The remote control of mobile robot on the Internet

Master Thesis
Computer Engineering
Nr: E3501D

Shengtong Zhong

Högskolan Dalarna, Sweden
May 2007
Dedicated…
To my dear parents
# DEGREE PROJECT
in Computer Engineering

<table>
<thead>
<tr>
<th>Programme</th>
<th>Reg number</th>
<th>Extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Master of Science in Computer Engineering</td>
<td>E3501D</td>
<td>30 ECTS</td>
</tr>
<tr>
<td>Name of student</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shengtong Zhong</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervisor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dr. Pascal Rebreyend</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company/Department</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laboratoire d'Informatique des Systèmes Complexes, Université Bretagne Occidentale, France</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supervisor at the Company/Department</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prof. Philippe Le Parc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The remote control of mobile robot on the Internet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keywords</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote Control, Teleoperation, Internet telerobotics, Mobile robot, Path error, Speed Limit Module, Delay Approximator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abstract

During last decades, the Internet teleobotics has been growing at an enormous rate due to the rapid improvement of Internet technology. This paper presents the internet-based remote control of mobile robot. To face unpredictable Internet delays and possible connection rupture, a direct continuous control based teleoperation architecture with “Speed Limit Module” (SLM) and “Delay Approximator” (DA) is proposed. This direct continuous control architecture guarantees the path error of the robot motion is restricted within the path error tolerance of the application. Experiment results show the feasibility and effectiveness of this direct Internet control architecture in the real Internet environment.

Acknowledgement

I would like to express my thanks for Prof. Philippe Le Parc, my dear supervisor for his sincere supports at the Department of Computer Science, Université de Bretagne Occidentale, from who I learnt a lot on the aspect of doing research. Thanks to my local supervisor at Department of Computer Engineering at Dalarna University, Dr. Pascal Rebreyend who I regard as one of my best friends other than a teacher. Thanks for your sincere help during the last two years.

Also my thanks to Dr. Jean Vareille, Prof. Mark Dougherty and Dr. Hasan Fleyeh, you are all very good teachers and I benefit a lot from your tuitions.

Special thanks to Yang Shen who did a lot to my first paper as well as this thesis. Thanks for the help from my good friends: Yi Feng, Ping Zhao, Liwen Liang.

Finally, my thanks (with kiss of course) to my beloved Lily and dear parents who are the driving force in my life.
## CONTENT

### CHAPTER 1 INTRODUCTION

1.1 MOTIVATION ........................................................................................................ 5
1.2 TELEROBOTICS .................................................................................................... 5
   1.2.1 Teleoperation .................................................................................................. 6
   1.2.2 Robotics .......................................................................................................... 6
1.3 CLASSIFICATION OF TELEROBOTICS SYSTEM ............................................. 6
1.4 INTERNET TELEROBOTICS ................................................................................ 7

### CHAPTER 2 RELATED WORK OF INTERNET TELEROBOTICS

2.1 MERCURY PROJECT AND TELEGARDEN ..................................................... 11
   2.1.1 Mercury Project ............................................................................................ 11
   2.1.2 Telegarden ..................................................................................................... 11
2.2 AUSTRALIAN TELEROBOT ON THE WEB .................................................... 13
2.3 RHINO ................................................................................................................ 14
2.4 XAVIER .............................................................................................................. 15
2.5 PUMA PAINT ................................................................................................. 16
2.6 KHEP ON THE WEB ....................................................................................... 17

### CHAPTER 3 MIABOT PROJECT ANALYSIS

3.1 MIABOT ................................................................................................................ 20
   3.1.1 Features of Miabot ......................................................................................... 21
      3.1.1.1 Drive Chain ............................................................................................... 21
      3.1.1.2 Position Encoders .................................................................................... 21
      3.1.1.3 Batteries ..................................................................................................... 22
      3.1.1.4 Communications Board .......................................................................... 22
      3.1.1.5 Expansion Port ........................................................................................ 22
      3.1.1.6 Parameter Specification ........................................................................... 22
   3.1.2 Why Miabot .................................................................................................. 23
3.2 SATURNE SYSTEM ................................................................. 23
  3.2.1 Generic architecture ....................................................... 23
  3.2.2 Software architecture ..................................................... 25

3.3 MISSIONS OF MIABOT PROJECT ............................................. 27
  3.3.1 A new Internet telerobotics application as teaching resource .. 27
  3.3.2 Validate the Saturne system ............................................. 27
  3.3.3 Using direct continuous in Internet telerobotics .................. 28
  3.3.4 Algorithms in Saturne system ......................................... 29
  3.3.5 Extended work ............................................................... 30

3.4 TECHNICAL CONCERNS ......................................................... 31
  3.4.1 Multimedia ..................................................................... 31
  3.4.2 Web server ...................................................................... 33
  3.4.3 Graphical User Interface (GUI) ......................................... 34
    3.4.3.1 CGI ........................................................................... 34
    3.4.3.2 Java Applet .............................................................. 34
    3.4.3.3 Server push ............................................................. 34
    3.4.3.4 Frames ..................................................................... 35
  3.4.4 Miabot serial link communication ..................................... 35
  3.4.5 Core of system communication ....................................... 36
  3.4.6 Control strategies in Miabot project ................................. 37
    3.4.6.1 “Move and Wait” strategy ....................................... 37
    3.4.6.2 “Speed control” strategy ......................................... 39
  3.4.7 Speed Limit Module (SLM) and Delay Approximator (DA) .. 41
    3.4.7.1 GEMMA-Q approach .............................................. 41
    3.4.7.2 Speed Limit Module (SLM) ...................................... 41
    3.4.7.3 Delay Approximator (DA) principals ......................... 44
    3.4.7.4 Global view ............................................................ 46

CHAPTER 4 EXPERIMENT

4.1 FEASIBILITY TEST OF MIABOT APPLICATION .......................... 49
  4.1.1 24 hours a day ............................................................... 50
  4.1.2 Continuous command ..................................................... 51

4.2 EFFECTIVENESS TEST ........................................................ 51
  4.2.1 Parameter choice and theoretical analysis ........................ 52
4.2.2 Practical test and analysis ................................................................. 54
  4.2.2.1 Stable delay test .............................................................................. 54
  4.2.2.1 Changeable delay test ...................................................................... 57

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 FINDINGS AND CONTRIBUTIONS .................................................... 62
5.2 APPLICATIONS .................................................................................. 62
5.3 FUTURE WORK ................................................................................... 62

REFERENCE

APPENDIX A

RELATED PUBLICATIONS ........................................................................ 71
SOURCE CODE SAMPLE ........................................................................ 71
Chapter 1

Introduction
1.1 Motivation

There are many physical difficulties to perform operations by human-beings due to the limitation like safety, time, cost etc. These operations are impossible to handle by human-beings, but robotics can help people in this situation. And then a new filed come into practice. Telerobotics, control the robotics from a distance mainly with wireless connection, Internet connection or Deep Space Network etc, is a good choice at this circumstance.

The operations can be achieved at a distance by a human operator. In some cases, the robotics is fully autonomy, and operator only need to supervise the course. This is not enough in most operations since tasks are always dynamic in real situation and require immediate feedback from operator.

One of the most famous telerobotics projects is the Space Telerobotics Program of National Aeronautics and Space Administration (NASA), USA. The program has a driving force to push the development of telerobotics, which aims to integrate robotics and teleoperations. When the famous Mars rover succeeded in science exploiting in Mars, it’s no doubt everybody believe that there will be a bright future for telerobotics.

The research of telerobotics mainly focused on merging robotics and teleoperation, it also involves many other research aspects due to different purposes of projects. This can be concluded as “To develop, integrate and demonstrate the science and technology of remote manipulation”.

1.2 Telerobotics

Telerobotics concerns the remote control of robotics at a distance mainly with the wireless connection or Internet connection etc. The major concerns in telerobotics include teleoperation and robotics.
1.2.1 Teleoperation

Teleoperations means perform a action at a distance. The distance here is not so clear: it could be a physical distance where the robotics is separated from operator with a large geographical distance, or it could be a distance in concept, e.g. a surgeon may conduct surgery with remote medical device in the room.

“A teleoperator is a machine that extends a person’s sensing and/or manipulating capability to a location remote from that person” (Sheridan, 1992). If such a device has the ability to perform autonomous work, it is called a telerobot.

1.2.2 Robotics

Robotics is around the science and technology about robot including design, fabrication, application etc. It involves aspects of sensors, actuators, control system etc. This is a very wide area. In this paper, there is no discussion in this aspect, and the chosen robotics is bought from commercial company.

1.3 Classification of telerobotics system

Conway (Conway and al., 1990) defines the following classification of telerobotic control scheme base on the method a system is controlled:

“Direct continuous control: The remote device follows the input from the controller; also known as master/slave control.

Shared continuous control: Control is at a higher level than direct position servoing; i.e. the device may vary from course if it encounters an obstacle.
**Discrete command control:** The controller is able to carry out discrete commands without intervention. This implies a higher level of capability in the remote portion of the controller as it must be able to carry out the command without help.

**Supervisory control:** The remote device operates in a largely autonomous mode and only interacts with the human when it encounters an unexpected situation.

**Learning Control:** The remote device is given intelligence that allows it to learn from human inputs and sensor information and subsequently to generate behavior in similar situations without human intervention."

Naturally, each telerobotics control scheme has its own feature, to choose the most suitable method according to project is rather important. Sometimes, there is no significant difference among the chosen methods; it can be a unique method which mixes some good points of different control schemes.

### 1.4 Internet telerobotics

In wikipedia, the definition of Internet is: “*The Internet is a worldwide, publicly accessible network of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol (IP). It is a "network of networks" that consists of millions of smaller domestic, academic, business, and government networks, which together carry various information and services, such as electronic mail, online chat, file transfer, and the interlinked web pages and other documents of the World Wide Web (WWW). "* Internet is very popular in our regular life.

Internet telerobotics is remote control of robots from a distance over Internet. The reason to choose Internet as the communication channel mainly lies in the following advantages of Internet:
● accessible from nearly every point of the Earth, using various technologies from basic Telephone lines to satellite transmission systems.

● becoming faster and faster, cheaper and cheaper, day after day, with the developing of new technologies and the increase in number of Internet links.

● easy to use and also easy to deploy, navigators are now belonging to classical software shipped with any computers and web servers are also following the same way.

In the section 1.3 classification of telerobotics system, different control schemes are introduced according to method of the system is controlled. Here, what should we consider when using Internet as the communication channel in telerobotics? Is there a most suitable control scheme for Internet telerobotics?

An important aspect of Internet telerobotics that affects the choice of control scheme is Internet time delay. The Internet doesn’t provide the guaranteed quality of service (QoS), which means the control schemes, must work under the unpredictable Internet time delay, or possible connection rupture. In some situations, the network is quite stable with small value of network delay. But since there is delay, the influence to the teleoperation is unavoidable.

“Direct continuous control scheme is not well suited, as it is prone to instability problems under time delay” (Sheridan, 1993). But the Internet technology has improved a lot during last twenty-four years. Internet Service Provider (ISP) has enhanced bandwidth of Internet significantly as well as equipped with more advanced hardware.

Shared continuous control and discrete continuous control is quite similar in some sense. The command is at a high level and it’s less sensitive to Internet delay as the high level command could me a large combination of movement.

Supervisory control is like the user only watches the robot perform a specified task, and
only need to handle the exceptions. This control is at a high level and user has less interactive with the robot.

Learning control refers to the system can the intelligence and can learn from the previous experience. It main depends on the sensors and actuators as well as the evolutionary algorithm in the system.

The choosing of a control scheme in an Internet telerobotics depends on the application requirement. A chosen scheme might be a combination of many schemes. In a word, the target is to choose the most suitable scheme to the application.

E.g. I used a 33.6-K modem to connect Internet some 8 years ago when I was still a student in high school; now the ordinary Internet access in my hometown is ADSL with 2M bandwidth. These improvements make it possible to use direct continuous control scheme in Internet telerobotics. However, Internet still doesn’t provide quality of service (QoS), so large time delay or connection rapture are still the dilemmas in such control scheme. Although the Internet connection is rather good inside Europe or North American, the situation is rather bad in Africa and Asia due to the limitation of hardware and long distance between continents.

In the hardware consideration, some countries in Africa and Asia, they are still using low speed Internet connection link, in the same time ISP may only provide limited bandwidth.
Chapter 2

Related work of Internet telerobotics
To my knowledge, the research of Internet te lerobotics has more than ten years’ history. This chapter is an extensive research review of Internet telerobotics, mainly in the area of purpose of these Internet telerobotics projects as well as its research contribution.

2.1 Mercury project and Telegarden

2.1.1 Mercury Project

The *Mercury Project* (Goldberg and al. 1994) is believed to be first system that allowed Internet users to remotely control robotics via Internet. This project launched the first system allows users to alter the real world

This project is initialized by an interdisciplinary team of anthropologists and computer scientists. They want to explore an area dubbed “Mercury site”. Because nobody can work in this dangerous area, the remote robotics is a good choice. The remote control of the robot is designed to excavate the surface with short burst of compressed air, and then the surface is revealed and the relevant data can be collected by the anthropologists. After the success of the initial exploration, the site is open to all the researchers who are interested in having a remote control of the robot via Internet.

The successes of *Mercury Project* is not only on its excavation purpose, but also showed the possibility of control the robot via Internet. This is the milestone on Internet telerobotics, more and more Internet telerobotics projects were launched in the later years.

2.1.2 Telegarden

*Telegarden* (A robotic garden, Figure 1) is second Internet telerobotics from Goldberg
and al. 1994. This *Telegarden* system allows Internet users to view and interact with a remote garden filled with living plants. Users can plant, water, and monitor the progress of seedlings via the tender movements of an industrial robot arm.

![Figure 1: Telegarden](image)

The *Telegarden* provides users the ability to communicate among each other in the time of controlling the robotic arm. The project was created to push the possibility of web interaction using customizable user preference, online chatting, web cameras, etc. This is the tendency in the research of Internet telerobotics until now. In a word, it’s not enough to only launch Internet telerobotics, the high quality of Internet telerobotics is a long run consideration which means it integrates possible science and technology to build a more powerful functions system with easier control Interface.
2.2 Australian Telerobot on the Web

The Australian Telerobot on the Web project is established by Taylor at The University of Western Australia in 1994. A six degree of freedom robotic arm is controlled through Taylor’s Internet telerobotics system. The robotic arm can play the wooden block on a table. The user connects the server via Internet connection, and can log to the system with an identification check. This project is very successfully due to it’s an interesting wooden block game as well as users can play it via Internet.

Figure 2: Australian Telerobot on the Web

The Figure 2 above is an image of the remote robotic arm system. To establish a telelabs is another purpose of this research which aims to enhance the student learning experience. The idea is to develop new styles of student learning method which provide maximum flexibility of using the teaching resources.
Sometimes, the laboratory equipments are very expensive and not all the institution has enough budgets to set up such equipments. In the same time, in the lab which has such equipments on-going, these equipments are not used frequently. The concept of Telelabs can definitely help intuitions to share the limited lab resource and make the best usage of them.

The robotic arm is also quite popular in the industrial area and this is also believed to be the first Internet industrial robot. This research showed how complex machines could be successfully operated by one or more simultaneous users using low bandwidth connections provided by the Internet connection. In the later years, more researchers and engineers form industrial area contribute to the Internet robotics.

2.3 RHINO

The launch of RHINO project (Figure 3) indicates the possible potential of Internet robotics in daily life. RHINO project is initially launched for a museum of contemporary technology in Bonn Germany.

Visitors of the "Deutsches Museum Bonn" will have the opportunity to be shown through the museum by a mobile robot “RHINO”. RHINO can provide the user with the information they concerned as well as more information in deep upon request.

The RHINO project is a joint project between the "Institut für Informatik III" of the Rheinische Friedrich-Wilhelms-Universität Bonn and the Carnegie Mellon University USA.

The control scheme of RHINO towards the learning control and the robot is expected to learn from its previous experience. The importance of this project lies in that it involves the Internet telerobotics to daily life of human being and commercial purpose.
2.4 Xavier

Carnegie Mellon University (CMU) developed a mobile robot *Xavier* (Simmons and al., 1999) which uses batteries power. Since it’s the batteries power supply, the batteries need to be changed from time to time. So the robot can be online only for a certain time with a fixed time table. The system has worked for more than 10 years, but the online time limitation indeed prevents some users. (*Xavier* is now working in a new project aims to develop an architecture that will allow multiple independent robots of different types to work together as a unified team. This is called Distributed Robot Architecture (DIRA) at CMU).

The research focus of *Xavier* is to study the local intelligence of the robot as well as users’ interface. The research team has considered the supervisory control which aims to give robot the command at a higher level. This scheme can reduce the influence of
Internet time delay, but prevents more interaction between users and robot. In this situation, users can’t interact with the robot immediately.

Another important findings of this project lies in the system has improved a lot after the public test. Because the users may user the system in their own styles and many bugs or exceptions may be found in this situation. These feedbacks are essential for improving the system. It’s no doubt that the system can be well improved (more useful feedback) if it can be put online for longer time. A consideration of putting the Internet telerobotics twenty-four hours’ online is realized by later researchers in this area.

The Xavier project leads researchers to consider the control algorithm of the Internet telerobotics. Users need fast feedback (image) when controlling the robots on the web facing unpredictable Internet time delay (limited bandwidth). The supervisory control can indeed reduce the bandwidth requirement, but at the same time reduces interactivity. These existed problems are highly concerned in the later years’ research of Internet telerobotics.

2.5 PumaPaint

The PumaPaint project (Stein, 1998; DePasquale and al., 1997; PumaPaint) came out from the collaboration between University of Wisconsin and Wilkes University. The computer science department at UW needed a PUMA robot as undergraduate teaching resource. But if they develop and maintain the system themselves, it’s money and time costly. So, they decide to share the installed one at Wilkes University. Students from University of Wisconsin can access the PUMA robot (located at Wilkes) via Internet connection.

The system is developed in Java considering the cross platform advantage of java as well as the reusability of Java program. The Java Virtual Machines (JVM) is involved in the development, every machine need to install this before using the system, but the JVM is quite popular in most of web browsers.
The highlight of *PumaPaint* project lies that it’s shared lab equipment between Universities. Students can not only use the system, but also take part in the developing new function of the system. The idea to let student involve in the project is quite popular in the later years’ research. This is the best way to help student to understand the knowledge around Internet telerobotics. In the same time, the system was developed in Java. In *PumaPaint* project, the encountered problem with Java application is rather important; it helped the researchers to have a relevant careful consideration when developing similar system with Java.

### 2.6 Khep on the Web

Khep on the Web – robot maze is first established Internet telerobotics system by Michel and al., 1997; Siegwart and Saucy, 1999 at the Swiss Federal Institute of Technology Lausanne (EPFL). This research team has made a lot of robot on the web, and Khep is the first and most well known one.
The Khep on the Web is very successful mainly on the developing technologies it
chooses. The control of the robot based the inference of the user according to the
uncalibrated video streaming (images), that is to say, the movement of the robot is
relative to its current state.

The project involves many advanced technologies like CGI, Java, Java applet, frames etc.
The control scheme is mainly based on the direct continuous control. The user has more
interaction with the system, but it’s sensitive to the Internet delay as the user has an
immediate feedback to the system. The chosen technologies targets to simplify the
system and to reduce the transmission as much as possible since the users’ control is very
sensitive to the Internet delay e.g. reduce the quality of the image (higher compression)
and using frames to separate parts with different update requirement.

This project showed the possibility of using direct continuous control scheme in Internet
telerobotics even though facing the unpredictable Internet delay. Advanced technology
can help to reduce the network transmission so as to reduce the influence of Internet
delay.
Chapter 3

Miabot Project Analysis
The Laboratoire d'Informatique des Systèmes Complexes (LISyC) of Université de Bretagne Occidentale (UBO), France has succeeded in launching some Internet telerobotics with industrial machines, including a prototype machine and a robotic arm with five degree of freedom etc., which can be controlled by users via Internet connection since 2001. The goal is to investigate the remote control of industrial machines using an Internet connection. Another consideration is to using a teleoperation system as part of automated system teaching, this is proved to be very efficient since student has more chance to connect the automated system.

In November of 2006, LISyC bought one robot call Miabot from Merlinrobotics, UK. This is a two-wheeled robot with Bluetooth connection. The lab considers to launch the remote control of Miabot based on its previous remote control system Saturne, and then to verify the computability of Saturne system for new Internet telerobotics application as well as employ some efficient algorithm in the remote control system application. This project would also be included in the Internet teleoperation teaching source after its launch.

The thesis continues the Internet telerobotics research at LISyC, UBO, France (Le Parc, Vareille, and al. 2001). The purpose of the thesis is to implement new Internet robotics application and employ possible solution to alleviate the influence of unpredictable Internet time delay or connection rupture.

### 3.1 Miabot

The Miabot (by Merlinrobotics, UK, Figure 5) is a fully autonomous miniature mobile robot. The latest BT version features bi-directional Bluetooth communications, which provides a robust frequency hopping wireless communications protocol at 2.4GHz. Miabots are ideal robots for use as part of technology class tutorials, for research and development, and for high-speed and agility requirements such as robot competitive events. Universities already use Miabots worldwide for a wide variety of applications including FIRA (Federation of International Robot-soccer Association) robot soccer.
competitions, intelligent behaviors, robot swarming experiments and mobile robot navigation experiments. Miabots can also be employed as super mice, for line/maze following experiments etc.

![Miabot](image)

Figure 5: Miabot

### 3.1.1 Features of Miabot

#### 3.1.1.1 Drive Chain

The motors are driven by 6 x 1.2v (AA) cells through a low-resistance driver I.C. with a slow-acting current limit at about 5A. Maximum speed of an unloaded motor is in the region of 6-8000 rpm.

#### 3.1.1.2 Position Encoders

The motor shafts drive the wheels through an 8:1 gearing. The motors incorporate quadrature encoders giving 512 position-pulses per rotation. The wheels are 52mm in diameter, so one encoder pulse corresponds to just under 0.04mm of movement.
3.1.1.3 Batteries

Each robot contains two 3’AA-cell battery packs (nominal 1.2v per cell, 1300mAh). The robot is supplied with a NIMH fast-charger that can charge these completely in about 1-2hrs. Batteries will last about 1hr typical continuous use (much more if not moving.). Higher-capacity batteries are also available in the same physical size.

3.1.1.4 Communications Board

A Bluetooth communications card is incorporated within each robot. This enables the host PC to communicate with the robot by converting the Bluetooth link to logic-level serial signals connecting to the main board processor – A PC Bluetooth dongle is supplied that plugs into the USB port on the PC. This can support wireless links with up to seven robots at once.

3.1.1.5 Expansion Port

This is used for development programming and for connecting to external devices such as sensors. The connector takes a 16-way IDC plug or board-stacking connector.

3.1.1.6 Parameter Specification

The parameter specification is shown below in Table 1.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Atmel ATMega64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>14.5 MIPS (RISC)</td>
</tr>
<tr>
<td>Program Memory</td>
<td>64Kb Flash</td>
</tr>
<tr>
<td>Non-volatile storage</td>
<td>2KB EEPROM</td>
</tr>
<tr>
<td>Data Memory</td>
<td>4Kb SRAM</td>
</tr>
<tr>
<td>Features</td>
<td>In System Programmable</td>
</tr>
<tr>
<td>Expansion Port</td>
<td>8 User I/O or 10 Bit A/D</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Drive Train</td>
<td>Two Wheel,</td>
</tr>
<tr>
<td></td>
<td>Optical encoder resolution of 0.04 mm</td>
</tr>
<tr>
<td>Speed</td>
<td>3.5 m/s</td>
</tr>
<tr>
<td>Communications</td>
<td>Bluetooth, 11.5 Kb/sec</td>
</tr>
<tr>
<td>Batteries</td>
<td>6 x 1.2 V, NiMH, rechargable</td>
</tr>
<tr>
<td>Development</td>
<td>Standard toolset includes GCC ‘C’ compiler &amp; link</td>
</tr>
<tr>
<td>Simulator</td>
<td>FIRA simulation engine</td>
</tr>
</tbody>
</table>

### 3.1.2 Why Miabot

The reasons to choose Miabot as the robot in the project are:

- affordable, it’s around 500 euro, and not so expensive.
- Miabot has Bluetooth connections; this means the robot is free of ware in the local sites.
- its connection board supports up to seven robots to be controlled in the same time. This is an ideally good choice in a later research expansion, i.e. multi robots control.
- the expansion port is used for development programming and for connecting to external devices such as sensors or actuators.

### 3.2 Saturne system

#### 3.2.1 Generic architecture

From the experiences made on the web, it is possible to describe the general Saturne system (software architecture) of an Internet telerobotics application (Figure 6).
remote user is located on the right side. The user is using a classical Internet browser and is connected through a modem line, a DSL connection or using a wide area network. On the left side is shown the machine to control. It is connected, using a known link (like a RS232 serial link); to an application server which will deliver commands send by the user and will propagate values of internal machine’s sensors. Generally, multimedia devices are also used to forward images and sounds from the system through a multimedia server. Finally, a web server (HTTP) is used to welcome the operators, to download the necessary web applications and to propose static information pages about the system.

Figure 6: Generic architecture

The user, through his Internet navigator, addresses a request to a Web Server (step 1) and downloads an application, such as a Java applet (step 2), on his own workstation. A connection is then established towards the Application Server in charge of robots and client management (step 3). At the same time, another connection with Multimedia Server is also established in the form of exchanging media signals. The user is now able to control the remote robot upon request.

Different technologies have been used to program such an application. First experiences
where using Common Gateway Interface (CGI) scripts and clickable images. Nowadays, application servers are developed mainly in Java and Java applets is used on the client side. The choice of the latter was rather obvious to take advantage of a high level language that may easily cope with Internet browser. The following is description of the software architecture proposed for Internet telerobotics.

![Figure 7: Saturne software architecture](image)

### 3.2.2 Software architecture

The software architecture (Figure 7) is based on a set of communicating processes. On the left side, the application server is presented. It is composed of a set of permanent processes called “Groom”, “Connection Manager” and “Device Manager”. They are respectively designed to receive connection requests, to manage the different users according to the control algorithm module, and to handle the machines. The connection between the “Device Manager” and the machines (robots) is made by specific processes called “Tool Interface” who are acting as device drivers. “Local Client Manager”, “NI Sender”, “NI Receiver” and “Pinger” are dynamic processes which are launched every time when a new user get the right to enter the system. Communication between these
processes is realized with the help of Java pipes. The right side of the figure shows the user side. “NI Sender” and “NI Receiver” are used to communicate, via TCP/IP sockets with the server side. The “Ponger” process is connected to its corresponding “Pinger” process. “Tool GUI” are graphical applets used to display information coming from the remote controlled system and to get commands from the user. ”Local Client Manager” is acting as a router between the different processes.

A detailed presentation of this architecture may be found in (Ogor and al., 2001) (Le Parc and al., 2002). There are two aspects that differentiate approach from those generally used in Internet telerobotics:

- Plug and Play devices: the architecture and the different processes have been defined to be able to control several devices. Except “Tool Interface” and “Tool GUI”, everything is generic and to add a new machine, just these two programs have to be re-written. A configuration file is added to the application to tell the system which machines have to be controlled, and their corresponding interface and GUI are launched.

- Network sensor: as explained before, on the one hand, Internet is not providing quality of services and on the other one, Internet telerobotics applications have to be safe. To try to manage this problem, it is necessary to take into account the quality of the network. A first step is to get a measurement of it.

In this approach, it is realized by the two processes called “Pinger” and “Ponger”. The first one sends a request to the second one, which is supposed to answer immediately and so on. The Round Time Trip (RTT) between the server side and the client side is then calculated. Of course the obtained values give us an image of the past, but as explained later in this article, it also gives an idea of the future. Moreover, while waiting the answer (resp. the request), the “Pinger” (resp. the “Ponger”) may launch watchdogs that which start up actions when time limits are exceeded.

The software architecture is proved to be efficient in previous launched Internet
telerobotics (including a milling machine and a robotic arm with five degree of freedom). A new application can be launched by a change in the Tool Interface part.

3.3 Missions of Miabot project

3.3.1 A new Internet telerobotics application as teaching resource

The LISyC of UBO France decides to develop a new Internet telerobotics application Miabot project. Miabot project will be conducted as a teaching resource later in the informatics department of UBO France. After its launch, the student could have more chance to remotely control the robot via Internet connection; it will enhance the students’ understanding about Internet telerobotics as well as hand on practice knowledge.

3.3.2 Validate the Saturne system

Saturne system (Le Parc and al, 2001) is a successful Internet telerobotics system. The LISyC of UBO France has successfully launched some applications with Saturne system, including a robotic arm with five degree of freedom and a light milling machine ISEL CMP3020 (http://www.islautomation.de) and etc.

These applications demonstrated the feasibility to carry Internet robotics with Saturne system. With years working of these applications, the running of Saturne system still keeps stable. As described above in 3.2.2, the Saturne system features to “Plug and Play Device”, it’s time to implement other robotics with Saturne system. The process of Miabot project is a validation of the Saturne system. The LISyC wants to know the compatibility of Saturne system as well as improving the system.
The best way to validate a system is to find someone who is unknown to the system to launch an application, and to observe how much time needed to launch an application based on the system as well as the problems aroused. Because, if the people has no previous knowledge of the system, he/she can ignore some fixed idea of previous system designer. There are more chances to find out weakness of the system so as to improve it, and the time needed to launch the new application is more general.

3.3.3 Using direct continuous scheme in Internet telerobotics

In section 1.3, different control scheme of telerobotics system, classified by the method by which it is controlled, are introduced. Then the control scheme choice in Internet telerobotics is discussed in section 1.4. The idea is that it’s hard to say which control scheme is suitable for Internet telerobotics, but there should be a control scheme which is fit for the specific Internet telerobotics application. It depends on the application and its purpose.

In the entire chapter 2, many Internet telerobotics projects are discussed; it’s quite obvious that every project has its own control scheme which makes the project successful, i.e. PumaPaint uses direct continuous control scheme in the project and Australian telerobot on the Web uses shared continuous control scheme. It’s rather hard to say a project is using a specific control scheme, because it always involves the features of other control scheme. In a word, a project towards a control scheme and this scheme has a dominating role in the project, but there are also some features or control techniques from other control schemes. To find the most suitable control scheme for the specific project is the essence of Internet telerobotics applications.

In the Miabot project, the aim is to launch the new robot based on the Saturne system, and then the user could remotely control the robot via Internet connection. Meanwhile, the system could provide the user with live video streaming, and the user need to infer the movement of the robot himself. In the initial idea, the Miabot project only provide the
user with low level command like move, turn, stop. Based on this feature, the user need to give an immediately and continuously feedback to the system according to the live video stream, the operations are time sensitive, so the control scheme is mainly with direct continuous control scheme.

But the Internet doesn’t provide quality of service (QoS); the Internet time delay and connection rupture are unpredictable. The direct continuous control is thought to be weak when the Internet time delay is rather large and changeable before. The LISyC of UBO France has investigated long period time of Internet condition in previous Internet telerobotics applications, and the Internet network may allow remote control of mechanical systems: round trip time (RTT) values are decreasing and rather stable (Le Parc and al, 2005). Nevertheless, the unpredictable nature of Internet connection is unavoidable, so the development methodologies and software architectures should take this into account. It’s also a verification of the previous investigation on Internet connection condition result from LISyC of UBO France. In Miabot project, direct continuous control scheme is expected to be proved feasible as the Internet condition becomes better and better nowadays.

3.3.4 Algorithms in Saturne system

In Miabot project, the main direction is fixed with using direct continuous control. Although the RTT is decreasing and rather stable in the Internet connection, the unpredictable nature of Internet is still an obstacle in Internet telerobotics especially with direct continuous control scheme which is very sensible to Internet time delay.

The previous launched Internet telerobotics applications with Saturne system at LISyC UBO France, including milling machine and robotic arms, are mainly using shared continuous control scheme and supervisory control which means control is at a higher level and the device (robotics) has reverent largely autonomy. These control modes are comparatively insensible to unpredictable Internet time delay or connection rupture. The
Internet nature has been considered in the previous application when develop the Saturne system and respective application, but it’s not enough in the Miabot project with Saturne system since the direct continuous scheme is more sensible to unpredictable Internet condition compared to previous applications.

So, it’s quite important to consider the unpredictable Internet nature in the Miabot application, especially based on previous Saturne system. The idea is to improve the control algorithms of Saturne system in necessary part so as to alleviate the influence of unpredictable Internet time delay or connection rupture, finally a successfully direct continuous control in the Miabot project based on Saturne system is expected to achieve.

### 3.3.5 Extended work

Miabot is a two-wheeled robot which is widely used in robot soccer organized by Federation of International Robot – soccer association (FIRA). Using Miabot as the robot in the project can help the LISyC have a deep understanding about the mechanical condition of robot, and it’s also possible to build a robot soccer team later if possible. The robot soccer is well organized by FIRA, many Universities from all continents involves in this competition. The objectives of robot soccer from FIFA include:

- to take the spirit of science and technology to the young generation and laymen.
- to promote the development of autonomous multi-agent robotic system that can cooperate with each other and to contribute to the state-of-the-art technology improvement in this specialized field.
- to bring together skilled researchers and students from different backgrounds such as robotics, sensor fusion, intelligent control, communication, image processing, mechatronics, computer technology, artificial life, etc. into a new and growing interdisciplinary field of intelligent autonomous soccer-robots to play the game of soccer.
If the LISyC can organize students and researchers to work in this field, the lab can largely broaden its research area and have more chance to involve in well organized international communication in robotics research since there is a FIRA World Congress once a year.

3.4 Technical concerns

3.4.1 Multimedia

The system need to provide the user with live image or live chat etc. Then the chosen driver for Multimedia server is Webcam32. Webcam32 is web camera software by Surveyor Corporation which is easy to capture live streaming video and broadcast it on the web pages, the reasons of choosing webcam32 lies in the following features of this software:

- Upload refreshed images using ftp dial-up connection or network (you select the upload interval, whether minutes or hours)
- Or broadcast live streaming video using javapush
- Chat feature allows live text conversation while broadcasting
- No plug-in or any other software necessary to view image

These features are enough to support the multimedia part of the Miabot project as an Internet telerobotics. The Miabot project is an Internet telerobotics, it provide the live streaming video to users via Internet. The transmission of live streaming video is a heavy load work in Internet connection, so the chosen upload interval (or Javapush interval) and the qualities of image are very important. These works can be done empirically after launching the system with sufficient test. In the Miabot project, the following preference is used in Webcam32 (Table 2):
Table 2: Webcam32 preference

<table>
<thead>
<tr>
<th>Web32 Preference</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame delay</td>
<td>50 mSecs</td>
</tr>
<tr>
<td>Image Format</td>
<td>320x240 RGB 15bit</td>
</tr>
<tr>
<td>JPEG compression quality</td>
<td>50%, 10349 bytes</td>
</tr>
<tr>
<td>Brightness</td>
<td>3381</td>
</tr>
<tr>
<td>Contrast</td>
<td>40%</td>
</tr>
<tr>
<td>Saturation</td>
<td>41%</td>
</tr>
<tr>
<td>Single frame</td>
<td>enabled</td>
</tr>
<tr>
<td>Applet</td>
<td>enabled</td>
</tr>
<tr>
<td>Server push</td>
<td>enabled</td>
</tr>
</tbody>
</table>

The values of set preference are based on empirical test and previous system knowledge, the purpose is to reduce the TCP/IP transmission between server and client and provide a reasonable live video streaming on users’ aspect. These values can be improved or modified due to different environments or usages. In a ward, reasonable video stream quality plus reasonable transmission speed are the key considerations.

The camera (Figure 8) used in the Miabot project is Sony EVI-D31. It is probably the most famous stand-alone PTZ camera in the world. The Sony EVI-D31 camera unit has been used throughout the videoconferencing market by just about every major manufacturer. Popularized by the Polycom range of viewstations, this camera has excellent image quality, 12x zoom, composite and S-Video outputs as well as an inbuilt microphone pre-amp.

Internally, the camera also features motion detection and image tracking along with a host of other functions. The unit is controlled via the hand held IR remote or by using the in-built RS232 VISCA interface connected to any PC or MAC.
Internally, the camera also features motion detection and image tracking along with a host of other functions. The unit is controlled via the hand held IR remote or by using the in-built RS232 VISCA interface connected to any PC or MAC.

The EVI-D31 Camera can be controlled through the VISCA input on the rear of the camera. VISCA is an RS-232 protocol. A cable (CAB90038) is provided to connect between a PC/Laptop COM1 (9 pin serial) port and the EVI-D31 camera.

Based on the advantages of Sony EVI-D31, it’s possible to communicate with the camera through its RS-232 protocol (serial link) since the target is to build clickable image control mode so as to the user can easily control the camera.

### 3.4.2 Web server

To consider the cost and compatibility cross the operating system, the Apache HTTP Server is the choice of Web server in Miabot project. To simplify the Miabot project, we try to use some common software, it make Miabot project more extensible for later colleague or other researchers as these choice are commonly used.
3.4.3 Graphical User Interface (GUI)

The GUI is an important part in the system; it’s the interface between system and user. The main techniques used in GUI are Common Gateway Interface (CGI) script, Java applet and frames. The target of designing GUI is let the user has a better control of the system with necessary functions provided. To design a simple and user-friendly GUI is the purpose.

3.4.3.1 CGI

CGI is a standard web related protocol. A user sends a request from web browser to a server which passes the request to an external application with CGI technology. Finally the server returns the result of from the external application to web browser.

3.4.3.2 Java Applet

A Java applet is an applet which can run in a Web browser using a Java Virtual Machine (JVM).

A Java applet is simple to make it work on Linux, Windows and Mac OS i.e. to make it cross platform, so it is supported by most of web browsers. The applet can work on all the installed version of Java with a sandbox mode without users’ security approval. The running JVM can load the applet in short time.

The advantages of Java applet is obviously suitable to the Miabot project, the Java applet is downloaded to user’s web browser from host server after request.

3.4.3.3 Server push

Server push (or webcasting) is commonly used in World Wide Web and related to HTTP protocol. The idea is the publisher informs the user with “I have some service for you”
and provides it directly to the user, on the contrary, the usual way should be the user requests a service from the server.

In the Miabot project, The GUI provide the live video streaming to the user side, here the server push is used so to provide the user with the latest image of the camera. But the dilemma is the server push is continuous even when the robot is not moving. This can result in unnecessary image downloads when nothing has changed in the remote environment.

### 3.4.3.4 Frames

HTML frames allow the transmission of an independent or sub window which communicate with server separately.

In the Miabot project, the target is alleviating the influence of unpredictable Internet delay. One consideration is to reduce the transmission between users and server. In 3.4.1, the preference value choice of Webcam32 also aims to reduce the network transmission with the method like highly compressing the JPEG image etc. Here, In the User side, the live video streaming is provided with server push and give user latest image of remote robot, it’s rather important to separate this part from the other control part in the GUI. With frame, live camera window is refreshed frequently as it's necessary. In the same time, there is no need to reload or refresh other unchanged control part. This will definitely reduce the network transmission and it’s also a good way to reduce the influence of unpredictable Internet nature.

### 3.4.4 Miabot serial link communication

A Bluetooth communications card is incorporated within each Miabot robot. This enables the host PC to communicate with the robot by converting the Bluetooth link to logic-level serial signals connecting to the main board processor. The communication between robot
and Saturne system is RS232 serial link. Because the Saturne system is developed in Java language, and then the Java communication API Javacom20-win32 is chosen.

The Java communications API is used to write platform-independent communications applications in Miabot application, because the Miabot server is supposed to work cross platform.

With this Java communication API, the Saturne system can communicate with Miabot freely (Bluetooth connection physically). In Miabot project, the commutation is mainly from Saturne system to Miabot, but it’s also possible for Miabot to send back message to Saturne system if necessary.

The serial link is between Saturne system and Miabot. The user could address command via the applet on GUI in his web browser, after the server (Saturne system) receives the command; it forwards the command to the Miabot motion. There are delays in the serial link communication between Saturne and Miabot, but this delay is rather small compared to the delay in the Internet communication. So this issue is not our consideration in this project.

### 3.4.5 Core of system communication

The core of system communication uses Java pipe. A pipe consists of a pair of channels: A writable sink channel and a readable source channel. Once some bytes are written to the sink channel they can be read from source channel in exactly the order in which they were written.

Whether or not a thread writing bytes to a pipe will block until another thread reads those bytes, or some previously-written bytes, from the pipe is system-dependent and therefore unspecified. Many pipe implementations will buffer up to a certain number of bytes between the sink and source channels, but such buffering should not be assumed.
In the Miabot project, the target is providing a remote control system with many users simultaneously. Although there is only one user is allowed to take the control of the system and others can only observer at the time while queuing for his control operation of the system, all the users has it’s own identity and memory in the system side, so there are lots of communication between users and server. Considering the priority, these messages have their own relative priority level, e.g. the user who is in control of the system may get the highest communication priority while sending or receiving information.

The core of system communication uses a pipe architecture, that is to say all the communication entity in the system (like Device Manager, Local Client Manager etc.) are initialed by the Groom and then waiting message, there are pipes between entities if there are communication requirement. Since all the entities are connected directly or intermediately, the complex pipes connection nests are the communication channel inside the Saturne system. e.g. “Pinger” can send a message to “Connection Manager” directly, and it can also send a message to “Device Manager” with the intermediate communication of “Connection Manager”. If the message enters an entity, the entity will perform respectively according to the message.

3.4.6 Control strategies in Miabot project

3.4.6.1 “Move and Wait” strategy

The “Move and Wait” strategy (Sheridan, 1992), which is typical for space robots with long distance communication has been applied first. In the Miabot case, five commands: “move forward” (MF), “move backward” (MB), “turn left” (TL), “turn right” (TR) and “stop” (ST) have been defined. These commands are enough to perform any complex task, but the user has to send lots of commands and change the move or turn parameters from time to time to meet the requirement.
In Figure 9, the robot is moving from $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F$. This complex task requires the following commands with respective parameters (Table 3).

![Diagram of robot movements](image)

**Figure 9: A serial movement task**

<table>
<thead>
<tr>
<th>Sub task</th>
<th>Parameter</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>MF</td>
<td>Distance $d_1$</td>
<td>$A \rightarrow B$</td>
</tr>
<tr>
<td>TR</td>
<td>Degree $a_1$</td>
<td>$B$</td>
</tr>
<tr>
<td>MF</td>
<td>Distance $d_2$</td>
<td>$B \rightarrow C$</td>
</tr>
<tr>
<td>TR</td>
<td>Degree $a_2$</td>
<td>$C$</td>
</tr>
<tr>
<td>MF</td>
<td>Distance $d_3$</td>
<td>$C \rightarrow D$</td>
</tr>
<tr>
<td>TL</td>
<td>Degree $a_3$</td>
<td>$D$</td>
</tr>
<tr>
<td>MF</td>
<td>Distance $d_4$</td>
<td>$D \rightarrow E$</td>
</tr>
<tr>
<td>TL</td>
<td>Degree $a_4$</td>
<td>$E$</td>
</tr>
<tr>
<td>MF</td>
<td>Distance $d_5$</td>
<td>$E \rightarrow F$</td>
</tr>
</tbody>
</table>

The user needs eighteen operations, including change parameter (distance and degree) and send command, to achieve this task. The execution result (path) is accurate if each sub task is exactly performed. Due to the physical mechanism of the robot (motor, gear, wheel, etc.), it is possible to have path error, but this situation is not concerned in this paper. Massive operations by the user is the basis of this strategy, which is insensitive to...
Internet time delay as the user has to wait the completely execution of previous command. The task process is not fluent; the execution time of entire task is much longer than the sum of each sub task time.

The “Mover and Wait” strategy is the basic control method on the user side. It’s simple and easy for user to master, but it requires a lot of changes in a complex task since the user need to change the move distance or turning degree. This is the most classic control method in Internet telerobotics, and it’s insensitive to Internet delay.

3.4.6.2 “Speed control” strategy

To reduce user’s workload and towards a fluent process during the complex task, a “Speed control” strategy, similar to real driving, is introduced. This is the another chosen control method of the direct continuous control of Miabot. “No-stop move forward” (NSMF), “no-stop move backward” (NSMB), “turn left” (TL), “turn right” (TR) and “stop” (ST) commands are defined in this strategy. When the robot executes a turning command, it stops running first, perform the entire turning next, and then runs in previous speed again. The user may choose different running speed as well as turning degree. Required commands to accomplish the task (Figure 9) from Table 4 are shown below.

<table>
<thead>
<tr>
<th>Sub task</th>
<th>Parameter</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSMF</td>
<td>Speed $v$</td>
<td>A→B</td>
</tr>
<tr>
<td>TR</td>
<td>Degree $a_1$</td>
<td>B</td>
</tr>
<tr>
<td>TR</td>
<td>Degree $a_2$</td>
<td>B→C</td>
</tr>
<tr>
<td>TR</td>
<td>Degree $a_3$</td>
<td>C</td>
</tr>
<tr>
<td>TL</td>
<td>Degree $a_4$</td>
<td>C→D</td>
</tr>
<tr>
<td>TL</td>
<td>Degree $a_5$</td>
<td>D</td>
</tr>
<tr>
<td>ST</td>
<td></td>
<td>D→E</td>
</tr>
<tr>
<td>ST</td>
<td></td>
<td>E→F</td>
</tr>
</tbody>
</table>

Table 4: Sub tasks with “Speed Control” strategy
The significant reduce of the user’s operation is observed. The robot is running with speed \( v \) during the whole task even though there is a turning/stop point. When the user sends a turning/stop command to robot based on the uncalibrated live video streaming, the command reaches the robot with a discrete time delay (RTT) \( \Delta t \) due to Internet time delay. The robot may run an extra distance \( \Delta d \) before it performs the turning/stop.

\[
\Delta d = v \times \Delta t
\]

Where

\( v \) indicates the current speed of the robot, \( \Delta t \) is the current Round Trip Time of Internet time delay.

As Internet doesn’t provide a guaranteed Quality of Service (QoS), the time delay is unpredictable. Figure 10 is an execution result to the task of Figure 12 with “Speed control” strategy. The path error is accumulative, and path deviation is significant which may result in the failure of the task.

![Figure 10: A path result with “Speed Control”](image)

Facing the discrete Internet time delay, to minimize the path deviation (error) and toward a fluent and successful task implementation with “Speed control” strategy, the quality
level idea of GEMMA-Q (Quality of Service GEMMA; Ogor, 2001) which derived from GEMMA (ADEPA, 1982) a widely used tool in French industrial, is introduced. Next section describes the Speed Limit Module (SLM) with quality level and how this works together with Delay Approximator (DA) to alleviate the influence of discrete Internet time delay. It’s functionality to change the robot speed automatically to restrict the path error within path error tolerance $\delta$ at every turning/stop point.

3.4.7 Speed Limit Module (SLM) and Delay Approximator (DA)

3.4.7.1 GEMMA-Q approach

The GEMMA methodology is developed to model the different phase of an automated system. It is a base for programming canvas proposing to the integration of a new automated system in the teleoperation architecture. Unfortunately, the GEMMA is only subjected to evolutions on automated system itself or the action undertaken by user owing the system. It doesn’t consider the event on the network communication quality.

The GEMMA-Q methodology derived from GEMMA has been proposed. The basic idea is to define the quality of communication as a discrete value. Following the value, the automated system has to follow different rules to ensure the security (path error). According to the type of quality, these rules might be different. In case of a good quality, the automated system must evolutes freely and no constraint is present. If the user has a slightly deteriorated communication quality, the user might keep the control and the automated system has to work in a deteriorated mode where, for example, the movement speed may be slowed down.

3.4.7.2 Speed Limit Module (SLM)
The GEMMA-Q has been used in the application over robot arm and prototype machine (Le Parc and al., 2001). In this paper, which aims to control a simple robot, use only the quality level idea of GEMMA-Q to design SLM. \( n \) levels of communication quality are introduced which corresponds Internet time delay zone (Time D.). Each time delay zone has its own sub path error tolerance \( \Delta \varepsilon_j \) (Sub T.) and Max Speed \( v_j \) as shown in the Table 5.

Table 5: Quality levels in SLM

<table>
<thead>
<tr>
<th>Qual.</th>
<th>Time D.</th>
<th>Sub T.</th>
<th>Max Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_0 )</td>
<td>( 0 \sim t_0 )</td>
<td>( \Delta \delta_0 )</td>
<td>( v_0 = \Delta \delta_0 / t_0 )</td>
</tr>
<tr>
<td>( Q_1 )</td>
<td>( t_0 \sim t_1 )</td>
<td>( \Delta \delta_1 )</td>
<td>( v_1 = \Delta \delta_1 / (t_1 - t_0) )</td>
</tr>
<tr>
<td>( Q_2 )</td>
<td>( t_1 \sim t_2 )</td>
<td>( \Delta \delta_2 )</td>
<td>( v_2 = \Delta \delta_2 / (t_2 - t_1) )</td>
</tr>
<tr>
<td>( Q_3 )</td>
<td>( t_2 \sim t_3 )</td>
<td>( \Delta \delta_3 )</td>
<td>( v_3 = \Delta \delta_3 / (t_3 - t_2) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( Q_{n-1} )</td>
<td>( t_{n-1} \sim t_n )</td>
<td>( \Delta \delta_{n-1} )</td>
<td>( v_{n-1} = \Delta \delta_{n-1} / (t_n - t_{n-1}) )</td>
</tr>
<tr>
<td>( Q_n )</td>
<td>( \geq t_n )</td>
<td>( 0 )</td>
<td>( v_n = 0 )</td>
</tr>
</tbody>
</table>

When the quality level is changing, the Application Server evaluates the Current Robot Speed (CRS) and the Max Speed of this quality level (MSoT). If CRS \( \leq \) MSoT, no command is sent to robot; else the Application Server sends MSoT command to robot (change CRS to MSoT).

MSoT is calculated by the time delay zone and sub tolerance which are defined by user according to application. In order to meet the path error tolerance \( \delta \) at the turning/stop point, the following constraints are used when designing SLM:

\[
\begin{align*}
\sum_{j=0}^{n} \Delta \delta_j & \leq \delta \\
v_{j+1} & \leq v_j \quad j \in (0,1,\ldots,n-1)
\end{align*}
\]

The above constraint guarantees the path error at the turning/stop point is restricted within \( \delta \) in any Internet time delay situation. \( Q_0 \) is the best quality level, its Max
Speed $v_0$ is the fastest running speed of the robot. $Q_n$ is the disconnection situation, and the robot stops running immediately ($v_n = 0$). The following is how the constraint works.

There are two kinds of situation when the robot runs under the speed limit rules.

1. Unstable network delay
   
   The robot is running in the different time delay quality levels between two continuous RTT. The worst case is from $Q_0$ to $Q_n$:
   
   $$\Delta d = \Delta \delta_0 + \Delta \delta_1 + \cdots + \Delta \delta_n = \sum_{j=0}^{n} \Delta \delta_j \leq \delta$$

2. Stable network delay
   
   It means the robot is running in the same time delay quality level $Q_i$ between two continuous RTT. Then
   
   $$\Delta d = v_i \times t_i \leq v_0 \times t_0 + v_1 \times (t_1 - t_0) + \cdots + v_i \times (t_i - t_{i-1}) = \Delta \delta_0 + \Delta \delta_1 + \cdots + \Delta \delta_i = \sum_{j=0}^{i} \Delta \delta_j \leq \delta$$

The path error $\Delta d$ is within the restriction of path error tolerance $\delta$ in both situations. Then the path error at every turning/stop point in the task is guaranteed and the successfully continuous control of the robot is achieved.
With the constraints, the different quality levels with its respective time delay zone value, sub tolerance and max speed are defined according to different application. e.g.: the normal quality level of Internet time delay is $Q_j$ in the application, its sub tolerance $\Delta \delta_j$ should take a larger percentage of $\delta$. It means the robot is preferred to have larger sub tolerance in the normal quality level; in the same time, speed limit rules guarantees $\Delta d \leq \delta$ in any situation.

In Figure 11, the path deviation at every turning/stop point is restricted within $\delta$. Then a successful task is achieved using “Speed control” with SLM. The above describes the detail SLM with the quality level idea, next part emphasize on how the gets the quality level information of current time delays from DA.

![Figure 11: A path result using “Speed Control” with SLM](image)

In the real environment control, the SLM can dynamically change the running speed of robot so as to reduce the path error. In the same time, the robot is safe if the user disconnect to the server.

### 3.4.7.3 Delay Approximator (DA) principals

The key part of DA is a watchdog. In the generic architecture, Pinger-Ponger is the mechanism to provide the Internet time delay information (RTT) to DA. The current RTT
$\Delta t_n$ is calculated as follows:

$$\Delta t_n = t_{pong} - t_{ping}$$

$t_{ping}$ is the time when watchdog receives a ping information from “Pinger”, and $t_{pong}$ is the time when watchdog receives a pong information from “Ponger”.

Pinger-Ponger only provides the RTT when it gets one. The Internet time delay is unpredictable, so there is no idea when Pinger-Ponger gets new information. When “Pinger” sends a ping to “Ponger”, it also informs DA, and the watchdog is initialized. In the same time, if “Pinger” receives pong from the “Ponger”, it also informs watchdog.

There is no idea about the future, but the current situation is supervised by DA as following principals:

1. **DA receives information from Pinger-Ponger.**

   Pinger-Ponger informs DA of receiving the pong, and then DA forwards the quality level of calculated $\Delta t_n$ to SLM. Meanwhile the previous RTT memory is set to be the current one, the quality level of previous RTT is changed to the quality level of $\Delta t_n$, reset the watchdog (“Pinger” sends out a new ping as soon as it receives the pong).

2. **No information from Pinger-Ponger and a new quality level is activated by watchdog**

   - The quality level of current RTT is no worse than that of previous RTT

   There is no action and DA keeps supervising.

   - The quality level of current RTT is worse than that of previous RTT

   When a new quality level is activated by watchdog and its worse than previous one, $t$ indicates a change of worse quality level and DA forwards the new quality level to
SLM. DA keeps supervising.

DA keeps supervising the Internet delay situation all the time. It provides the real Internet time delay or current time delay to SLM. The Delay Approximator works for supervising the current delay time from the time “Pinger” sends out a ping. If a new quality level change is observed by DA, it inform the system the situation and the system will forward a respective command to robot according to SLM.

3.4.7.4 Global view

In figure 12, SLM receives quality level information of current time delay which is decided by DA, and then SLM applies the speed limit rules accordingly. This guarantees the robot is running in a proper speed that meets the restriction of path error tolerance at every turning/stop point. Then a continuous control of robot motion with “Speed Control” strategy is achieved.

DA is working like a watchdog inside the system; it provides the delay zone change to system dynamically. SLM receives quality level information of current time delay which is decided by DA, and then SLM applies the speed limit rules accordingly. This system has been successfully launched at LISyC of UBO, France. The system has three parts,
including Client, Server and Controlled system (two wheeled robot). SLM and DA are implemented inside Tool Interface Block between Server and Controlled system.
Chapter 4

Experiment
The Miabot has been launched successfully at LISyC of UBO, France. The launch of the robot is the first phase of project. After launch the robot based on Saturne system, the following step is to improve the algorithm in the control strategies, and then SLM and DA based architecture are introduced. The SLM and DA have been merged into the system successfully. The Saturne system has the feature “plug and play the device”, so the whole project is to write a “driver” for Miabot, and most of programming are in this part. The improved algorithm (SLM, DA), which aim to reduce the influence the Internet delay, is also included in the “driver”. The reason is that LISyC want lest modification of previous Saturne system since it’s running in stable for years.

![Miabot user interface](image)

Figure 13: Miabot user interface

### 4.1 Feasibility test of Miabot application

The initial idea is to launch a remote control of mobile robot application, so the first experiment in Miabot project is to test the feasibility of remote control of the Miabot. In Figure 13, it shows the user interface of Miabot application (if you have interest, you can...
visit http://similimi.univ-brest.fr/v1.6/miabot to have a live control).

The Miabot application has been launched for around 5 months; many users visited the project website and used Miabot application during this period of time. In the past months, some modifications and improvement are done to the application system; the application system is running stable twenty-four hours a day currently. The followings are some problem encountered as well as improvement.

### 4.1.1 Twenty-four hours online a day

In the original Miabot set, the battery can only support two hours working of the robot. This problem also happened in famous Xavier project, and this problem is regarded as main obstacle which prevents the further user of the system. A fixed working time table for the system is a solution, but the battery still need to be charged regularly. It’s not possible for us to remember it everyday, it indeed result in failing to open the system to the user in the limited working time table.

The Miabot is working in a small area which is a rectangle area. Since the working area of Miabot is small, and it’s possible to plug the power charge line when it’s working. The solution is considered, but it still doesn’t work. Because when you plug the power charge line, the robot will be only charged once. After the battery is fully charged, there is no charge any more. Now, it seems that the working time of the application is a big problem. But later, based on the experience of many times’ charging operation of the robot, we find out that if the power charge line is disconnected to the robot and plug it again, the robot can be charged again. So a power timer is used in this situation, the power charge line is connected to a power timer. The power timer will switch the power periodically (turn off first, and turn on in the next second), and then the continuous charge to the robot is achieved, the Miabot can work twenty-four hours a day.

Now the application is online twenty-four hours a day, more users can use the system as
they want; especially the place has time difference with Europe. More tests of the system means more feedback are expecting to receive, this will definitely help the improvement of the system. The project aims to be provided a teaching resource to the student, with the ability of working twenty-four hours a day, there are more flexibility for students to explore this system.

4.1.2 Continuous command

There is no buffer in the previous system to store the command received from the user, and the Saturne system just forwards every command to the Miabot directly. In the early application, we find out that if continuous command are addressed from user, and if the interval time between command are small, that is to easy, the second command is sent to the Miabot before it fully perform the first one. In the direction commands, this is allowed to happen since user can change the direction frequently. But in the turning degree command (e.g. turn right 90 degree), the whole turning action takes a little while, if the Miabot receives another command before its fully turning, it will perform next command. This will be a problem in the real operation. In order to keep the flexibility in the direction command as well keep a fully performance of turning command, the system will buffer the command if previous command is a turning one until reaching the required action time. The required turning action time is calculated based on the mechanism of the robot and turning degree it requires.

The feasibility of the Miabot application is obvious, but the system should be put online for more test as to improve the performance.

4.2 Effectiveness test

The Effectiveness test of Miabot application refers to the test of Miabot application system when facing unpredictable Internet delay. The target is to validate the SLM and DA based architecture is effective to reduce the influence of unpredictable Internet time...
delay.

This will be divided into two phases. Users will visit and use the system from all over the world, and feedbacks are expected to receive from them. In the same time, some experiment will be performed locally to test the system, namely, simulated delay will be added to the local user, and the data will also be collected and analyzed to improve the system. The pre

4.2.1 Parameter choice and theoretical analysis

In this experiment, a simple SLM mechanism is chosen. The consideration is to simplify the test as well as expecting the positive result to validate the effectiveness of SLM and DA based architecture while facing unpredictable Internet time delay. The sub tolerance is calculated as:

\[ \Delta \delta_i = \frac{\pi \cdot r \cdot v_i \cdot t_i}{60 \cdot 1000} \]

Where \( r \) is the diameter of robot wheel, \( v_i \) is the Max Speed of \( Q_i \), \( t_i \) is the total working time in \( Q_i \), and then the total tolerance \( \varepsilon \) is:

\[ \delta = \sum_{i=0}^{n} \Delta \delta_i \]

The diameter \( r \) of the robot wheel is 52 mm, and the corresponding Max Speed of each quality level and its time delay zone have been chosen as the following Table 6, and the sub tolerances are calculated based on the chosen time delay zone and Max Speed.

<table>
<thead>
<tr>
<th>Qual.</th>
<th>Time D. (ms)</th>
<th>( t_i ) (ms)</th>
<th>Max Speed (rpm)</th>
<th>Sub T. (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Högskolan Dalarna</td>
<td>52</td>
<td></td>
<td></td>
<td>Tel:+46-23-778800</td>
</tr>
<tr>
<td>Röda Vägen 3</td>
<td></td>
<td></td>
<td></td>
<td>Fax:+46-23-778050</td>
</tr>
<tr>
<td>S-781 88 Borlänge</td>
<td></td>
<td></td>
<td></td>
<td><a href="http://www.du.se">http://www.du.se</a></td>
</tr>
</tbody>
</table>
In the above SLM parameter table, the sub tolerances are calculated according to the parameter choice of each quality level. The total tolerance of this SLM is 98 mm. The global highest speed in this SLM is 110 rpm.

$$Q_0 \quad 0 \sim 150 \quad 151 \quad v_0 = 110 \quad 45$$

$$Q_1 \quad 151 \sim 400 \quad 250 \quad v_1 = 60 \quad 40$$

$$Q_2 \quad 401 \sim 650 \quad 250 \quad v_2 = 20 \quad 13$$

$$Q_3 \quad 651 \sim 900 \quad 250 \quad v_3 = 0 \quad 0$$

$$Q_4 \quad \geq 901 \quad 0 \quad v_4 = 0 \quad 0$$

In this SLM, the worst case (theoretically) is the quality level is changing from 0 to 4 (disconnection), and the maximum path error distance is 98 mm. The Table 7 is a comparison between the system with and without SLM and DA.

<table>
<thead>
<tr>
<th>Speed (rpm)</th>
<th>Time (ms)</th>
<th>Worst Path error without SLM and DA (mm)</th>
<th>Worst Path error with SLM and DA (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>900</td>
<td>269</td>
<td>98</td>
</tr>
<tr>
<td>60</td>
<td>900</td>
<td>147</td>
<td>77.5</td>
</tr>
<tr>
<td>20</td>
<td>900</td>
<td>49</td>
<td>49</td>
</tr>
</tbody>
</table>

In the Figure 14, the difference of worst path error in the situation with or without SLM and DA is obvious. The SLM and DA based architecture can reduce the worst path error significantly. The SLM can be defined according to the application and it’s quite flexible. So the SLM and DA based architecture can reduce the influence of Internet time delay. Another import idea is that, in the SLM and DA based architecture, the robot will stop if the connection with user deteriorates to the disconnection level, but the robot in the system without DA and SLM will continue the previous running and it’s dangerous in some cases.
4.2.2 Practical test and analysis

The practical test follows the SLM parameters in section 4.2.1, and there are two tasks designed for the practical test. The tests will be focused on stable Internet delay (fixed delay time) and unstable Internet delay (changeable).

4.2.2.1 Stable delay test

In there two similar tasks (Figure 15, Figure 16): task one and two, the experiment results
are similar, only the result of task one is shown and discussed later.

![Figure 15: task one of practical test](image1)

![Figure 16: task two of practical test](image2)

In figure 15, this is a task which requires the robot to run a circle along the stake A, B, C, D. It starts from stake A and returns to A finally. The SLM has been chosen in Table 6,
and the running speed in this task is fixed to 60 rounds per minute (rpm). The reason to choose this speed has two reasons: 1. if the speed is too fast, user can’t react and finish the task even there is no Internet delay; 2. the chosen speed should clearly reflect the effectiveness of SLM scheme.

The first test is to let the Internet time delay fixed to 0. This is tested in the local network, and this is only an ideally assumption, there is delay even it’s local network. But, these delay are small compared to the delay zone in the $Q_0$ here. The result of the test is shown below in Table 8.

<table>
<thead>
<tr>
<th>Turning (degrees)</th>
<th>Succeed (times)</th>
<th>Fail (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>48</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>93</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The fixed running speed is 60 rpm, and this speed is lower than the Max Speed of $Q_0$. The robot is running with speed 60 rpm all the time (the local network has no significant traffic in the test).

The original chosen degree is 30, 45 and 90, but all the tasks have been failed in the beginning. The reason is the degree lost in the real turning movement due the friction of wheel movement and mechanism of the robot. So the degree lost is found within 2 ~ 4 degree empirically after many tests. Because the task is performed in a short and there is no enough time to change the turning degree parameter, all the turning degree is fixed in the test.

In the test result there are some fails in the degree 30 and 45. The reason is the user isn’t familiar with the task and it requires turn 30 degree 3 times to reach the goal to turn a 90 in total. This operation may result in the inaccurate in the turning action.

The second test of task one is to set the delay time to the fixed value 250 ms, so the robot
is running in the $Q_1$. This test is also achieved locally and a constant 250 ms is added to the network delay. Because the Max Speed of $Q_1$ is 60 rpm, the robot can still run in 60 rpm all the time. The test result is shown in Table 9 below:

<table>
<thead>
<tr>
<th>Turning (degrees)</th>
<th>Succeed (times)</th>
<th>Fail (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>93</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The test result is better than last, because the user is familiar with this task and can indeed have some experience. But the turning degree still fails some test.

The third test of task one is set the delay time to the fixed value 501 ms, so the robot is running in $Q_2$. Because the Max Speed of $Q_2$ is 20 rpm, the robot can only run in 20 rpm all the time. The test result is shown in Table 10 below:

<table>
<thead>
<tr>
<th>Turning (degrees)</th>
<th>Succeed (times)</th>
<th>Fail (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The test result is quite good, when the running speed is slow; the user definitely has a very good control of the robot. The turning degree has comparatively less influence to the control when the robot speed is slow.

### 4.2.2.1 Changeable delay test

The last test of the task one is with a changeable Internet delay. It is tested locally, and a periodically changeable delay is added to the network and achieves the simulated environment. The simulated delay works as in Figure 17. The robot is working in the
$Q_0$ (with delay 0) from 0 ~ 150 ms; in the $Q_1$ (with delay 151) from 151~400 ms; in the $Q_2$ (with delay 401) from 401~650 ms; in the $Q_3$ (with delay 651) from 650~900 ms; in the $Q_4$ (with delay 901) from 900~1350 ms. This is a delay simulation loop, the it generates the Internet delay periodically.

Figure 17: Simulated changeable Internet delay

In Figure 18, it’s obvious that the chosen speed 60 rpm is no more than the Max Speed of $Q_0$ and $Q_1$, and the robot runs in 60 rpm from 0 ~ 400 ms. The robot runs in 20 rpm in $Q_2$ from 401 ~ 650 ms, and stops in $Q_3$ and $Q_4$ from 651 ~ 1350 ms.
The test has been done based on the task one, the result is shown below in Table 11.

Table 11: test result of task one when delay is changeable

<table>
<thead>
<tr>
<th>Turning (degrees)</th>
<th>Succeed (times)</th>
<th>Fail (times)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

The result shows that the test is very successful in the changeable delay simulations. The robot speed is slowing down until the delay quality becomes better again, this will surely...
solve the situation when the user suddenly disconnect with the system, the robot will slow down until to stop at this circumstance.

The test is performed locally with simulated Internet delay; it aims to simulate some cases of users with different connection condition.

Based on the result of stable delay (fixed delay value) and changeable delay test, the SLM and DA based architecture shows the effectiveness in the real application.

The project has two main phases. The launch of Miabot application based on Saturne system is the first phase, it has been accomplished successfully. The system is online twenty-four hours a day stably. Many users have visited the system and many valuable feedbacks are collected. So the system is improving all the time.

The second phase is the effectiveness test. Although some experiments have been performed and relevant data are collected to improve the system, it’s not enough since the system need a long period of time in many aspects. The next step is to invite user from different part of the world to test the system and improve the system. It takes a long time to finish this, so the system is believed to be effective now based on the theoretical analysis and limited experiments.
Chapter 5

Conclusion and Future work
5.1 Findings and Contributions

This project has launched a remote control of two-wheeled mobile robot via the Internet connection. This application system is tested, and the application is successful. Now it’s used as a teaching resource at informatics department of UBO, France.

In the same time, according to the Miabot project, the Saturne system is validated as a successful Internet teleoperation system. The initial launch of the Miabot application takes around one month and a half by a person who has no previous knowledge of system. The feature “plug and play the devices” is very efficient in the Saturne system. A new application can be launched with less modification/creation of system. It’s more like to write a “driver” for the application.

Meanwhile, the LISyC has previous investigated the Internet connection condition in Internet teleoperation, and found out that the RTT is decreasing and stable. The feasibility of Miabot application with direct continuous control scheme has demonstrated correctness of the previous study. Finally, the SLM and DA module which are introduced in the control algorithm helps the system to reduce the influence of unpredictable Internet delay. The test result has shown the effectiveness of the method theoretically and practically.

5.2 Applications

The Saturne is system is good choice when launching a new Internet telerobotics application. It takes short time for a people to finish without previous knowledge to finish the application. The system is working in stable during the long period of test.

5.3 Future work

The prior work need is to continue the effectiveness test. The target is to invite different
users from different part of the world (normally they have different connection speed to the server at LISyC). After getting the feedbacks of these users, more information can be collected to improve the system stability and effectiveness when facing unpredictable Internet delay. In the same time, the system can collect visitors connection history automatically, so the statistical result can be observed in the future.

In the proposed SLM and DA based architecture, there is only one SLM and it’s only suitable for a user with certain delay. The future consideration is to have many SLMs in the system in the same time. Different user groups which are distinguished by average Internet delay may be allocated with different SLM which have the better service for user group. This can be done dynamically, even though the SLM for the same user may be changed if his average Internet time delay has a significant change which indicates another SLM will be chosen to him dynamically as to provide better service to him. In the same time, the SLM module can also be improved after further study.

In the future, to integrate more robots in the system simultaneously to have robot collaborative research is also a good direction as well as adding sensor and actuator to Miabot.
Reference


Francisco, CA (2000).


University of Western Australia (1999).


[34] “Mecury porject”, http://www.usc.edu/dept/raiders.


Appendix A
Related Publications


Source code sample

```java
//MiabotToll.java
//Created by Anson, last edit date: 2007-03-25
package javaprod.tool;

import java.io.*;
import java.util.*;
import javaprod.system.*;
import javax.comm.*;
import javaprod.tool.*;

public class MiabotTool extends ToolInterface implements MiabotData, SystemData {

    protected boolean _open = true;

    protected PipedInputStream _pipeSCEtoTI;
    protected PipedInputStream _pipeWDtoTI;

    protected MiabotSerial _serialComMiabot;
    protected Watchdog _watchdog;
    // protected int _nbTools;

    protected int _GemmaStatus = 0; // set initial connection in zone 0;
    protected int lastCommand = Stop;
    protected String _speed = "0";
    protected String _lastSpeed = "0";

    protected int[] _maxSpeed = {110, 60, 20, 0, 0, 0};

    /**
     * constructor.
     **/
    public MiabotTool(int indexTool, PipedInputStream pipeTItoDM, PipedOutputStream pipeDMtoTI,
                        SystemSemaphore semDM, SystemSemaphore semTI) throws ExceptionTeleproductic {

        try{
```

Högskolan Dalarna 71 Tel:+46-23-778800
Röda Vägen 3 Fax:+46-23-778050
S-781 88 Borlänge http://www.du.se
// System.out.println("[Miabot] Init Miabot");
//
// try {
//    _nbTools = Integer.parseInt(properties.getString("NbTools").trim());
// }
// catch(Exception e) {
// }

_indexTool = indexTool;

_pipeTItoDM = new PipedOutputStream(pipeTItoDM);
_pipeDMtoTI = new PipedInputStream(pipeDMtoTI);
_pipeSCetoTI = new PipedInputStream();
_pipeWDtoTI = new PipedInputStream();

_semDM = semDM;
_semTI = semTI;

_serialComMiabot = new MiabotSerial(_pipeSCetoTI, _semTI, _indexTool);

_watchdog = new Watchdog(_pipeWDtoTI, _semTI);

this.start();

} catch(Exception e) {
  trace.add("[Miabot] Error when creating Miabot");
  closeMiabot();
  throw new ExceptionTeleproductic();
}

public void run() {

  try {
    trace.add("[Miabot] Start Miabot");

    while (_open) {

      Message.waitMessage(_semTI);
      //System.out.println("Try to Analyse");
      if (_pipeSCetoTI.available() > 0)
        analyse(new Message(_pipeSCetoTI));
      else if (_pipeDMtoTI.available() > 0)
        analyse(new Message(_pipeDMtoTI));
      else if (_pipeWDtoTI.available() > 0) {
        //System.out.println("[Miabot] receive information from Watchdog");
        analyse(new Message(_pipeWDtoTI));
    }
  }
}
try {
    Message message = new Message(TOOL_INTERFACE, _indexTool, DEVICE_MANAGER, NOT_AFFECTED, END, NO_VALUE);
    message.sendMessageV(_pipeTItoDM, _semDM);
} catch (Exception ee) {}

closeMiabot();
}

void analyse(Message message) throws Exception{
    int dataCtrl = message.GetDataCtrl();
    String dataValue = message.GetDataValue();
    String dataValueCopy = dataValue;
    String finalCommand = "";
    int timer = 0;

    //System.out.println("Sending command" + dataCtrl);
    if(dataCtrl == Left || dataCtrl == Right || dataCtrl == Forward
        || dataCtrl == Back || dataCtrl == NoStopForward || dataCtrl == NoStopBack ||
        dataCtrl == Stop) {
        StringTokenizer parserTmp = new StringTokenizer(dataValueCopy, "@");
        String tmp = parserTmp.nextToken();
        _speed = parserTmp.nextToken();
        //System.out.println("[MiabotTool] " + _speed + _speed);

        StringTokenizer parser = new StringTokenizer(dataValue, "]");
        String degree = parser.nextToken();
        double timerTmp = Integer.valueOf(degree.substring(3, degree.length())) * 27 / 400 +
            1;
        timerTmp = 6.6 * timerTmp;
        timer = (int) timerTmp + 50; //secure enough turning time
        //System.out.println("[MiabotTool] " + degree + " -- " + timer);

        if(dataCtrl == Forward || dataCtrl == Back) {
            if(dataCtrl == Forward) {
                finalCommand = message.GetDataValue() + "]"
            }
            lastCommand = Forward;
        }
    }

else {
    finalCommand = message.GetDataValue() + "[v];
    lastCommand = Back;
}

_serialComMiabot.sendStringMiabot(finalCommand);
}
else if (dataCtrl == Left || dataCtrl == Right) {
    if(dataCtrl == Left)
        finalCommand = message.GetDataValue() + "[<];
    else
        finalCommand = message.GetDataValue() + "[>];

    _serialComMiabot.sendStringMiabot(finalCommand);

    try{
        Thread.sleep(timer);
    }
    catch (InterruptedException t) {
        System.out.println("[MiabotGUI] Sleep Error");
    }

    if(lastCommand == NoStopForward) {
        finalCommand = "[=" + _lastSpeed + "," + _lastSpeed + "]";
        _serialComMiabot.sendStringMiabot(finalCommand);
    } else if (lastCommand == NoStopBack) {
        finalCommand = "[=-" + _lastSpeed + ",-" + _lastSpeed + "]";
        _serialComMiabot.sendStringMiabot(finalCommand);
    } else {
        if(dataCtrl == Left)
            lastCommand = Left;
        else
            lastCommand = Right;
    }
}
else if(dataCtrl == NoStopForward || dataCtrl == NoStopBack) {
    int thisMaxSpeed = _maxSpeed[_GemmaStatus];
    if(Integer.valueOf(_speed) <= thisMaxSpeed)
        _lastSpeed = _speed;
    else
        _lastSpeed = String.valueOf(thisMaxSpeed);

    if(dataCtrl == NoStopForward) {
        finalCommand = "[=" + _lastSpeed + "," + _lastSpeed + "]";
        lastCommand = NoStopForward;
    } else {
        finalCommand = "[=-" + _lastSpeed + ",-" + _lastSpeed + "]";
        lastCommand = NoStopBack;
    }
```java
    _serialComMiabot.sendStringMiabot(finalCommand);

} else if (dataCtrl == Stop) {
    //System.out.println("Sending Stop Command!!!!");
    finalCommand = "[s]";
    _serialComMiabot.sendStringMiabot(finalCommand);

    lastCommand = Stop;
    _speed = "0";
    _lastSpeed = "0";
}

Message messageToSend = new Message(TOOL_INTERFACE, _indexTool,
    TOOL_GUI_BROADCAST_WITH_CONTROLER, _indexTool,
    Miabot_RECEIVED, NO_VALUE);
messageToSend.sendMessageV(_pipeTItoDM, _semDM);
} else if(dataCtrl == NEW_DELAY_ZONE){
    _GemmaStatus = Integer.valueOf(dataValue);
    //System.out.println("[MiabotTool] Now you are in zone" + _GemmaStatus);

    //Speed Limit

    int thisMaxSpeed = _maxSpeed[_GemmaStatus];
    if(Integer.valueOf(_speed) <= thisMaxSpeed)
        _lastSpeed = _speed;
    else
        _lastSpeed = String.valueOf(thisMaxSpeed);

    if(lastCommand == NoStopBack) {
        finalCommand = "[-=+ _lastSpeed + ","] + _lastSpeed + "]";
        _serialComMiabot.sendStringMiabot(finalCommand);
    } else if (lastCommand == NoStopForward) {
        finalCommand = "[=+- _lastSpeed + ","] + _lastSpeed + "]";
        _serialComMiabot.sendStringMiabot(finalCommand);
    }
}
else if(dataCtrl == GET_PING){
    /*synchronized (_watchdog) {
        _watchdog.pleaseWait = false;
        _watchdog.notify();
    }*/
    _watchdog.resume();
    //System.out.println("[MiabotTool] Get Ping ");
} else if(dataCtrl == GET_PONG){
    /*synchronized (_watchdog) {
        _watchdog.pleaseWait = false;
    }*/
    _watchdog.resume();
    //System.out.println("[MiabotTool] Get Ping ");
```
_watchdog.wait();
*/
    _watchdog.interrupt();
    //System.out.println("[MiabotTool] Get Pong ");
} else if (dataCtrl == END) {
    closeMiabot();
}

/*if (dataCtrl == DELAY_TIME_OUT) {
 *
    Message messageToSend = new Message(TOOL_INTERFACE, _indexTool,
    TOOL_GUI_BROADCAST_WITH_CONTROLER, _indexTool,
    CTRL_RELEASE_OK, NO_VALUE);
    messageToSend.sendMessageV(_pipeTItoDM, _semDM);
}*/

void closeMiabot() {
    try{
        _open = false;
        trace.add("[Miabot] Miabot is ending....");
        _serialComMiabot.close();
        _pipeSCEtoTI.close();
        _pipeWDtoTI.close();
        trace.add("[EVID31] Miabot ended.... bye");
    } catch (Exception e) {
        trace.add("[Miabot] Can not close the pipes on Miabot.");
    }
}