was to find some software which decodes the data that the 3D mouse sends and try to understand the correspondence between the movements of the mouse and the codes received.

Disassemble the device:

The first solution that I took was the dismantling of the device to see what there was inside with the aim of trying to understand how the 3D mouse works internally. We can say that the device is divided in 3 parts.

1. The first part is composed of the infra-red LED diodes and the sensors which detect the infra-red light. There are six infra-red diodes and just in front of each of them there is a sensor.
2. The second part is the external part that you touch when you are handling the device. This plastic piece is found between the sensors and LED diodes.
3. The third and last part is the electronics. This is composed by two PCB boards. These two parts are connected by a plastic connector and one of them has a connection with the USB connector.

- Motion Control of Industrial Robots by 3D Mouse.

At this point, in order to continue in the same way, the idea was to forget the second PCB board, which has the microcontroller and processes the data of the first PCB board and focus only in the first PCB board.

To do so, the signals would be taken directly from the red plastic connector of the first PCB board and the data would be acquired through an external microcontroller; the latter should be built in an external board with its additional components and then, it should be connected to the USB port.

This option can be possible only if we have the description of the electronic circuits that the mouse is composed. To get this knowledge it was necessary to decode track by track with the multimeter checking the continuity of the electronic circuit. Here is the scheme of the circuit decoded using OrCAD Capture:

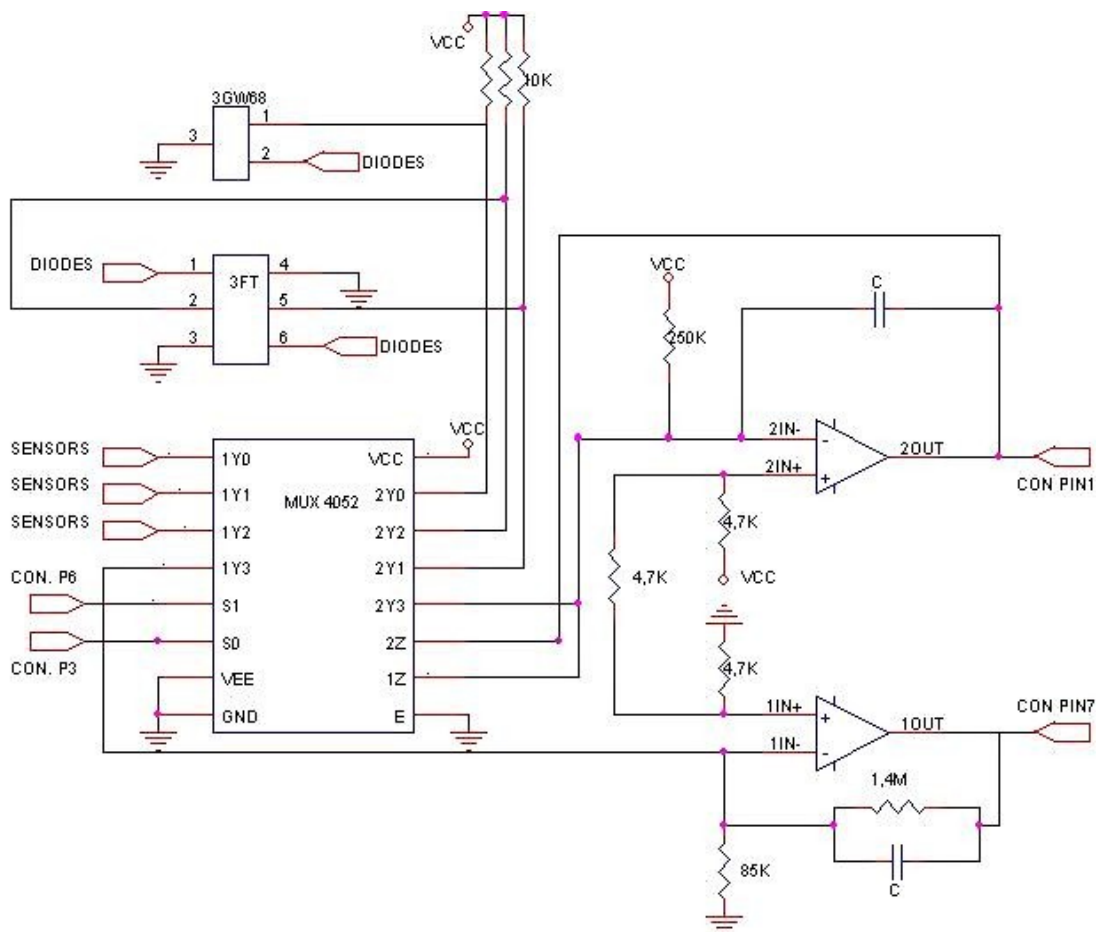


Figure 8: Scheme of the electronic circuit.

- Motion Control of Industrial Robots by 3D Mouse.

This possibility was an interesting solution, but before starting to implement it, I decided to try to investigate the other possible alternative; that is, the use of a USB Traffic Analyser.

Use a USB Traffic Analyser:

The second possibility or different solution for this project was to find some software which analyses the USB port traffic. If we could manage to understand the data that the mouse sends through the USB port with each movement, later we would be able to handle the robot arm according to the movements of the 3D mouse.

The first step was to find the most suitable software which would allow us to see the codes sent. All the programs used were trial versions:

- Advanced USB Port monitor:[6] The trial version of this software gives us the data trend, how many bits per second are sent by the device to the PC and vice versa, and another amount of data that is not useful for the task that we need.
- Device Monitoring Studio 5.2.2:[7] This software is one of the best options because it has exactly what we need to analyse the traffic through the USB port:
Indeed, it has two tools that, when combined, give us the exact information that we need to decode the movements of the 3D mouse.
 1. URB View: USB Request Block. This visualizer decodes raw packets and control requests sent to and received from the USB device.
 2. HID View: This visualizer decodes HID specific packets and data structures and displays them in the comfortable way. It is ideal for devices belonging to Human Interface Device class.

According to the results found and regarding all the possibilities to develop this project, we can conclude that the most suitable solution is to use the Device Monitoring Studio. The use of this software will only provide us with the knowledge of the way the data is sent through the USB port to the computer.

Once we have understood how the mouse sends the data and which byte corresponds to each movement, we can continue to the next step.

- Motion Control of Industrial Robots by 3D Mouse.

2.2. Technical details of 3D mouse.

2.2.1. Kinematics.

The key component to carry out this project is the 3D mouse Space Navigator, developed by Logitech.

In the following section we are going to study and describe the kinematics of this 3D mouse; that is, to disassemble each part of the mouse and see them separately.

- ♦ LED diodes and sensor's part:
Looking at this part, we can see how the mouse translates mechanical movements into electrical pulses that the computer can understand.

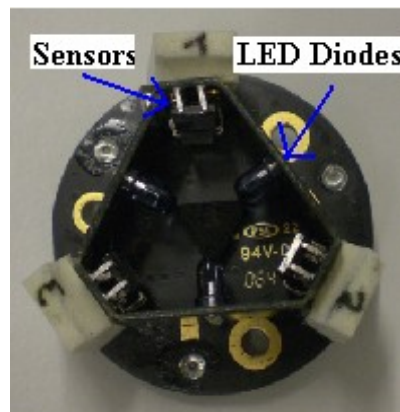


Figure 9: Sensors and LED diodes part

As we can see on the picture above, this part of the mouse:

- ♦ Has a triangular shape.
- ♦ Has a pair of sensors in each corner.
- ♦ Has a pair of LED diodes in the middle of each side of the triangle.

The way this circuit works is the following: each sensor has its respective infra-red LED diode in front of it. When the sensor receives the infra-red light it has a value, and when it does not receive the light it changes.

- Motion Control of Industrial Robots by 3D Mouse.

- Handle part:

This is the part that you can touch when you are handling the mouse. It has a plastic column between each pair of sensors and LED diodes with a thin hole. This hole lets pass the beam of light of each LED diode to the sensor, only if the current position is the initial position.

The three bottom holes have a vertical orientation and the three top holes have a horizontal orientation.

If you move the handle in some direction, the beam of light that the sensor is supposed to detect is going to be cut off and the sensor's value is going to change. With this information we can detect all the directions of movement. There are three for the translation movements and another three for the rotation movements.



Figure 10: Handle part

2.2.2. Electronic circuit.

The electronics of this device is divided in two different parts; that is, two different PCB boards.

- The first PCB board is integrated in the same part as the sensors and the LED diodes. The task of this part is to process the information that the sensors supply. On this PCB, there are two dual 4 channel analogue multiplexers, two operational amplifiers, resistors and capacitors. All of these components have to process the information that goes directly through the plastic connector to the other PCB.

- Motion Control of Industrial Robots by 3D Mouse.

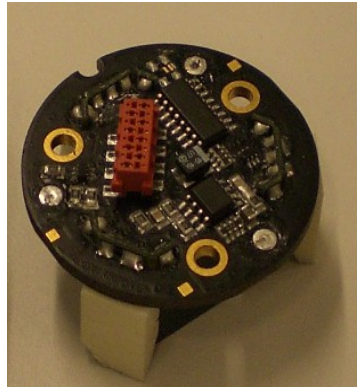


Figure 11

- When the feasibility study was made, the electronics and the circuits of this part had to be translating. This was done with a multimeter checking the continuity of all the tracks of the electronic circuit.
- The second PCB has one 8 bits microcontroller, ST72F621J4T1 by ST Microelectronics, which manages the information supplied through the connector by the sensors, and sends the respective codes and commands through the USB port.
 1. The red connector that we can see on the picture below is the part which connects both PCB boards.
 2. The white connector is the part which connects the second PCB board with the USB connector that is going to make the communication between the mouse and the computer.
 3. The two yellow circles surrounded by a white square that we can see on both sides of the PCB board are the two buttons that the 3D mouse has.

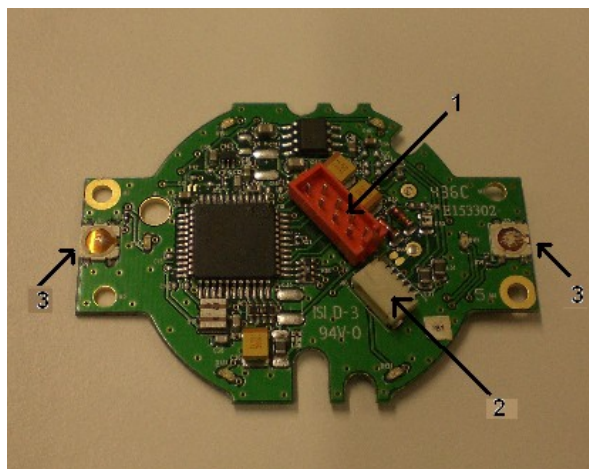


Figure 12

- Motion Control of Industrial Robots by 3D Mouse.

2.2.3. Communication protocol.

At the beginning of this project, one of the most important tasks was to find out how the device communicates with the computer; that is, the communication protocol. It is necessary to know which are the commands or the data that the device is sending with each movement. With this information, later, we can translate the movements of the mouse into orders for the robot.

The option that was taken to carry out this task was the USB traffic analyser. With this tool, we can see and record the stream of bytes that the mouse sends in each moment.

The tool that I have used to analyse the USB traffic is Device Monitoring Studio and I decided to make some previous trials or experiments before starting directly with the 3D mouse. These experiments were made with a 2D mouse.

2.2.3.1. Preliminary study with 2D mouse.

The software that we are going to use, Device Monitoring Studio, has an interface where we can use at the same time two very useful tools: URB View and HID View. Those tools are described above in the section: goal of the study.

The 2D mouse that has been used to make this experiment is provided by Kensington. Each time that you make a movement, the mouse sends a stream of 4 bytes. Each one of these bytes has a different meaning that we are going to see in the explanation below.

<u>1st Byte</u>	<u>2nd Byte</u>	<u>3rd Byte</u>	<u>4th Byte</u>
Left Click Right Click Central Click	Left-Right Movement	Up-Down Movement	Central Wheel Movement Up-Down

★ *First Byte:*

The first byte is dedicated to control the three buttons that the mouse has:

Left Click	Central Click	Right Click
01 00 00 00	04 00 00 00	02 00 00 00

- Motion Control of Industrial Robots by 3D Mouse.

★ **Second Byte:**

The second byte provides us with the information about the left-right movement.

When we move the mouse to the right the second byte is 01, but if we move the mouse fast, the data that will be sent will be 02. This value will increase each time that we move the mouse faster.

If we move the mouse to the left the second byte will be FF, in hexadecimal format, and if we move faster the mouse the value will be reduced according to the speed of the movement.

+ Faster Left Mov.	Faster Left Movement	Left Movement	Right Movement	Faster Right Movement	+Faster Right Mov.
00 FD 00 00	00 FE 00 00	00 FF 00 00	00 01 00 00	00 02 00 00	00 03 00 00

★ **Third Byte:**

This third byte is pretty similar to the second one. The only difference between them is that this one is dedicated to the up-down movement.

+ Faster Up Mov.	Faster Up Movement	Up Movement	Down Movement	Faster Down Movement	+Faster Down Mov.
00 00 FD 00	00 00 FE 00	00 00 FF 00	00 00 01 00	00 00 02 00	00 00 03 00

★ **Fourth Byte:**

The last byte is used for controlling the central wheel. This byte is very simple; it does not change if we move the wheel faster. It only changes of value when we change the direction of the wheel.

Up Movement	Down Movement
00 00 00 01	00 00 00 FF

- Motion Control of Industrial Robots by 3D Mouse.

◆ This is the overview of how the 2D mouse works:

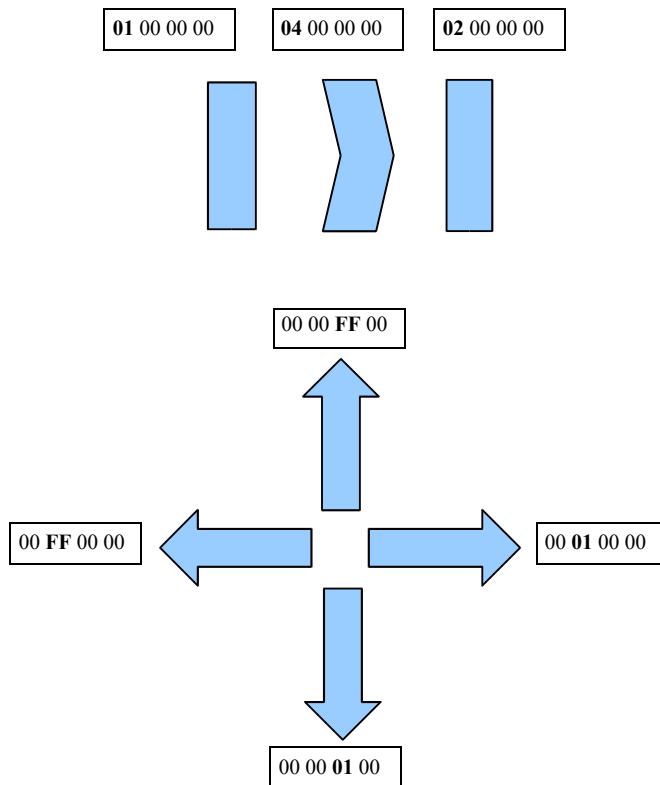


Figure 13: 2D mouse communication protocol

2.2.3.2 Conclusion of the preliminary study.

As we can see in the information presented above, the 2D mouse uses 4 bytes to transmit all the information: the 3 buttons, the left-right movement, the up-down movement and the wheel.

The first conclusion that we can draw is that the information will be organized in the same way for the 3D mouse. We only have to find out which byte corresponds to each movement.

- Motion Control of Industrial Robots by 3D Mouse.

2.2.3.3. Communication Protocol of the 3D mouse.

If we pay attention to the architecture of the mouse, we can divide the possible movement in 2 groups: translation and rotation movement.

We will have the X, Y and Z axes for the translation movement and the Rx, Ry and Rz axes for the rotation movement. That is, 6 different movements which enable us to reach any position of the space.

Using the HID View we can know the value between 500 and -500 of each coordinate. It means that we can assign each movement to its respective coordinate: X, Y, Z, Rx, Ry or Rz.

The URB View gives us the bytes that the mouse sends with each movement of the handle.

The stream of bytes sent has the following characteristics:

- *If you do not touch the handle of the mouse, any data is sent through the USB port.*
- *If you are touching the handle of the mouse, a stream of bytes is sent **each 15 milliseconds**.*
- *Every stream of bytes has **7 bytes**.*
- *The first byte is used to distinguish the rotation movement from the translation movement*
 1. *If the first byte is **01**, this stream of bytes gives the information about the **translation**.*
 2. *If the first byte is **02**, this stream of bytes gives the information about the **rotation**.*

1. **Translation Movement:**

When a translation movement byte is sent, the first byte is always 01. We have 6 other bytes; 3 of them are used to count the number of degrees, and the 3 other ones are used as overflow bytes because with one byte only, we cannot measure more than 256 positive degrees.

These 3 overflow bytes give us the possibility to measure positions bigger than 256 degrees, as well as the opportunity to measure negative positions.

- Motion Control of Industrial Robots by 3D Mouse.

The overflow bytes can have these different values and each one has the following meaning:

VALUE	MEANING
00	Positive value No overflow
01	Positive value Overflow.
FF	Negative value No overflow
FE	Negative value Overflow

In the table below, we can see the connection between the bytes and the axes:

1 st Byte	2 nd Byte	3 rd Byte	4 th Byte	5 th Byte	6 th Byte	7 th Byte
Translation Byte→01	X- Counter	X- Overflow	Y- Counter	Y- Overflow	Z- Counter	Z- Overflow

To understand better how the overflow works we can look at the example below:

X-Counter	X-Overflow	Number of degrees
A5	00	A5h=165 positive degrees.
06	01	6h+256=262 positive degrees
4B	FF	4Bh=75 → 256-75=181 negative degrees
BE	FE	BEh=190 → (256-190)=66 → 66+256=322 negative degrees

- Motion Control of Industrial Robots by 3D Mouse.

This graphical explanation shows us which byte corresponds to each movement. We can see a box next to each arrow.

- This box always has in the first position the number 01 because this is the translation movement.
- If you see dots, it is because these bytes don't correspond to this movement.
- The XX represents the value in degrees that we have at any moment.
- The FF or the 00 value give us the direction of the axis in which we are.

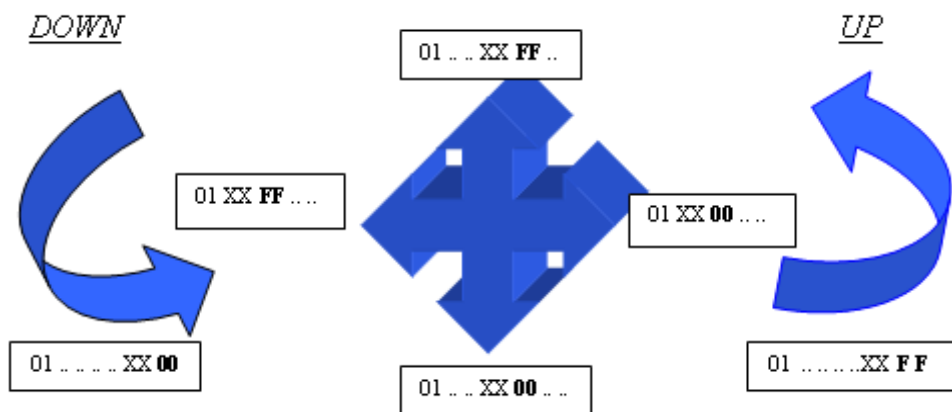


Figure 14: Schematics of the translation movement.

2. Rotation Movement:

The rotation movement byte is exactly the same as the translation movement byte. The only difference is that the first byte, in this case, is 02.

In the following table, we can see the correspondence:

1 st Byte	2 nd Byte	3 rd Byte	4 th Byte	5 th Byte	6 th Byte	7 th Byte
Rotation Byte→02	Rx- Counter	Rx- Overflow	Ry- Counter	Ry- Overflow	Rz- Counter	Rz- Overflow

- Motion Control of Industrial Robots by 3D Mouse.

This drawing represents the rotational movement and follows the same rules as the previous one:

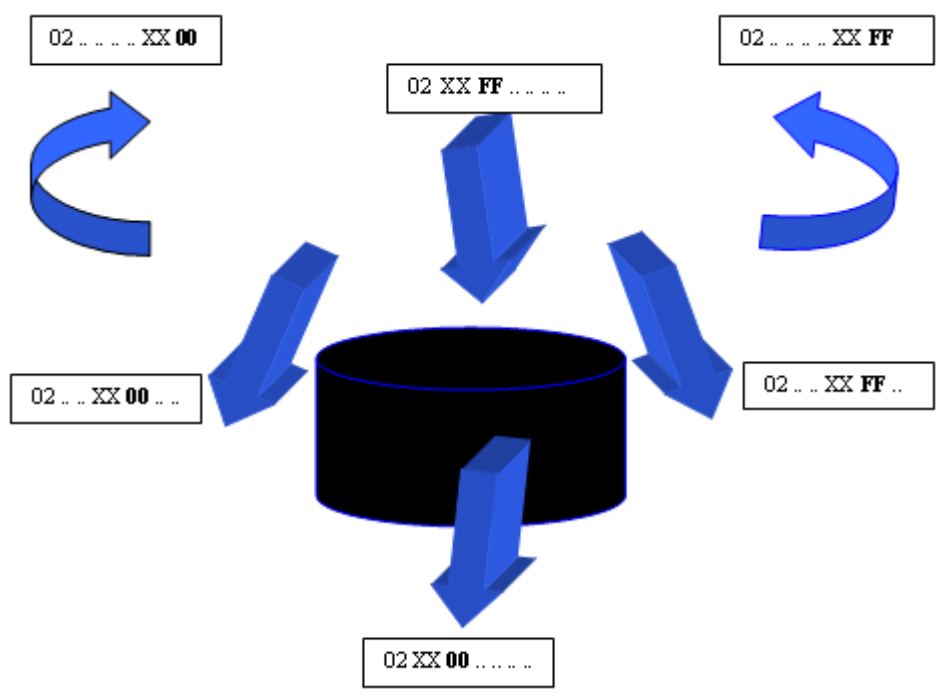


Figure 15: Schematics of the rotation movement.

- Motion Control of Industrial Robots by 3D Mouse.

3. Technical description of the developed system

At this point, we know everything about the electronics of the internal circuits that compose the 3D mouse, the communication protocol that the mouse uses when it sends the commands concerning the movements of the handle, and the robot to which the mouse will be attached, with all the possible movements and rotations of all its joints.

It is now the moment to develop the real application that we are going to use to handle the arm of the robot.

To carry out this task, we had to choose the most suitable software and programming language:

3.1. Software.

The software used to develop the project depends a lot on the programming language chosen. There were two candidates: Eclipse IDE for Java Developers if we work in Java and Microsoft Visual Studio if we work in C++.

1. Eclipse is the software platform most used if we use Java. This program has comprising application frameworks and runtime libraries. This platform also contains an integrated development environment (IDE). But the main feature that we can mention about Eclipse is that it is open source software, and the most important is that it is free, and everybody can use it.
2. Microsoft Visual Studio: This software does not only support C++ as programming language, with it, we can also use several languages. It is the main integrated development environment that Microsoft can offer us. The software can be used to develop graphical user interface and console applications. At this point we can say that this is the best option if we want to develop the project in C++.

The available version of this program that we have in the laboratory is Microsoft Visual Studio .NET 2003.

- Motion Control of Industrial Robots by 3D Mouse.

3.2. Programming language.

Looking at the programming languages there were only two real possible candidates: Java and C++.

3. To read the data sent through the USB port in Java, there are a lot of possibilities. In our case, the program has to be developed under Windows and not under Linux. If we use this operative system, there are hundreds of examples and projects that we can consult if we have any doubt. The main drawback when using Java as programming language is that it is difficult to develop any application if we do not work under Linux. However I found a project called Java USB API for Windows that is part of the open source project jUSB. Once all these information found, we could start to work with Java.
4. The other option was C++. This is a widespread programming language in which I could find a lot examples and advice from other colleagues when I was stuck at one point. Moreover, I have much more skills in C++ than in Java.

I evaluated all the possibilities explained above and I decided that the best idea was to use C++ as programming language and Microsoft Visual Studio .NET 2003 as software to develop this project.

3.3. General structure.

The application that I have made to carry out this project has two different parts. The first is the part where the data of the 3D mouse 3DConnexion of Logitech is read. To do that, the Logitech web page was very useful because I could find a lot of information and examples in the SDK (Software Development Kit) part.

The other part is the one where the data read of the mouse is represented in 2D graphs. To do that, I found some help searching for useful information on Google as well as some examples that I could use as a template to develop the application with the features that I needed.

- Motion Control of Industrial Robots by 3D Mouse.

3.3.1 Specifications of the project.

The aim of this project is supposed to handle a robot arm using the programming teaching method Walk-through. It means that the 3D mouse will be attached at the end of the robot arm. The programming teaching method chosen will change the way to program the application and the commands sent to the robot.

In this case, we have the robot ABB IRB-140 manufactured by ABB that we can see on the picture:

- This robot is a 6 axes robot as it is described in the respective previous section.
- The point is that this project will not develop the final application to handle the 6 axes robot arm.
- This project is a previous study where we will get the knowledge needed to drive directly the motion of this ABB robot based on a commercial 3D mouse.
- The mouse will be attached to the robot, allowing that a human operator can pull/push/rotate the mouse.
- As the mouse will be attached to the robot, the programming method used will be walk-through.



Figure 16: ABB IRB-140 Robot

The main features of the project are:

1. The robot has one maximum of 6 axes/joints that we can drive.
2. The study that is going to be done will be with one joint.
3. This joint will be attached, and it will have a fixed point that it will not be possible to be moved.
4. Regarding the previous specifications, to have a mouse with 6 different coordinates that can be moved in 6 different direction of the space is not totally necessary.
5. For that reason the translation and rotation movement will be considerate the same.

- Motion Control of Industrial Robots by 3D Mouse.

3.3.2 Reading the 3D mouse commands.

Reading the commands that the 3D mouse sends with each movement of the handle was one of the most difficult tasks of this project. It was also the part that took me the most time and for which I had to research more. The first attempts consisted in trying to use Java but finally, with the help found in the 3DConnexion web page, I could find some C++ functions that let me read the data of the mouse and I could convert this data into the respective coordinates of the space: three coordinates for the translation movement and another three for the coordinates of the rotation movement.

I found the most significant information by searching in the 3DxInput API. [5]

API is the abbreviation of Application Program Interface, a set of routines, protocols, and tools used to built software applications. A good API makes the development of a program easier by providing all the building blocks. Then, a programmer puts the blocks together.

Most operating environments, such as MS-Windows, provide an API so that programmers can write applications that are consistent with the operating environment. Although APIs are designed for programmers, they are ultimately good for users because they guarantee that all programs using a common API will have similar interfaces. This makes the learning of new programs easier for users. [2]

On the previous API, 3DxInput, we can find all the information that we need about the 3D mouse and the way it works. But there were only a few functions that were useful in the final application. In this paper, I am going to talk only about two of them only. The one that gets the translation data and the one that gets the rotation data:







I. MathFrameTranslation:

As we will see in the following source code of the C++ function, this function is a very simple function that just stores the value of the current coordinates in one variable.

```
void MathFrameTranslation(MathFrame*Frame, real X, real Y, realZ)
{
    Frame->MathTranslation[0] = X;
    Frame->MathTranslation[1] = Y;
    Frame->MathTranslation[2] = Z;
}
```

- The table below displays which translation movement corresponds to each coordinate.

- Motion Control of Industrial Robots by 3D Mouse.

TRANSLATION Handle Movement.	Value of respective variable.
	<p>Positive X-Axis 0 to 1</p>
	<p>Negative X-Axis -1 to 0</p>
	<p>Positive Y-Axis 0 to 1</p>
	<p>Negative Y-Axis -1 to 0</p>
	<p>Positive Z-Axis 0 to 1</p>
	<p>Negative Z-Axis -1 to 0</p>

- Motion Control of Industrial Robots by 3D Mouse.
 - The three coordinates X, Y and Z can have positive and negative values. The X axis has its maximum value in 1 and the minimum in -1.
 - If we release the handle of the mouse the value is 0.
 - If we push the handle in the positive X axis direction the value of this variable will increase.
 - If we reach the maximum position of the joystick the value of the variable will be 1.
 - The following data is the information that we can find in the 3DxInput API:

Sensor: Translation

Description

This property returns the translation component of the sensor data. This is a read only property.

Syntax (COM)

result = ISensorPtr ->get_Translation(&IVector3DPtr)

Property: IVector3DPtr Pointer to an Vector3D component interface.

Return: (HRESULT) result S_OK if successful

II. MathFrameRotation:

This function is pretty similar to the previous one. Its role is to get the data of the rotational movements. It is almost the same as the translation function; the only difference is that the rotation function makes some calculation.

- First, it gets the three coordinates and calculates the sine and cosine of those values.
- Then the function performs the rotation value with the suitable mathematical operations.

- Motion Control of Industrial Robots by 3D Mouse.

We can see the implementation of the function below:

```
void MathFrameRotation(MathFrame*Frame,realRoll,real Pitch, real Yaw)
{
    real sa,ca, sb,cb, sc,cc;
    real sacc, cacc, sasc, cacc;

    sa = (real)-sin(Yaw);          ca = (real) cos(Yaw);
    sb = (real)-sin(Pitch);        cb = (real) cos(Pitch);
    sc = (real) sin(Roll);         cc = (real) cos(Roll);

    Frame->MathRotation[0][0] = ca*cb;
    Frame->MathRotation[1][0] = -sa*cb;
    Frame->MathRotation[2][0] = -sb;

    sacc = sa*cc;
    cacc = ca*cc;
    sasc = sa*sc;
    cacc = ca*cc;

    Frame->MathRotation[0][1] = sacc - cacc*sb;
    Frame->MathRotation[1][1] = cacc + sacc*sb;
    Frame->MathRotation[2][1] = -cb*sc;







    Frame->MathRotation[0][2] = sasc + cacc*sb;
    Frame->MathRotation[1][2] = cacc - sacc*sb;
    Frame->MathRotation[2][2] = cb*cc;

}
```

- In the following table, we find the same information as we have in the translation movement table:

The rotation movement follows the same rules as the translation movement.

- Motion Control of Industrial Robots by 3D Mouse.

ROTATION Handle Movement.	Value of respective coordinate.
	<p align="center"> Positive X-Axis 0 to 1 </p>
	<p align="center"> Negative X-Axis -1 to 0 </p>
	<p align="center"> Positive Y-Axis 0 to 1 </p>
	<p align="center"> Negative Y-Axis -1 to 0 </p>
	<p align="center"> Positive Z-Axis 0 to 1 </p>
	<p align="center"> Negative Z-Axis -1 to 0 </p>

- Motion Control of Industrial Robots by 3D Mouse.

- The next data is the information that we can find in the 3DxInput API about this function:

Sensor: Rotation

Description

This property returns the rotation component of the sensor data. This is a read only property.

Syntax (COM)

result = ISensorPtr->get_Rotation(&IAngleAxisPtr)

Property: IAngleAxisPtr Pointer to an AngleAxis component interface

Return: (HRESULT) result S_OK if successful

- Now that we know which functions are used to read the USB port, the way these functions work and which values we are going to get, it is time to draw some conclusions.
 - There are two functions, which can be positive or negative, that give us the value of the three coordinates of the space for the translation movement and three values for the coordinates of the rotation movement.
 - We know thanks to the USB traffic analyzer that I used at the beginning of this project, the 3D mouse sends the data of the current position of the handle each 15 milliseconds.
 - In the next part of the project, we will use those functions to get the current position of the joystick at any time. These functions will be combined with the other function necessities to reach the purpose of this project.

3.3.3 Representation of the data.

Now that we have the values of the coordinates, we can continue with the second part of the project; that is, the representation of this data.

The way I am going to do this is in 2D graphs. More specifically I am going to do three charts where the three possible combinations of coordinates will be shown.

Firstly, I am going to explain how the charts are made. The features and the main differences between the two types of charts that compose the application are also going to be explained.

The final application uses Microsoft Foundation Class Library, MFC, which is a library that wraps portions of the Windows API in C++ classes, including functionality that enables them to use a default application framework. Classes are defined for many of the handle-managed Windows objects and also for predefined windows and common controls. There are MFC classes for using GUI elements, accessing databases, handling Windows messages from other applications, and dealing with keyboard/mouse input. [3]

- Motion Control of Industrial Robots by 3D Mouse.

The main class used in this project is the class used to build a chart: CChart.

I. CChart:

This is the only special class that is used in this project. It is derived from the CWnd class. This class provides us with the functionality of plotting a 2D graphs in the principal window of the application.

The CChart class has a lot of functions that give us the possibility to make the graph as we need. First of all, it is necessary to call these three functions that will do the following role:

- InitialUpdate(); //This function enables us to create and customize the chart.
- OnTimer(); // The data that we want to plot is put on this function.
- OnPrint(CDC *pDC); // This function is used to print on the chart.

— To customize the chart several functions have to be used. Some of the most important are going to be explained below.

1. To set the axes label we can call this function:

```
SetAxisLabel(Cstring strLabelX , Cstring strLabelY)
```

2. The function that has to be used to set the chart title is:

```
SetChartTitle(Cstring str)
```

3. The range of each axis can be fixed calling this function. In the example below the range is 100 like in the charts of the application.

```
SetRange(double Xmin, double Xmax, double Ymin, double Ymax)
```

```
SetRange(-100,100,-100,100)
```

4. To set the style of the axes we have to call this function. The correspondence can be seen below. All the graphs in this application will be with the style 4 quadrant.

```
SetAxisStyle(int nStyle)//0: Single Quadrant
```

```
//1: Double Quadrant
```

```
//2: 4 Quadrant
```

5. Using this function we can set the grid scale and the label that will be plotted on screen.

```
SetGridNumber(int nGridX , int nGridY)
```

- Motion Control of Industrial Robots by 3D Mouse.

6. We can also personalize the chart choosing the colours that we want.

```
m_BGColor = RGB(0,0,0,)           // Set background colour to black
m_AxisColor = RGB(255,0,0);       // Set axis colour to red
m_GridColor = RGB(120,120,120);   // Set grid colour to grey.
```

7. All the other functions were used to customize the chart but the most important function in the application is the following:

```
BOOL SetXYValue(double x, double y, int index, int iSerieIdx );
```

- This function is called each time that a command is received through the USB port; that is, if we push the handle in any direction and do not release it, each 15 milliseconds.
- The function is included in the function OnTimer(), that is called each 15 milliseconds if the handle is not released.
- To get the aspect of one vector, that is a line, the SetXYValue function is called 10 times. Otherwise, only a dot will be represented in the chart. Calling several times the function allows us to draw a line. Moreover, this line will be at the desired position and with the desired length.

- Now that we know all the functions available to customize our charts, we can look at the following picture, which is an example of the charts that we can find in the application.

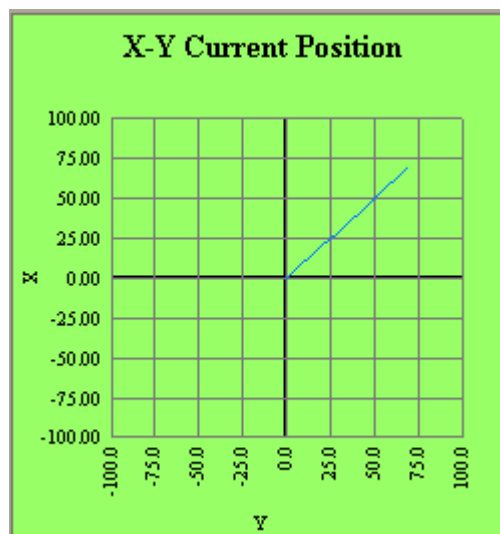


Figure 17: Aspect of the charts of the final application.

- Motion Control of Industrial Robots by 3D Mouse.

3.3.4 Technical aspects of the representation.

At this point, we know how to get the data of the mouse, how to plot a chart and how to plot the values of the coordinates in this chart; it is now time to speak about some technical aspects that must be taken into consideration.

3.3.4.1. Translation vs. Rotation:

As we already know, in this project the translation and rotation movements are the same. It means that if we make a positive X axis translation movement and if we make a positive X axis rotation movement, we are going to obtain the same result.

? But what happens when the translation and rotation movements of the same axis do not have the same sign?

- In the program of the application there are 6 variables to store the current values of the coordinates. Three of them are for the translation movement and the other three for the rotation movement.
- These variables can have a value from -1 to 1. To have the possibility to distinguish between translation and rotation movement this is what I did.
- Before sending the value of the variables to the function that plots them, the two values are added. The final result will be the one which is going to be sent to be plotted.
- If the result of the addition is more than 1 or -1, the value that is sent to be plotted is reduced to 1 or -1 respectively.
- If the value of the translation X-axis is 1, which is the maximum; and the rotation X-axis is -0.75, the length of the vector plotted will be a 25% of the maximum length.

3.3.4.2. Handle Vibrations:

A problem that I could notice when I was implementing the application was that when you are pushing the handle of the mouse and then you release it, there are small vibrations that enable the mouse to send commands. These commands move the arrow of the vector whereas it was not the order given.

The solution taken to avoid this problem was the following:

1. All the values for which the absolute value does not reach more than 5% of the maximum value will be treated as zero.
2. This solves all the problems caused by the vibrations as the values received because of these vibrations never reach values bigger than the 5% of the maximum.
3. An error of 5% of the total is acceptable. This error does not affect the use of the final application and it does not affect the reliability of the application either.

- Motion Control of Industrial Robots by 3D Mouse.

4. Experiment with the system:

In the main window of the application of the project there are 6 different charts. These charts are divided in two groups. Both groups have 3 charts. Each chart corresponds to each pair coordinates represented by: X-Z, X-Y and Y-Z.

4.1 Current position:

The aim of this project is not to simulate the real movement of the ABB robot with its 6 joints. This project is just an approximation, where a low cost system for driving the movements of the robot will be tested.

The real aim of the project is to handle one joint with the low cost driving system that in this case, is the 3D mouse.

The specifications of the project say that the mouse must be attached to the robot. This leads to the fact that the programming method used has to be Walk-through.

- The most important aspect that we should keep in mind when the walk-through method is used is that we have two different points of coordinates, one point of coordinates for the robot and another one for the mouse.
- The point is that when the mouse is attached to the robot and we move its handle, the coordinates of the mouse will change.
- When we move the arm of the robot, the mouse is moving with it, and the coordinates of the mouse change with each movement of the robot.
- For this reason, it is totally necessary to know all the time the current position of the mouse. With this information we will be able to send the correct commands to the robot.

This type of chart, called in the application 'current position', shows the position that the mouse has at any moment when the handle is moving.

- There is an arrow that starts at the position 0, 0. At the beginning, if we do not push the mouse the arrow has length 0.
- If we push the mouse, the arrow will be bigger in the direction pushed.
- If we release the handle, the arrow will not return to the initial position. This would be the natural result because when the joystick is released, the commands received are 0, which means that the arrow has length 0.
- This is the main feature of this type of chart; when the handle is released, the arrow stays at the same position as before.

- Motion Control of Industrial Robots by 3D Mouse.
 - This means that we know all the time the current position of the mouse.
 - The charts of current position are made with a maximum value of 100 and a minimum value of -100.
 - This simulates the real action of the robot. Because a real robot has a maximum position, when you reach this position the joint of the robot cannot continue moving.
 - When the maximum position is reached, the length of the vector does not increase anymore.

An example of the charts explained above is the following:

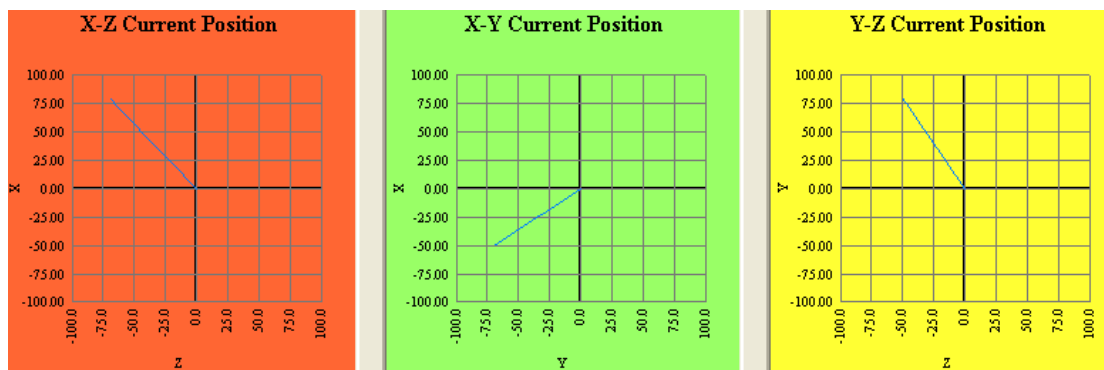


Figure 18: Current Position charts.

Looking at these three charts, we can see:

- The three coordinates X, Y and Z are represented in two of the three charts.
- In the first and the second charts, the X coordinate is in the horizontal axis. We can see that the value of the X-axis is the same in both charts. The only difference is that in the first one, the vertical axis is the Z coordinate and it is positive; and in the second chart the vertical axis is the Y coordinate and it is negative.
- With all these information, the current position can be reconstructed and know at every moment the position of the end of the robot and the position of the 3D mouse that we are handling.

- Motion Control of Industrial Robots by 3D Mouse.

4.2. Vector state:

The other type of chart that there is in the application is called vector state. The charts are pretty similar to the current position charts that are described above.

- In this type of charts the state of the vector, or the state/value of the coordinates of the mouse is/are represented.
- That is, the axis range of these charts is from -1 to 1. This is the range of value that the coordinates received through the USB port can have.
- The pairs of coordinates are the same as the type seen before; each current position chart has its respective vector state chart just below it.

On the following image we can see the vector state's charts:

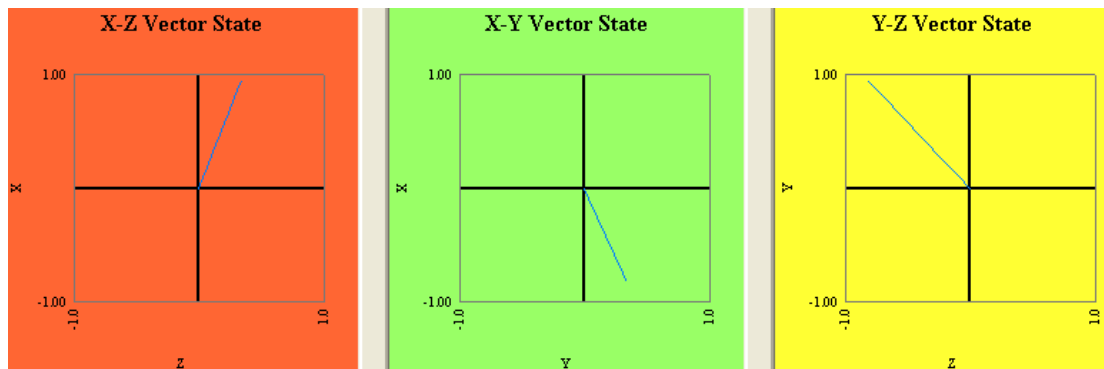


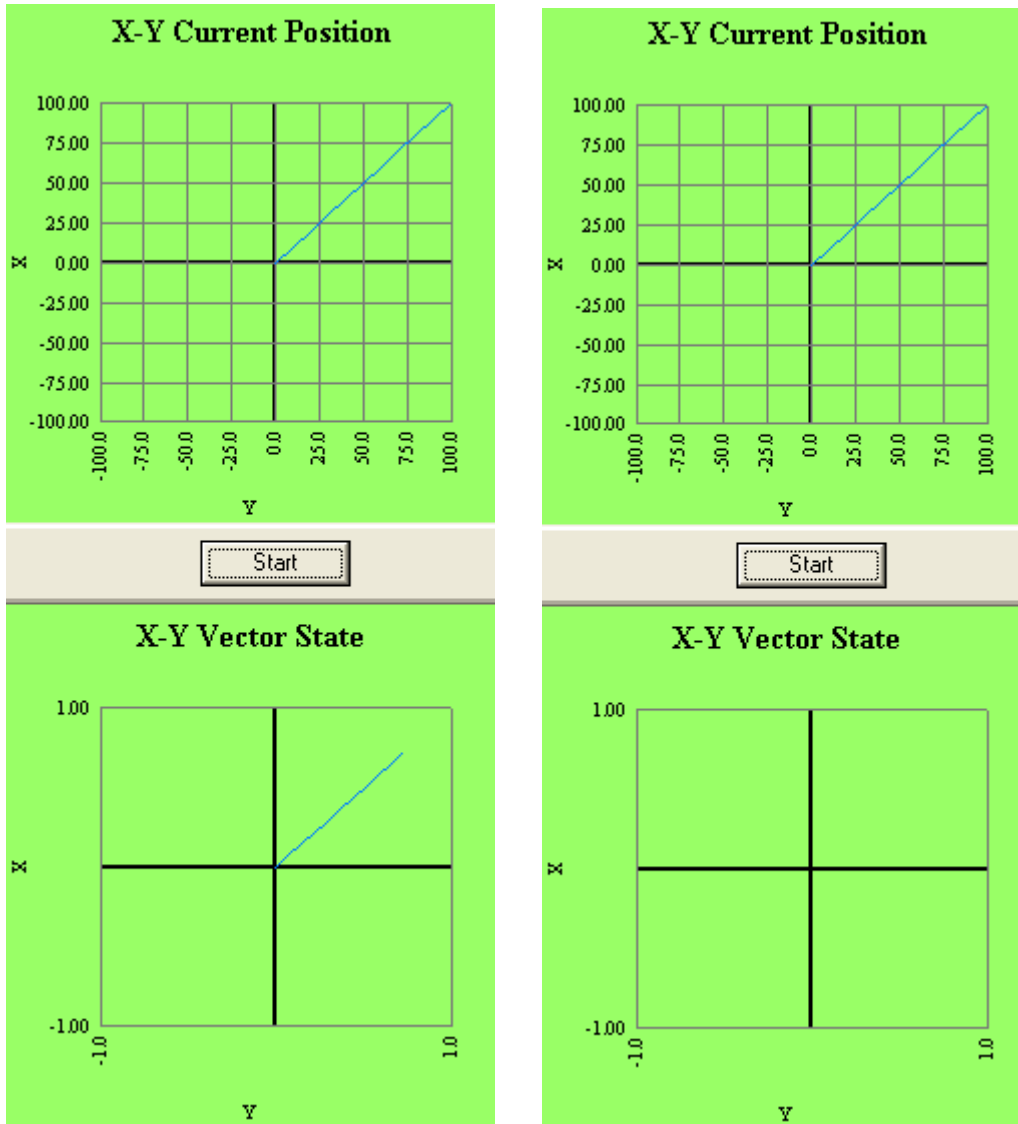
Figure 19: Vector State charts.

As we can see on the picture this type of chart works in a pretty similar way as the current position chart. The next lines give a brief description of its main features:

1. When the handle is pushed until the maximum position that we can reach, the arrow of the vector will have a length of 1, which is the maximum value.
2. If the handle of the mouse is pushed until half of the maximum position that the mouse can reach, the arrow of the vector that is represented will reach a length of 0.5 on the chart.
3. If the handle is pushed, the vector is represented on the chart and then when the handle is released, the vector will decrease until 0.
4. This is the main difference between these charts. This type, that is to say when the handle is released, does not keep the arrow on the screen. It gives us the information of the commands that the mouse sends at every moment.
5. This type of chart represents the information of the data that the mouse sends at every moment. In conclusion, the vector state charts represent the coordinates of the handle of the mouse, and the current position charts represent the coordinates of the end of the robot, or in other words, the physical position of the mouse.

- Motion Control of Industrial Robots by 3D Mouse.

On the next images we can see the differences between both types of charts:



Pushing the mouse

Releasing the mouse

Figure 20: Comparison between types of charts.

The first image shows the state of the charts when the handle is pushed. When the handle is released, we can see the difference between both types. In the current position chart the vector stays in its last position and in the vector state chart the vector returns to the initial 0 position.

- Motion Control of Industrial Robots by 3D Mouse.

4.3. Test scenario.

Scenario testing is a software testing activity that uses scenario tests, or simply scenarios, which are based on a hypothetical story to help a person to think through a complex problem or system. They can be as simple as a diagram for a testing environment or they could be a description written in prose [4]

These are the steps that a test scenario is composed of:

4.3.1. Purpose of the test:

The purpose of the test is to make a practical demonstration of what is explained in all this paper. This paper explains in detail all the steps to follow to reach this point; that is the test scenario.

This test is the practical part of the project, where the results of all this work will be tested.

4.3.2. Hardware requirements:

The test scenario does not have any special hardware requirements. The principal device used to make the test is the 3D mouse 3DConnexion manufactured by Logitech. Moreover, a personal computer or laptop with Windows operative system is needed. This test will not succeed if our application is running under Linux.

4.3.3. Software requirements:

The software requirements to make this test scenario are very simple. We need the software Microsoft Visual Studio .NET 2003. Inside this software, the project is a C++ project MFC application. The Microsoft Visual Studio version recommended is the previous one, but a newer version of the one used in this project is also valid.

The drivers of the 3D mouse are also required. These drivers are given by Logitech. They are not editable and we need to have them installed on the computer under the test that is going to run.

4.3.4. Description of how to perform the test:

1. The first thing that has to be done is open the project called 3DMouse with the designed software Microsoft Visual Studio .NET 2003.
2. The second step is built the project and on the Debug tab click Start.

- Motion Control of Industrial Robots by 3D Mouse.

3. In this moment the principal window of the application will be appear on the computer, in mode full screen.

4. The main window has 6 charts and 2 buttons. The buttons are Start and Restart.



To start with the test the start button must be clicked.



As we know in the current position charts, when we release the mouse the vector stays on the last position. If we want to restart the position of the vector and return it back to the initial 0 position, the Restart button must be pushed.

5. When the Start button is pushed the real test starts. In this moment is when we can move the handle of the 3D mouse in all the possible directions of the space and see the results on the screen.

4.3.5. Expected results or succeed criteria for the test:

The final result obtained is pretty satisfactory. This test scenario gave us the possibility to move the handle of the mouse and to see in real time how the vectors of the pairs of coordinates move on the charts.

This image shows the aspect of the final application with the 6 charts:

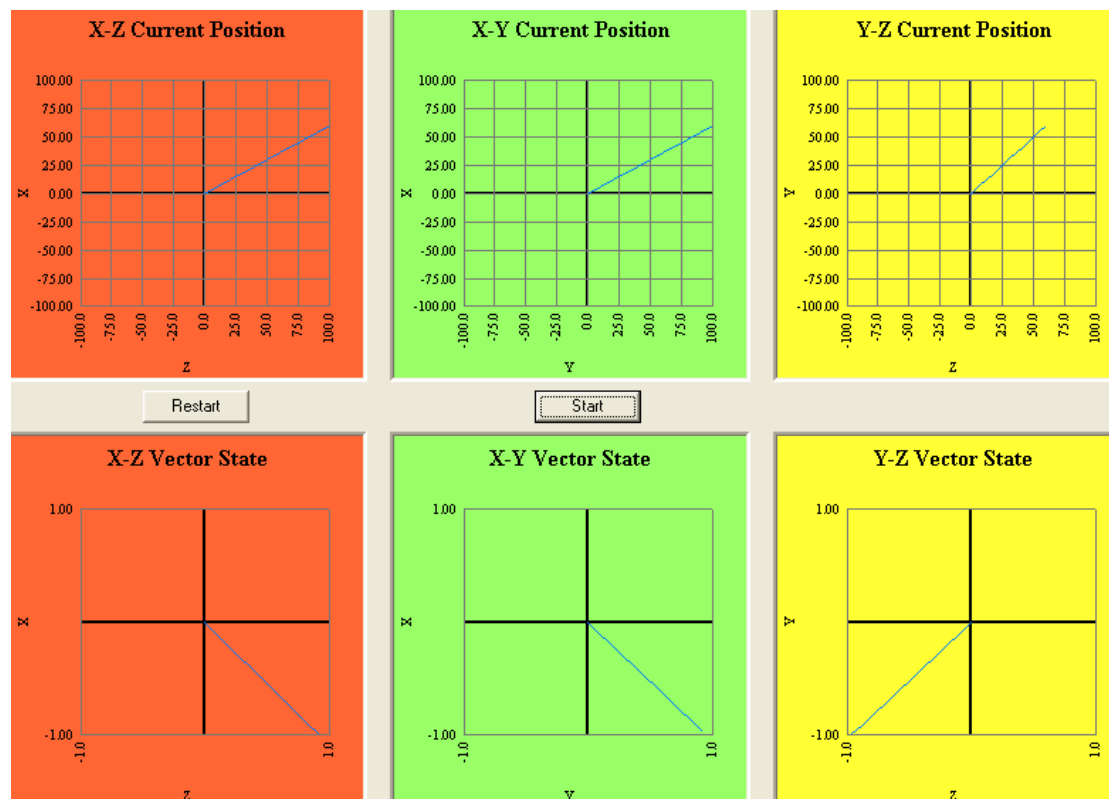


Figure 21: View of the final application.

- Motion Control of Industrial Robots by 3D Mouse.

Another very important feature that the application has is the possibility to see the values of the 6 coordinates, translation and rotation, on the output window. In this way the graphical results can be contrasted with the numerical value of all the coordinates.

The following image shows the output window:

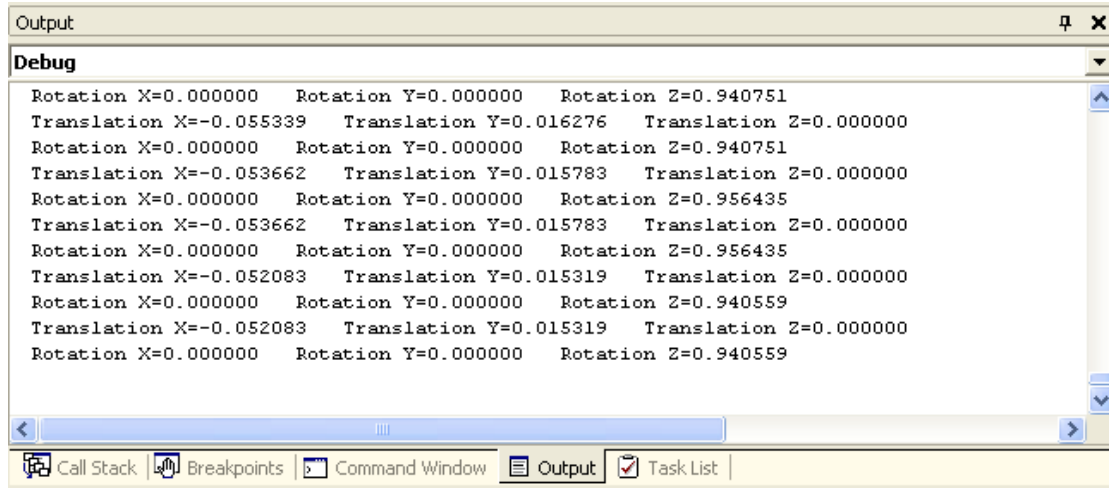


Figure 22: Image of the output window.

- Motion Control of Industrial Robots by 3D Mouse.

5. Conclusions.

5.1. Project evaluation.

Now that the project is almost finished, some very interesting and useful conclusions can be drawn.

1. At the very beginning of the project a feasibility study was made to set which method was the best to carry out the task of reading the data through the USB port.

A 3D mouse was disassembled and its electrical components and circuits studied in detail. Finally, the option of reading the data using a USB traffic analyzer was taken as the best option. Making this feasibility study, now we can say which option is the best and why.

2. Another very important point that we have executed is to transcribe the communication protocol that the 3D mouse uses. Thanks to the USB traffic analyzer we know the meaning of the codes received with each movement that is made with the handle of the mouse.
3. The available robot ABB IRB-140 has six joints and the available mouse has six possible directions. But the project I have developed is to handle one joint. This is for these reasons that:
 - The programming method used is Walk-through and not Lead-through. This makes the programming totally different because it is necessary to know the current position of the end of the robot, or the physical position of the mouse at every moment.
 - If only one joint is developed, most of the work is done because when we know how a joint works, to combine the rest of the joints in one application is not a very complicated work.
4. The final application already available has a lot of advantages. In my opinion, if you need to handle a robot with a 3D mouse, if you have an application where the movements of the joystick and their values are monitoring and we can see them on the screen of the computer it is a very useful and essential tool.
5. As a final thing, in this project there are two types of charts, current position charts and vector state charts. These are the main conclusions that these charts give us:
 - The current position charts tell us the current position of the end of the robot, and in other words, the physical position of the mouse. As the programming method used is Walk-through, we need to know this information at every mo-

- Motion Control of Industrial Robots by 3D Mouse.

ment. The final application gives us this information monitoring on a 2D chart.

- The vector state charts give us the coordinates of the handle of the mouse. With it, we know the position of the handle at every moment. This information is also very important because these coordinates will be the commands sent to the robot to move its arm.
- The information of the current position charts will give us the data needed to modify the information of the other type of charts according to the position of the robot's arm.

5.2. Options for future development.

- The primitive aim of this project was to develop a low cost system for driving directly the robot with the tool chosen, in this case the 3D mouse, attached to the robot.
- At the beginning, I did not know anything about the way I could manage the mouse. Now that the feasibility test has been done, these results could be used in other similar applications that would need the 3D mouse used in this project.
- This project handles one possible joint out of six possible. To extend the number of joints used is not very complicated. Now that we know how one joint works, we can implement the use of the six joints of the robot developing a not very complex work.
- An application that monitors the position of the robot's arm and the coordinates of the handle of the mouse has been done. This is a very useful tool to continue developing the handling of the six joints of the robot because the position of the robot and the position of the joystick are shown in real time on the screen. If the use of the six joints is added to the application, this part does not need to be changed deeply. Moreover, if we use the 3D mouse with another aim, the options that this application already gives us are very useful.

- Motion Control of Industrial Robots by 3D Mouse.

6. References.

- [1] http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html.
Definition of the respective programming methods. 27/08/08
- [2] <http://www.webopedia.com/TERM/A/API.html>.
Definition of API 27/08/08
- [3] http://en.wikipedia.org/wiki/Microsoft_Foundation_Class_Library.
Definition of MFC 27/08/08
- [4] http://en.wikipedia.org/wiki/Test_scenario.
Definition of test scenario 27/08/08
- [5] <http://www.steelbrothers.ch/jusb/api/usb/windows/related-docs/JavaUSBforWindowsWeb.pdf>
Java API. 27/08/08
- [6] <http://www.aggsoft.com/usb-port-monitor.htm>
Downloaded of the software USB Port Monitor. 27/08/08
- [7] <http://www.hhdsoftware.com/Products/home/usb-monitor.html>
Downloaded of the software Device Monitoring Studio. 27/08/08
- <http://www.robotsltd.co.uk/robot-guide.htm>
Chart with movement of the joints of the robot. 27/08/08
- www.abb.com
General information about the ABB robot. 27/08/08
- <http://www.robots.com/abb.php?robot=irb+140>
Global description of the robot. 27/08/08
- <http://html-color-codes.com/rgb.html>
Colour code for the charts. 27/08/08