UNIVERSITÀ DI PISA

Dipartimento di Oncologia, dei Trapianti
e delle Nuove Tecnologie in Medicina

Corso di Dottorato in Tecnologie per la Salute:
Valutazione e Gestione delle Innovazioni nel Settore Biomedicale

XIX Ciclo

Ph.D. Thesis

Platforms for Prototyping Minimally Invasive Instruments

Francesco Focacci

Ph.D. Coordinator:

Prof. Andrea Pietrabissa

Tutors:

Prof. Paolo Dario

Dr. Oliver Tonet

March 2007

Contents

Acknowledgements

I would like to extend my sincere thanks to Professor Andrea Pietrabissa, Professor Paolo Dario, Dr. Oliver Tonet, Dr. Giuseppe Megali and to the members of the CRIM Lab of Scuola Superiore Sant'Anna.
I would like to thank Marco Piccigallo and Claudio Quaglia that are the designers of the mechanical parts used in this work. I would like to thank also Luca Citi, Martina Marinelli, Lorenza Mattei, Filippo Cavallo, Cinzia Freschi and Stefano Sinigaglia.
This work has been supported in part by the FIRB-2001 Project "ApprEndo" (no. RBNE013TYM) and by *EndoCAS*, the Centre of Excellence for Computer-Assisted Surgery (COFINLAB-2001 no. CLAB01PALK), both funded by MIUR, the Italian Ministry of Education, University and Research.

Summary

The introduction of new technologies in medicine is often an issue because there are many stages to go through, from the idea to the approval by ethical committees and mass production. All projects start with the problem analysis and very often in the academic field off-the-shelf solutions are not suitable. This is not only a drawback but represents a big opportunity to investigate new technological paths and to try to add new contributions to research. The first steps are related to the definition of requirements and specifications, which is one of the most time consuming parts. An in-depth technological investigation has to be carried out together with the study of commercial existing instruments, including patents, and research activities.

This work covers the first steps of the development of a medical device, dealing with the tools that can help to reduce the time for producing the laboratory prototype. These tools can involve electronics and software for the creation of a "universal" hardware platform that can be used for many robotic applications, adapting only few components for the specific scenario. The platform is created by setting up a traditional computer with operating system and acquisition channels aimed at opening the system toward the real environment. On this platform algorithms can be implemented rapidly, allowing to assess the feasibility of an idea. This approach lets the designer concentrate on the application rather than on the selection of the appropriate hardware electronics every time that a new project starts. Requisites, specifications and components can change but the platform on which to create the prototype does not need to be modified. After completing the first prototype with external controller, the control system can be embedded: when the specifications and the components such as motors and sensors are definitive then the design of the embedded hardware can start. The designer selects a microcontroller or microprocessor chip, designs the supporting electronics around it. In many cases the created instrument has to be portable and handy for the medical doctor.

This work treats the study and the development of robotic instruments for minimally invasive surgery (MIS). These devices have motors and sensors in close contact with the patient and that can be controlled autonomously, this means that hazards exist. Robotic systems produce large amount of heat, large voltages and movements of mechanical parts that can cause harm. The control part relative to motors movements or sensors reading needs to be tested. In a perfect world motors coupled with the mechanical parts as well as sensors have ideal behavior, everything that is designed on the paper is perfectly translated into practice. Unfortunately all the parts

that compose a robotic device are affected by noise or we may simply not know the exact behavior of the system that we are trying to control. Different strategies have to be implemented for testing purposes and if this step is not sufficient, changes in project specifications have to be performed, for instance changing actuators or sensors. It would be useful to be able to experiment different solutions just after the ending of the design phase or even better when the design phase is still running. The time needed for passing from the design to the first prototype is not negligible because often in the medical field, and particularly for minimally invasive surgery, mechanical parts are very complicated in terms of size, shape and materials. A five years period from the concept idea to the real functioning laboratory prototype is not uncommon.

The thread of this work concerns the development of robotic instruments for laparoscopy trying to obtain a platform modular, efficient and easy to use, able to deal with all the parts that compose a robotic system, such as motors, sensors and user interfaces. The aim is to speed up the very early stages of the development of a new robotic instrument. In the first part an overview of the existing instruments for minimally invasive interventions that can be found as commercial or research products is given. An introduction related to hardware electronics is presented with the requirements and the specific characteristics needed for a robotic application. The second part focuses on specific projects in MIS. The hardware used for the projects is described together with the software tools used to develop applications that manage the inputs and the outputs needed for the robotic systems and the user interfaces. Subsequent chapters describe each project in detail.

The first project concerns the study and the development of a lightweight hand-held robotic instrument for laparoscopy. Motivations are related to the lack of dexterous hand-held laparoscopic instruments. Teleoperated robotic systems bring some advantages at the cost of longer setup times and of cluttering the already crowded operating table, pushing the surgeon away from the patient: the surgeon operates at a console in a corner of the operating room and only the robot is in direct contact with the patient. Experienced surgeons tend to agree that in many procedures the benefits provided by those teleoperation systems are not really needed during the whole surgical procedure, and they tend to prefer the traditional hands-on approach for routine tasks. Led by these considerations, the aim of this project concerns the develop of a hand-held surgical robot that can be operated by a surgeon with only one hand, while the other hand is free to use a traditional endoscopic instrument.

The second project concerns the study and the presentation of a prototype of a robotic endoscope with enhanced resolution. Visual acuity of artificial vision systems in endoscopy surgery is currently well below the human eye and this is due to the limited resolution of both cameras and displays. The application field of the system could be laparoscopic surgery, in which it achieves about 7 times higher resolution than typical commercial systems.

The third project concerns the development of a system able to detect the inspiration and the expiration phases. The analysis of the breathing frequency is useful

for surgeon gesture evaluation. The aim is to evaluate the weariness of the surgeon, since breathing can be related to fatigue. This system could also be used for the ergonomic assessment of new surgical instruments.

Final considerations concern the approach used for the development of the prototypes. The aim is to rapidly implement and test control algorithms, virtual environments and custom electronics dedicated to the instruments. Typical microcontroller boards require less than one hour of components soldering and microcontroller software implementation requires only the time needed to write the program. The computing power of a PC is also of immediate use with the high flexibility furnished by the data acquisition boards used for the projects. The main advantage is the possibility to concentrate efforts on application development without worrying about the hardware.

Chapter 1

Instruments for minimally-invasive surgery

1.1 Introduction

Laparoscopic surgery has had one sensitive increase in the last ten years. In 1992 the 70% of the interventions to the gall bladder in the United States, Europe and in Japan were carried out with the laparoscopic technique. The fundamental advantages, respect to classic surgery, are constituted by a minor post-operative trauma and a better recovery time. The patients have contributed in a meaningful way to the success of such technique, in fact it has been an increasing demand on dealing the pathologies with mimimally invasive techniques. Unfortunately the costs are not diminished with the advent of laparoscopy. Moreover the surgeon has lost skill, tactile feedback and a perfect coordination between hands and eyes, peculiarities that the surgeon had with the traditional techniques. These problems are due to the construction of laparoscopic instruments. The studied solutions in order to try to improve the laparoscopic instruments are essentially two: the development of flexible mechanical instruments and the study of computer-assisted devices and surgeon aid-systems. With these last systems the surgeon works to an interface while the tip follows its movements or executes predetermined actions in an autonomous way. The computer-assisted instruments are often cumbersome and much expensive, for which theirs use it is limited some surgical fields. In the next few years, probably, we will assist to the substitution of the rigid laparoscopic instruments with new ones that, thanks to the computer aid, will allow the surgeon to gain sensibility through force feedback, at the same time with lower costs and an easy way to sterilize, to transport and to prepare [5].

1.2 History

The history of laparoscopic surgery begins perhaps with the first speculum, the rectal one, invented from Ippocrate (460-375 a.c.), which dealt some internal pathologies

with minimally invasive techniques. The speculum is an instrument that is used in order to watch inside of cavity, and is constituted from two or three metallic sheets, that concur to delicately spread the tissues. In the ruins of Pompei a three valve copy was found, demonstrating the interest of the doctors for the study of inner organs. The true endoscopy begins with the Filippo Bozzini's invention (1773-1809). He developed an apparatus, called "Lichtleiter", that allowed to illuminate in indirect way the tip of the instrument and to bring back the image to the examiner's eye. The apparatus of Bozzini never had clinical employment for rivalry within the Medical Academy of Vienna, in which the instrument was introduced.

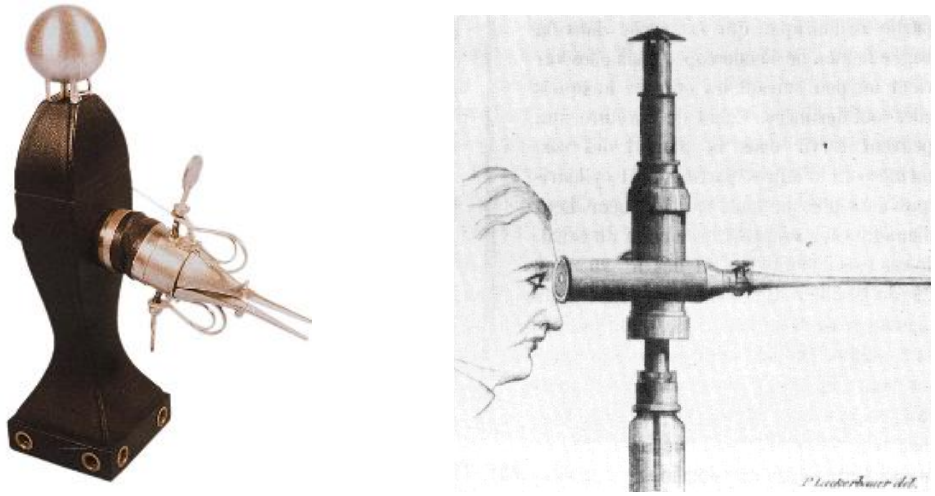


Figure 1.1. Lichtleiter Endoscope

From Boston, John D. Fisher (1798-1850) described an endoscope, used initially for the vagina, but subsequently modified for the exploration of the bladder. In 1853, a French surgeon, Antoine Jean Desormeaux, first employed the Bozzini's Licthleiter on a patient. The instrument, through a series of mirrors carried the light of a flame, obtained from the combustion of alcohol and turpentine. The instrument therefore reached very hot temperatures and burns were the more frequent complications, for this the employment was limited. After the discovery of the incandescence bulb Maximilian Nitze (1848-1906) elaborated some changes to the Bozzini's endoscope but only in 1883 Newman, in Glasgow, succeeded to insert in the endoscope tip one miniaturized light bulb. In 1901, George Kelling, surgeon in Dresda, coined the term of "coelioscopie", in order to indicate the technique that allowed to examine to the abdominal cavity of some dogs using a cystoscope. Kelling obtained a pneumoperitoneal insufflating air through a sterile gauze filter. In literature it is unknown if he never employed this technique on the man. To the end of 1910 in Stockholm HC Jacobaeus published in the review Munchener Medizinische Wochenschrift an article on the employment of the "laparothorakoskopie", in which an endoscope instrument was employed also in the thorax. Kelling and Jacobaeus contended for the paternity of the instrument and its employment, even if Kelling never did not succeed

to demonstrate to have really practiced it on the man. In John Hopkins Hospital, Bertram M. Berheim become acquainted of the studies of Jacobaeus and Kelling and he performed the first laparoscopy intervention in the United States, using, in 1911, a proctoscopio. A radiologist named Eight Goetzes developed an instrument for the creation of the pneumoperitoneus, and he predicted the use of it in laparoscopic field. In 1920 B.H.Orndoff, internist in Chicago, reported the first series of 45 cases of diagnostic peritoneoscopies, and he invented a trocar with a trunk of cone head. In the same period the applications and the innovations were succeeded. Heinz Kalk, founder of the German school of Laparoscopy, in 1929 introduced an endoscope with 135 degrees lens and an access with multiple trocar. J.C.Ruddok, internist, developed a pliers able to coagulate. In 1938 the hungariam Janos Veress developed a needle with a safe head, that had conceived for practising the therapeutic pneumothorax in the tuberculosis. He used it in over 2000 cases, but he never suggested the employment of it in laparoscopy. In 1944 Raoul Palmer effected the first pelvic exploration for the study of uterus and ovaries, employing the position of Trendelemburg. The most important invention was perhaps made by Harold H. Hopkins, after 1945 and concerned optical fibers and cylindrical lenses. From that moment, for over 40 years the laparoscopic field was almost exclusive appanage of gynecologists and internists. In 1966 Kurt Semm, German gynecologist, invented the automatic insufflator and performed numerous interventions on the pelvis with laparoscopic technique. Semm also performed an appendectomy that was judged too much dangerous from the German academy of Medicine that expelled him. In the United States Semm had the opportunity to operate and to invent numerous tools for laparoscopic use.

1.3 Laparoscopic interventions

Currently the laparoscopic technique consists in the use of stiff and long instruments to effect the intervention: dissectors, forceps, hooks, pliers and clip appliers. These are inserted in the abdomen sliding them inside the trocars, after having "inflated" the abdominal cavity and made small incisions. Another important tool is the endoscope that, once introduced in the abdomen through a small incision, it transmits the bidimensional images of the organs on a video screen. There are endoscopes that have mounted ccd sensors that capture in digital format the image carried by the optical lenses. There are also endoscopes with the ccd sensor mounted on the distal part making unnecessary the use of the optical lenses. The surgeon can see the surgical field through a dedicated monitor. The other necessary tools are:

- a carbonic anhydride insufflator and an insufflator (Veress needle) needle to induce the pneumoperitoneus, this for inflating the abdominal cavity;
- a "cold light" bright source to bring the light on the laparoscope tip;
- special tools for irrigation and aspiration, useful when it is necessary to dampen the zone to be treated or to inhale possible losses of blood;

Figure 1.2. Single use dilating trocar

- electrocoagulation, for suturing the smaller vases;
- the trocars (with valves), through which introducing the tools.

The term "trochar" was coined in 1706; it derives from French "trois" (three) + "carre" (side). The trocar is a hollow cylinder with a sharply pointed end, often three-sided, that is used to introduce laparoscopic instruments into body cavities. The trocar is a portal for the subsequent placement of other instruments, such as a forceps, clip appliers etc (in Fig. 1.3 are represented some laparoscopic instruments).

In the operating room there two columns are attached to the ceiling, one for every side of the operating table. There is also a display commanded by the surgeon with which he checks all the desired parameters. The most important interventions performed with laparoscopic techniques are:

- cholecystectomy;
- nephrectomy;
- adrenal gland removal;
- left pancreas removal;
- appendectomy;
- inguinal hernia;
- gastro-esophageal reflux desease;
- cardio-thoracic surgery.

Figure 1.3. Laparoscopic instruments

Figure 1.4. Laparoscopic column

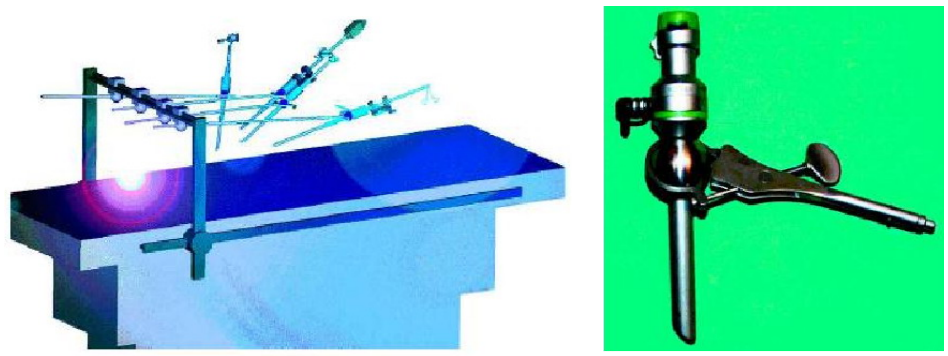
1.4 Mechanical manipulators

The development of mechanic instruments able to give to the surgeon more degrees of freedom than traditional laparoscopic instruments is still evolving in the study of devices that are more coherent with the needs of the physician. It is not in fact necessary for example to have great orientation angles in all the directions. The tendency is to project mechanical manipulators that are simple to use and to maintain. We can identify which are the fundamental characteristics([21]):

- diameter analogous to the commonly used laparoscopic instrumentation;
- instruments that are simple to sterilize;
- low cost;
- easy to use for the surgeon;
- affordable;
- possibility to connect with high frequency electrocauterizing instruments.

In some scientific articles are presented mechanical manipulators with deflectable tip. For instance in [21] it is described an endoscopic instrument that has the characteristic to have a deflectable articulation among $\pm 60°$. The tip is tilted through a handle with a sphere joint which rotates and transmits the motion to the final link. The system is entirely mechanic with cables that allow the articulation to steer. In this project the authors have focused their work on the possibility to simply clean and use it, considering that it exists only one command for orientation and rotation around the instrument axis. A further work is reported in [42], in which the authors have studied a support system for laparoscopic instruments. This allows to sustain all the instruments used by the surgeon. The system is integral with the operating table and in the point in which the single device is fixed it is present a "ball trocar" with an adjustable clutch. This allows the surgeon to regulate all the instruments in the desired position without the necessity to free them or to block them every time that are moved. In [42] it is developed a mechanical manipulator that consists of a handle inspired to the instruments used for open surgery. In fact the ergonomic principle is to transmit the movements of the surgeon's wrist to the deflectable tip. The critical aspects concern the precise transmission of the tip rotation: this requires rigidity to always give the feeling of immediateness in the control. The complete system (support system and manipulator) allows to have six degrees of freedom.

In commerce there are some devices that have a steerable tip. These are normal laparoscopic instruments to which it is added a mechanical mechanism that allows to earn two degrees of freedom (see Fig. 1.7). Another example of mechanical manipulator is the Radius Surgical System from Tuebingen Scientific Medical GmbH that has been designed to dramatically improve endoscopic suturing and other surgical maneuvers. Instruments with mechanical driving are not intuitive since the



(a) (b)

Figure 1.5. Support system described in [42](a) and ball trocar detail (b)

Figure 1.6. Radius Surgical System

mapping of degrees of freedom (DOFs) has to respect mechanical constraint and cannot be designed freely.

The development of the mechanical manipulators for endoscopic surgery seems to be interesting. Such tools can constitute a valid alternative to the robotic manipulators in applications where a microsurgery manipulation is not necessary and traditional rigid instruments are not completely satisfactory. The sphere of use could

Figure 1.7. ENDOPATH ETS-Flex Articulating Endoscopic Linear Cutters by Johnson & Johnson

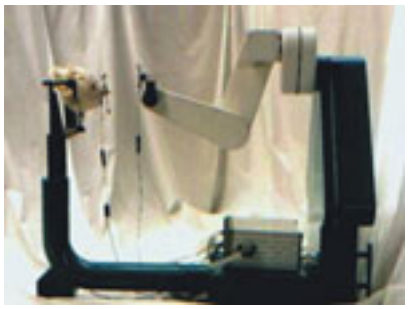


Figure 1.8. Neuromate by Integrated Surgical System

regard laparoscopic procedures for urological, gynecological and digestive tract.

1.5 Robotic manipulators

The use of robots in surgery is still been having a development from the half of the eighties. The number of robotic systems studied and developed in research fields is higher in comparison to the commercial offer. The advantages of a robotic system concern the accurate positioning, the ability to perform difficult trajectories and the repeatability of the movements. There is also the possibility to limit the use of the robot to a "safe" area for the patient [11]. The first robot interventions were carried out for neurosurgery applications and consisted in instrument positioning on the skull and then in instrument insertion to reach the area of interest inside the brain. A recent example of such type of system is the Neuromate (http://www.robodoc.com/) by Integrated Surgical System, that is used for neurosurgery (see Fig. 1.8). The robot is switched off after the positioning task and the physician can proceed with the manipulation of the instrument located at the end of the robotic arm (powered robot but passively used).

A different approach is constituted by the active robots that have a great construction complexity because they are in direct contact with the patient and they interact with him [18], in fact they continue to be operating during the whole intervention. Today this is the most used design and construction approach. Surgical robots can be classified in various ways, according to the design characteristics, to the degree of autonomy, to the anatomical district in which they can work or to the surgical technique. One possible classification can be made on the role of the robot inside the integrated surgical system [45]:

- CAD/CAM Surgical Systems;
- Surgeon aid systems.

For the first ones there is an integrated approach that expects a preoperative planning with integration and use of bi-dimensional or three-dimensional images together with other data relative to the patient. The aim is to recreate a model of the patient that can be used by the computer using images and other anatomical informations. In the operating room, during the intervention, there is the registration of the created model using on line data measured at the moment through a localization system: infrared, X rays or ultrasounds. Subsequently the surgical plan update is performed with the verification that the robotic procedure is correct, and after this phase the intervention is carried out. The difficulties of such procedure concern the different typologies of images to integrate (X rays, magnetic resonance, echography, etc.), preoperative as well as intra-operative. A suitable calibration is necessary to get good results in terms of precision and accuracy. Neuromate is an example of such category of systems. The aim of surgeon aid systems is to improve ergonomics in laparoscopic surgery and to give to the physician additional functionalities and abilities with respect to the traditional laparoscopic procedures. The application fields of such systems concern laparoscopy and microsurgery. An example is constituted by the "assistant" robot AESOP® that is able to sustain the endoscope and to direct it according to the directives transmitted by the surgeon with the voice (see Fig. 1.9). AESOP has a kinematic architecture with a passive remote center of motion (RCM), in fact the insertion point of the trocar into the patient, called pivot, is used to orient the endoscope. Such robot is composed from the first three active joints and from the last two passive rotational joints. The rotation is possible thanks to the constraint constituted by the pivot point. Other robots have different active RCM architecture, due to the different kinematic project; the rotation around the pivot is performed thanks to a parallelogram mechanical structure. Other examples of surgeon aid systems are constituted by teleoperated systems that usually have more capabilities than AESOP. They replicate the movements made by the surgeon augmenting his capabilities. A further classification can be made on the degree of "intelligence" of the system [16]:

- Hand-Held instruments;
- Teleoperated robotic systems for minimally invasive surgery;

Figure 1.9. AESOP by Computer Motion

- Autonomous surgical robots for minimally invasive surgery.

Hand-held instruments are instruments guided from the surgeon's hand and they are able of augmenting surgeon's capabilities, correcting his actions, amplifying or diminishing the interaction with the surgical field or restoring degrees of freedom. Hand-held instruments can be mechanical or mechatronic, for the last ones there are many research studies even if these are not still been using in the clinical practice. An example of hand-held mechatronic prototype is described in [13] and [17]. This instrument has been developed for arthroscopy and has one cable actuated steerable tip and incorporates sensors for tip position monitoring and for contact monitoring with surrounding tissues. The main feature of this arthroscope is that it is possible to implement a semi-automatic procedure in order to avoid contacts between the tip and delicate tissues like cartilages and ligaments that are selected during the preoperative phase. The complete system expects the construction of a model with diagnostic images that are used in the operating room and matched with the intra operative instrument localization. Micron [2] is another example of mechatronic instrument (see Fig. 1.10) and is designed in order to eliminate the surgeon's hand tremor in ophthalmological surgery. Equipped with an accelerometer and three gyroscopes it can reduce the tremor of 45% in monodimensional tests and 37% in three-dimensional tests. Teleoperated robotic systems are composed by two unit: the master, in which the surgeon operates and the slave, that is in direct contact with the patient. The surgeon carries out the tasks at the master console while the slave unit follows the movements of the surgeon. Finally the autonomous systems are able to carry out tasks without the participation of the surgeon. The tasks are well planned during the preoperative phase. Teleoperated and autonomous systems will be described more in detail later.

Figure 1.10. Micron

Figure 1.11. ROBODOC by Integrated Surgical System http://www.robodoc.com/

1.5.1 CAD/CAM systems

CAD/CAM and autonomous systems have functional analogies that concern the planning phase of the surgical intervention. As already pointed out the analogies are based on the integration of the 2-D and 3-D preoperative images with intra operative data. CAD/CAM systems constitute a category that can include autonomous systems in which the idea is to place the robot and then start the procedure that is executed autonomously by the robot while the surgeon monitors the procedure. The fields of application of these systems are those in which there are fixed structures like as an example in neurosurgery and orthopedic surgery. The surgeon plans during the preoperative phase the task that the robot will have to execute. ROBODOC by Integrated Surgical System (see Fig. 1.11) and CASPAR by U.R.S.-Ortho are examples of such type of systems and are used respectively for femur and knee in orthopedic surgery. The limb of the patient is fixed in a prearranged position and

Figure 1.12. CyberKnife by Accuray

therefore the cut of the bone can start. Recently a new system for orthopaedic surgery has been developed from the KAIST (South Korea), this is called Arthrobot and works on the femur. Although it has not been still used on real patients the experimental results are remarkable in terms of error and contact surface. CyberKnife (http://www.accuray.com/) by Accuray (see Fig. 1.12) is another example and is used for robotic radiosurgery.

On the robotic arm is mounted a linear accelerator and the system is designed to treat tumors anywhere in the body with sub-millimeter accuracy. Using image guidance technology and computer controlled robotics, the CyberKnife is designed to continuously track, detect and correct for tumor and patient movement throughout the treatment. Because of its extreme precision, the CyberKnife does not require invasive head or body frames to stabilize patient movement, vastly increasing the system flexibility. Unlike traditional radiosurgery systems that can only treat tumors in the head and neck, the CyberKnife can treat both intracranial and extracranial tumors. Other applications include spine, lung, prostate, liver and pancreas tumor. The autonomous systems for deformable organs endoscopy constitute an optimal opportunity in order to perform minimally invasive interventions in organs that vary their dimension. Preoperative images are not always reliable in order to take decisions on the intervention, it can be useful to have an autonomous robot that is able to interact with the environment. Cardiovascular catheters do not fall properly in this category even if they have been equipped with therapeutic instruments, sensors and shape memory materials in order to implement an active guidance system. The catheters in fact are controlled directly from the surgeon. In CAD/CAM category there are also robotic systems that can support and position an instrument that is in direct contact with the patient and the instrument is controlled directly by the surgeon. The instrument handle is situated at the end of the robot and is equipped with force sensors in order to follow the movements made by the surgeon. ACROBOT by ACROBOT Company (see Fig. 1.13) is another example of such robot systems and is designed to carry out knee prosthesis implant intervention. The active robot is

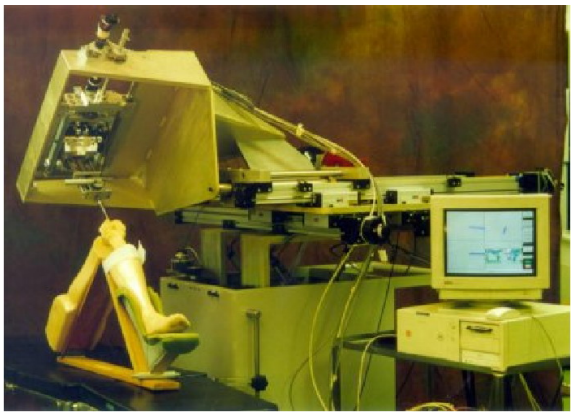


Figure 1.13. ACROBOT by ACROBOT Company

located at the end of the positioning system and is controlled by the surgeon. The operative workspace is confined, by hardware and software, to a certain volume in space. The device does not move autonomously, although it could be programmed to do so; it reacts to the actions of the surgeon holding the handle attached to the device. It aids motion, if the surgeon is moving the tool inside an allowed spatial volume; it prevents motion outside this volume called safety area.

The "active constraints" are defined in the preoperative phase, those zones cannot be reached from the instrument. Moreover the best robot positioning is decided also in this phase.

1.5.2 Surgeon aid systems, teleoperated robots

Teleoperated master-slave manipulators have the "master" console at which the surgeon works, while the "slave" part acts directly on the patient. Such type of approach is an open-loop control, closed from the surgeon through the display view. The tools on which the surgeon acts can be joysticks or ergonomic grips in order to simulate an instrument (see Fig. 1.14).

In commerce two products exist that operate with the master-slave modality: the ZEUS system by Motion Computer (see Fig. 1.15) and the daVinci surgical system by Intuitive Surgical Inc. (see Fig. 1.16). The master console of the daVinci surgical system (see Fig. 1.14) is composed by two grips that are used to read the position of the surgeon's hands and to map their position with the "endowrist" position in the slave part. The two "endowrists" are the end effectors of the robot. The endoscope has two integrated optical systems are used to reconstruct a three-dimensional image in the monitor of the master console. The vision cone is less than the one offered by a single optical endoscope.

Many teleoperated robot projects were aimed at the possibility to allow teleoperated interventions. The surgeon seat to the master console that could be in a

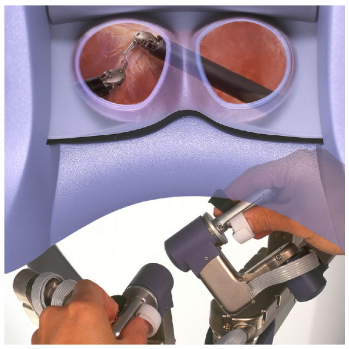


Figure 1.14. Master console of daVinci surgical system by Intuitive Surgical Inc.

Figure 1.15. ZEUS system

different place from that of the slave part and a remote connection is established between the two part of the robot. The delay time and the possible interruptions on the remote connection led to neglect the teleoperated interventions in favor of major ergonomics obtainable with master-slave robotic systems. Laprotek system by endoVia Medical (endovia.millersystems.com) is another system formed by a master console separated from the slave part (see Fig. 1.17). Compact drive units are attached to the sides of the operating table during use and are connected to the surgical arms by a cable mechanism that eliminates the need for electrical connections between the surgical arms and the computer control system. Sensors on the drive motors provide force feedback to the surgeon's control handles, thus providing tactile clues to help the surgeon manipulate the instruments.

There are also mechanical teleoperated systems and in particular in [26] is described a system for the control of two seven degrees of freedom endoscopic instruments (see Fig. 1.18). The system is fixed at the operating table, the surgeon is sitting and through the grips transfers the movement to the tip of the instruments.

Figure 1.16. daVinci surgical system by Intuitive Surgical Inc.

Figure 1.17. Laprotek system by endoVia Medical

A monitor is placed in optimal direction for the surgeon. The surgeon's hands are always held in a parallel position respect to the tip instruments position.

1.6 Capsular endoscopy

Semiautomatic endoscopes represent a step ahead because they can move in small spaces with arbitrary directions. Their locomotion is worm like. The endoscope presented in [14], in [15] and in [39] are examples of this approach. Three types of autonomous robots for colonoscopy have been studied at Korea Institute of Science

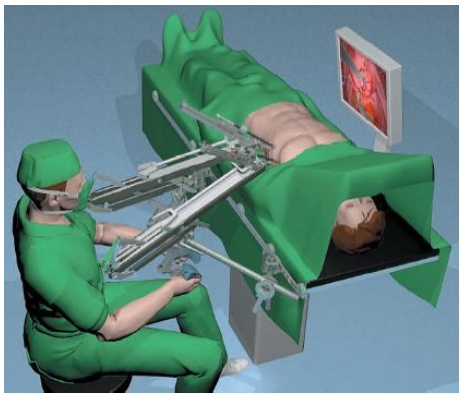


Figure 1.18. Mechanical teleoperated manipulator [26]

and Technologies: one uses wheels in order to move; the second uses a pneumatic actuator [60] while the third uses a millepede inspired locomotion [10].

The studies carried out on pills for endoscopic exploration represent a further step. Capsules are miniaturized instruments for colon inspection and they are powered by batteries. Two commercial pills exist, one from Olympus and the other from Given Imaging (see Fig. 1.19). The main capsules project specifications are reported in Tab 1.1. Many research works are involved in the study of legged endoscopic capsules as the one presented in [43], able to move by means of legs. The contact between the leg and the tissue allows the capsule to slide inside the colon.

(a)

(b)

Figure 1.19. EndoCapsule by Olympus (a) and PillCam by Given Imaging (b)

	Given Imaging	Olympus	RF System Lab	IMC
Operative Environment	Colon (PillCam TM COLON)	All Gastrointestinal Tract	All Gastrointestinal Tract (Norika)	All Gastrointestinal Tract (MiRO)
Dimensions (L x D)	11 x 26 mm	11 x 26 mm	9 x 23 mm	11 x 25,4 mm
Optical sensor	—	CCD	CCD	CMOS
Pixel	—	—	410,000	102,400
Illumination	6 white LED	6 white LED	4 strobed LED	4 white LED
Frame Rate	2 images/sec	5 images/sec	30 images/sec	2 images/sec
Power Source	—	Wireless Power Transmission	Wireless Power Transmission	Micro Battery, Fuel Cells
Source of motion	—	Peristalsis + External Magnetic Field	Peristalsis + External Magnetic Field	Peristalsis + Active Locomotion Mechanism
Motion Control	—	Forward, Reverse, Rotation	Only Rotation	Forward, Reverse,Rotation + Stopping + Clamping

Table 1.1. Main capsules specifications.

Chapter 2

Design of medical mechatronic instruments

2.1 Introduction

The requisites for a general purpose "rapid prototyping platform" are pointed out in this chapter together with a list of possible technological solutions. Prototype fabrication is one of the most important step in every research project because it is required in order to test the goodness of the hypothesis or to investigate new solutions. It would be useful to prove the feasibility of a project in very short time from the concept idea to the first prototype. The development phases of a new medical instrument are presented in Fig. 2.1 and it is not uncommon that the first three phases occupy on third of the scale. Software an hardware tools can be considered for accelerating the development process, they can be used for simulation as well as for data acquisition. Mechanical design software tools are not analysed in this work, that is more oriented to electronics aspects. Model characterization of the system is very important in order to perform simulations and therefore the tuning of system parameters. MATLAB is one of the main software tools used for simulation, it has great capabilities in terms of mathematical computations. Typical application fields are signal and image processing, statistical elaboration and simulation of complex control algorithms. Scilab (www.scilab.org) is an open-source software used for numerical computations and control applications. Specific toolboxes are available for classic and robust control, data interpolation, graphics and others. Simulation is very important for having a rough estimation of the behavior of the system that will be fabricated. The next step is the prototype fabrication: this is one of the most time consuming phases because mechanical parts require the use of machinery and skilled labour. Rapid prototyping machines represent a valid option for the prototype development. They accept in input CAD drawings and produce in output the mechanical part made of resin or other materials in short time, they create physical models directly from digital data in hours instead of days.

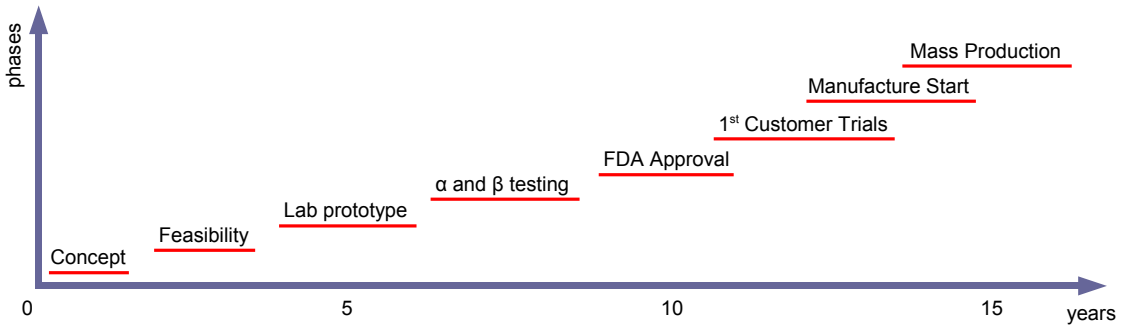


Figure 2.1. Instrument development phases.

As a drawback the mechanical parts produced are not very strength and they can only be used for the very "first prototype".

The control unit is the system "brain" and his task is to coordinate appropriately all the system components with the aim to ensure the correct behavior of the device. The controller can be implemented in simulation environment bu the validation phase can be performed only when mechanical system is built and actuators and sensors are chosen.

2.2 Modeling medical mechatronic instruments

Medical mechatronic instruments can be modeled using the scheme depicted in Fig. 2.2. Mechatronics is defined as the integration of methodologies and techniques coming from different fields like mechanical and electronic engineering and computer science. Deep integration of the fundamental components like mechanical parts, sensors, actuators, controllers and power supply has to be taken into account in the instrument design phase. In traditional industrial environment, robots are programmed to complete tasks with no human interaction. Medical robotic instruments have to be safe enough to allow the interaction with tissues and organs.

The power supply is located at the centre of the diagram in Fig. 2.2 because it has to provide energy to all the system components. Low energy sources must be employed when in contact with patients, isolating the high energy part from the low energy, depending from the micro-shock or macro-shock risks. Moreover in autonomous medical robots the available power is very low, like in capsular endoscopy in which images acquisition is required, maybe in some cases together with locomotion fighting with small sizes and therefore small batteries.

The control unit is the core of the mechatronic system since it generates suitable signals for the actuators after reading sensors. Low-level controller are more related to the hardware: for example the motor positioning calculations must be performed by the low-level control because several timing constraints exist. Graphical user interfaces or more abstract algorithms can be delegated to the high-level control that

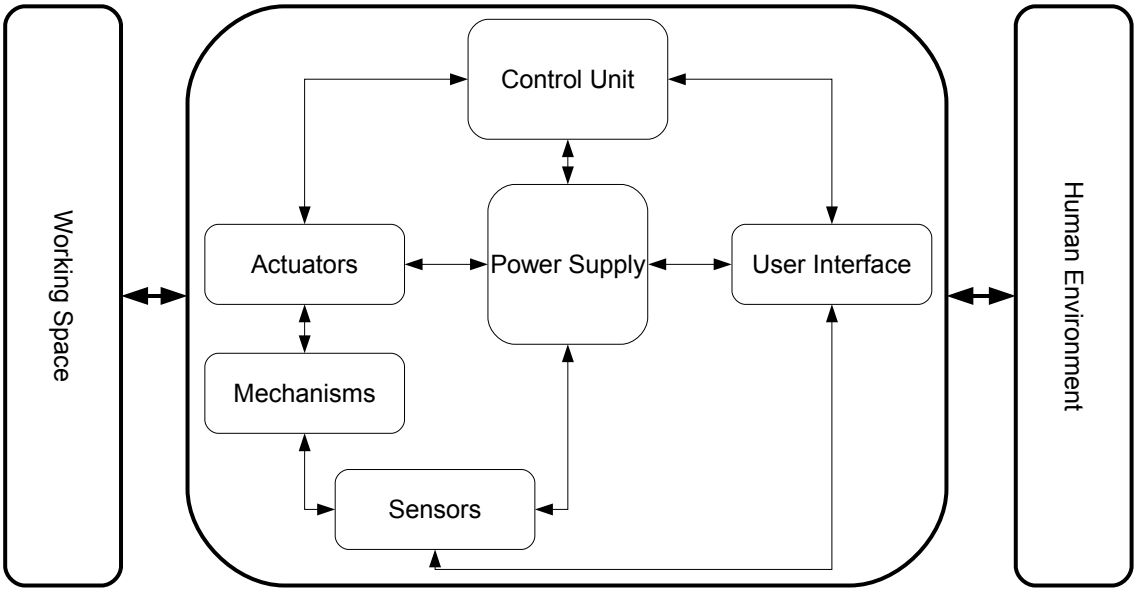


Figure 2.2. Generic robotic instrument schematic.

has to perform a high number and often more complex calculations with less timing constraints. The more the task complexity arises the less the timing constraints will be narrow.

Control systems are used to regulate an enormous variety of machines, products, and processes. They control quantities such as motion, temperature, heat flow, fluid flow, fluid pressure, tension, voltage, and current. Most concepts in control theory are based on having sensors to measure the quantity under control. In fact, control theory is often taught assuming the availability of near-perfect feedback signals. Unfortunately, such an assumption is often invalid. Physical sensors have shortcomings that can degrade a control system. There are at least four common problems caused by sensors. First, they are expensive and this can substantially raise the total cost of a control system. In many cases, the sensors and their associated cabling are among the most expensive components in the system. Second, sensors and their associated wiring reduce the reliability of control systems. Third, some signals are impractical to measure. The objects being measured may be inaccessible for such reasons as harsh environments and relative motion between the controller and the sensor (for example, when trying to measure the temperature of a motor rotor). Fourth, sensors usually induce significant errors such as stochastic noise, cyclical errors, and limited responsiveness. Every prototype, in medical robotics as well as for other fields, needs motors that have to be powered and sensors that have to be read. Many data acquisition systems exist and they can be combined in many configuration. Personal computers with data acquisition electronic boards can be useful instruments for testing control strategies, data monitoring, user interfaces but they can be used only in a research environment. Deeply embedded systems

have microcontrollers or DSPs (DSP Digital Signal Processor) that perform rapid calculation dedicated to the instrument behavior. When a system is fully developed and tested it is possible to port its "operating system" from the testing platform to an embedded hardware.

2.3 Requirements

The requirements for a mechatronic prototyping platform are related to the flexibility. In many cases we may not known the exact configuration of the system. In chapter 5 the bioinspired endoscope is described and the relative motor driver board gives back an impulse every time that the motor performs a revolution. This measure is not affordable anymore because it is not based on "real" position sensors but on counter-electromotive voltage that is generally used for low-cost position sensing. This problem can be overcame by adding two sensors and therefore two additional signals to have to be read and processed. The detail of this project are reported in chapter 5. This example has highlighted the need for very flexible acquisition systems. General Data Acquisition board (DAQ board) are very important because they have a great number of digital input/output channels, analog input channels and counters/timers and analog output channels. Analog inputs usually are specified as single-ended or differential input channels. Single-ended (SE) channels share the same ground point. Differential inputs (DE) have different reference points for each input, and therefore need two channels. In general, SE channels are appropriate when the input signals are greater than 1 V, the signal source is not so far in terms of distance, and all inputs share a common reference. DE inputs have better noise immunity and typically prevent ground loops. While not strictly part of data acquisition, many products offer analog outputs on the same board. In almost all cases, a data acquisition board has either 2 or 4 outputs. If more are needed, a dedicated analog output board may be required. A somewhat separate class of digital I/O is pulse inputs and outputs, which typically is associated with frequency, counting, or totalization applications. Pulse inputs might be used to count the rotations of a turbine flowmeter; pulse outputs might be used to drive a stepping motor. Counter/timer channels are very useful in order to interface with motor encoders or to generate pulse trains or pulse width modulated signals. Pulse inputs are handled in the same way as digital logic inputs, but the output of the sensing circuit is normally connected to a counter rather than a specific bit position in the input register. Successive pulses increment or decrement the counter. Add an elapsed time measure and a frequency or pulse rate can readily be determined.

All these features can be found in microcontrollers or in DSPs but for these devices a deep knowledge of the internal architecture, the programming language like assembler is necessary. PCI DAQ boards represents another possible solution, they are plugged in a traditional PC on the PCI bus and they can be accessed for reading all the input/output channels using libraries supported in C/C++ programming languages. In this case the traditional tools for software development like Visual

Studio with C++ programming language can be used for accessing the DAQ board and implement the board registers reading or writing. Often the DAQ boards have special features that allow to acquire samples using the on-board clock and to write the samples in a buffer that the user can access through software. This feature is very important because the operating systems like Windows or Linux are not real-time operating systems and they can not guarantee the timing requirements for sampling. Real-time operating systems are used in industry for controlling robots or other machines and they are generally commercial. Free real-time operating systems exist, they are in tight relationship with the used hardware, therefore the user must deeply know the structure of the operating system and how it operates to customize it. Implementing the control of a robotic system with a Windows based PC it is not a correct approach because robots have very narrow timing constraints. Code Development on a Windows based PC is suitable for implementing high-level control algorithms in which timing constraints are not very narrow and in which great complexity exists, while the real-time operations have to be delegated to specific hardware.

2.4 Rapid prototyping machines

Rapid prototyping is the name given to a host of related technologies that are used to fabricate physical objects directly from CAD data sources. These methods are unique in that they add and bond materials in layers to form objects. Such systems are also known by the names additive fabrication, three dimensional printing, solid free form fabrication and layered manufacturing. They offer advantages in many applications compared to classical subtractive fabrication methods such as milling or turning:

- Objects can be formed with any geometric complexity or intricacy without the need for elaborate machine setup or final assembly;
- Objects can be made from multiple materials, or as composites, or materials can even be varied in a controlled fashion at any location in an object;
- Additive fabrication systems reduce the construction of complex objects to a manageable, straightforward, and relatively fast process.

These properties have resulted in their wide use as a way to reduce time to market in manufacturing. Today's systems are heavily used by engineers to better understand and communicate their product designs as well as to make rapid tooling to manufacture those products. Surgeons, architects, artists and individuals from many other disciplines also routinely use the technology. The names of specific processes themselves are also often used as synonyms for the entire field of rapid prototyping. Among these are stereolithography (SLA for stereolithography apparatus), selective laser sintering (SLS), fused deposition modeling (FDM), laminated object manufacturing (LOM), inkjet-based systems and three dimensional printing (3DP). Each of

these technologies and the many other rapid prototyping processes has its singular strengths and weaknesses. Stereolithography is the most widely used rapid prototyping technology, it builds plastic parts or objects a layer at a time by tracing a laser beam on the surface of a vat of liquid photopolymer. This class of materials, originally developed for the printing and packaging industries, quickly solidifies wherever the laser beam strikes the surface of the liquid. Once one layer is completely traced, it's lowered a small distance into the vat and a second layer is traced right on top of the first. The self-adhesive property of the material causes the layers to bond to one another and eventually form a complete, three-dimensional object after many such layers are formed. Some objects have overhangs or undercuts which must be supported during the fabrication process by support structures. These are either manually or automatically designed and fabricated right along with the object. Upon completion of the fabrication process, the object is elevated from the vat and the supports are cut off. Stereolithography generally is considered to provide the greatest accuracy and best surface finish of any rapid prototyping technology. Over the years, a wide range of materials with properties mimicking those of several engineering thermoplastics have been developed. Limited selectively color changing materials for biomedical and other applications are available, and ceramic materials are currently being developed. The technology is also notable for the large object sizes that are possible. On the negative side, working with liquid materials can be messy and parts often require a post-curing operation in a separate oven-like apparatus for complete cure and stability.

SLS technology uses thermoplastic powder that is spread by a roller over the surface of a build cylinder. The piston in the cylinder moves down one object layer thickness to accommodate the new layer of powder. The powder delivery system is similar in function to the build cylinder. Here, a piston moves upward incrementally to supply a measured quantity of powder for each layer. A laser beam is then traced over the surface of this tightly compacted powder to selectively melt and bond it to form a layer of the object. The fabrication chamber is maintained at a temperature just below the melting point of the powder so that heat from the laser need only elevate the temperature slightly to cause sintering. This greatly speeds up the process that is repeated until the entire object is fabricated. After the object is fully formed, the piston is raised to elevate it. Excess powder is simply brushed away and final manual finishing may be carried out. No supports are required with this method since overhangs and undercuts are supported by the solid powder bed. It may take a considerable length of cool-down time before the part can be removed from the machine. Large parts with thin sections may require as much as two days of cooling time. SLS offers the key advantage of making functional parts in essentially final materials. However, the system is mechanically more complex than stereolithography and most other technologies. A variety of thermoplastic materials such as nylon, glass filled nylon, and polystyrene are available. Surface finishes and accuracy are not quite as good as with stereolithography, but material properties can be quite close to those of the intrinsic materials. The method has also been extended to provide direct fabrication of metal and ceramic objects and tools. Since

the objects are sintered they are porous. It may be necessary to infiltrate the part, especially metals, with another material to improve mechanical characteristics.

FDM is the second most widely used rapid prototyping technology, after stereolithography. A plastic filament is unwound from a coil and supplies material to an extrusion nozzle. The nozzle is heated to melt the plastic and has a mechanism which allows the flow of the melted plastic to be turned on and off. The nozzle is mounted to a mechanical stage which can be moved in both horizontal and vertical directions. As the nozzle is moved over the table in the required geometry, it deposits a thin bead of extruded plastic to form each layer. The plastic hardens immediately after being squirted from the nozzle and bonds to the layer below. The entire system is contained within a chamber which is held at a temperature just below the melting point of the plastic. Several materials are available for the process including ABS and investment casting wax. ABS offers good strength, and more recently polycarbonate and poly(phenyl)sulfone materials have been introduced which extend the capabilities of the method further in terms of strength and temperature range. Support structures are fabricated for overhanging geometries and are later removed by breaking them away from the object. A water-soluble support material which can simply be washed away is also available. The method is office-friendly and quiet. FDM is fairly fast for small parts on the order of a few cubic inches, or those that have tall, thin form-factors. It can be very slow for parts with wide cross sections, however. The finish of parts produced with the method have been greatly improved over the years, but are not quite on a par with stereolithography. The closest competitor to the FDM process is probably three dimensional printing. However, FDM offers greater strength and a wider range of materials than at least the implementations of 3DP from Z Corp. which are most closely comparable.

In **LOM** technology profiles of object cross sections are cut from paper or other web material using a laser. The paper is unwound from a feed roll onto the stack and first bonded to the previous layer using a heated roller which melts a plastic coating on the bottom side of the paper. The profiles are then traced by an optics system that is mounted to an X-Y stage. After cutting of the layer is complete, excess paper is cut away to separate the layer from the web. Waste paper is wound on a take-up roll. The method is self-supporting for overhangs and undercuts. Areas of cross sections which have to be removed in the final object are heavily cross-hatched with the laser to facilitate removal. It can be time consuming to remove extra material for some geometries, however. In general, the finish, accuracy and stability of paper objects are not as good as for materials used with other RP methods. However, material costs are very low, and objects have the look and feel of wood and can be worked and finished in the same manner. This has fostered applications such as patterns for sand castings. While there are limitations on materials, work has been done with plastics, composites, ceramics and metals. Some of these materials are available on a limited commercial basis. Variations on this method have been developed by many companies and research groups. For example, Kira's Paper Lamination Technology (PLT) uses a knife to cut each layer instead of a laser and applies adhesive to bond layers using the xerographic process. There are

also variations which seek to increase speed and/or material versatility by cutting the edges of thick layers diagonally to avoid stair stepping.

Thermal Phase Change Inkjets uses a single jet each for a plastic build material and a wax-like support material, which are held in a melted liquid state in reservoirs. The liquids are fed to individual jetting heads which squirt tiny droplets of the materials as they are moved in X-Y fashion in the required pattern to form a layer of the object. The materials harden by rapidly dropping in temperature as they are deposited. After an entire layer of the object is formed by jetting, a milling head is passed over the layer to make it a uniform thickness. Particles are vacuumed away as the milling head cuts and are captured in a filter. The process is repeated to form the entire object. After the object is completed, the wax support material is either melted or dissolved away. The most outstanding characteristic of the Solidscape company system is the ability to produce extremely fine resolution and surface finishes, essentially equivalent to CNC machines. However, the technique is very slow for large objects. While the size of the machine and materials are office-friendly, the use of a milling head creates noise which may be objectionable in an office environment. Materials selection also is very limited. Other manufacturers use considerably different inkjet techniques, but all rely on squirting a build material in a liquid or melted state which cools or otherwise hardens to form a solid on impact. 3D Systems produces an inkjet machine called the ThermoJet Modeler® which utilizes several hundred nozzles in a wide head configuration. It uses a hair-like matrix of build material to provide support for overhangs which can be easily brushed off once the object is complete. This machine is much faster than the Solidscape approach, but does not offer as good a surface finish or resolution. All thermal phase change inkjets have material limitations and make fragile parts. The applications range from concept models to precise casting patterns for industry and the arts, particularly jewelry.

Three dimensional printing was developed at MIT. It's often used as a direct manufacturing process as well as for rapid prototyping. The process starts by depositing a layer of powder object material at the top of a fabrication chamber. To accomplish this, a measured quantity of powder is first dispensed from a similar supply chamber by moving a piston upward incrementally. The roller then distributes and compresses the powder at the top of the fabrication chamber. The multi-channel jetting head subsequently deposits a liquid adhesive in a two dimensional pattern onto the layer of the powder which becomes bonded in the areas where the adhesive is deposited, to form a layer of the object. Once a layer is completed, the fabrication piston moves down by the thickness of a layer, and the process is repeated until the entire object is formed within the powder bed. After completion, the object is elevated and the extra powder brushed away leaving a "green" object. No external supports are required during fabrication since the powder bed supports overhangs. Three dimensional printing offers the advantages of speedy fabrication and low materials cost. In fact, it's probably the fastest of all RP methods. Recently color output has also become available. However, there are limitations on resolution, surface finish, part fragility and available materials. The closest competitor to this

process is probably fused deposition modeling.

Laser Engineered Net Shaping® and similar laser powder forming technologies are gaining in importance and are in early stages of commercialization. A high power laser is used to melt metal powder supplied coaxially to the focus of the laser beam through a deposition head. The laser beam typically travels through the center of the head and is focused to a small spot by one or more lenses. The X-Y table is moved in raster fashion to fabricate each layer of the object. The head is moved up vertically as each layer is completed. Metal powders are delivered and distributed around the circumference of the head either by gravity, or by using a pressurized carrier gas. An inert shroud gas is often used to shield the melt pool from atmospheric oxygen for better control of properties, and to promote layer to layer adhesion by providing better surface wetting. A variety of materials can be used such as stainless steel, Inconel, copper, aluminum etc. Of particular interest are reactive materials such as titanium. Materials composition can be changed dynamically and continuously, leading to objects with properties that might be mutually exclusive using classical fabrication methods. The strength of the technology lies in the ability to fabricate fully-dense metal parts with good metallurgical properties at reasonable speeds. Objects fabricated are near net shape, but generally will require finish machining. They have good grain structure, and have properties similar to, or even better than the intrinsic materials. Selective laser sintering is at present the only other commercialized RP process that can produce metal parts directly. However, laser powder forming methods have fewer material limitations than SLS, don't require secondary firing operations as some of those processes do, and can also be used to repair parts as well as fabricate them.

2.5 Introduction to real-time systems

The aim of this introduction is to give to the reader practical informations on the hardware and the software that can be used to build up a substantial control system. After some knowledges on real-time computing systems, some hardware platforms are taken into account in order to implement control systems, from microcontroller, to PC104 boards, to Single-Board Computer. These are important to build up compact and embedded systems. Linux based PCs with real-time operating systems are taken into account for the big potential represented by the possibility to have in a common desktop PC a very smart platform to test and prototype control systems for many applications. Operating system is the software component of a computer system that is responsible for the management and coordination of activities and the sharing of the resources of the computer. Real-time computing systems have the peculiarity to react within precise time constraints during the interaction with the real world. This does not means that a real-time system is fast in an absolute way but that is able to process informations from the environment in useful time to guarantee the correct functioning of the whole system. The objective of fast computing is to minimize the average response time of a given set of processes and this is not

acceptable for the real-time computing that is focused on the process deadlines observance. The average response time is not able to guarantee the individual time constraints of each process. For example if in an aircraft a process deadline is missed this may cause catastrophic consequences. A deadline is represented by the time before which a process should complete its execution. The words task and process in this work are used as synonyms and the meaning is the computation that is executed by the central processing unit (CPU) in a sequential fashion. Real-time tasks are usually distinguished in two classes:

- **Hard**: in which the completion after the deadline can cause catastrophic consequences. In this case any process should be guaranteed in the worst-case scenario;
- **Soft**: if the deadline is missed the performances of the system are decreased but the correct behavior is not compromised.

A schedule is an assignment of tasks to the processor, so that is executed until completion. More formally a schedule can be defined as a function $\sigma : \Re^+ \rightarrow N$ such that $\forall t \in \Re^+, \exists t_1, t_2$ such that $t \in [t_1, t_1)$ and $\forall t' \in [t_1, t_2)$ $\sigma(t) = \sigma(t')$. In other words, $\sigma(t)$ is an integer step function and, $\sigma(t) = k$, with $k > 0$, means that task J_k is executing at time t, while $\sigma(t) = 0$ means that the CPU is idle. The scheduling policy is the criterion with which the CPU is assigned to the various tasks. The scheduling algorithm is the set of rules that, at any time, determines the order in which tasks are executed. The specific operation of allocating the CPU to a task by the scheduling algorithm is called dispatching. The main algorithm classes are:

- **Preemptive**: the running task can be interrupted at any time to assign the processor to another active task.
- **Non-preemptive**: the running task, once started, is executed by the processor until completion.
- **Static**: the scheduling decisions are based on fixed parameters, assigned to task before their activation.
- **Dynamic**: the scheduling decisions are based on dynamic parameters that may change during system evolution.
- **Off-line**: the scheduling algorithm is executed on the entire task set before actual task activation. The schedule generated is stored in a table and later executed by a dispatcher.
- **On-line**: the scheduling decisions are taken at runtime every time a new task enters the system or when a running task terminates.
- **Optimal**: the algorithm is said to be optimal if it minimizes some given cost function defined over the task set.

- **Heuristic**: the algorithm is said to be heuristic if it tends toward but does not guarantee to find the optimal schedule.

A real system must deal with both periodic and aperiodic tasks. Typically periodic tasks are related to sensory data acquisition, control loops, low-level servoing and so on, this represent the major computational demand because they have to be executed at specific rates. Aperiodic tasks are related to events called interrupts and the arrival time is unknown. In a real application it is plausible that aperiodic tasks are treated like soft tasks and periodic tasks like hard tasks under dynamic priority assignments. A resource is any software structure that can be used by a process to advance in execution and can be typically a set of variables, a data structure, a set of registers or a file. The resource can be "private" when is dedicated to a particular process or "shared" when more than one task access to it. A shared resource is named "exclusive" if it is protected against concurrent accesses. A task that needs to enter in a critical section must wait that the exclusive resource becomes "free", otherwise it called "blocked". An operating system provides general synchronization tool called "semaphores" that can be used by tasks to build critical section for accessing resources. Moreover there are resource access protocols in order to solve problems that arise when concurrent task use shared resources in exclusive mode and to respect the maximum blocking time for each task for ensuring the time constraints.

2.6 Different approaches for the implementation

2.6.1 Embedded PC

In the past it was more common for designers to select a microcontroller or microprocessor chip, design the supporting electronics around it and after add the special inputs and outputs required by the application. In many cases this is the optimal solution to a given problem because the design is fully customized to match exactly the project requirements. However this approach can take substantial time to get from the drawing board to the market. Reducing the time to market is one of the biggest reasons for selecting an off-the-shelf embedded PC also for medical applications. In fact the core processor is already plugged in the system and integrated in a chipset. This also reduces the time spent on developing firmware, because embedded PCs typically already have basic input/output system (BIOS) firmware that initialize the core components, tests critical subsystems and loads the application program. All these features are present with bounded costs. Developers can base their activities on the particular application without worry about the hardware. For example today many modern medical equipment have a full-color graphical display in order to give to the user more informations and in embedded PCs there are VGA, CRT and flat-panels LCD interfaces that can be used in combination with powerful graphics softwares already available on desktop PC. The features that embedded PC share with desktop PC are (see Table 2.1):

- x86 processor.
- Chipset, including timers, direct memory access (DMA) controllers, and interrupt controllers.
- Serial ports.
- Printer ports.
- Ethernet connection.
- Video.

x86 Processor. The x86 family of microprocessors has gone through many changes since the original Intel (Santa Clara, CA) 8086 processor. From the 8086 to the 286, 386, 486, Pentium, Pentium II, Pentium III, and Pentium IV (as well as the 186 family), performance has increased exponentially over the past 20 years. In addition, other cpu manufacturers such as National Semiconductor (Santa Clara, CA), AMD (Sunnyvale, CA), and STMicroelectronics (Geneva, Switzerland) have created x86-compatible processors that have higher levels of integration or lower power consumption. However, the most important factor in determining which cpu to select is whether or not that cpu has the performance to keep up with the application. This is not a simple matter, because it is dependent on the cpu type, clock speed, and other factors such as the amount of cache and the chipset. Benchmarks are generally of little use because they rarely measure the things that are critical to a specific application. In any case, it is always a good idea to get at least twice as much computing power as is thought to be required. This allows for errors in estimation of the processor required, and it allows room to add features later. If the application requires a great deal of floating-point or long integer arithmetic, a math coprocessor is often required. It is still possible to do these calculations without a coprocessor because this can be emulated in software. However, emulation is much slower than handling these routines in hardware. Math coprocessors are standard in Intel processors from the 486DX and up (except for the 486SX, which was not widely used in embedded PCs). Processors of the 386 variety and lower require an external coprocessor. In most cases, if a coprocessor is required, a 486DX or higher is appropriate because external coprocessors are not widely available and are generally not cost-effective.

Chipset. The chipset in an embedded PC provides much of the glue logic that connects the cpu, the memory, and the I/O together in order to have a functioning PC. The chipset can be a single chip or multiple chips that are separate from the cpu, or it may be integrated into the cpu. The Intel 186 family and 386EX integrate the chipset into the processor. AMD's Elan family, National Semiconductor's Geode, and STMicroelectronics's STPC similarly integrate chipset functionality into the processor. From an application standpoint, this integration does not have an effect on how development takes place. However, it can greatly reduce the number of chips

required. This can either reduce the size of the embedded PC board or create room on the board for additional features.

Serial Ports. Serial ports are one of the most commonly used interfaces to an embedded PC. They have the advantage of being a simple, well-defined interface with low- to medium-speed data rates. They can be used to communicate with other pieces of equipment, such as a laptop computer for analyzing and displaying data. Alternatively, the serial communications can take place entirely inside a single piece of equipment, linking together different internal modules. There are two commonly used universal asynchronous receiver-transmitters (UARTs) in embedded PCs. These are the 16450 and the 16550, which are almost identical. The difference between the two is that the 16550 has 8-byte first in, first outs (FIFOs) at the transmit and receive buffers. This helps if the data rate is high or if the software disables interrupts for a period greater than the time between two characters being received. If the 16450, which has no FIFOs, receives a character in the receive buffer and does not read that character before the next character is received, that first character will be lost. It is also important that the voltage levels be matched between serial devices. The types that are typically used are RS-232 and RS-422/RS-485. RS-232 is the most common interface and is found on desktop computers and embedded PCs. RS-422/RS-485 is not often found on desktop PCs, but its differential signals have higher noise immunity and are often used for faster communication interfaces with embedded PCs.

Printer Ports. Printer ports on embedded PC boards generally follow the Centronics standard that is common in desktop PCs. Naturally, this can be used to drive an actual printer if the equipment needs to create a hard copy of the output. For example, the results of a treadmill stress test can be printed immediately after running the test. However, in many cases, there is no need for an actual printer in the system because a network printer is used, or simply because no hard copy is required. If the printer port is not used for a printer, it can be used for transistor-transistor logic-level digital I/O. Typically, a printer port can be used for 8 bidirectional I/O lines, 5 input lines, and 4 output lines.

Ethernet Connection. The prevalence of Ethernet connections on embedded PC boards is due in part to the popularity of Ethernet in desktop PCs. In medical equipment, it can be used for connecting to networks to access patient records, such as in cases where x-ray or magnetic-resonance-imaging graphic files are transferred directly from the equipment to the patient database. Ethernet can also be used for high-speed data transfer between different parts of a single piece of equipment. The most common types of Ethernet available on embedded PC boards are 10Base-T and 100Base-TX. These both use an unshielded twisted-pair cable to plug into a standard RJ-45 modular connector. The difference between the two is the speed. A 10Base-T connection transfers data at 10 Mb/sec, whereas 100Base-TX transfers data at 100 Mb/sec. Therefore, the main thing to consider in selection is the amount of data to be transferred.

Video. VGA displays are supported by many embedded PC boards. This allows complex displays of information to be presented to the end-user. This can be

a graphical user interface or a text display of information updated in real time. For example, some medical equipment uses Microsoft Windows as an operating system in order to provide a familiar interface. In general, the important parameters are the type of display and the resolution. The major types of displays supported are cathode-ray tubes (CRTs) and flat-panel LCDs. Flat-panel displays can be further classified into thin-film transistor or active-matrix color, supertwist nematic or passive-matrix color, or monochrome. There are other types of flat-panel displays, such as plasma displays, but many of these displays have interfaces that emulate an LCD's signals and thus can be treated the same as LCDs. Each type of display has its own set of advantages and disadvantages with respect to cost, temperature range, brightness, and viewing angle. It is important that the display selection and embedded PC selection are coordinated because each one affects the other. The resolution will usually drive flat-panel display selection more than it does CRT selection. Most modern VGA CRT monitors are multifrequency and can accommodate many different resolutions. Flat-panel displays, on the other hand, have a fixed resolution and generally increase in price as the resolution goes up.

Table 2.1. Embedded PCs main features [29]

Desktop and Embedded PC Feature	**Selection Criteria**
x86 processor	·Speed ·Math coprocessor
Chipset	·Integration
Serial ports	·Type (16450 or 16550) ·Levels (RS-232 or RS-422/RS-485)
Printer ports	·Will it be used for a printer? ·Will it be used for digital I/O?
Ethernet connection	·Speed
Video	·Display type (CRT or flat-panel LCD) ·Resolution

However in embedded PCs there are other features that are not present in common desktop PC. These regard the possibility to interface the CPU with the external environment for reading sensors, for driving other devices such as relays or motors etc. Some of these features can be summarized (see Table 2.2):

- Digital I/O.
- Analog I/O.
- Solid-state disks.

- Different system buses such as PC/104, CompactPCI, etc.

Digital I/O. Digital I/O is one of the most commonly used features of an embedded PC. It can be used for turning on pumps, reading panel switches, controlling panel light-emitting diodes, interfacing to alphanumeric LCDs, and a host of other purposes. Digital I/O is easy to use from both a hardware and a software point of view. The main characteristics that need to be examined are the voltage level, current source and sink capability, and initialization states. Different things need to be considered in the case of digital inputs and digital outputs. In the case of digital inputs, the main things to be concerned with are the voltage threshold and the high and low voltage limits. In most embedded PCs, a 0- to 5-V TTL input level is standard, although occasionally there may be some other voltage levels. Interfacing is generally a simple matter of matching the voltages of the source with the digital input and perhaps adding a pull-up or pull-down resistor to avoid floating inputs. In the case of digital outputs, not only are the voltage levels important, but the current source or sink capability is important as well. A 0- to 5-V TTL output level is generally the standard, but the current-handling capabilities can vary substantially from board to board, or even between digital outputs on the same board. In embedded PCs, an 82C55 chip is often used to provide 24 bits of digital I/O. An 82C55 chip has a source and sink capability of about 2.5 mA. On the other hand, if a chip in the 74ACTxx family of logic is used to provide digital outputs, the current capability will be in the neighborhood of 24 mA. Paying close attention to these numbers will determine whether a digital output can be used directly or must be buffered with off-board circuitry. One special caution is required in the case of bidirectional I/O pins that are used as digital outputs, as might occur in the case of an 82C55 chip. In most cases, bidirectional pins are configured as floating inputs upon power-up or reset of the board. If a bidirectional pin controls a device that must have a known state on power-up or reset, the pin must have a pull-up or pull-down resistor added to make sure that there is a valid voltage level instead of a floating pin. The resistor should be connected so that it pulls the pin to the same state that it would have when the pin is reconfigured to be an output.

Analog I/O. Analog I/O is often required for interfacing with the real world. Analog inputs can be used to read in temperatures, flow rates, or other parameters by interfacing with thermocouples and other types of sensors. Analog outputs can be used to control heaters or to adjust ultrasound levels. The range of applications is endless. Not all embedded PC boards have analog I/O on them, but if it is needed, it is generally less expensive to buy a board that has analog I/O built in than to buy a separate analog I/O board to interface with the cpu board. The important parameters to examine in both analog outputs and analog inputs are the range, resolution, input type (single ended or differential), and conversion times. Analog inputs with 12-bit resolution are commonly found on embedded PC boards. If the range is 0 to +5 V, a 12-bit resolution translates to being able to resolve changes of 1.22 mV. This is more than enough resolution for most applications. If higher resolution is required, an add-on analog-input board will most likely need to be

installed. The conversion time is important in applications that need analog inputs sampled at fast rates and at regular intervals. However, a few other parameters can affect the actual sampling rate. The way in which the conversion is started can be important. In some cases, the conversion can be started at regular intervals by using a hardware counter or timer. At the end of the conversion, the hardware triggers an interrupt or DMA cycle to store the data. On other boards where there is no direct counter or timer starting the analog-to-digital (A/D) conversions, the conversions can still be started indirectly by having the counter or timer cause an interrupt at a regular interval and then having the interrupt service routine start the A/D conversion.

Solid-State Disks. On desktop PCs, integrated drive electronics (IDE) drives are currently the dominant drive type because they offer the lowest cost per megabyte of storage. In some embedded PC applications, IDE drives may still be used because of their low cost and high capacity. However, in many applications, IDE drives may not be the best choice because of their physical size, lack of speed, power consumption, or susceptibility to shock or vibration. In these cases, some type of solid-state disk is more suitable. There are different types of solid-state disks available, with different sets of advantages and disadvantages. They may be based on a resident flash array, in which individual flash components are installed on the board. A solid-state disk can be a flash module, which is installed on the embedded PC board and interfaces using standard memory signals. Another option is to use a removable module such as a PC Card or CompactFlash card. With a resident flash array, the advantage is low cost, but it typically has less storage capacity than other methods. In addition, it may not be supported by all operating systems. If the size of the application is small enough and if the operating system supports it, a resident flash array is the most cost-effective solid-state disk. Many embedded PC boards support flash modules that interface to standard memory signals because these boards quite often have Joint Electron Device Engineering Council (JEDEC)-standard memory sockets already installed for static random-access memory (SRAM) or erasable programmable read-only memory (EPROM). Again, it is important that the operating system selected has the proper drivers to use such a flash module. PC Cards or CompactFlash cards are sometimes supported by embedded PC boards because they are readily available and their standard interface, which emulates an IDE drive, is supported by many different operating systems. Because these cards are removable, programs may be easily installed, and stored data may be easily off-loaded. However, the large physical size of these cards may be a hindrance in some applications.

Different System Buses. In desktop PCs, the buses that are available for expansion cards are the peripheral component interconnect (PCI) bus and the industry standard architecture (ISA) bus. In the embedded PC arena, there are some boards that use these buses, but there are also buses designed specifically for embedded applications. The advantage of desktop PC buses is in the variety of low-cost boards available. The disadvantages are that these boards are physically larger and mechanically more susceptible to vibration or shock. PC/104 boards have a smaller

size and a set of bus signals that is compatible with the desktop PC ISA-bus signals. PC/104 boards are available for many different applications such as motor control and sound. The PC/104-plus standard adds a connector with PCI signals in order to support higher-speed boards such as video frame grabbers. CompactPCI and secondary transmitted-data buses are standards used for embedded PC board applications in which each board is inserted in a slot in a card rack. Although these are very good for applications in which boards need to be easily changeable or in which lots of I/O is required, they can be bulky for some deeply embedded applications. Other system bus configurations are available, from proprietary buses to busless systems. When evaluating buses, the first thing to determine is whether any type of expansion bus is necessary. If one is necessary for custom I/O, then the bus should be one that supports the needed throughput and a set of signals for which it is easy to design interface circuitry. If standard I/O boards are going to be plugged in, then it should be a bus that has a sufficient variety of boards from which to choose. Finally, the bus should physically fit into the enclosure of the final system. [29]

Table 2.2. Embedded PCs extended features [29]

Embedded PC Feature	**Selection Criteria**
Digital I/O	·Voltage level ·Current source and sink capability ·Initialization state
Analog I/O	·Resolution ·Range ·Input type (single ended or differential) ·Conversion time ·Method of starting conversions (hardware or software)
Solid-state disks	·Capacity ·Physical size ·Removability
System bus	·Physical size ·Availability of plug-in boards ·Access for changing individual boards ·Amount of I/O required ·Throughput

2.6.2 Single-Board Computers

Single-board computers (SBCs) are complete computers built on a single circuit board. The design is centered on a single microprocessor with RAM, IO and all other features needed to be a functional computer in one board. Single board computers are most commonly used in industrial environment where they are used in rack-mount format for process control or embedded within other devices to provide control and interfacing. Because of the very high levels of integration, reduced component counts and reduced connector counts, SBCs are often smaller, lighter, more power efficient and more reliable than comparable multi-board computers. One of the drawbacks is represented by their highly integrated nature. This means that upgrading an SBC is normally impossible, if there is a failure or an upgrade needed, the entire SBC normally has to be replaced. Currently the most common variety of SBC in use is of a specific form factor similar to other plug-in cards and is intended to be used in a backplane. Some architectures are dependent entirely on single-board computers, such as CompactPCI, PXI, VMEbus, VXI, PICMG architecture, etc. Some single-board computers also exist as form factors that stack like building blocks, and do not have the form of a traditional backplane. Examples of stacking SBC form factors include PC/104, PC/104-Plus, PCI-104, EPIC, and EBX; these systems are commonly available for use in embedded control systems. Stack-type SBCs often have memory provided on plug-cards such as SIMMs and DIMMs, however they can still be regarded as SBCs because although the memory modules are technically additional circuit boards, they have no extra functionality beyond providing memory and are basically just carriers for the RAM chips. Hard drive circuit boards are also not counted for determining if a computer is an SBC or not for two reasons, firstly because the HDD is regarded as a single block storage unit, and secondly because the SBC may not require a hard drive at all as most can be booted from their network connections. The PC/104 standard is very interesting because there is a Consortium of over 75 members worldwide who have joined together to establish and maintain standards as well as for PC/104-Plus, PCI-104, EBX and EPIC. Users are guaranteed that these standards are respected; two different boards are compatible in terms of power consumption, connectors and interfacing. Typical applications include vending machines, test equipment, medical instruments, communications devices, vehicular systems, data loggers, industrial control systems, full motion video, image processing, PCI bus adapters and bridges, etc. The reason for using this kind of embedded systems is related to their similarity with PC processing performances but in smaller sizes, smaller amount of power consumption, the developer can concentrate on the application rather than spending time in designing the electronic board or compiling a compatible kernel and interfacing all the needed peripherals.

FOX Board

FOX Board is a very small size board (just 66 x 72 mm) based on the ETRAX 100LX microprocessor with MMU (Memory Management Unit) made by Axis, the world leader on network cameras and printer servers. It runs a real Linux with complete glibc C libraries. FOX Board is useful either as a stand alone device for network applications like micro web server, proxy, router, etc. or as a socket module to integrate in the user application board. Designed to meet demands for low cost, easy implementation and superior network performance, the ETRAX 100LX is Axis' sixth-generation optimized system-on-a-chip solution for putting peripherals on the network. The ETRAX 100LX was developed using 0.25μm ASIC technology with the best price/performance ratio available today. The sixth generation of the chip was specifically designed with Linux in mind and includes an MMU (Memory Management Unit) for that purpose. The latest edition of Axis' ETRAX chip was designed with a number of basic criteria in mind:

- **Support higher bandwidth networks**. The increasing use of network topologies such as Fast Ethernet has created the requirement to support faster speeds in Axis products. To achieve a higher data transfer rate, both the CPU and DMA functions were integrated. This has enabled Axis to simplify the design, reducing necessary program memory by a factor of 30 percent over a typical 32-bit RISC processor while lowering the cost.

- **Optimize performance**. In order to saturate a 100 Mbit network, Axis created a packet burst architecture featuring a zero-copy network DMA structure. The integration of this structure into the overall architecture results in a network device "system-on-a-chip" capable of supporting high performance while reducing the load on the 100 MIPS-rated CPU. The overall approach is one suited for connectivity rather than computation, supports data transfer rates of up to 200 Mbit/s (100 Mbit Ethernet full duplex), as well as a wide range of network device applications.

- **Reliability, stability and rapid development**. An ASIC approach provides the ability to build in functionality typically found in high-end communications devices. ETRAX 100LX-based products and embedded systems include a number of management utilities such as:

  - A patent pending bootstrap function so units can be booted remotely over the network, even if they have no program code in memory.
  - A patent pending logic analyzer function for cache monitoring and real-time debugging.
  - Watchdog timer providing self-diagnostics and increased reliability.
  - A consistent development environment: The ETRAX 100LX is backwards compatible with the ETRAX 4, in order to ensure that OEM partners are able to preserve their earlier development investments.

The innovative 100 MIPS 32-bit RISC design delivers compact code and exceptional price/performance at low power consumption. An 8-kbyte on-chip cache helps to take full advantage of the CPU performance. ETRAX 100LX can run the real Linux 2.4/2.6 kernels (more info on http://developer.axis.com/). ETRAX 100LX has almost everything you need included:

- 32 bit RISC CPU core.
- 10/100 MBit Ethernet controller.
- 4 asynchronous serial ports.
- 2 synchronous serial ports.
- 2 USB ports.
- 2 Parallel ports.
- 4 ATA (IDE) ports.
- 2 Narrow SCSI ports (or 1 Wide).
- Support for SDRAM, Flash, EEPROM, SRAM, ...

FOX VHDL board is the same size of the FOX Board and can be mounted through J6 and J7 (see Fig. 2.3) extension headers by two 20x2 pin connectors. This board uses a powerful ProAsic 3 FPGA made by Actel with 250K gates that can be programmed at run-time directly from the FoxBoard without any additional hardware programming tool with user designed hardware circuits and peripherals. One of the first examples of how to use this board is a concept VHDL implementation of a VGA video interface. The main scope of the FOX VHDL Board is to enable a different model of development for the hardware/software design in a Linux based project. Instead of trying to transform the operating system in to a real time one, to control fast peripherals and protocols, the aim is to couple the Linux processor with a fast, reprogrammable hardware logic so as to have a standard interface toward the operating system and with specialized programmed logic toward the external hardware. This allows to manage with the hard real time part of the application against fast hardware peripherals using custom logic at hardware speed and to free the operating system from the stress to follow directly the fast peripherals. If you also consider the possibility to reprogram the hardware logic, the dynamic reconfiguration of the hardware could bring new opportunities in the design patterns where the frontier between hardware and software is easily adapted to the real necessities, and flexibility becomes a real asset for the electronic and information technology designer. The FOX VHDL Board is a good opportunity to experiment fast custom logic circuits in relation to a Linux platform thus enabling users to develop the hardware they need by themselves with the possibility to reconfigure it at will, even from remote. This opens a whole field of opportunity toward

hardware and software development. Hardware description The main component of the new FOXVHDL Board is the Actel A3P250 ProAsic3 FPGA with 250,000 gates in a FBGA package with 256 ball grid array pins in the small size of 17 by 17 mm. Then two banks of 256K x 16 bits CMOS SRAM are present, with access time 20ns in 44-pin TSOP package. The two banks are separately addressable by the FPGA. The other hardware on board is devoted to the interfacing of video output monitors. In fact a resistor network connected to the FPGA realizes a D/A converter for three video channels that can be connected to a standard VGA monitor: The video circuit logic will all be inside the FPGA, programmed in VHDL with an open project developed in these pages as one of the first applications of this VHDL Development System. Another important part is a 40 pins header step 0.1in (2.54mm) (JP3) that delivers 35 signals to/from the FPGA matrix logic with capabilities of more than 300 MHz signals and standard 3.3Volts logic and LVDS interfaces. Through this header we plan to introduce small boards with direct connection to cheap color LCD panels as another kind of video output soon. The headers JP1 and JP2 (see Fig. 2.3) of the FOXVHDL Board are for connecting the board directly to the FOX Board. A series of signals for the data and control interface to and from the FOX VHDL board with the FOX Board are mapped on those headers (more info on http://www.acmesystems.it/).

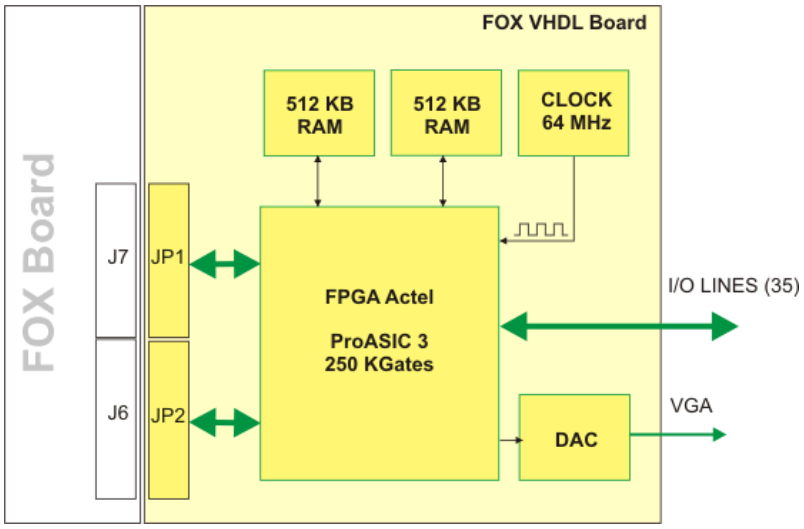


Figure 2.3. VHDL FOX board schematic

VHDL is a language for describing digital electronic systems. It arose out of the United States Government's Very High Speed Integrated Circuits (VHSIC) program, initiated in 1980. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed, and subsequently adopted as a standard by the Institute of Electrical

and Electronic Engineers (IEEE) in the US. VHDL is designed to fill a number of needs in the design process. Firstly, it allows description of the structure of a design, that is how it is decomposed into sub-designs, and how those sub-designs are interconnected. Secondly, it allows the specification of the function of designs using familiar programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping [3].

2.6.3 Microcontrollers and DSPs

A microcontroller (or MCU) is a computer-on-a-chip. It is a type of microprocessor emphasizing self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor (the kind used in a PC). A typical microcontroller contains all the memory and interfaces needed for a simple application, whereas a general purpose microprocessor requires additional chips to provide these functions. A microcontroller is a single integrated circuit, commonly with the following features:

- central processing unit, ranging from small and simple 4-bit processors to sophisticated 32 or 64 bit processors;
- input/output interfaces such as serial ports, digital I/O ports, analog to digital converters and capture/compare/pulse width modulation units or serial peripheral interface (SPI);
- features such as timers, watchdog timer, brown-out reset;
- RAM for data storage;
- ROM, EPROM, EEPROM or Flash memory for program storage;
- clock generator, often an oscillator for a quartz timing crystal, resonator or RC circuit.

This integration drastically reduces the number of chips and the amount of wiring and PCB space that would be needed to produce equivalent systems using separate chips and have proved to be highly popular in embedded systems since their introduction in the 1970s. Microcontrollers take the largest share of sales in the wider microprocessor market. Over 50% are "simple" controllers, and another 20% are more specialized digital signal processors (DSPs). A typical home, in a developed country, is likely to have only one or two general-purpose microprocessors but somewhere between one and two dozen microcontrollers. A typical mid range automobile has as many as 50 or more microcontrollers. They can also be found in almost any electrical device: washing machines, microwave ovens, telephones etc.

Digital signal processors (DSPs) are specialized microprocessor designed specifically for digital signal processing, generally in real-time. DSPs provide really fast

computing, an entry level processor can reach the speed of 30MIPS (million of instruction per second) at the cost of less than 10 euro per unit.

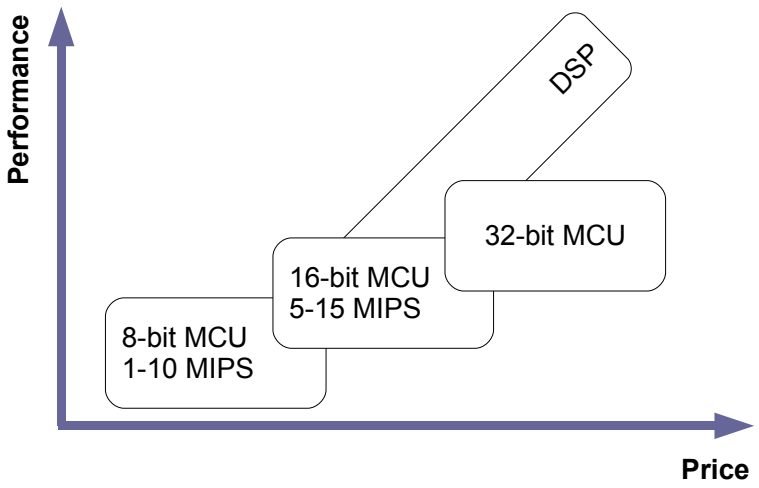


Figure 2.4. Comparison between microcontrollers and DSPs

The dsPIC30F represents an example of DSP family by Microchip, it has a 16-bit (data) modified Harvard architecture with an enhanced instruction set, including significant support for DSP. Harvard architecture means that there is a physically separate storage and signal pathways for instructions and data. The CPU has a 24-bit instruction word, with a variable length opcode field. The program counter (PC) is 24-bits wide and addresses up to 4M x 24 bits of user program memory space. A single cycle instruction pre-fetch mechanism is used to help maintain throughput and provides predictable execution. All instructions execute in a single cycle, with the exception of instructions that change the program flow, the double-word move (MOV.D) instruction and the table instructions. Overhead free program loop constructs are supported using the DO and REPEAT instructions, both of which are interruptible at any point. The dsPIC30F devices have sixteen 16-bit working registers in the programmer's model. Each of the working registers can act as a data, address, or address offset register. The 16th working register (W15) operates as a software stack pointer for interrupts and calls. The dsPIC30F instruction set has two classes of instructions: the MCU class of instructions and the DSP class of instructions. These two instruction classes are seamlessly integrated into the architecture and execute from a single execution unit. The instruction set includes many Addressing modes and was designed for optimum C compiler efficiency

2.6.4 Real-Time PCs

The operating system (OS) acts as a host for application programs that are run on the machine. As a host, one of the purposes of an operating system is to handle the

details of the operation of the hardware. This relieves application programs from having to manage these details and makes it easier to write applications. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even modern video game consoles, use an operating system of some type. Operating systems offer a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls. By invoking these interfaces, the application can request a service from the operating system, pass parameters, and receive the results of the operation. Users may also interact with the operating system by typing commands or using a graphical user interface (GUI). For hand-held and desktop computers, the GUI is generally considered part of the operating system. For large multiuser systems, the GUI is generally implemented as an application program that runs outside the operating system. Modern operating systems provide the capability of running multiple application programs simultaneously, which is referred to as multiprogramming. Each program running is represented by a process in the operating system. The operating system provides an execution environment for each process by sharing the hardware resources so that each application does not need to be aware of the execution of other processes. The CPU of the computer can be used by only one program at a time. The operating system can share the CPU among the processes by using a technique known as time slicing. In this manner, the processes take turns using the CPU. Single-user desktop PCs may simplify this further by granting the CPU to whichever application the user has currently selected and allowing the user to switch between applications at will. The main memory of a computer (referred to as random access memory, or RAM) is a finite resource. The operating system is responsible for sharing the memory among the currently running processes. When a user initiates an application, the operating system decides where to place it in memory and may allocate additional memory to the application if it requests it. The operating system may use capabilities in the hardware to prevent one application from overwriting the memory of another. This provides security and prevents applications from interfering with one another. The details of device management are left to the operating system. The operating system provides a set of APIs to the applications for accessing input/output (I/O) devices in a consistent and relatively simple manner regardless of the specifics of the underlying hardware. The operating system itself will generally use a software component called a device driver to control an I/O device. This allows the operating system to be upgraded to support new devices as they become available. In addition to a device driver for the network I/O device, the operating system includes software known as a network protocol and makes various network utilities available to the user. Operating systems provide security by preventing unauthorized access to the computer resources. Many operating systems also prevent users of a computer from accidentally or intentionally interfering with each other. The security policies that an operating system enforces range from none in the case of a video game console, to simple password protection for hand-held and desktop computers, to very elaborate schemes for use in high-security environments. Windows and Linux are general purpose operating systems and in particular Linux

has non-pre-emptable kernel: it deals the resources (processor, memory, peripheral devices and so on) but it does not interrupt kernel activities. Linux basic user space scheduler is of the time slicing type: it gives more or less equal time slices to different tasks. It is possible to change the priorities of user space tasks to some extent but not enough to make the scheduling deterministic. Other reasons why Linux is a poor RTOS are the unpredictable delays caused by non-pre-emptable operations running in kernel space. Indeed, nobody can understand the kernel sufficiently well to be able to predict how long a certain operation is going to take. All remarks above hold for all general purpose operating systems, such as Windows, AIX, IRIX, HP-UX, Solaris, etc. DOS was much closer to being an RTOS than Linux, because its scheduler was less advanced, and it had fewer system services to look after but it cannot deal with more than one tasks. Because none of the desktop or server operating systems is a good candidate for real-time and/or embedded applications, several companies have started to develop special purpose operating systems, often for quite small markets. Many of them are UNIX-like, but they are not mutually compatible. The market is very fragmented, with several dozens of RTOSs, none of which holds a majority of the market. At least, this was the case before Linux appeared on the radar of real-time and embedded system companies. Since about the year 2000, the market has seen lots of mergers and acquisitions, and substantial efforts from the established RTOS companies to become as "Linux-compliant" as possible. Anyway, quite a lot of Free Software efforts have started to contribute software in the area of real-time and embedded systems. These contributions can be classified as follows:

- Eliminating functionalities from the standard Linux kernel: this approach aims at reducing the memory footprint of the operating system, and is hence mainly focused on embedded systems. uCLinux (www.uclinux.org) is an example. Other projects develop small and simple C libraries, because the current versions of the GNU tools have become quite large; for example, BusyBox (www.busybox.net) (a replacement for most of the utilities one usually finds in the GNU fileutils, shellutils, etc.); uclibc (www.uclibc.org) (a small version of the general C library);

- Patches to the standard Linux kernel: this approach replaces the standard scheduler of Linux with a more deterministic scheduling algorithm, and adds scheduling points to the Linux source tree, in order to make the kernel more responsive;

- Real-time patches underneath the Linux kernel: this approach runs Linux as a low-priority process in a small real-time kernel. This kernel takes over the real hardware from Linux, and replaces it with a software simulation. The two major examples that follow this road are RTLinux (www.rtlinux.org/) and RTAI (www.rtai.org/).

- Linux-independent operating systems: these projects have been developed

completely independently from Linux. S.Ha.R.K. is an example of this category of OS developed at RETIS Lab at Scuola Superiore Sant'Anna (www.shark.sssup.it).

Chapter 3

Development of a medical robotic platform

3.1 I/O Hardware

The choice of electronic hardware is affected by the mechanical parts, the actuators and sensors used for the projects. As first requirement the electronic data acquisition board must read encoders that are integral with electric motors and it has to give in output a measure related to the drive shaft position. The most common type of incremental encoder uses two output channels (A and B) to sense position. Using two code tracks with sectors positioned 90 degrees out of phase, the two output channels of the quadrature encoder indicate both position and direction of rotation. If A leads B, for example, the disk is rotating in a clockwise direction. If B leads A, then the disk is rotating in a counter-clockwise direction. Monitoring both the number of pulses and the relative phase of signals A and B, it is possible to track both the position and direction of rotation (see fig. 3.1). Some quadrature encoders also include a third output channel, called a zero or index or reference signal, which supplies a single pulse per revolution. This single pulse is used for precise determination of a reference position.

Robotic systems have actuators that are controlled in position or speed and a power module is necessary to separate motors from the control unit. The classical method to drive motors is to use analog amplifiers that give in output continuous voltages, the more the value is high and the more the speed will be high. Pulse width modulated (PWM, see fig. 3.2) signals typically associated with the direction signal represents another technique. The duty cycle **dc** is defined as the ratio between the time in which the signal remains at high level state and the signal period. There are two modalities that can be used with PWM technique:

- **sign/magnitude PWM** in which the **dc** is related to the motor speed, the more **dc** is high and the more the speed will be high. An additional digital signal is used for the direction;
- **locked anti-phase PWM** that use only one signal, if the **dc** is greater than

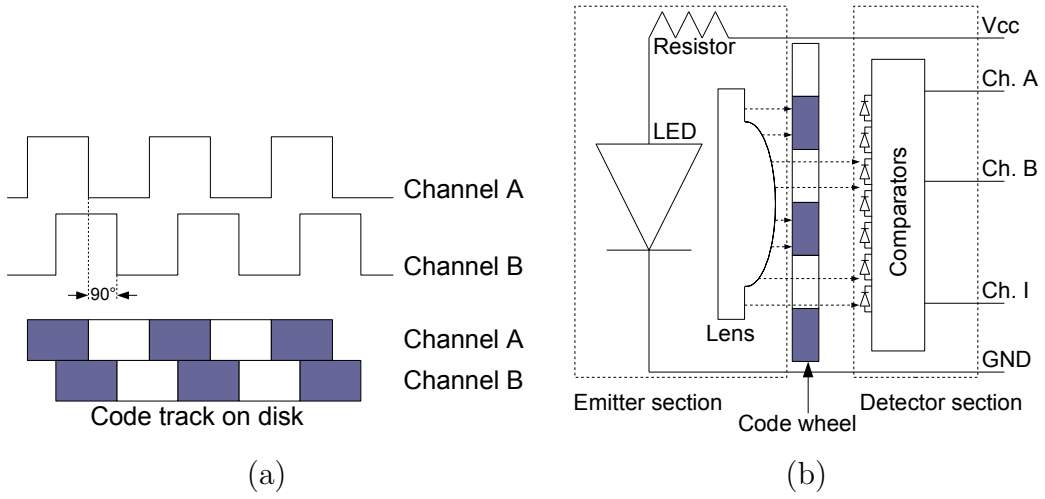


Figure 3.1. Encoder output signals (a) and block diagram of real quadrature encoders(b)

50% the motor will move in one direction and the dc will represent the speed inetnsity while if the dc is less than 50% the motor will move in the opposite direction.

The main advantage in PWM technique is related to power considerations, in fact the use of analog amplifiers results in a more power dissipation and motor heating. With PWM signals a very high energy efficiency can be achieved with a quiet behavior of the motor. Electrical motors are naturally low-pass filtering systems and if the f_{PWM} is in the order of the kHz the current will be quite constant and proportional to the dc value. A hissing sound is audible if $f_{\mathrm{PWM}} < 20kHz$. The speed or position control needs to be performed in a hard real-time task and typically the choice falls on specific devices like microcontrollers that accept in input the speed or position command and perform the control loop reading the encoders mounted on the motors. Classical controllers for DC motors are based on Proportional Integral control algorithms, this means that the output signals toward motors are calculated summing a term proportional to the position/speed error with another term proportional to the position/speed integral error. Direct encoder reading gives a position measure while the speed has to be calculated by deriving the direct measure. The position (speed) command given to the microcontrollers could be an analog signal proportional to the desired motor position or speed but also could be another PWM signal.

The motion controller employed for driving the DC motors employed for this work are MCDC2805 by Faulhaber. These are Proportional Integral motion controller. The board, used for interfacing with the motion controllers, is a National Instruments 6624 PCI version that is an Industrial 8-Channel Counter/Timer Board. The main features are:

- Event counting, period/frequency measurement, encoder position, pulse width

measurement, pulse generation;

- 400KHz maximum frequency with 48 VDC voltage range on inputs and outputs;
- Eight counter/timers with 26 channel-channel isolated inputs and 8 channel-channel isolated outputs;
- Proprietary drivers for working in C++ under Visual Studio .NET.

This board has been used to read encoders, to generate PWM signals for motor controllers and to perform pulse width measurements under Windows XP operating system because the device is not supported in real-time operating systems.

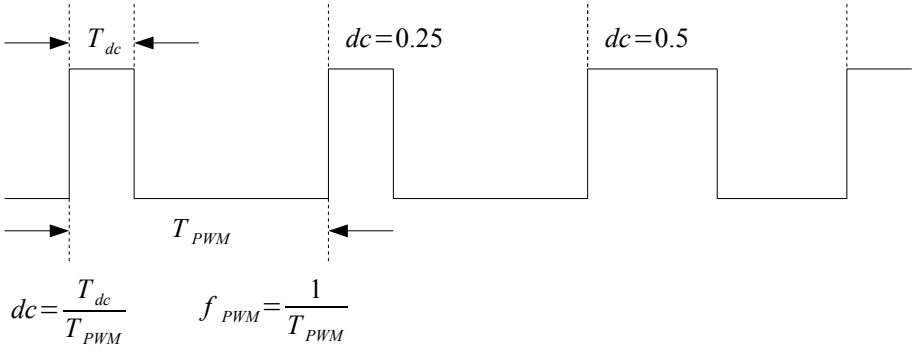


Figure 3.2. Pulse Width Modulated signal and duty cycle

The 6624 is very useful when more than two motors are needed, often an analog output board should be better for sending position/speed commands to the motor controllers but these boards have only one or two counter/timer channels so it is not possible to read more than one or two encoders.

Applications that need lots of sensors require the presence of digital input/output, analog input and analog output channels. Microcontrollers could be an interesting solution because they have all these features and moreover can communicate with PC through serial port. The used microcontroller is the PIC16F876 from Microchip. The cost of the PIC16F876 is less than 10 euros and the operating frequency can be up to 20MHz. It is a RISC processor (reduced instruction set controller) that means that it has 30 basic instructions. Three I/O ports are included and the first port has also a 10-bit analog to digital converter module. The universal asynchronous receiver transmitter module is able to implement the RS-232 communication protocol (the protocol of PC serial port), very useful to send data toward the PC. The PWM module is used to generate two pulse width modulated (PWM) signals that are used to pilot the power modules of motors. Timers are used for counting purposes or to generate interrupts useful to perform sampling applications that have to be performed at constant time rates. The use of low cost DSPs constitutes

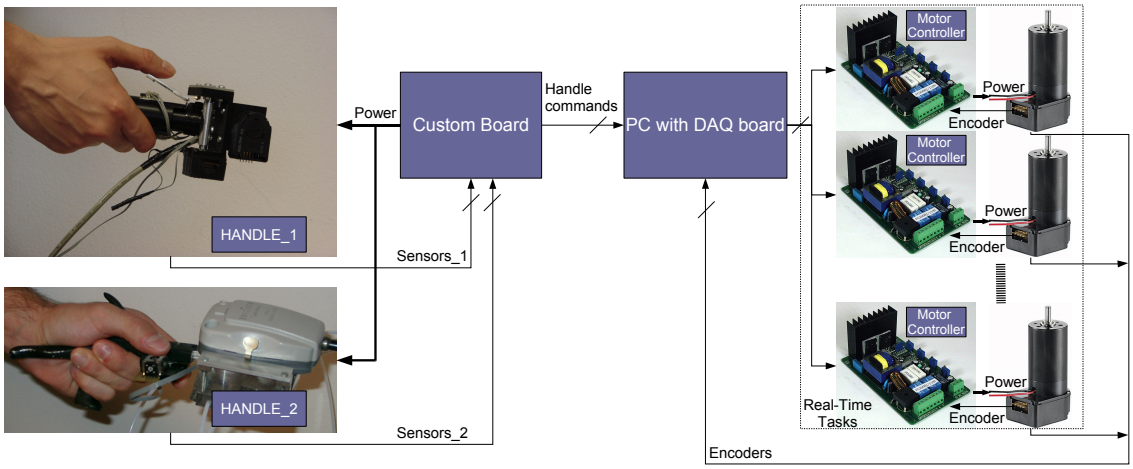


Figure 3.3. Functional schematic of a robotic system

another approach similar to the previous one, they have more features than microcontrollers as already described in chapter 2. The used DSP is the dsPIC30F2010 by Microchip, a low cost device with many features like the floating point calculations support and the encoder input channel. The last considered solution regards the use of a DAQ board such as the National Instrument 6025E PCI version and the main features are:

- 16 12-bit analog input;
- Two 12-bit analog outputs;
- 32 digital I/O lines;
- Two 24-bit counters.
- Proprietary drivers for working in C++ under Visual Studio .NET;
- COMEDI libraries for working in C with real-time operating systems like Linux RTAI and S.Ha.R.K.. These libraries can be used also to produce code for xPC target applications.

3.1.1 Serial communications

RS-232 communication is asynchronous, this means that the clock signal is not sent with the data. Each word is synchronized using it's start bit, and an internal clock on each side (receiver and transmitter devices), keeps tabs on the timing. The diagram depicted in Fig. 3.4 shows the expected waveform from the UART when using the common 8N1 format. 8N1 signifies 8 Data bits, No Parity and 1 Stop Bit. The RS-232 line, when idle is in the Mark State (Logic 1). A transmission starts

with a start bit which is (Logic 0). Then each bit is sent down the line, one at a time. The LSB (Least Significant Bit) is sent first. A Stop Bit (Logic 1) is then appended to the signal to make up the transmission.

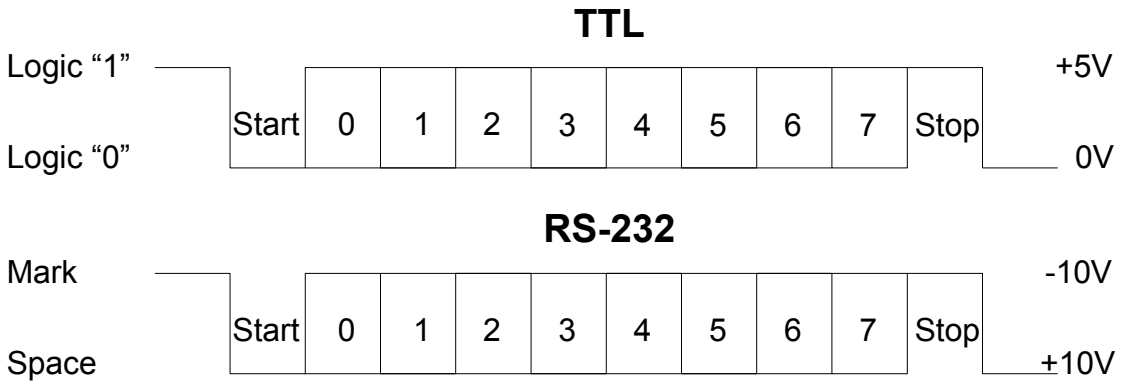


Figure 3.4. Timing diagram for TTL (top) and RS-232 (bottom) signals

The diagram, shows the next bit after the Stop Bit to be Logic 0. This must mean another word is following, and this is it's Start Bit. If there is no more data coming then the receive line will stay in it's idle state(logic 1). A "Break" Signal exists and this happens when the data line is held in a Logic 0 state for a time long enough to send an entire word. Therefore if the line is not set back into an idle state, then the receiving end will interpret this as a break signal. The data sent using this method, is said to be framed. That is the data is framed between a Start and Stop Bit. Should the Stop Bit be received as a Logic 0, then a framing error will occur. This is common, when both sides are communicating at different speeds.

The diagram depicted in Fig. 3.4 is only relevant for the signal immediately at the UART. RS-232 logic levels uses +3 to +25 volts to signify a "Space" (Logic 0) and -3 to -25 volts for a "Mark" (logic 1). Any voltage in between these regions (ie between +3 and -3 Volts) is undefined. There are other lines on the RS-232 port which, in essence are Parallel lines. These lines (RTS, CTS, DCD, DSR, DTR, RTS and RI) are also at RS-232 Logic Levels. Almost all digital devices require either TTL or CMOS logic levels. Therefore the first step to connecting a device to the RS-232 port is to transform the RS-232 levels back into 0 and 5 Volts and this is done by RS-232 Level Converters.

Two common RS-232 Level Converters are the 1488 RS-232 Driver and the 1489 RS-232 Receiver. Each package contains 4 inverters of the one type, either Drivers or Receivers. The driver requires two supply rails, +7.5 to +15v and -7.5 to -15v. This may pose a problem in many instances where only a single supply of +5V is present. Another device is the MAX-232 that includes a Charge Pump, which generates +10V and -10V from a single 5v supply. This integrated circuit also includes two receivers and two transmitters in the same package. MAX232 is useful

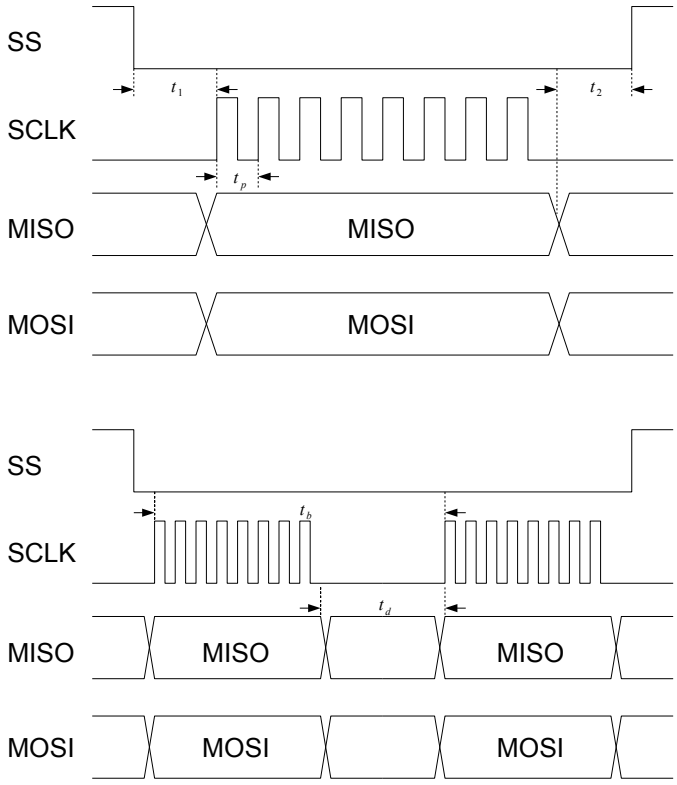


Figure 3.5. SPI timing diagram

in many cases when only the Transmit and Receive data Lines are used without implementing Flow Control.

Serial ports use two-level (binary) signaling, so the data rate in bits per second is equal to the symbol rate in baud (baud is a measure of symbol rate). Common bit rates per second for asynchronous start/stop communication are 300, 1200, 2400, 9600, 19200 baud. Serial ports on PCs and on modern microcontroller or DSPs can reach up to 115200 bits per second.

SPI Interface or Serial Peripheral Interface bus is as a serial interface, this means that data is shifted out (and in) one bit at a time. It is intended for transmission of data from a master device to/from one or more slave devices over short distances at speeds greater than 1MHz. The timing diagrams are depicted in Fig. 3.5.

It is simply based on an 8 bit shift register shifting data out on a single pin and shifting data in on another pin. SPI main use is to replace parallel interfaces so routing parallel buses around a board is not needed anymore. The timing constraints are reported in table 3.1

IIC is a multi-master serial computer bus invented by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cellphone. The name stands for Inter-Integrated Circuit and is pronounced I-squared-C and

Table 3.1. SPI timing specifications

Symbol	Parameter	Min	Max	Units
t_1	SS assertion to first clock	10	20	μs
t_2	Last clock to SS deassertion	5	10	μs
t_p	Clock period	125	8000	μs
t_d	Setup time (Master)	7	9	μs
t_d	Setup time (Slave)	4	n/a	μs
t_b	Time between start of bytes (Slave)	10	n/a	μs

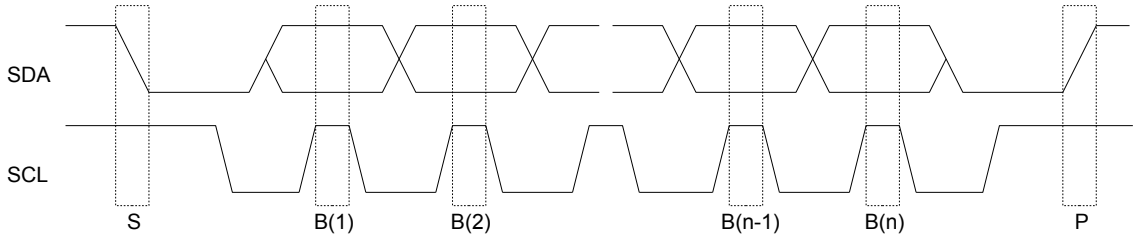


Figure 3.6. IIC timing diagram

also, incorrectly, I-two-C. As of October 1, 2006, no licensing fees are required to implement the IIC protocol. However, fees are still required in order to obtain IIC slave addresses. Data transfer is started with the START bit (S) when SDA is pulled low while SCL stays high. Then, SDA sets the transferred bit while SCL is low and the data is read (or received) when SCL goes high. When transfer is ended, a STOP bit (P) is send by pulling data line high while SCL is high (see Fig. 3.6). The IIC reference design has a 7-bit address space with 16 reserved addresses, so a maximum of 112 nodes can communicate on the same bus. The most common IIC bus modes are the 100 kbit/s standard mode and the 10 kbit/s low-speed mode, but clock frequencies down to zero are also allowed. Recent revisions of IIC can host more nodes and run faster (400 kbit/s Fast mode and 3.4 Mbit/s High Speed mode), and also support other extended features, such as 10-bit addressing.

3.2 Real-Time part

Two types of approaches heve been implemented, the first is related to the use of dedicated hardware like microcontrollers or DSPs that guarantee the observance of the timing constraints. The programming language is C and particularly it is used the free compiler CC5X and the free version of PIC30 for respectively microcontrollers and dsPIC30F2010. The need for a dedicated programmer is one of the

main drawbacks because it is required for transferring the written application on the device. This problem can be overcame with the use of a "bootloader" that is a sort of operating system that can handle signals from PC serial port in order to write the program memory with the user application. In this way the programmer is not needed anymore and major details are reported in appendix (micro). The second approach is related to the use of real-time PC like Linux RTAI and S.Ha.R.K., the programming language is C for both these solutions using gcc compiler. The Linux distribution is a Suse 9.3 Professional installed on a mobile PC with a Pentium III working at 1GHz. Under this distribution is installed another kernel and particularly the 3.2 test 3 version patched for RTAI support. A detailed installation procedure is reported in appendix (realtime). It is possible to use the digital input/output channels of the parallel port and the serial port in a real-time environment. S.Ha.R.K. operating system, version 1.5.2, is installed on a PC with a dual xeon processor working at ... and in this computer is plugged the National Instruments 6025E DAQ card in order to work in real-time modality. A detailed installation procedure is reported in appendix (realtime).

3.3 Non real-Time part

The non real-time parts are implemented in personal computer running Windows XP operating system and generally they are related to graphical user interfaces and to high level control strategies calculations. C++, under Visual Studio .NET development environment, is the used programming language. QT 3.2.1 non-commercial version is used for the implementation of graphical user interfaces (GUIs). The programs written for the projects are reported in appendix (codewindows). One of the used computers has a processor AMD 1.8 GHz and has plugged the National Instruments 6624 DAQ board. The board libraries used to work in C++ are furnished by National Instruments (NIDAQmx). It was not possible to use a real-time operating system (RTOS), in this computer, for the lack of libraries relative to the 6624 board and supported by any RTOS. The other computer is the same dual xeon used with S.Ha.R.K. using the DAQ board 6025E with NIDAQmx libraries to program the board in C++. All the projects presented in the next chapters are implemented with the use of this non real-time setup leaving the observance of timing constraints to dedicated hardware. The implementation on a real-time PC is presented at the end of chapters related to the projects.

Chapter 4

Leonardo. Hand-held robotic instrument for laparoscopic surgery

4.1 Introduction

Minimally invasive surgery interventions bring numerous benefits to the patient, but severely hamper surgeons' perception and motor skills. Technology can provide many different instruments and devices aimed at restoring, and possibly augmenting, visual feedback, haptic and tactile senses, motor coordination and dexterity: stereo endoscopes and displays, teleoperated master-slave systems, robotic interfaces and surgical instruments with many degrees of freedom (DoFs). Systems at the current state-of-the-art, like the da Vinci surgical system [23] by Intuitive Surgical Inc., bring some advantages at the cost of longer setup times and of cluttering the already crowded operating table, pushing the surgeon away from the patient: the surgeon operates at a console in a corner of the operating room and only the robot is in direct contact with the patient. Experienced surgeons tend to agree that in many procedures the benefits provided by those teleoperation systems are not really needed during the whole surgical procedure, and they tend to prefer the traditional hands-on approach for routine tasks. Mechanical instruments that gives more DoFs are used for knee surgery like the instrument presented in [50] or for laparoscopic surgery like the Radius Surgical System from Tuebingen Scientific Medical GmbH covered by the patent [6]. Robotic hand-held instruments have been developed but mainly these works have motors in the handle part leading to heavy devices. Examples of this approach are presented in [33] and in [56]. In [35] a robotic manipulator is described and the main feature is represented by the motor dislocation from the handle.

Led by these considerations, the aim of this project concerns the develop of a lightweight hand-held surgical robot that can be operated by a surgeon with only one hand, while the other hand is free to use a traditional endoscopic instrument. The

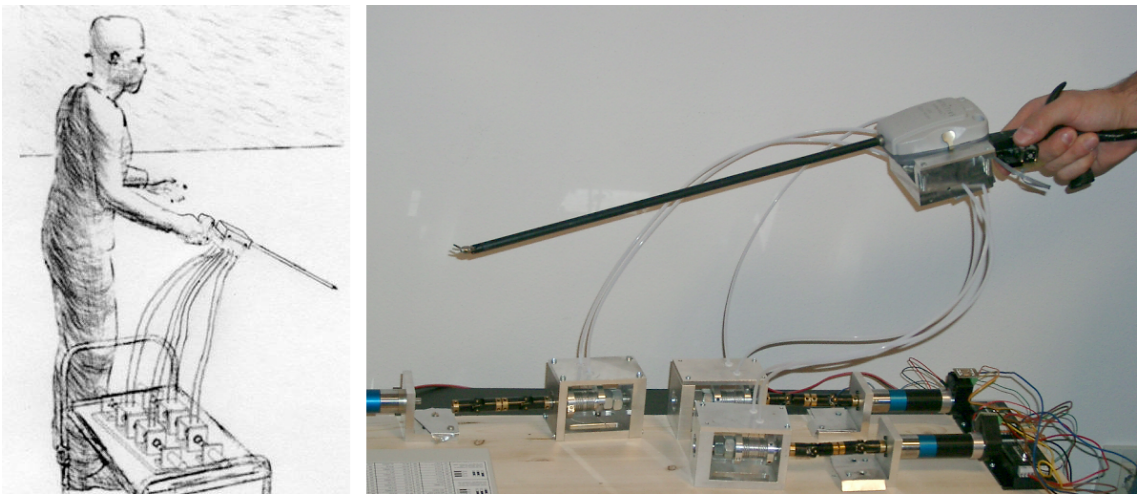


Figure 4.1. Concept drawing of the lightweight hand-held laparoscopic robot (left) and the first prototype (right), using the EndoWrist of the da Vinci system as end-effector.

robot consists of a master part (the handle) and a 6-DoF slave part (the instrument tip), connected together. [19]. The Figure 4.1 depicts the concept drawing together with an preliminary version of the prototype. Selecting the best way to control the instrument tip by means of the handle is one of the major issues in the design. A *control mode* is defined as a particular way to map the DoFs of the handle to the DoFs of the tip [30]. The first part of the project focuses on the study of the control mode that provides the most intuitive and efficient way to steer the instrument [48]. To this purpose a teleoperated system has been developed and it is composed of custom handles mounted on a haptic interface that also serves as 6-DoF digitizer. On this system, it has been implemented an exercise mimicking a surgical gesture requiring high dexterity, namely knot tying, and asked subjects to perform the exercise with four different control modes in order to select the most efficient. The second part of the project concerns details on the design and the fabrication of the hand-held robotic instruments for laparoscopy. Since MIS instruments are rigid or only limitedly flexible, some anatomical regions are not accessible. The insertion point acts as fulcrum constraint on the long, stiff instruments, causing non-intuitive effects on the tip movements, like movement inversion and velocity scaling. A dexterous robotic instrument can help to overcome the listed problems during some phases of the interventions in which the surgeon has to perform complex tasks, such as for suturing.

4.2 Evaluation of different control modes

The study is concerned with how surgical performance is influenced by the way the user controls the instrument tip through the handle. A teleoperated system

has been built in order to select among the control modes to map the DoFs of the instrument handle to those of the instrument tip [30]. For the comparative evaluation two control parameters are considered obtaining four different control modes. The first parameter is defined *PCM* (position control mode) and its values are *Tip* or *Handle*; the second parameter is called *ACM* (angle control mode) and its values are *Absolute* or *Relative*. PCM denotes how the position of the master is mapped to the position of the slave, while ACM specifies how the angular DoFs at the instrument tip are controlled. PCM=Tip means that the position of the master is directly mapped to the position of the tip. In other words, moving the master in one direction causes the tip to move in the same direction. By setting PCM=Handle it is introduced a fulcrum constraint at the trocar position, hence moving the master in one direction causes the tip to move in the opposite direction (as in laparoscopic surgery). By setting ACM=Absolute, the orientation of the tool tip in 3-D space will be exactly the same as the orientation of the handle. By setting ACM=Relative, the angles at the tip joints are controlled incrementally through a joystick (moving the joystick in one direction makes the corresponding tip angle increase in the same direction). By combining the two PCMs with the two ACMs, four different control modes are obtained and can be evaluated on the platform: *H_A* that means (PCM=Handle, ACM=Absolute), *T_A* (PCM=Tip, ACM=Absolute), *H_R* (PCM=Handle, ACM=Relative), and *T_R* (PCM=Tip, ACM=Relative). E. g. teleoperated master-slave systems like the da Vinci surgical system have *T_A* control mode, whereas the hand-held robot described in this chapter has *H_R*.

4.2.1 System description

Fig. 4.2 shows the architecture of the system used for the experiments. For the master system, two types of handle are used, one for each ACM. To control the slave angles of the instrument in the ACM=Relative configurations, a joystick, driven by the user's thumb, has been mounted on the foil grip used as ergonomic handle. The joystick is a two-axis potentiometer and communicates with the host PC through an electronic board connected to the serial port. Two switch have also been mounted in the foil grip in order to implement the stem roll movement in the virtual environment. For the ACM=Absolute configuration a tweezers-like handle has been fabricated. The handles were connected to a Phantom Premium 1.0 by Sensable Inc., which is a 6 DOF localizer and 3 DOF force feedback interface. The Phantom is used to measure the master position; the orientation encoders are used only when ACM=Absolute.

To simulate PCM=Tip the handle is free to move in space and the position of the Phantom end-effector is mapped to the instrument tip position directly. To replicate PCM=Handle, the handle (foil grip or tweezers) is attached to the Phantom end-effector and the Phantom stylus, extended with an aluminum cylinder, is inserted through a trocar on an abdomen replica. The four configurations are depicted in Fig. 4.3. It is worth noting that in *H_A*, an additional gimbal ring was needed, allowing to control the handle orientation independently from the aluminum cylinder,

constrained by the trocar.

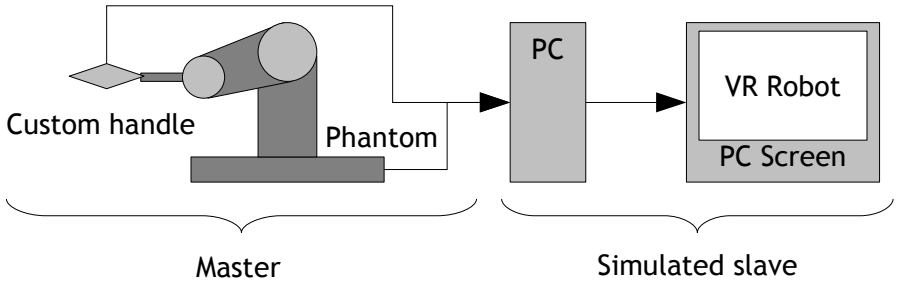


Figure 4.2. The architecture of the system, with sensorized master and simulated slave.

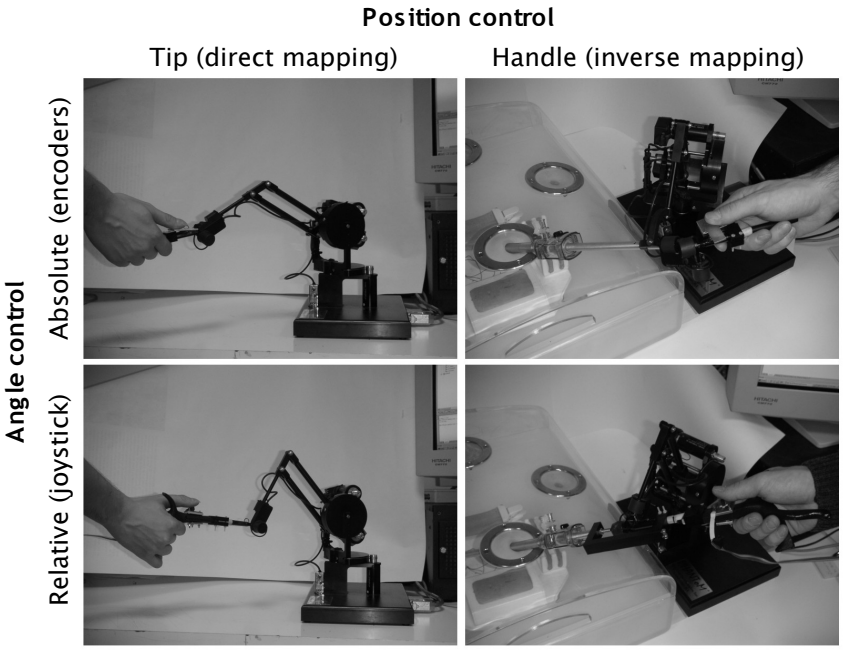


Figure 4.3. The four control modes and the experimental setup.

The slave robot kinematics has been simulated in a virtual reality environment (see Fig. 4.4.right). The robot design is derived from the end-effector of the da Vinci, called EndoWrist®. 3D objects from the CAD models of the mechanical components are exported and assembled in a 3D scene graph using the OpenGL Optimizer library. The inverse kinematics of the EndoWrist is calculated in order to map the end-effector position and orientation correctly and reproduce realistic animation in the virtual environment.

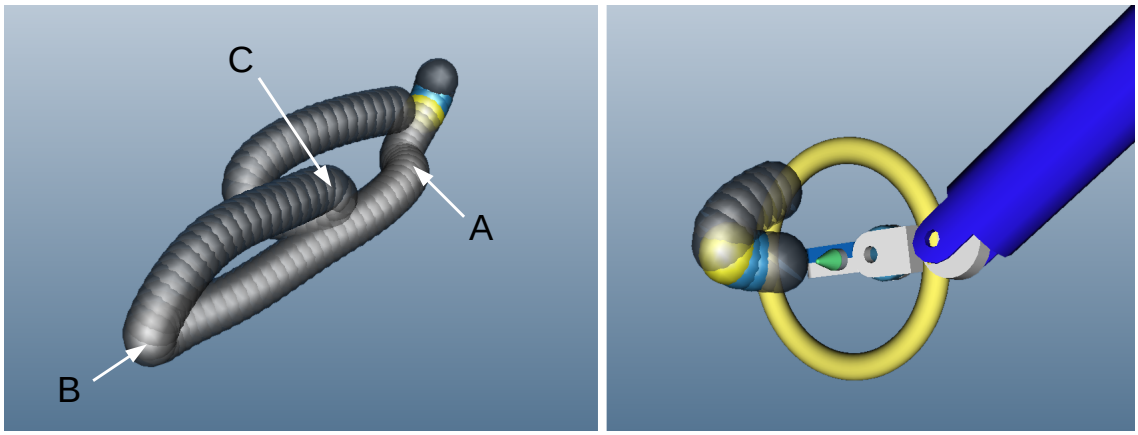


Figure 4.4. The trajectory mimicking a knot-tying task as displayed in the virtual environment (left) and the simulated slave robot approaching the active sphere (right).

4.2.2 Performance assessment

Performance assessment is carried out by measuring the accuracy of following the modeled trajectory in 3D space. The 4 control modes are compared asking test subjects to reproduce, by means of the different interfaces, a gesture that follows a complex path in 3-D space, thus requiring high dexterity and use of all 6 DoFs of the instrument tip. The trajectory to reproduce was created by sampling, at 60 Hz, the coordinates of the instrument tip while a surgeon was mimicking a knot-tying gesture on the platform itself. The data were smoothed with a low-pass filter, interpolated by means of a cubic spline, and downsampled to obtain a final trajectory composed of 100 points.

In the virtual environment the trajectory is shown as composed of a sequence of spheres that must be pierced by means of a needle placed in the slave forceps. A ring, perpendicular to the needle, is displayed as a visual aid to identify the needle orientation, as depicted in Fig. 4.4. Only a few of the spheres composing the trajectories are shown at the same time, to keep the display clear. One sphere is marked in bright yellow: this is the active sphere. To cross the active sphere successfully with the ring, the user must slide the needle past the sphere center, without touching the sphere with the ring, and keeping the ring normal to the direction tangent to the trajectory. Actually, a tolerance angle of 60 deg has been used. When the active sphere has been crossed, the following sphere is marked as active and so on, until the whole trajectory has been correctly followed. As additional requirement, to cross the actual active sphere successfully, some of the previous spheres must also have been crossed successfully. These spheres are highlighted when they have been crossed. Moving away from the trajectory resets the task and requires the user to cross some of the previous spheres again. In the following, this latter condition is referred as *off* state, whereas, the *on* state is defined when the user is performing the exercise

crossing the spheres correctly. The position and orientation of the instrument tip, together with a time stamp and all interactions with the spheres are logged on file for offline analysis.

A group of five subjects with only some experience on a laparoscopic trainer was asked to complete two trials with each of the four control modes. The trials were performed during two days, with no predefined order, allowing the subject at least 10 minutes of rest between the trials. Expert surgeons were not involved at this stage in order to avoid biasing the results with dexterous movement schemes expert surgeons have acquired during prolonged practice with laparoscopic instruments. The period of the simulation cycle is between 15 and 16 ms. The position and orientation of the tool were sampled from the Phantom interface at every cycle, sufficient to describe human gesture [8], especially fine movements, performed during surgery [24]. The acquired data was interpolated and resampled at a constant rate of 70 Hz. Only low-frequency components are present in surgeon's movements. However due to artifacts and some error of instruments, 3D position data is also contaminated by high-frequency noise. Since the data analysis involves first, second and third derivatives, the acquired data are off-line filtered using a numerical fourth-order low-pass Butterworth filter, with cut-off frequency of 15 Hz.

For the evaluation of the exercises, the following parameters have been computed: total time spent (t), total time spent in *on* state (t_{on}), total time spent in *off* state (t_{off}), percentage of time spent in off state ($t := t_{\text{off}}/(t_{\text{on}} + t_{\text{off}})$), number of errors, mean velocity, mean acceleration, and normalized jerk [12].

4.2.3 Final results

Fig. 4.5 shows the progression of the users during the exercises, separately for each control mode. The five users are called U1-U5. Only the second trial are plotted in order not to overload the figure; the first trial has roughly similar shape. The horizontal axis shows which of the 100 spheres that build the trajectory the user was interacting with. The vertical axis shows the time, in s. The almost horizontal segments correspond to parts of the trajectory where the user was progressing, crossing the spheres rapidly. The sloped segments correspond to difficult points in the trajectory, where the users spent much time in advancing to the next sphere. Comparing performance of users, there are mainly three difficult points, roughly corresponding to spheres number 15, 45, and 70. These points correspond to passages with higher curvature, as highlighted in Fig. 4.4 (left) with the letters A, B, and C. It is also possible to note that passage B is more difficult with ACM=Absolute (encoders), and passage C is more difficult with ACM=Relative (joystick).

Fig. 4.6 analyzes time and number of errors of each user, using the four control modes. Users U1 and U2 were faster in performing the exercises, and also made a smaller number of errors. This result was expected, since they are more familiar with laparoscopy and also had some previous experience with laparoscopy simulators. From the histograms, it looks like there is no single control mode preferred by all users, however it is still possible to highlight some basic trends.

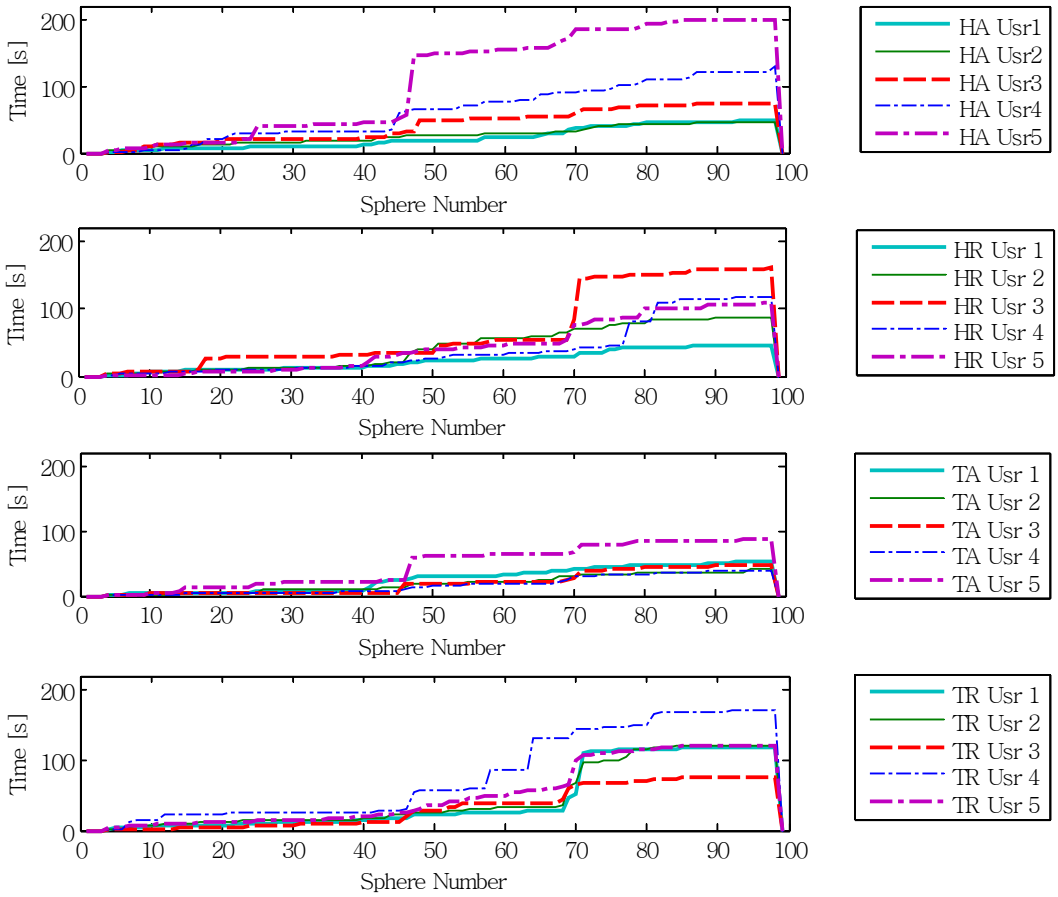


Figure 4.5. Plots showing the time spent to complete the exercise by the users with the four control modes. Steep parts highlight difficult points.

Concerning time, it looks like the *T_A* control mode allows all users to perform the exercise in the shortest time. The reason is that *T_A* is the most intuitive way to control the needle, since there is a direct mapping of both position and angles, and is equivalent to moving one's hand in free space. Note, however, that U1 spends most of the time in t_{off} state, trying to get back to where he made a mistake. Comparing times of the users U1 and U2, who are a little trained in laparoscopy, it can be seen that, apart from *T_R*, they score quite similar with the other control modes. In particular U1 (the more trained in the usage of the developed hand-held robot), performs as well with *H_R* than with *T_A*, but spending much less time in t_{off} state. U2 has similar performance with *H_R* than with *T_A*.

Concerning number of errors, *H_R* always is the 1st or 2nd choice of 4 out of 5 users. In particular, U1 and U2 make less errors with *H_R* than with *T_A*.

Concerning mean velocity $\bar{v}$, mean acceleration $\bar{a}$ and normalized jerk, trends are less evident. It appears as, while performing the exercise, $\bar{v}_{\text{on}}$ and $\bar{a}_{\text{on}}$ are less when PCM=Handle, suggesting that the trocar contributes to slow down, but also

control, the movements. Normalized jerk is higher when ACM=Relative, but this is easily explained since the movements of the joystick are, exactly, *in jerks*.

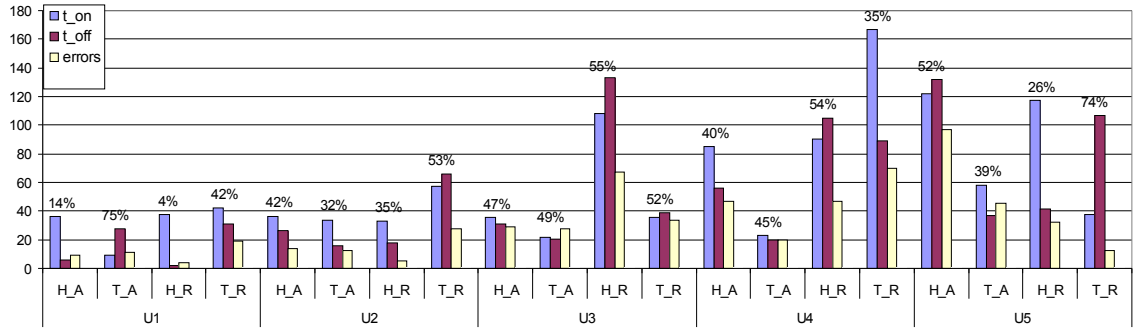


Figure 4.6. Histogram showing, for each user and control mode, t_{on}, t_{off} and the number of errors (see text for the description). The percentage shown is the ratio $t_{\text{off}}/t_{\text{total}}$.

4.3 First prototype

Technology can provide many different instruments and devices aimed at restoring, and possibly augmenting, the reduced surgeons' perception and motor skills. Some phases of laparoscopic interventions, such as suturing, require precise and dexterous movements that are difficult to perform by means of rigid instruments. Multi-DOF hand-held instruments and teleoperated systems have been developed to increase movement dexterity. The next sections focus on laparoscopic procedures and present a novel hand-held robotic instrument with additional degrees of freedom (DOF) at the instrument tip, that can be used by the surgeons during specific phases of the intervention, when they need additional dexterity, such as for suturing.

4.3.1 System description

A novel multi-DOF hand-held mechatronic instrument for laparoscopic procedures is presented. In its design the major drawbacks of currently available instruments and research prototypes have been taken into account, aiming at developing an instrument that allows the surgeon to perform dexterous movements while still being ergonomic and easy to set-up and use.

In particular the aim is to develop a hand-held dexterous instrument, that can be operated by the surgeon with one hand only while standing at the operating table and acting on a traditional laparoscopic instrument with the other hand. The instrument must be readily available and must not require long or complex set-up procedures: it must be possible to use the instrument only during some phases of

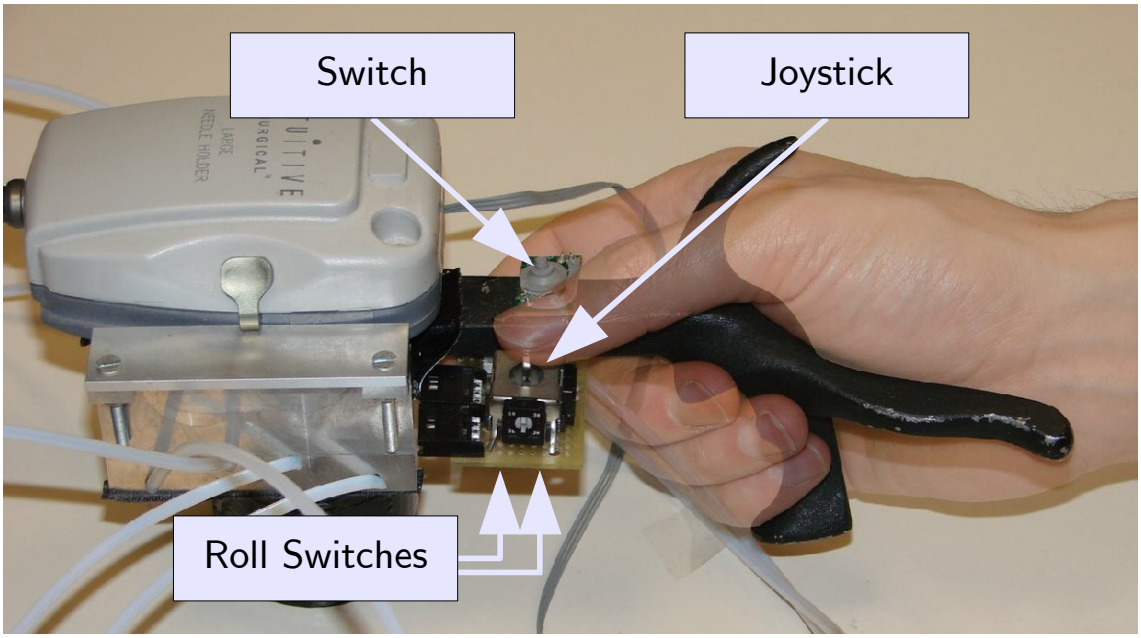


Figure 4.7. Instrument handle with joystick.

the intervention and then to revert rapidly to traditional instruments when the additional DOF are not needed. The instrument must be lightweight in order to hold down the surgeon's fatigue during long interventions. The instrument must have enough DOF to allow the surgeon to perform complex tasks, like stitching and knot tying, at all angles with respect to the instrument shaft. The commands to drive the DOF at the instrument tip must be intuitive and natural at all orientations of the instrument and must not require too much training to be mastered. It must be possible to easily switch the end-effectors of the instrument so that only one robot is needed in the operating room. Available end-effectors should include scissors, graspers, needle holders, dissectors, etc. The separation between instrument and end-effector also has the additional benefit of allowing different sterilization procedures and the disposal of the end-effector. The designed and fabricated instrument has an ergonomic handle and allows to mount the instruments of the da Vinci EndoWrist family. The motors used to steer the tip are dislocated from the handle by using flexible transmissions. The master part has a joystick that allows the surgeon to control the DOF of the end-effector. The handle is a foil grip, because it is very comfortable and there are many sizes that can be matched to the surgeon's hand [20]. The joystick is mounted on this foil grip, as shown in Fig. 4.7. The system has four motors and allows yaw, pitch, grasp and roll movements. Figure 4.8 shows a concept drawing of the instrument.

Motors are placed away from the tool handle and connected to it by means of a flexible transmission. A prototype has been built using a 4-DOF EndoWrist® instrument by Intuitive Surgical Inc. [32], that is connected to the motors using

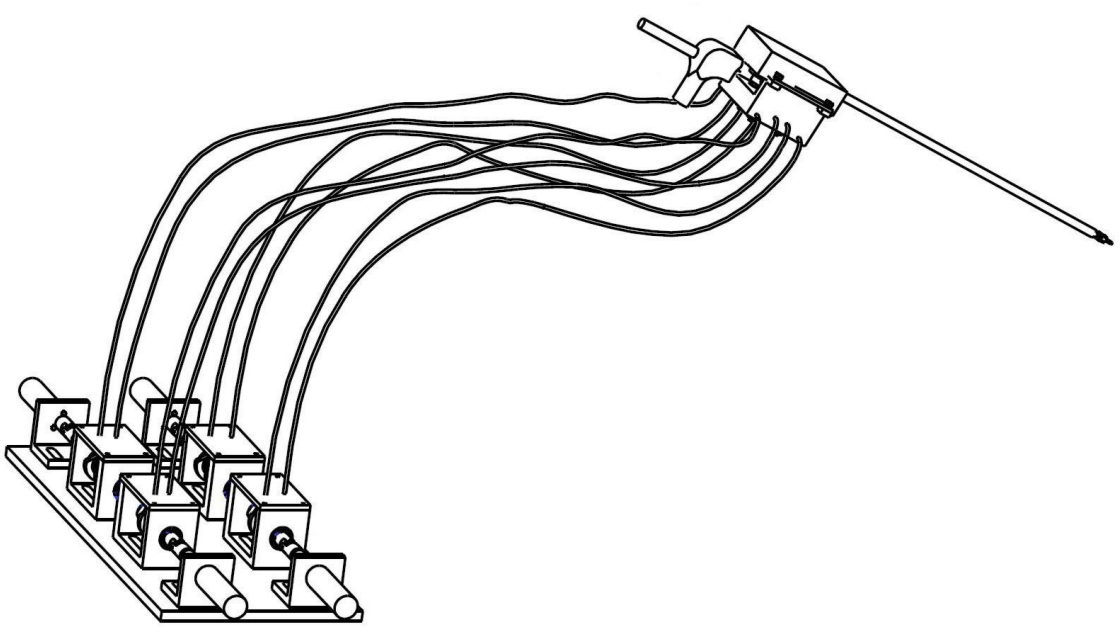


Figure 4.8. Concept drawing of the lightweight hand-held laparoscopic robot.

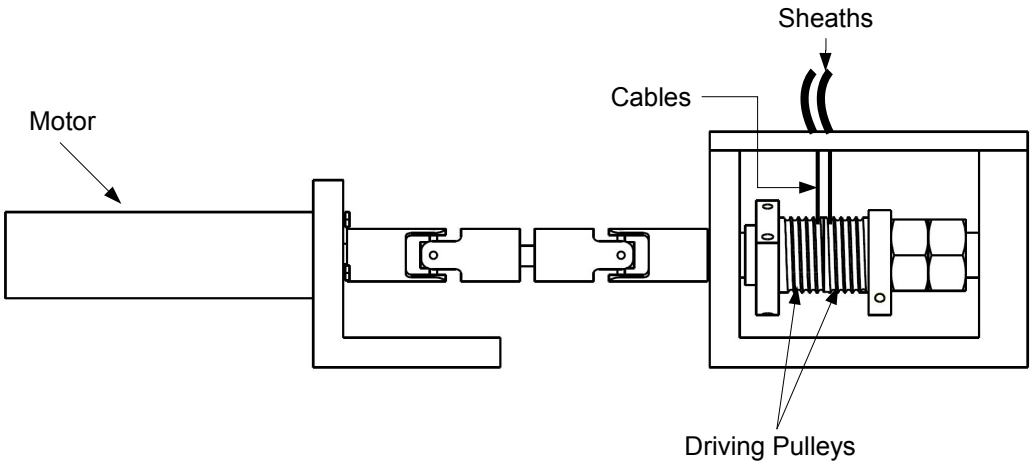


Figure 4.9. Detail on one motor and relative driving pulleys.

tendon-sheath transmissions, the same kind of transmission used in robotic hands. Cables are used to transmit movement from the motors to the tool, and sheaths are used in order to have a flexible transmission, while maintaining the cable in tension. The system is composed by 4 DC motors mounted on a frame and attached to the driving pulleys (see Fig. 4.8). For each of the driving pulleys a stainless steel

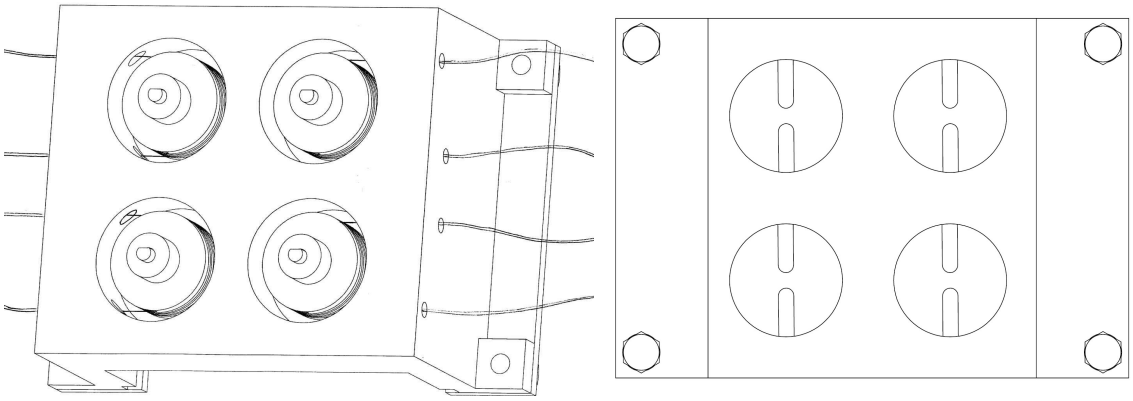


Figure 4.10. Support for the EndoWrist: bottom view (left) and top view (right).

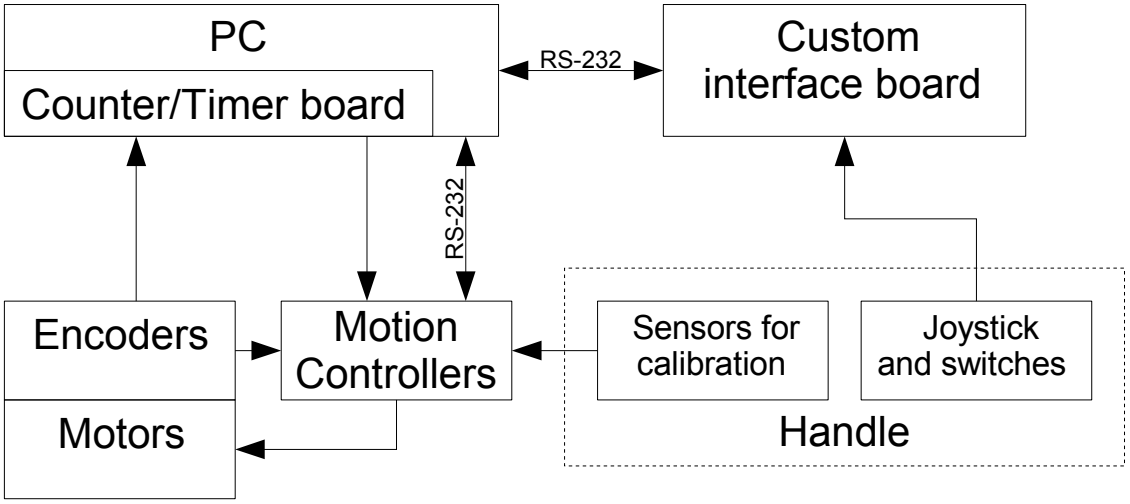


Figure 4.11. System control scheme.

cable is fixed with both ends and is also wound on a driven pulley (see Fig.4.9). Each driving pulley is split in two parts that can counterrotate allowing the cable to be pretensioned, as in [41]. Then these two parts are connected together after the device has been assembled. Another support was built for the driven pulleys and attached to the laparoscopic tool, where other 4 wheels are located, each connected to a driven pulley.

The design of the interface between the support and the tool tip is based on [46]. There are four driven pulleys, one for each driven pulley of the EndoWrist (see Fig. 4.10). The sheaths, two for each pulley, are attached to both supports and the cables slide inside them. The tendon-sheath driving system allows the tool to be moved independently from the position of the motors: these remain fixed to the

base. The weight of the prototype in aluminum of the hand-held instrument is about 300 g, which is lower than the weight of several other robotic devices in previous literature. An analog joystick is used to control the movement of the end-effector of the tool. The DC motors are manufactured by Faulhaber, model 2342018CR coupled with a gear reduction, series 23/1, ratio 43:1. This model has been taken into account after a rough calculation, by considering the order of magnitude of the forces applied to the laparoscopic instrument handles like reported in [7] and in [40]. The gear reduction ratio has been chosen on the basis of the maximum required output torque of about 350 mNm for the grasping function and on the basis of the maximum output speed of about 360 deg/s. The gear reduction ratio is found by dividing the maximum allowable speed for the gear reduction by the maximum speed required for the load and the decision fell on the precautionary value of 43:1 of the 23/1 series. The output power required by the motor is about 30 W for continuous operation. However, for the developed instrument, continuous operation is not required, therefore the choice fell on the 19 W motors, series 2342, powered with 18 V. The control part is formed by a Windows PC, four PI controllers and drivers for the motors. Inside the PC there is a dedicated PCI Counter-Timer board in order to generate the velocity commands or the position commands for the motors. This board is a NI-PCI 6624 manufactured by National Instruments. The motor drivers include the PI motion controllers and these are manufactured by Faulhaber, series MCDC2805. The input signals originate from the handle part of the instrument where a two-axes potentiometer joystick with a momentary switch and two other switches are located. The joystick signals are used to orient the tip of the EndoWrist, one for the pitch, one for the yaw movement and the switch present on the joystick is used for the grasping function. The other two switches are used for activating the roll movement of the stem: one switch is used to rotate in one direction and the other for the opposite direction. These last two buttons are located in the lower part of the handle. All these handle signals are processed by a custom interface board that sends commands to the PC through a RS-232 connection. The control scheme is shown in Fig. 4.11. These informations are related to the position of the joystick and the state of the switches and the software that runs on the PC will calculate the correct values for the velocity commands to send to the motor drivers on the basis of the received informations. Software has been developed in C++ language with Microsoft Visual Studio .NET 2003 and QT3 by Trolltech for the graphical user interface. The velocity commands are sent to the drivers with PWM signals, one for each motor, generated by four counter-timer channels of the PCI board. The position of the motors is read by the PC with the remaining four counter-timer channels connected directly to the encoders. Moreover the motion controllers are connected to the PC with a RS-232 connection. With this configuration the time critical tasks are carried out by the motion controllers that perform the velocity control, also taking into account the range of motion of each motor, while the generation of the command signals for the motors, that is a less time critical task, is left to the software. The problem is to generate signals for the four motors to move the end effector correctly. The kinematics of

the EndoWrist was calculated previously by moving each pulleys in a micrometric slide and photographing the tip position. Using an image analysis software the relationship between the EndoWrist pulleys and the parts that are in the tip was found. Figure 4.12 shows the internal view of the EndoWrist highlights the pulleys involved in the tip movement [53]. The relationships between the angles of the pulleys (ϑ_{130}, ϑ_{132}, ϑ_{136}, ϑ_{134}) and the angles of each part of the tip (ϑ_{154}, ϑ_{52}, ϑ_{581}, ϑ_{582}) are represented by the transmission ratio matrix (4.1).

$$\begin{pmatrix} \vartheta_{154} \\ \vartheta_{52} \\ \vartheta_{581} \\ \vartheta_{582} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1.54 \\ 0 & 0 & 1.03 & 0 \\ 0 & 1.15 & 0.743 & 0 \\ 1.15 & 0 & 0.743 & 0 \end{pmatrix} * \begin{pmatrix} \vartheta_{134} \\ \vartheta_{136} \\ \vartheta_{132} \\ \vartheta_{130} \end{pmatrix} \tag{4.1}$$

With this rough estimation the correct movement of the tip was implemented. It is possible to move the ϑ_{52} without changing ϑ_{581} and ϑ_{582} and this corresponds to the pitch movement if the tweezers are oriented horizontally. This movement is performed with three motors because of the correlation between the parts of the tip. A problem that arises with the use of the joystick is that when there the stem roll angle is non-zero, the joystick vertical and horizontal movements do not correspond to the tip vertical and horizontal movements. The tip direction is calculated considering the stem roll angle and the joystick values in order to maintain the correspondence between the horizontal and vertical movement of the joystick and the tip. Each time the system is powered on an initial calibration is performed because the tip position is unknown. For this purpose there are four omnipolar hall-effect digital switches, one for each handle driven pulley. These sensors are fixed to the EndoWrist support. On the driven pulleys there are four permanent magnets, one for each pulley. Initially, each motor goes to its limit position that is reached when the magnet stands in front of the hall-effect sensor; afterward, each motor goes to the position represented by the middle of his maximum range; then the system is ready to operate. Every motor range is known because it is measured from the EndoWrist pulleys. The speed of the motors that command ϑ_{134} and ϑ_{136} (that move the two parts forming the tweezer, angles ϑ_{582} and ϑ_{581}) is lower than the speed of the other two angles, in fact ϑ_{582} and ϑ_{581} depend respectively from ϑ_{136} and ϑ_{134} and both depend also from ϑ_{132}, therefore for these angles the maximum position limits are reached when ϑ_{132} is already at the limit switch. This particular procedure has been studied for the use of the EndoWrist and it is implemented directly in the motion controllers. This calibration procedure can be carried out every time that the instrument loses the coordination, in fact tendon-sheath drive systems are affected by friction and compliance and they introduce a hysteresis nonlinearity between the joint torque output and the actuator displacement, as reported in [27]. When the system is running, the calibration procedure can be called through the PC software that has a graphical user interface showing the main features like position and current consumption for each motors and permits also to set directly the parameters like tip velocity, motor enable or disable and to call the calibration procedure (see Fig. 4.14).

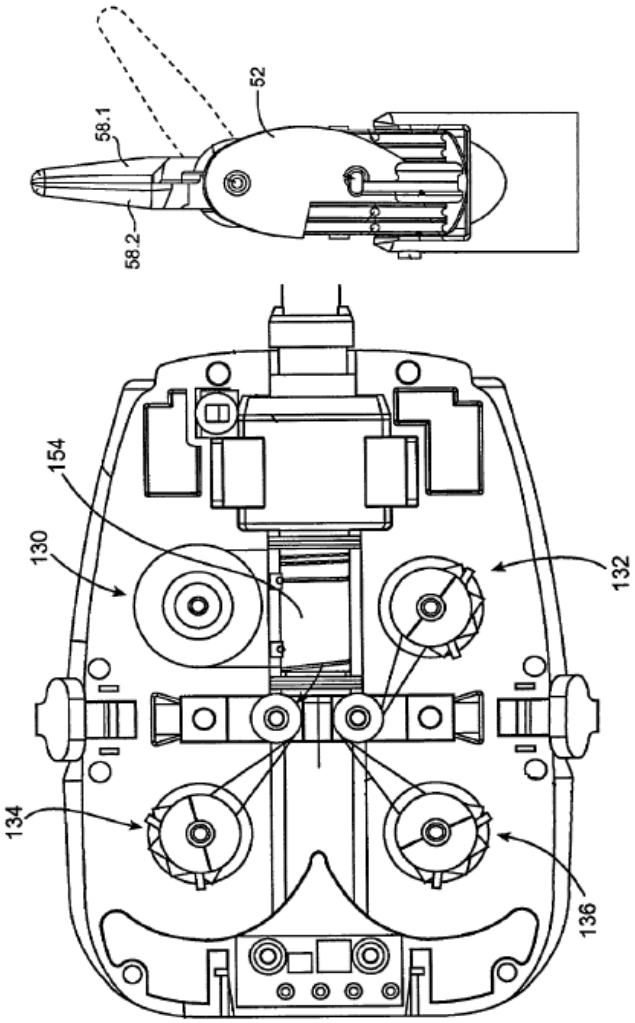


Figure 4.12. EndoWrist schematics (from [32]).

4.3.2 Considerations

With respect to mechanical instruments Leonardo offers a more natural mapping of the degrees of freedom, by allowing to use a 2-DOF joystick to orient the tool tip. The use of a joystick allows to map the orientation of the stick to the orientation of the tool tip, which seemed to be a natural and intuitive mapping. Moreover, the surgeon can decide if a forward movement of the joystick tip corresponds to an upward or downward movement of the end effector according to his own preferences and previous training. As previously discussed, purely mechanical tools usually require two separate controls for orienting the tool tip (e.g. two knobs or a knob and a lever), since a mechanical transmission connecting a multi-DOF handle to the tip would be too complex. Current mechatronic systems adopting the joystick

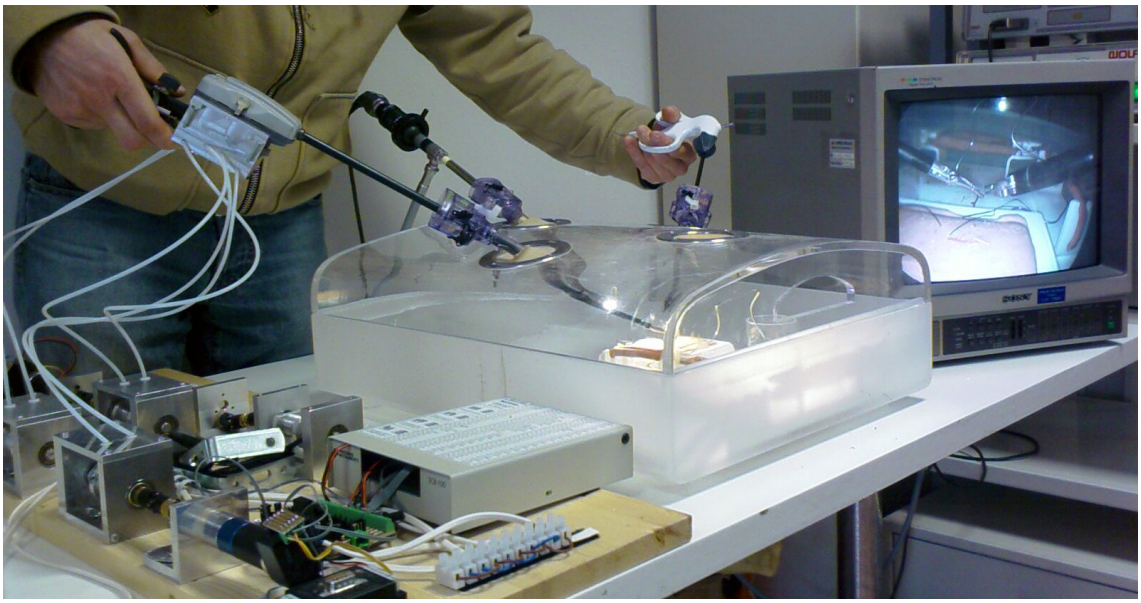


Figure 4.13. The first prototype used in a laparoscopic bench box.

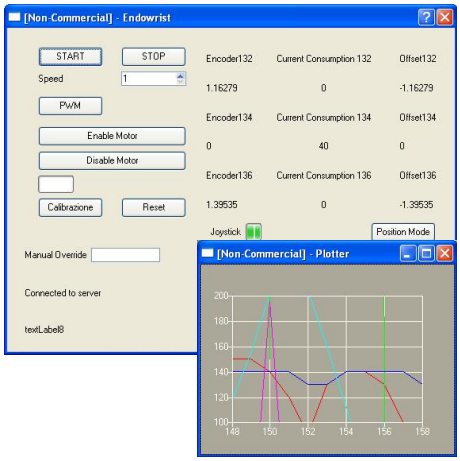


Figure 4.14. Graphical User Interface.

solution have the motors mounted on the instrument. By dislocating the motors, this system is more lightweight and ergonomic.

Two expert surgeons was asked to use the first prototype (shown in Fig. 4.13) to approach a suture from different angles, particularly 0, 45 and 90 degrees, and then to perform the same task with a traditional instrument (see Fig. 4.15). With Leonardo, the surgeons can grab the needle in the correct way at all angles, while approaching the suture at 90 degrees with the traditional instrument is much more difficult. The opinions of the surgeons are encouraging, because they found it very

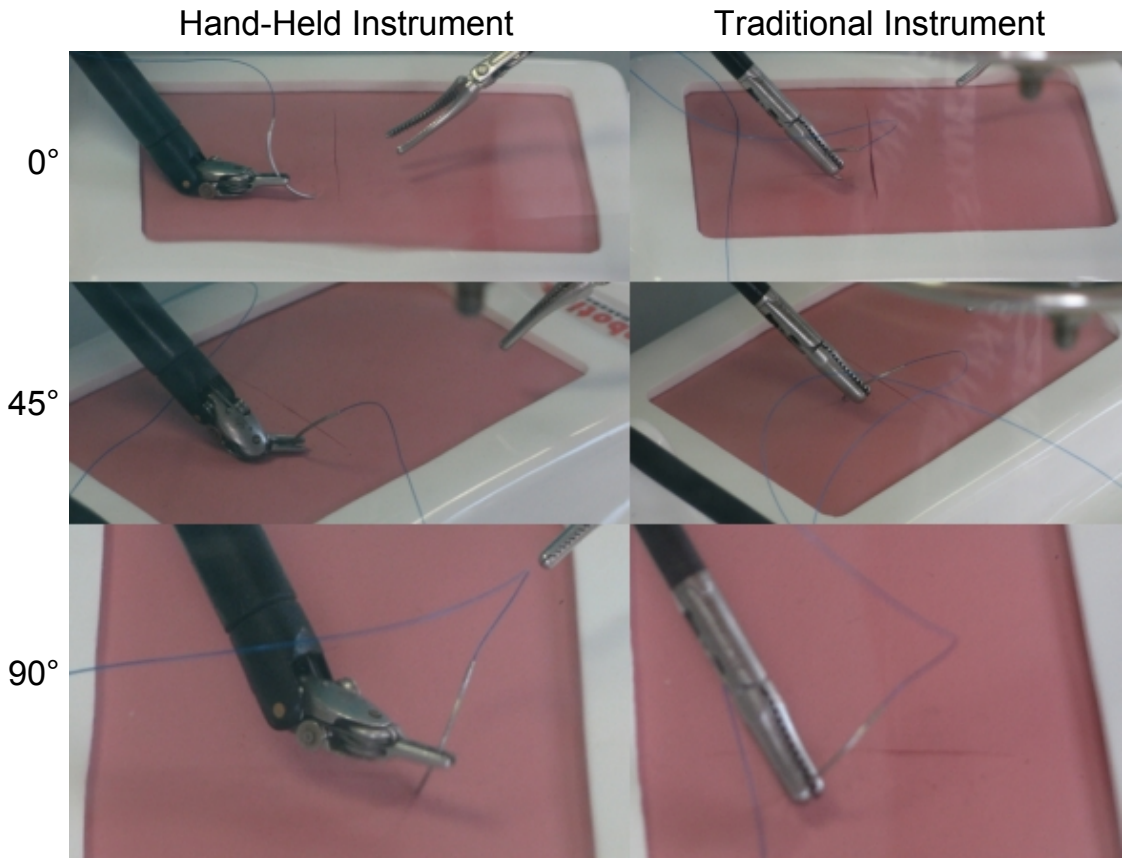


Figure 4.15. The different angles at which the suture was approached.

interesting to have many more DOF than with a traditional instrument, while standing near the operating table and also the possibility to change end-effector in a rapid way was appreciated. On the other hand, the surgeons did not agree with the use of a joystick to control this instrument, in fact they felt this control mode is not very intuitive because they were unable to control both DOFs of the tip orientation simultaneously and ended up controlling only one DOF at a time. It is worth noting, however, that the surgeons started using the instrument with no previous training at all and still managed to perform the sutures correctly.

4.4 Absolute positioning handle

The first prototype has been fabricated in order to test the feasibility of the project and the use of joystick like user interface has shown some limitations. In fact surgeon's opinion was not positive because joystick is a totally different interface from that employed in traditional laparoscopic instruments. The global instrument positioning is performed with the arm movement while the tip positioning is performed

by the surgeon's thumb, acting on the joystick. This consideration led to the study of a more ergonomic handle, basing the design on the study carried out on control modes for a hand-held robotic instrument described in the previous section. Excluding the "tip" position control modes (PCM=Tip) for their complexity in implementation for a hand-held instrument, the choice has fallen on *H_A* modality. This means that the tip orientation is directly mapped to the handle orientation while the global instrument positioning remains the same of the first prototype [44], [33]. The new handle is depicted in Fig. 4.16. Two encoders are used to read the handle yaw and pitch angles that correspond respectively to the tip yaw and pitch angles, while an infrared proximity sensor is used for the grasp angle. The instrument tip is normally open, the pressure on the grasp command causes the tip closing. The navigation switch (see Fig. 4.17) is used for rotating the instrument stem, the more the switch is activated in one direction the more the roll angle will be incremented in the same direction.

The definitive version of Leonardo includes the development of a new dexterous end-effector that substitutes the commercial one. The new end-effector has a roll-pitch roll configuration and the grasping function. Fig. 4.18 depicts the prototype that will be integrated with the new handle as can be seen in Fig. 4.19.

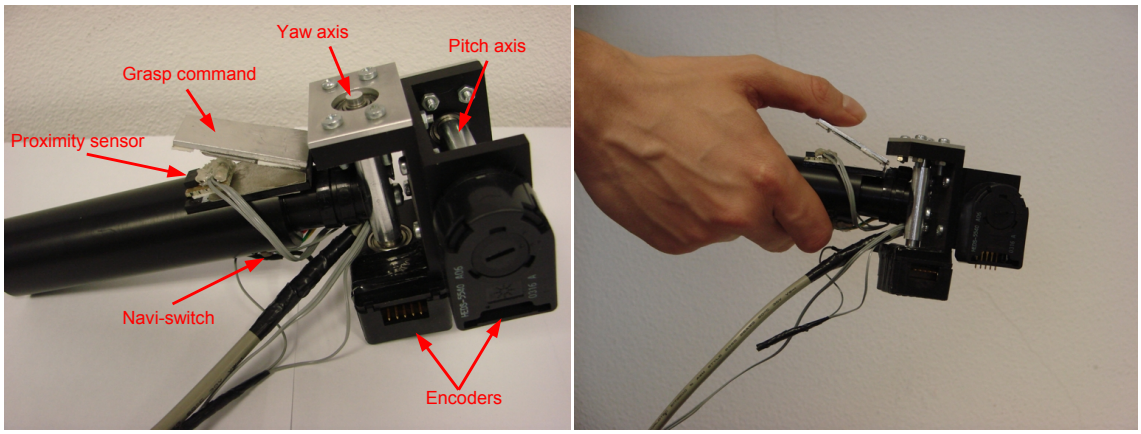


Figure 4.16. Prototype of the new handle for Leonardo

The custom electronic board (see Fig. 4.21) receives the signals from encoders, infrared sensor and navigation switch and then, after the elaboration, signals are sent to the National Instruments 6624 board that is plugged in the PC dedicated to calculate the handle position. In particular the infrared proximity sensor produces an analog output that is sampled by a microcontroller PIC16f876 with sampling time T_s=1ms and converted in a pulse width modulated signal of frequency f_{PWM_1}=1.22kHz. The navigation switch produces two digital signals that are read by the microcontroller with the same sampling time of the infrared sensor and converted in a second PWM signal of frequency f_{PWM_2}=1.22kHz. The duty cycles of

Figure 4.17. Navigation Switch

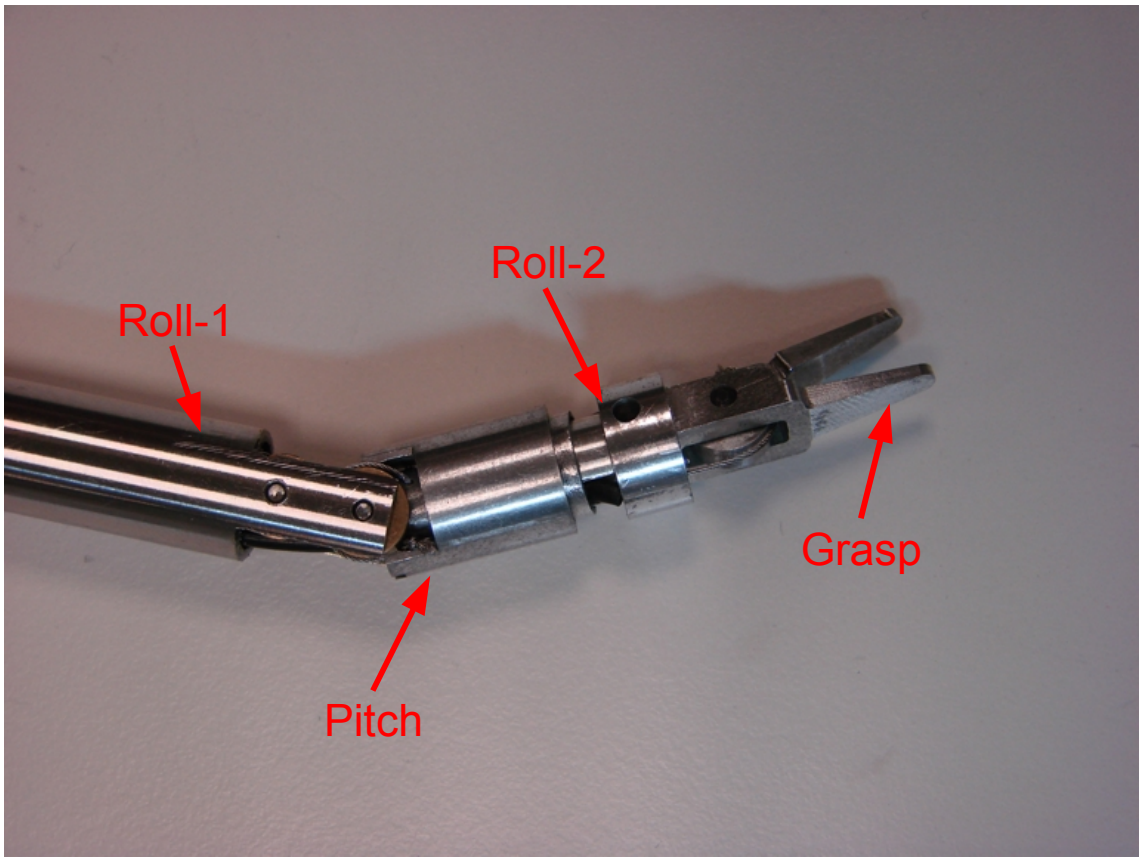


Figure 4.18. New end-effector for Leonardo

the two PWM signals represent respectively the opening angle of the tip and the roll angle of the instrument stem and the relative code implemented in the microcontroller is reported in listing 4.1. The two navi-switch digital signals are processed and a position information is generated, the more the switch is pressed in one direction the more the duty cycle of PWM_2 will be incremented or decremented depending

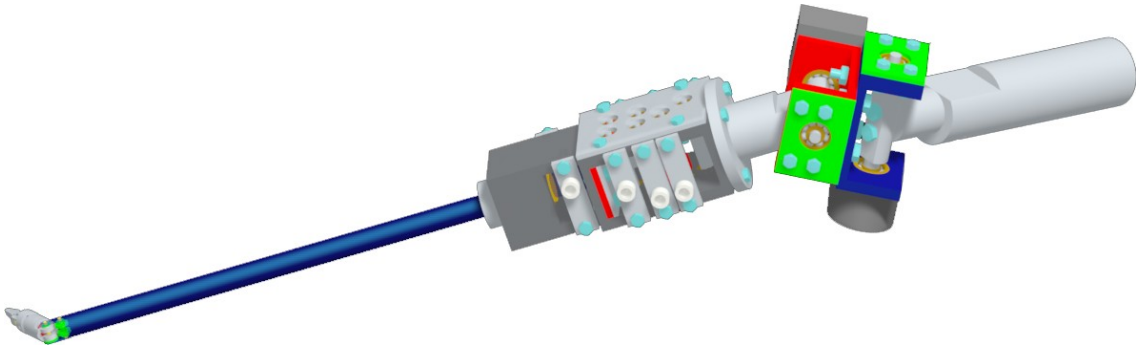


Figure 4.19. Leonardo2.

on the rotational direction. Figure 4.20 depicts the GUI relatives to the signals read from the handle.

The final version of the instrument will be more embedded because an instrument that needs complex components, like PCs, or complex wiring systems is not suitable for real use. This consideration is also very important thinking to the future development of a commercial product. The main requirements are relative to encoder reading, sensor reading, mathematical calculation and pulse width modulated signal generation. Digital signal processors can be used for substituting the PC in performing the control algorithms, since they fit quite well these specifications. They can perform encoder reading, floating point calculations, signal sampling and so on. Moreover DSPs have very compact dimensions and they can be fitted in small packages. The selected DSPs are from Microchip, model dsPIC30F2010 and for the laparoscopic instruments two of these DSPs are needed. One is dedicated to the first handle encoder (yaw angle) while the second DSP is dedicated to read the second handle encoder (pitch angle). The first dsPIC is also responsible for the generation of three PWM signals for three motors (yaw, pitch and grasp tip angles) while the other is responsible for the motor that actuates the roll angle and at the same time communicates the roll angle values to the first dsPIC (through IIC serial communication protocol) that calculates the three motor angles according to the direct kinematic of the robot. Fig. 4.22 depicts the proposed circuit schematic.

Listing 4.1. Analog and digital signals to PWM signals. PIC C code

```
if((tr_x>10)&&(tr_x<110)){
 y=tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+tr_x;
 y=y+22; // y=8*tr_x+22 linear interpolation
         // of the sampled analog signal
```

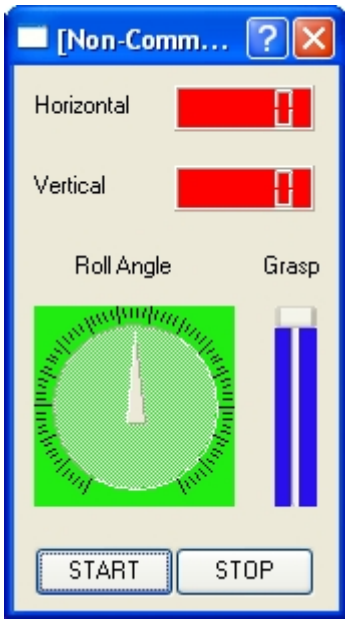


Figure 4.20. Graphic user interface for the new handle

```
        // relative to grasp angle
 CCP1CON.4=y.0;
 CCP1CON.5=y.1;
 CCPR1L.7=y.9;
 CCPR1L.6=y.8;
 CCPR1L.5=y.7;
 CCPR1L.4=y.6;
 CCPR1L.3=y.5;
 CCPR1L.2=y.4;
 CCPR1L.1=y.3;
 CCPR1L.0=y.2; // CCP1CON and CCPR1L
               // are registers relative
               // to the duty cycle
}
if(tr_x<=10){
 y=50;
 CCP1CON.4=y.0;
 CCP1CON.5=y.1;
 CCPR1L.7=y.9;
 CCPR1L.6=y.8;
 CCPR1L.5=y.7;
 CCPR1L.4=y.6;
 CCPR1L.3=y.5;
 CCPR1L.2=y.4;
 CCPR1L.1=y.3;
 CCPR1L.0=y.2; // duty cycle can not be
               // less than 20\% and
               // greater than 80\%
}
if((PORTB.4==0) && (y2<902) ) {
 y1++;
 y2=y1>>5; // duty cycle of the roll angle
           // it is incremented every 2^5
           // cycles
```

```
 CCP2CON.4=y2.0;
 CCP2CON.5=y2.1;
 CCPR2L.7=y2.9;
 CCPR2L.6=y2.8;
 CCPR2L.5=y2.7;
 CCPR2L.4=y2.6;
 CCPR2L.3=y2.5;
 CCPR2L.2=y2.4;
 CCPR2L.1=y2.3;
 CCPR2L.0=y2.2;
}
else if((PORTB.5==0) && (y2>102)) {
 y1--;
 y2=y1>>5; // duty cycle of the roll angle
           // it is decremented every 2^5
           // cycles
 CCP2CON.4=y2.0;
 CCP2CON.5=y2.1;
 CCPR2L.7=y2.9;
 CCPR2L.6=y2.8;
 CCPR2L.5=y2.7;
 CCPR2L.4=y2.6;
 CCPR2L.3=y2.5;
 CCPR2L.2=y2.4;
 CCPR2L.1=y2.3;
 CCPR2L.0=y2.2;
}
```

4.5 Conclusions

In this chapter the design of a hand-held robotic instrument for laparoscopic surgery has been presented together with the study on the best control mode for a laparoscopic instrument. Results showed that users with no experience in laparoscopy simulators performed quite well the implemented exercises in the virtual environment with the control mode corresponding to the developed prototype. The evaluation of the control modes benefited of the implemented simulation of the instrument end-effector since only four different handles have been created. This approach greatly sped up this phase. The developed prototype has a joystick interface that has not encountered the surgeon's favourable opinion. A different handle has been developed following the indications came out from the control modes study. The purpose is to give to the surgeon a more ergonomic interface, that can be used intuitively with only low practice. Finally CAD schematic and a picture of a new dexterous end-effector is presented in order to complete the development of a new laparoscopic instrument. Also a solution for embedding the whole system is briefly explained. This is a very important aspect from the commercial point of view since the next step of this project will be the in-vivo testing of the instrument.

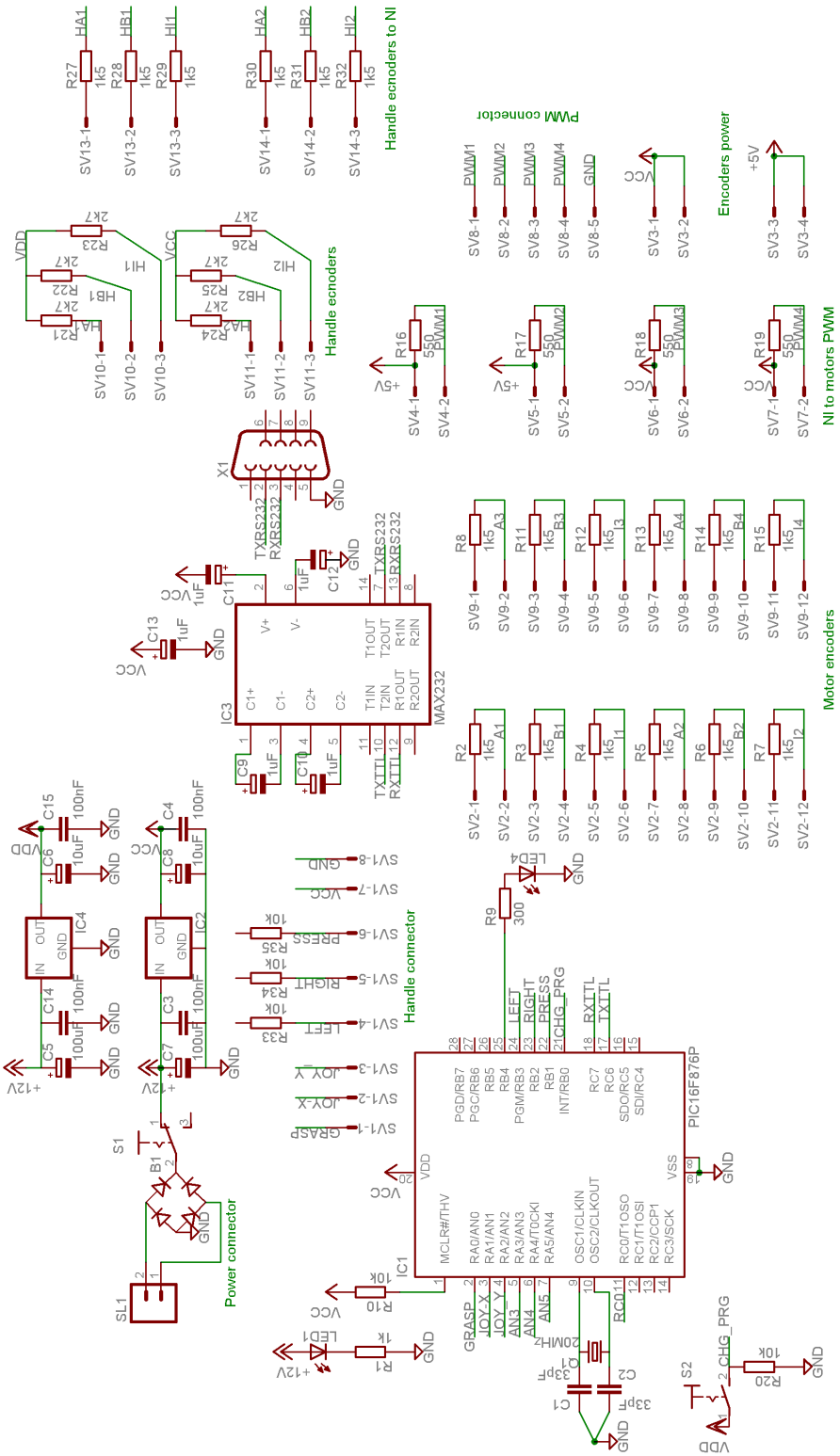


Figure 4.21. Schematic board for the new handle

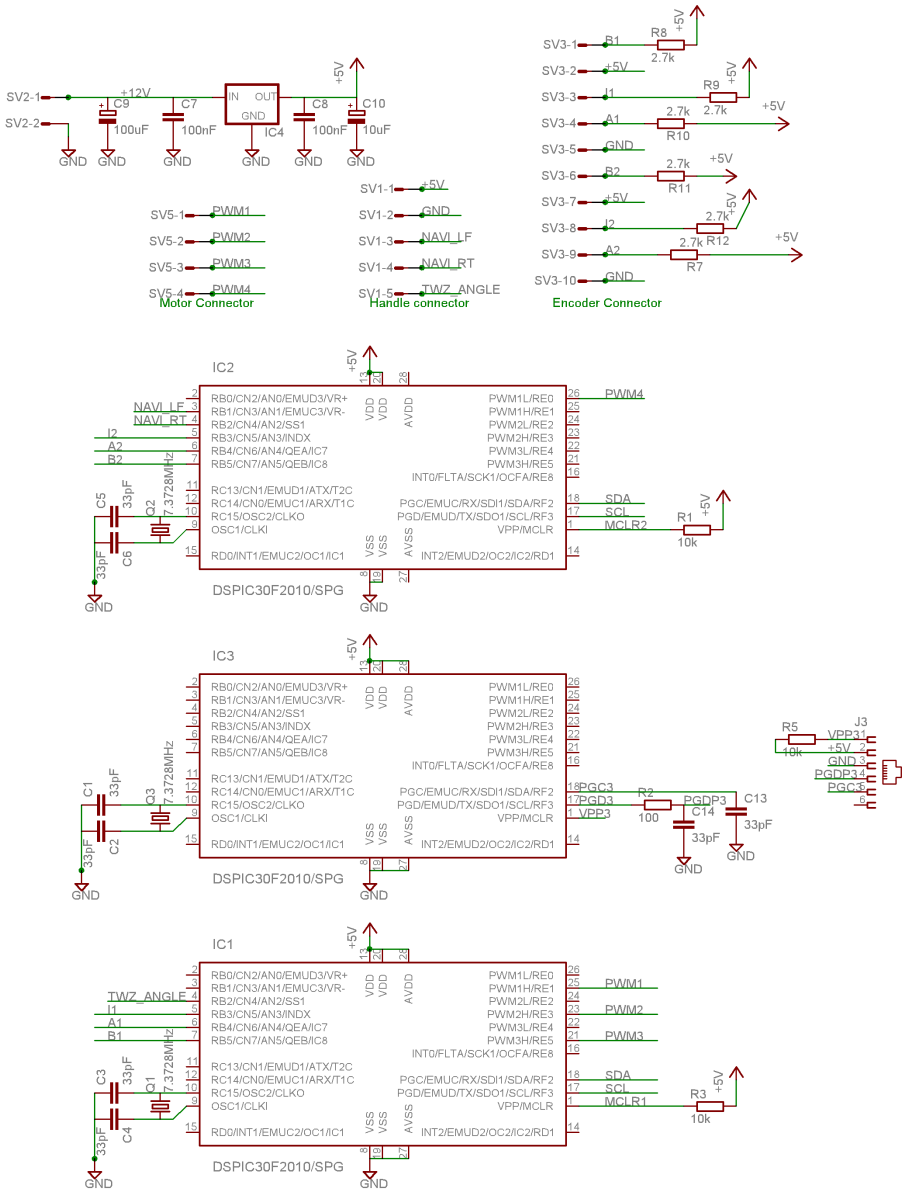


Figure 4.22. Schematic board for the embedded version of Leonardo

Chapter 5

Endoscope with enhanced resolution

5.1 Introduction

Visual acuity of artificial vision systems is currently well below the human eye. This is due to the limited resolution of both cameras and displays. Nonetheless, acuity is very important for visual inspection tasks. Visual acuity, in cameras as well as in animal eyes, can be increased by making smaller receptors or bigger eyes. In some scenarios, the size of the camera is constrained, so alternative solutions must be sought. In this work a robotic dual-camera vision system is presented with a design inspired to the visual system of jumping spiders (*Salticidae* family). The system is composed of telephoto camera whose field of view can be moved within the larger field of view of a wide-angle camera and allows to form a high-resolution image, i.e. an image at the resolution of the telephoto camera with the field of view of the wide-angle camera. The design of the robotic system is described, together with the direct and inverse kinematics, and the image acquisition and fusion algorithms that allow to build the high-resolution image. Sample images from experiments are also presented, together with a discussion on sources of errors and possible solutions.

5.2 Classic endoscopy

Commercial endoscopic systems can be divided in two big categories: rigid and flexible endoscopes. All the instruments have an illumination system, an image transmission system and the central processing unit. The illumination system is composed by a white light source like a xenon lamp that is connected to incoherent optical fibers. This solution is typical for rigid endoscopes while modern flexible instruments have light emitting diodes (LEDs) mounted directly in the tip. Traditional rigid endoscopes have an optical lens system that carries the image directly to the sensor that is placed in the distal part of the instrument. Charge-Coupled Devices (CCD) and Complementary Metal Oxid Semiconductor(CMOS) represent

the two categories of sensors employed for image acquisition. Recent research studies have demonstrated that CMOS sensors are approaching the quality level of CCD sensors with lower power consumption, size requirements and costs. The typical architecture of a rigid endoscope is depicted in Fig. 5.1, where optical fibers are employed for illumination and are collocated in the peripheral space. The optical lenses system is located in the center of the instrument and is composed with unitary magnification. Early rigid endoscopes had lenses separated by air (Niezte endoscopes) while modern instruments are composed by cylindrical lenses called "rod lenses" that allow to obtain superior light transmission. Bright images, better color correction and wider visual field are the main results achieved with technological improvements. All these advantages led to endoscopes with lower diameter. GRadient INdex (GRIN) endoscopes are composed by a single cylindrical lens, they have great optical quality but they need complex process fabrication limiting their use.

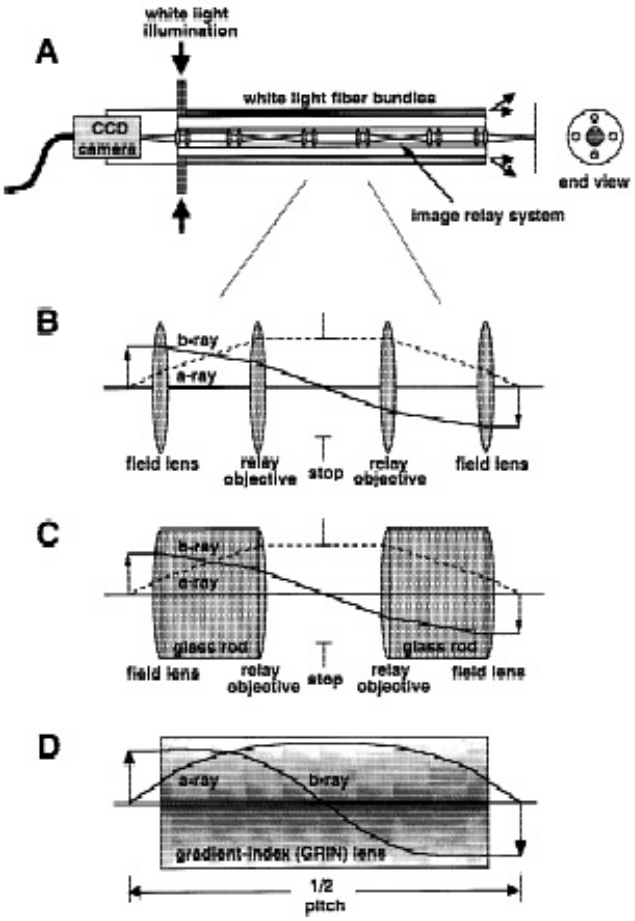


Figure 5.1. Typical architecture of a rigid endoscope (**A**). Evolution from the first systems (**B**), to "Hopkins" systems (**C**), to single lens endoscopes (**D**).

Flexible endoscopes can adapt their shape to the cavity in which they are inserted. Early flexible instruments had coherent optical fibers (diameter of the single optical fiber 10-125 μm) that carried out the images. The image sensor was placed in the distal part of the instrument (see Fig. 5.2(A)). Modern flexible endoscopes have the CCD sensor directly mounted on the tip (see Fig. 5.2(B)), minimizing image quality losses and distortions present in optical transmissions.

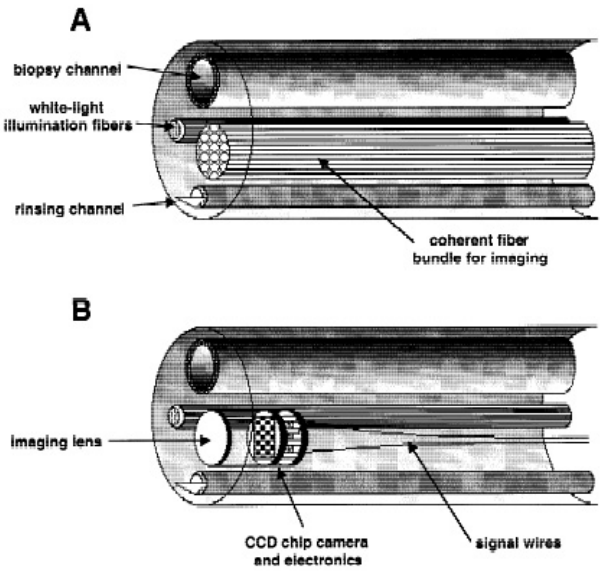
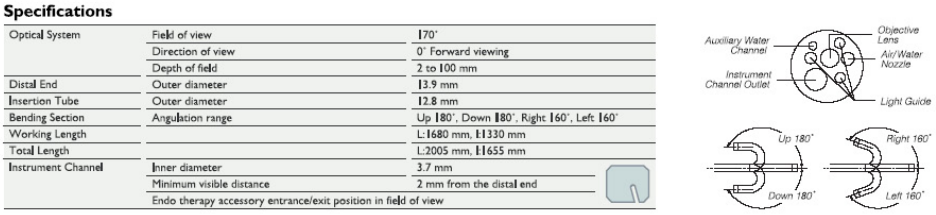


Figure 5.2. Optical image transmission (**A**) or electrical wired transmission (**B**).

The lack of optical fibers results in free space that can be used for placing small surgical instruments inside the endoscope as depicted in Fig. 5.2. Colonoscopes are flexible endoscopes used for both diagnostic and therapeutic purposes since micro-instruments are present in the tip for biopsy (see Fig. 5.3).

Figure 5.3. Flexible endoscope for diagnostic-therapeutic intervention: surgical instrument are inserted through the operative channel (www.fujinon.com).



Specifications

Optical System	Field of view	170°
	Direction of view	0° Forward viewing
	Depth of field	2 to 100 mm
Distal End	Outer diameter	13.9 mm
Insertion Tube	Outer diameter	12.8 mm
Bending Section	Angulation range	Up 180°, Down 180°, Right 160°, Left 160°
Working Length		L:1680 mm, H1330 mm
Total Length		L:2005 mm, H1655 mm
Instrument Channel	Inner diameter	3.7 mm
	Minimum visible distance	2 mm from the distal end
	Endo therapy accessory entrance/exit position in field of view	

Figure 5.4. Colonoscope Datasheet. Olympus CF Type H180AL/I.

Fig. 5.4 depicts a modern model of flexible endoscope (EVIS EXERA II Colonvideoscope Olympus CF Type H180AL/I). This instrument is equipped with a CCD HDTV sensor able to acquire images with 1080 lines instead of 576 lines obtained with PAL standard.

5.3 Scientific research

In minimally invasive surgery research works are involved in the improvements of image quality in terms of contrast, brightness and resolution, for diagnostic purposes as well as for intervention procedures. Some general points can be highlighted:

- Steady images should be acquired filtering the physiological tremor of the surgeon's assistant,
- The field of view has to be incremented in order to reduce the "keyhole" effect. The field of view of colonoscopes is wider (170°) than rigid laparoscopes (60°);
- The resolution should also be incremented because diagnoses are based on images. Modern flexible colonoscopes support HDTV (1080 lines) while in laparoscopy, commercial instruments have PAL (576 visible lines) or NTSC standard (484 lines);

In laparoscopy robotic systems could be useful for their accuracy and precision. AESOP represents an example of robotic camera holder that accepts in input voice commands from the surgeon for positioning and orienting the endoscope. LER [4] is a robot directly positioned on the insertion point of the endoscope in the abdomen and it can be employed to move the laparoscope with three degrees of freedom (DoFs). These systems can be employed for laparoscopic instruments tracking, involving image processing tasks for instruments detection. Wei et al. [55] have used colored markers fixed to the surgical instrument to identify them but the procedure could fail if more than one instrument are present in the scene. Wang et al. [54] proposed a color based approach to distinguish organs from the instruments but problems raise with color variability that can introduce uncertainty in localization tasks. Tonet et

al. [47], used colored markers to localize laparoscopic instruments in 3D space while Voros et al. [52] used the 3D projection of the trocar insertion point to localize the instrument tip without using markers or color identifiers. Laparoscopes have a narrow field of view (60°) and during the intervention the surgeon's assistant accommodates the endoscope according to the surgeon's directives. Instruments tracking allows to keep the tips in the centre of the field of vision and the use of micro-robot provided with one or two cameras could be a solution to implement this task. The idea is to develop robots that are located inside the abdominal cavity. In [34] a stereoscopic vision system is described. Two cameras are actuated by two motors and a wireless transmission system send the images to the central unit. The robot has 5 DOFs, a diameter of 1.75 cm and length of 19.81 cm (see Fig. 5.5). The cameras are withdrawn in order to make possible the robot insertion inside the body through the trocar and after the cameras are positioned along the orthogonal axis respect to the instrument body axis.

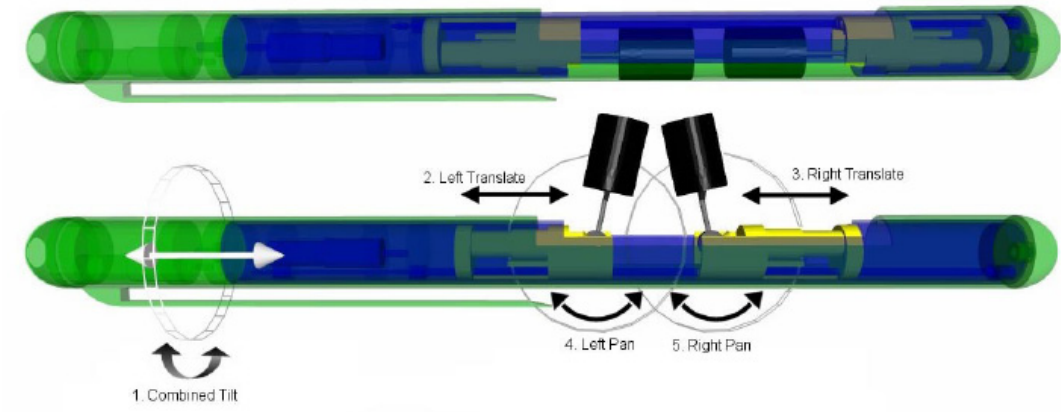


Figure 5.5. CAD scheme of the robot presented in [34]; the system with withdrawn cameras is depicted on top while on the bottom the two cameras are positioned for the stereoscopic vision.

Rentschler et al. [37] proposed a laparoscopic robot able to move inside the abdomen by means of two screw thread wheels and it has a wireless camera fixed on it (see fig. 5.6).

Kim et al. [28] proposed a fixed endoscope with mobile internal components in order to cover a field of vision of about 90° (see Fig. 5.7). Two prisms are moved by means of two motors (see Fig. 5.8) while the optical aperture remains constant This results in adding 19.5° to the field of vision.

Image-shift mechanisms are used to acquire two images at the same moment. One image channel has a wide field of vision while the other has a narrow field of vision and can be moved to cover the area of the first image. Two Porro prisms are employed to obtain the image shifting [57] and they are manually moved (see Fig. 5.9).

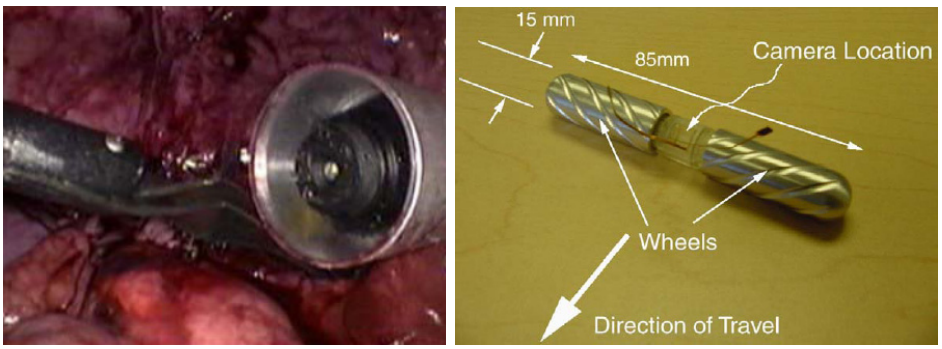


Figure 5.6. Micro-robot for laparoscopy [37]: the wireless camera is depicted on the right while the mobile robot is depicted on the left.

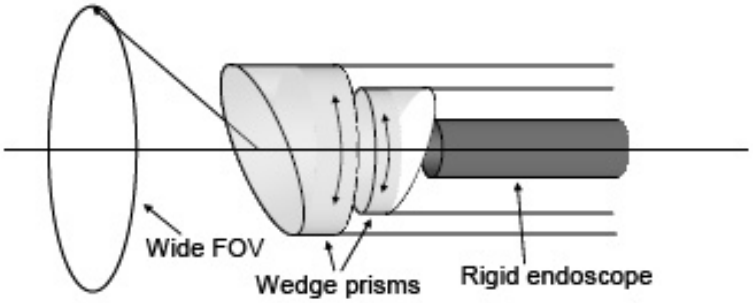


Figure 5.7. Simplified scheme of the system presented in [28] with two prisms.

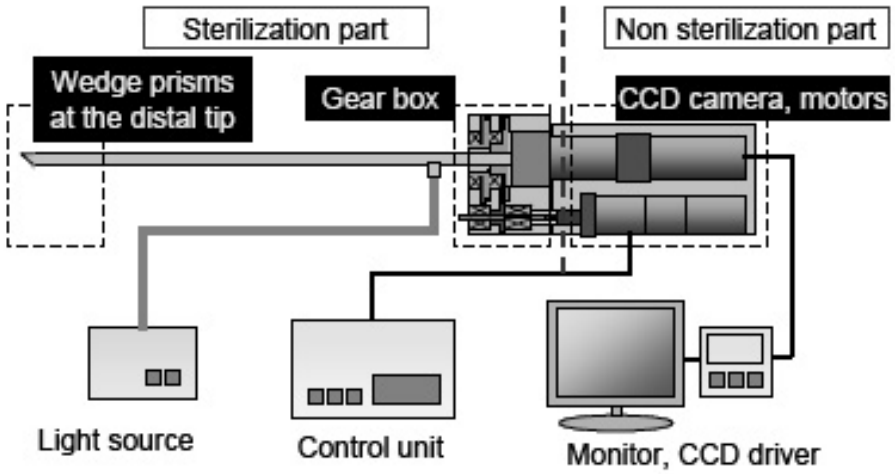


Figure 5.8. Ensoscope schematic view [28].

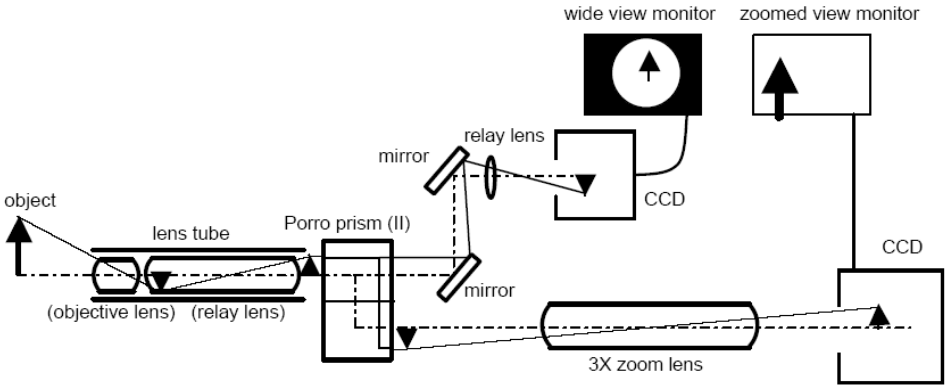


Figure 5.9. "Dual-view" endoscope with two Porro prisms [57].

The internal mechanism has mirrors and magnifier lenses that reflect images toward the two cameras. Two separate monitors show the two images to the surgeon.

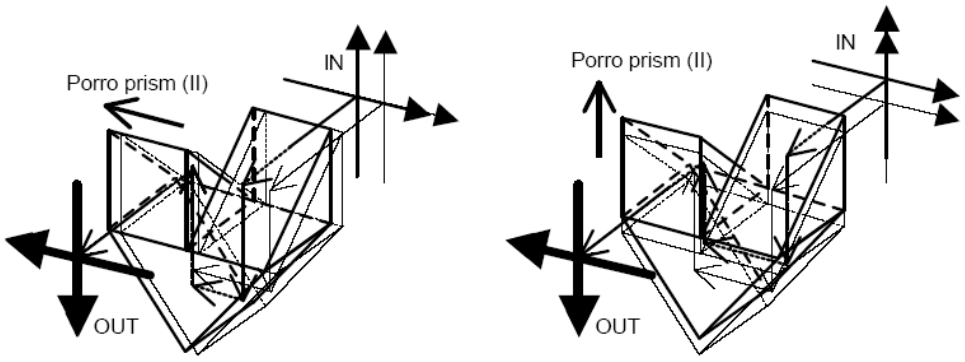


Figure 5.10. Detail on the mechanism for shifting the field of view horizontally and vertically with the prism movement.

Yelin et al. [59] propose a miniaturized endoscope able to give a 3D image, the instrument is composed by a single optical fiber that transports policrome light and each color is projected in a different space position. A three-dimensional image (see Fig. 5.11(a)) is obtained by moving also the optical fiber for the third dimension (see Fig. 5.11(b)).

Infrared Ray Electronic Endoscopy is a new technique [22] that is employed for lymph nodes identification using near-infrared wave lengths (805 nm). These wave lengths correspond to the absorption peak of ICG molecule with which the patient is previously treated. Damaged tissue will appear darker than sane tissue with near-infrared illumination (see Fig. 5.12).

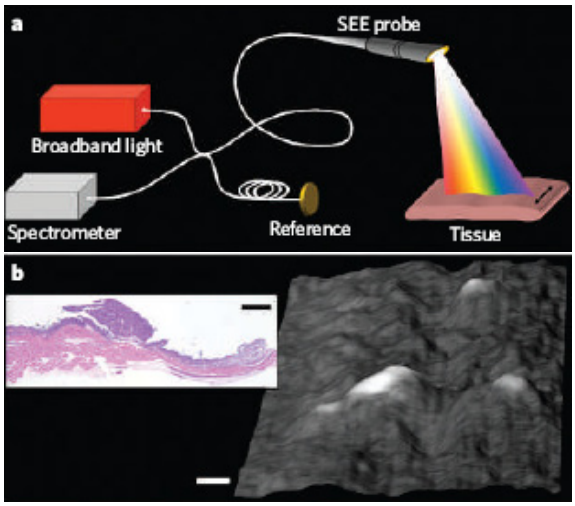


Figure 5.11. (a) Three-dimensional spectral encoding endoscope. (b) [!]hystologic and 3D *in-vivo* image.

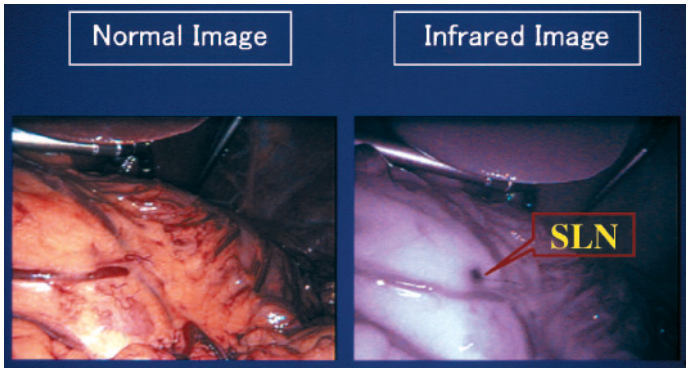


Figure 5.12. Comparison between the image obtained with visible light and the image obtained with near-infrared light after ICG treatment.

5.4 System description

The aim of this project is to build a vision system with enhanced resolution trying to use two small cameras instead of a single camera with a big optical unit. The nature has overcame size dimensions for the spider of "Salticidae" family by using more optical units, they have six principal eyes with high resolving power but narrow FOV (field of vision) and two secondary eyes with low resolving power and wide field of vision. The spider has a visual acuity higher than other insects, by combining this two vision systems. Following this approach, the developed endoscope is composed of a telephoto camera and a wide-angle camera. The former acts as salticids' principal eyes and is actuated to allow 2 degrees of freedom (DoF) of scanning movements. The latter acts as salticids' secondary eyes, allowing a low-resolution view of a large visual field. As in salticids, the secondary eye is used as movement detector. Rather than using the primary eye to scan continuously and periodically a wide FOV, we scan the large FOV of the secondary eye only once at the beginning and then only update those regions where the secondary eye has detected changes. This allows to keep the primary eye pointed at those regions where movement occurs, rather than wasting time to scan the whole FOV. The system is therefore more suitable for applications where the camera position is fixed: in mobile robots, the entire image changes at every movement and an update of the whole image would be needed.

Finally, unlike in [57], where the user is presented with two distinct wide and narrow FOV images, in this work the images of the primary eye are merged together into one big image that covers the whole FOV of the secondary eye. The scanning movements of the primary eye ensure that the whole image is always up to date. In other words, the system allows to form a high-resolution image (i.e. at the resolution of the primary eye) with a wide FOV (i.e. the FOV of the secondary eye).

Figure 5.13 shows a CAD drawing of the design. The *secondary eye* is composed of a CMOS sensor coupled with a pinhole optics (MO-S588 by MISUMI Corp.), with a short focal lenght (f=3.1 mm) and a wide FOV (61.3° horizontally and 46° vertically). The *primary eye* is composed of a CCD sensor (STC-R640C by Sentech, Sensor Technologies America, Inc.) coupled with a telephoto lens (L620 by Sentech), with a long focal length (f=16 mm) and a narrow FOV (11.6° horizontally and 8.7° vertically). Both cameras have analog NTSC output. These cameras and optics were chosen for their small size (8 mm diameter) and because they were readily available off-the-shelf: the aim is to demonstrate the dual-camera concept rather than developing a system tailored for a specific application. Therefore the main requirement is that the primary eye has a higher resolution than the secondary. The resolution of the primary eye is about 7 times higher. As in salticids, the FOVs of the primary and secondary eye overlap. The secondary eye is fixed, while the FOV of the primary eye can be moved horizontally and vertically, allowing scanning movements. In salticids, the lens of the primary eye is fixed, and six muscles attached to the eye tube can make the retina move with 3 DOF; in the system, both the sensor and the lens are fixed: instead, the movements of the visual field are obtained thanks to the movements of a first surface mirror that is placed in front of the CCD camera

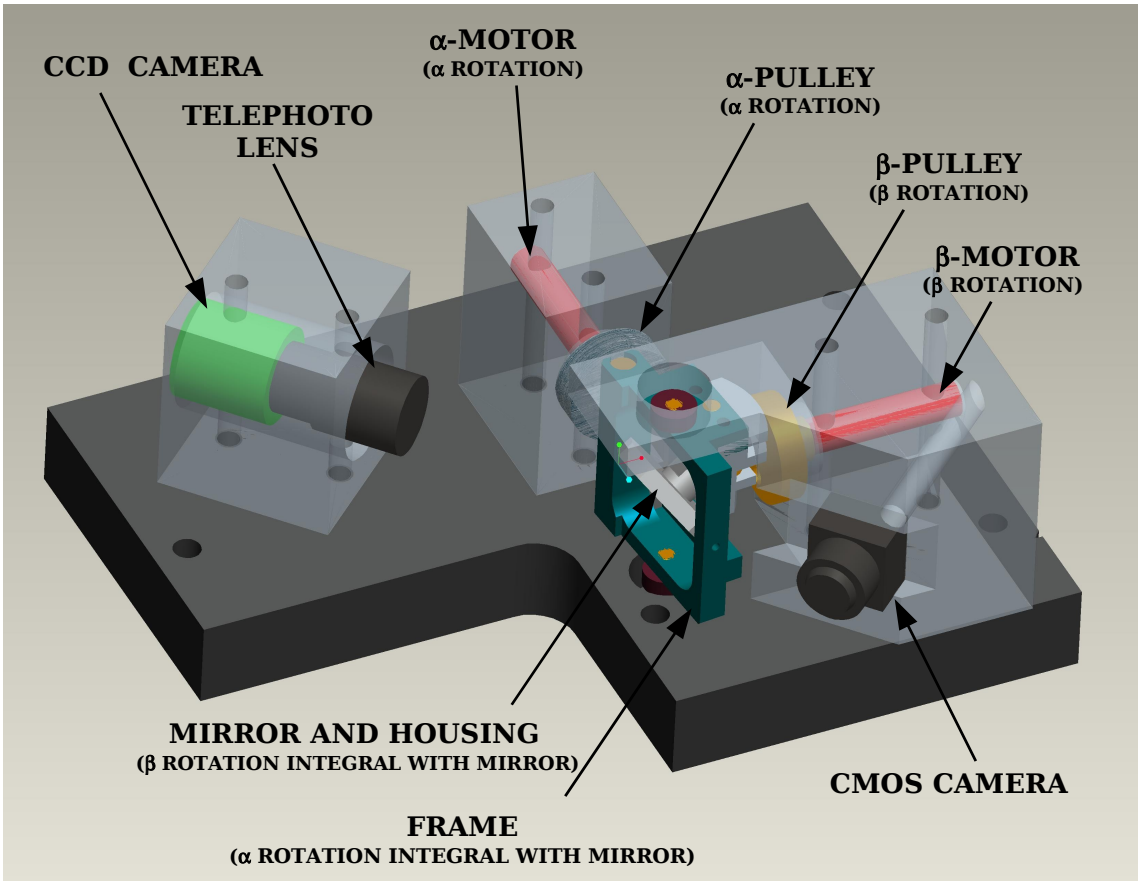


Figure 5.13. CAD drawing of the vision system, highlighting the main components.

(12.5 mm diameter and 0.8 mm thickness). By tilting the mirror in two directions the reflection angle varies and the FOV can be moved inside of the larger FOV of the secondary eye. The optical axis of the CCD camera and telephoto lens are mounted perpendicularly to the optical axis of the CMOS camera. Placing the mirror at 45° with respect to the vertical axis reflects the optical axis of the CCD camera parallel to the CMOS camera, so that the two "eyes" point in the same direction. The FOV of the primary eye is changed with the mirror movements rather than rotating the whole camera system (sensor, optics and electronics) because of the lower inertia [36] and absence of electric wires in moving parts. The mirror is mounted in a support that has 2 DOF and is actuated by two small brushless motors (SBL04-0829 PG04-337 by Namiki Precision Jewel Co., Ltd.) with SSD04 drivers (also by Namiki). These motors have a diameter of 4 mm and like the optical components were chosen with the aim of building a smaller device.

Figure 5.14 shows the hardware components of the vision system and also the software processing loop. The system is equipped with a main processing unit (MPU) that deals both with motor control and image fusion. The MPU periodically samples the image I_S from the CMOS sensor. When a change in a region of the

CMOS image is detected, the corresponding region in the high-resolution image I_H needs to be updated: the MPU computes the driving parameters (position and speed) for the motors in order to rotate the mirror at the right angles and to frame the desired area with the CCD camera. The image I_P is then sampled from the CCD sensor and patched on the high-resolution image I_H at the corresponding spot. Before doing this, the image needs to be corrected for radial distortion (camera calibration) and projective distortion (due to the different viewing angle of the two eyes). This loop is repeated until all changes on the secondary eye image have been re-sampled with the primary eye. The result of this mosaicking process is an updated high-resolution image (skip to Fig. 5.24 for an example).

To safely interface the MPU with the motors, a custom electronics board has been developed: it is used to buffer and decouple the input/output signals sent by the DAQ board to the motor drivers. The board also includes the power module for the motors and sensors and is able to read the motor counter-electromotive voltage that is related to the speed of the actuator. The output is a pulse signal in which every rising edge corresponds to a complete drive shaft revolution. The motor positioning is performed taking counting the pulses needed to reach the desired position, taking into account the reduction ratio of 337:1. The motor positions corresponding to the desired mirror orientation are calculated using the kinematic description of the system shown in Section 5.5. Moreover, two digital hall effect sensors have been used to identify a known calibration configuration, which is used to calibrate the system at power on.

The MPU is based on a personal computer (PC) running Microsoft Windows XP, the images are sampled by means of a Flashbus MV frame grabber (Integral Technologies, Inc.), the motors and position sensors are controlled by means of a PCI-NI6025E DAQ board (National Instruments Corp.). The processing loop has been implemented in C++ language with Microsoft Visual Studio .NET 2003; image processing has been performed with the Halcon 7.0 library (MVTec Software GmbH) and kinematics computation by means of MATLAB 7.1 (The Mathworks, Inc.) code, exported in C++.

Finally, Fig. 5.15 further clarifies the parallelism between the developed vision system and salticids' eyes, pointing out the corresponding components.

5.5 Mechanisms

Yaw movement of field of view is obtained by rotating the mirror around a vertical axis. The α-pulley is integral with the α-motor shaft and has an eccentric pin mounted on it. The pin is inserted orthogonally in a cylindrical slider that transmits the rotational movement to the mirror housing. Motor shaft continuous rotational motion is transformed in reciprocating translational motion of the cylindrical slider and in reciprocating rotational motion of the mirror slider, obtaining the mirror reciprocating yaw motion (see Fig. 5.16). Referring to Fig. 5.17 it is possible to calculate the reduction ratio of the mechanism τ_α. This is the relationship between

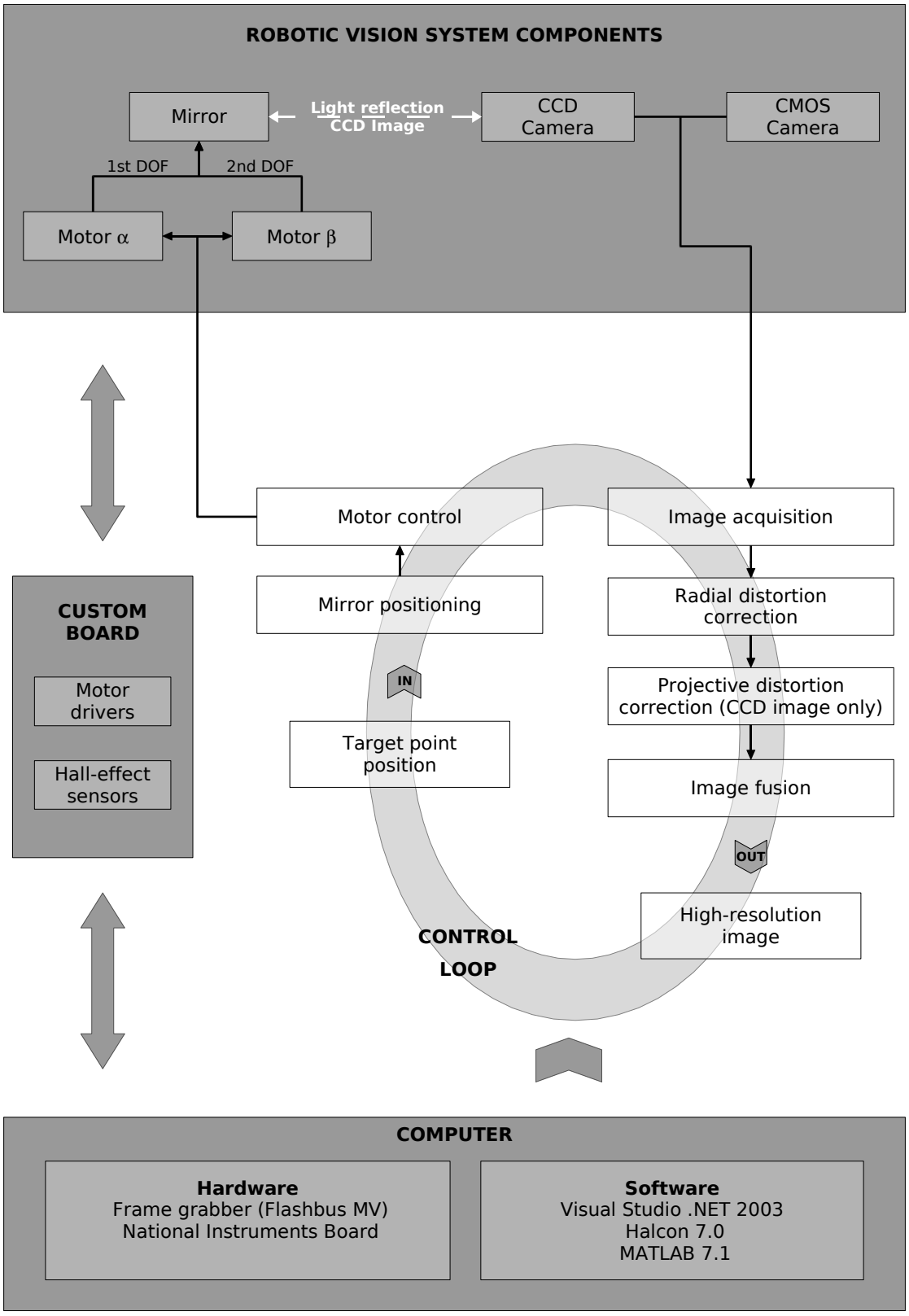


Figure 5.14. The hardware components and software processing stages of the vision system.

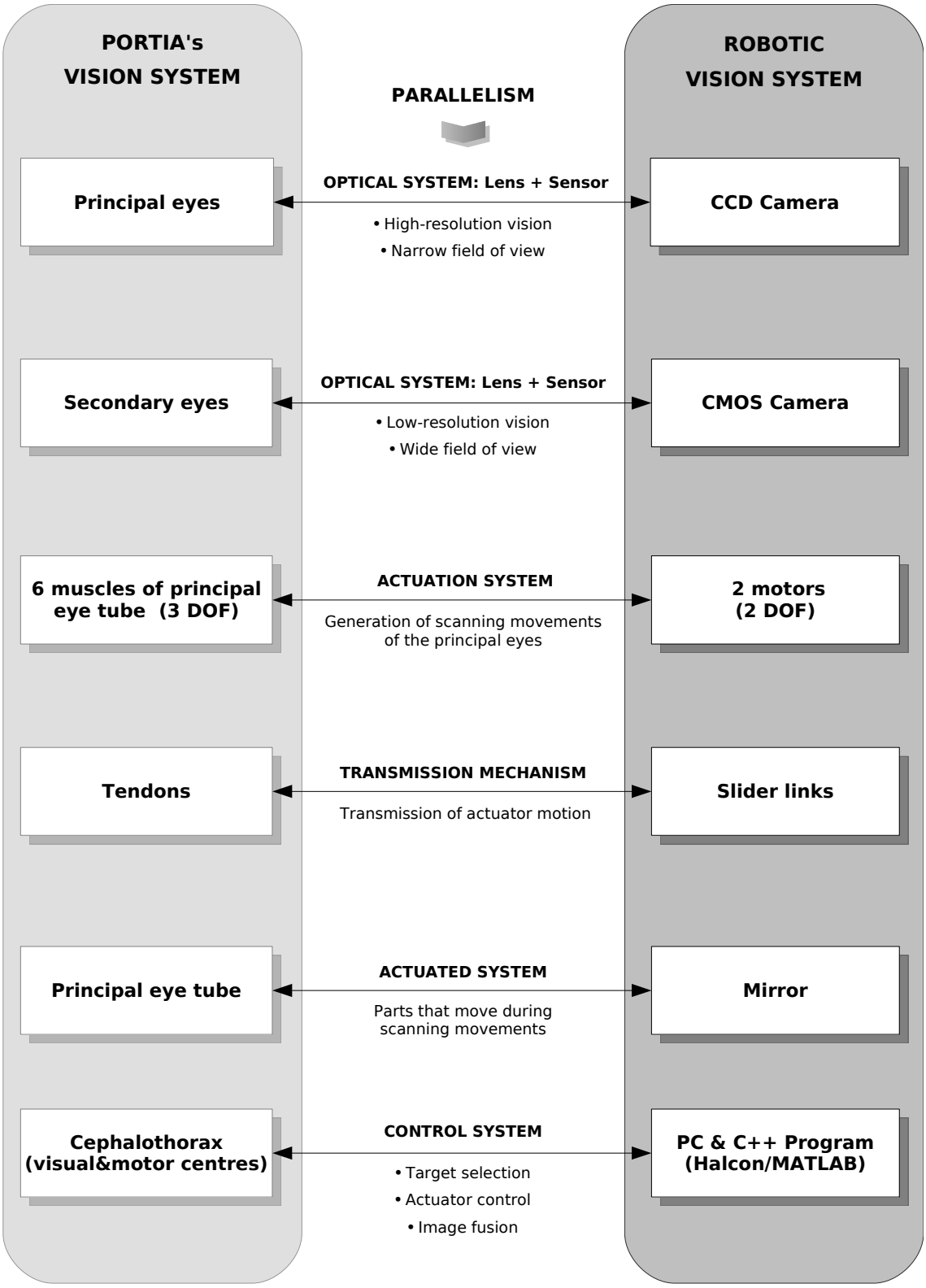


Figure 5.15. Parallelism between the developed vision system and salticids eyes.

the motor angle γ and the mirror angle α

$$d_\alpha = l_\alpha \cdot \sin\alpha \tag{5.1}$$

$$d_\alpha = r_\alpha \cdot \sin\gamma \tag{5.2}$$

From (5.1) and 5.2 we get:

$$\gamma = \arcsin(\frac{l_\alpha \cdot \sin\alpha}{r_\alpha}) \tag{5.3}$$

$$\alpha = \arcsin(\frac{r_\alpha \cdot \sin\gamma}{l_\alpha}) \tag{5.4}$$

The reduction ratio is:

$$\tau_\alpha = \frac{\gamma}{\alpha} = \frac{\arcsin(\frac{l_\alpha \cdot \sin\alpha}{r_\alpha})}{\alpha} = \frac{\gamma}{\arcsin(\frac{r_\alpha \cdot \sin\gamma}{l_\alpha})} \tag{5.5}$$

The reduction ratio has a nonlinear relationship with motor angle γ and α (see equation 5.5).

Tilt movement of the FOV is obtained by rotating the mirror around a horizontal axis. The β-pulley is integral with the β-motor shaft and has an eccentric pin mounted on it. This pin is inserted in the slider that is linked to the mirror housing by the mirror housing pin (see Fig. 5.18). The horizontal component of the force transmits a horizontal reciprocating translational motion to the pin while the vertical component of the force transmits the reciprocating tilt motion to the slider, obtaining the reciprocating mirror tilt motion. Referring to Fig. 5.17 the relationship between the shaft motor angle θ and the tilt angle called τ_β can be calculated.

$$d_\beta = l_\beta \cdot \sin\beta \tag{5.6}$$

$$d_\beta = r_\beta \cdot \sin\theta \tag{5.7}$$

From (5.6) and 5.7 we get:

$$\theta = \arcsin(\frac{l_\beta \cdot \sin\beta}{r_\beta}) \tag{5.8}$$

$$\beta = \arcsin(\frac{r_\beta \cdot \sin\theta}{l_\beta}) \tag{5.9}$$

The reduction ratio is:

$$\tau_\beta = \frac{\theta}{\beta} = \frac{\arcsin(\frac{l_\beta \cdot \sin\beta}{r_\beta})}{\beta} = \frac{\theta}{\arcsin(\frac{r_\beta \cdot \sin\theta}{l_\beta})} \tag{5.10}$$

From (5.10) it is possible to note that the reduction ratio has a nonlinear relationship with the motor angles θ and β.

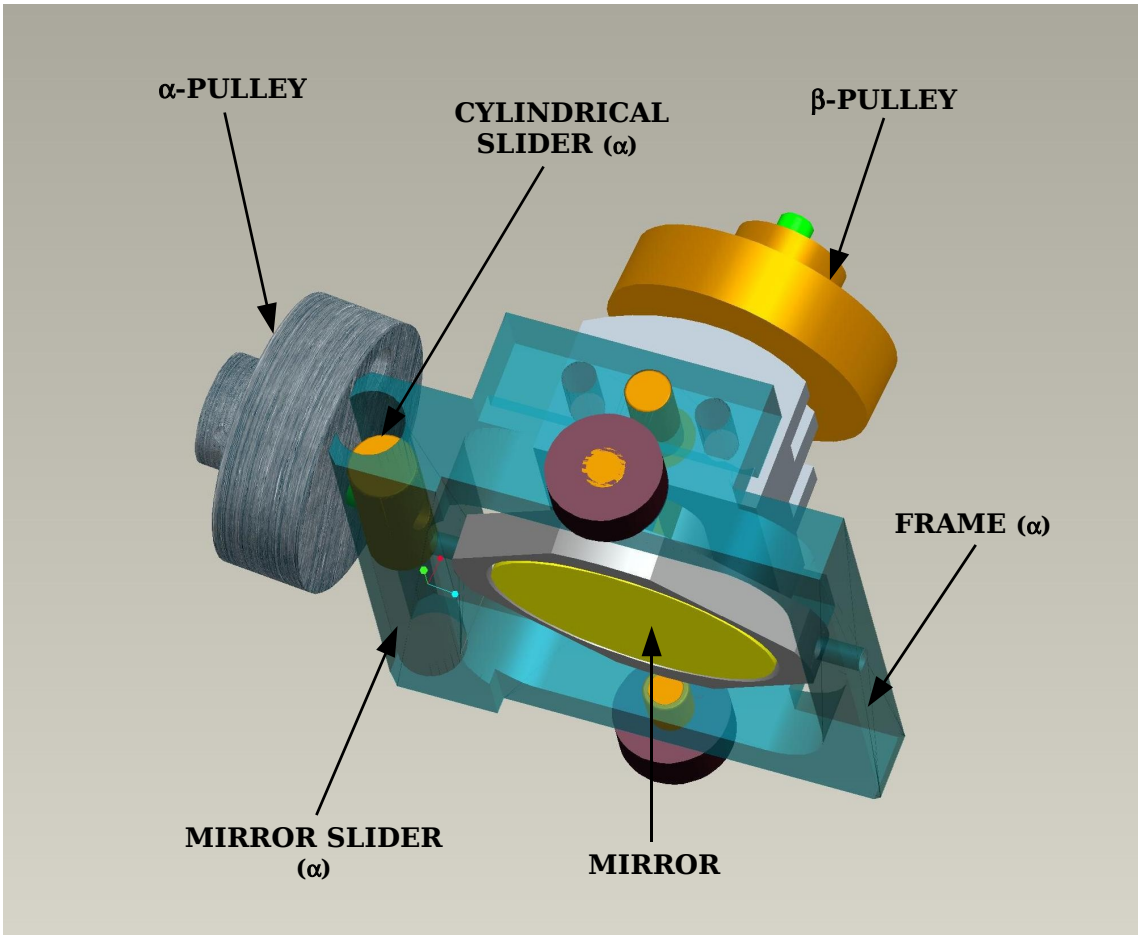


Figure 5.16. Yaw rotation mechanism (side view) corresponding to DOF α.

5.6 Direct and inverse kinematics

High resolution visualization of a target requires an adequate positioning of the mirror that reflects images toward the CCD camera. The mirror has 2 DOF and its spatial configuration is identified by the two angles α and β. The generic ray of light $\boldsymbol{r}$ coming from a point inside the field of view is reflected on a CCD pixel (image plane). For correct mirror positioning it is necessary to calculate the function of α and β that correlates the center of the sensor with a point in the field of view. Inverting this function the expression of the mirror configuration for the target desired visualization can be obtained. If the spatial configuration of the mirror is known, the point of incidence $\boldsymbol{m}$ of the ray $\boldsymbol{r}$ on the mirror surface can be calculated and the reflected ray $\boldsymbol{r}'$ is obtained by reversing the line of the incident ray with respect to the mirror normal. The target point $\boldsymbol{t}$ in the field of view is identified by intersecting the line of the incident ray with a plane π_{FOV} positioned at distance d from the CMOS camera and orthogonal to the camera axis (see Fig. 5.20).

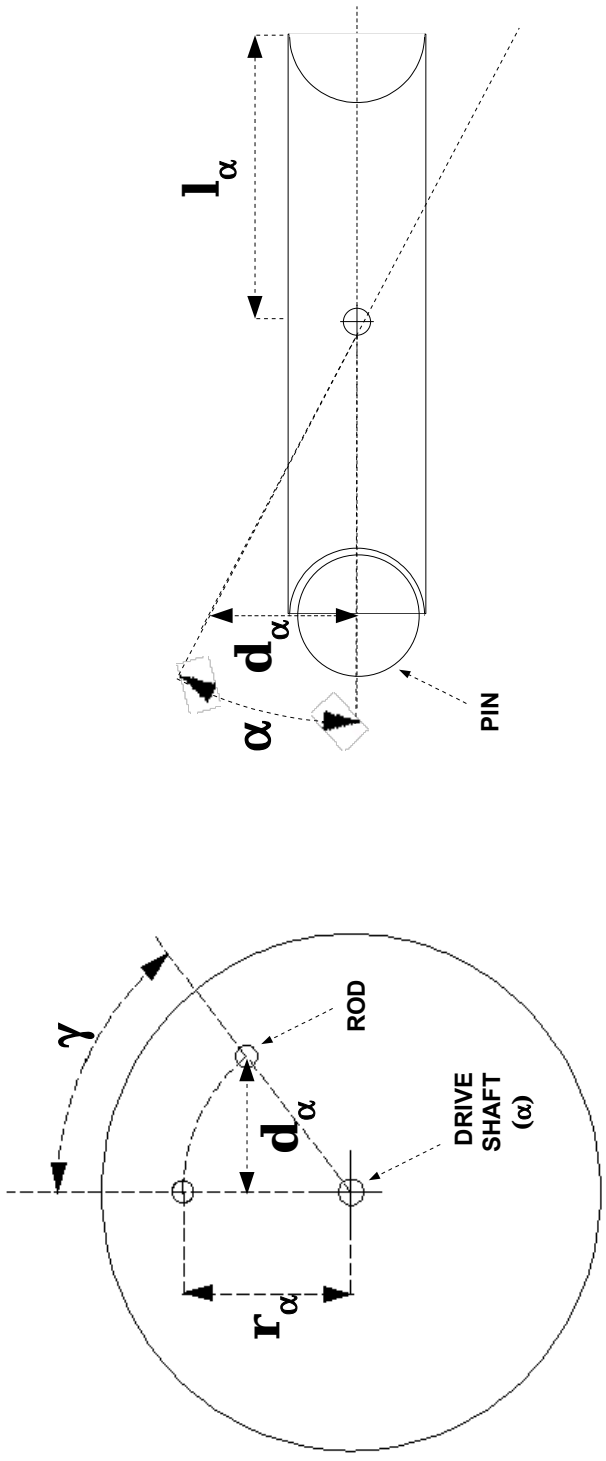


Figure 5.17. Yaw rotation scheme. Left: α-pulley (front view); Right: frame (top view).

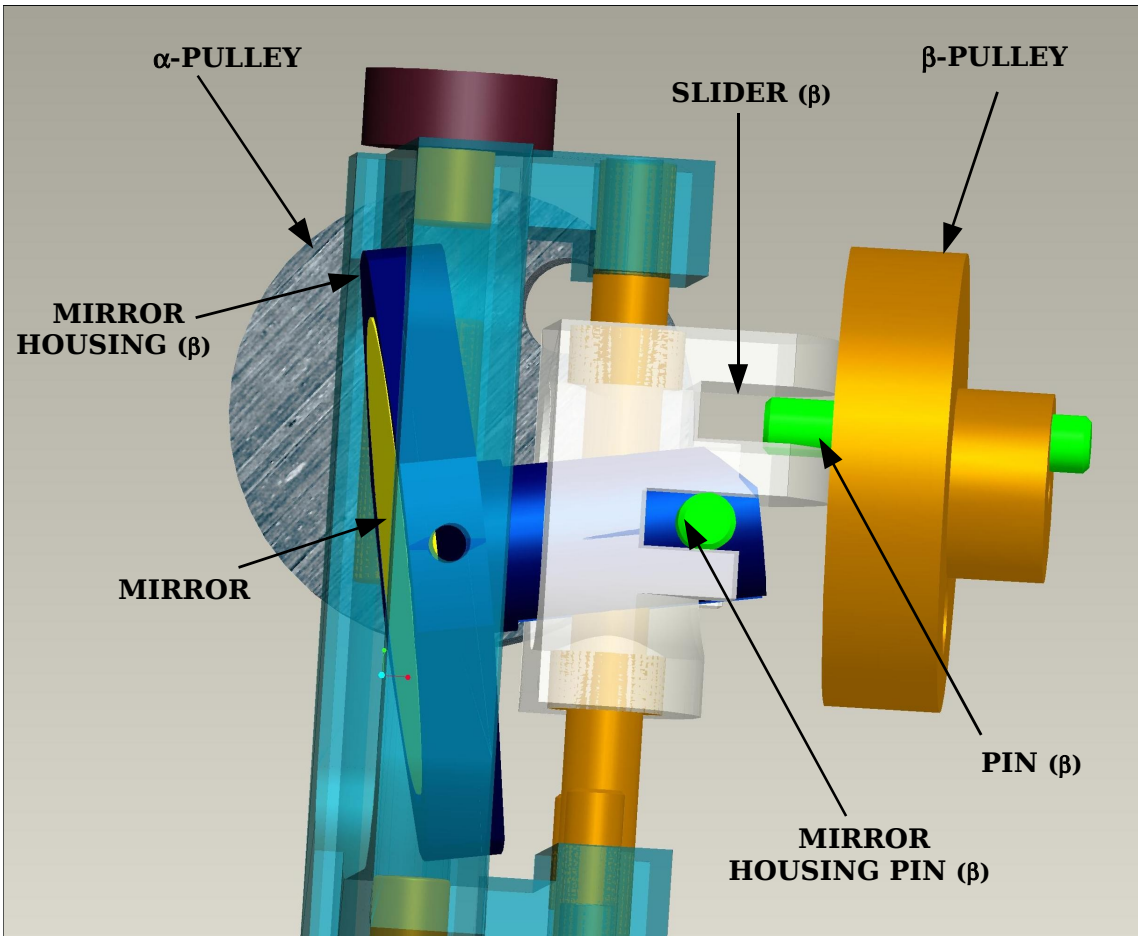


Figure 5.18. Mirror tilt mechanism (side view) corresponding to DOF β.

To calculate the geometric transformations the frames of reference shown in Fig. 5.20 are used. These frames are:

- G: fixed frame of reference with origin positioned in the center $\boldsymbol{o}_G$ of the CCD camera lens objective and oriented such as the xz plane is parallel to the CCD sensor plane;
- I: fixed frame of reference with origin $\boldsymbol{o}_I$ at the intersection of the CMOS camera axis with the plane π_{FOV};
- M: mobile frame of reference integral with the mirror. The x_M axis is orthogonal to the mirror plane and directed towards the field of view. The z_M axis coincides with the vertical axis of rotation;

The generic incident ray $\boldsymbol{r}$ is a straight line passing through a point $\boldsymbol{p}$ on the image plane of the CCD sensor and through the center of the objective $\boldsymbol{o}_G$:

$$\boldsymbol{r}^G = \lambda \boldsymbol{p}^G \qquad \forall \lambda \in \mathbb{R} \tag{5.11}$$

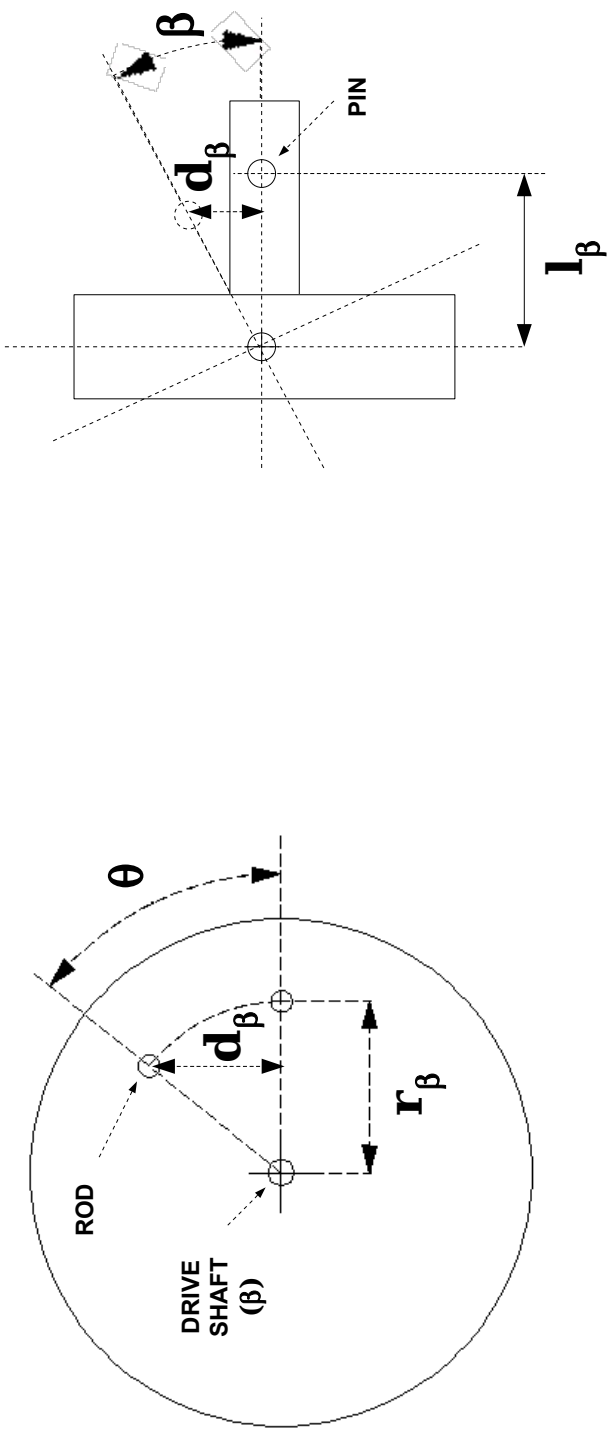


Figure 5.19. Mirror tilt scheme. Left: β-pulley (front view); Right: mirror housing (side view).

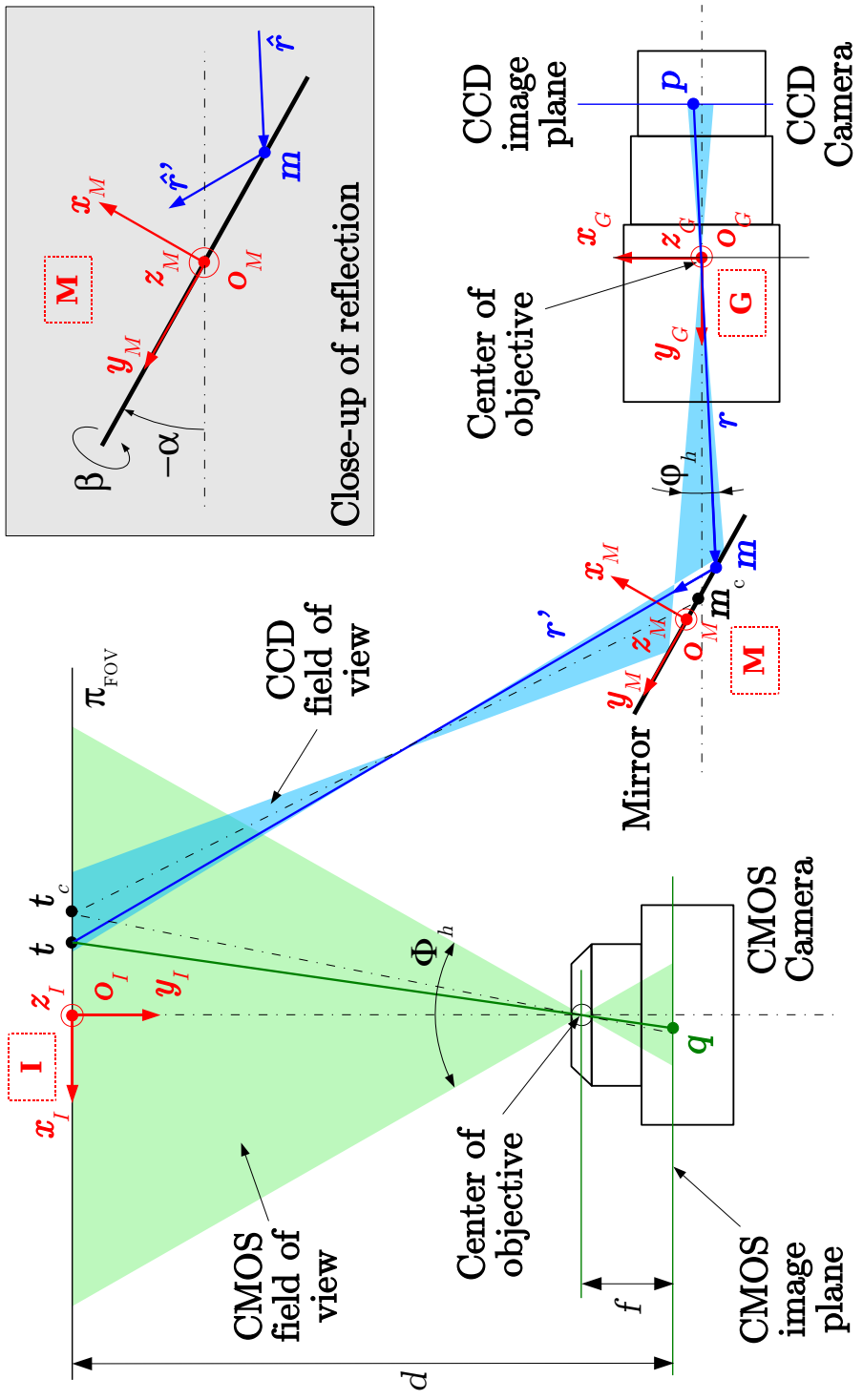


Figure 5.20. Top view of the vision system showing the field of view of the cameras and the optical path of the rays reaching the CCD sensor. Reference frames and vectors are explained in the text. The box in the upper right shows a close-up view of the mirror area.

where the superscript G means that $\boldsymbol{r}^G$ and $\boldsymbol{p}^G$ are written with respect to reference frame G. The mirror frame M is rotated with respect to the global frame G by an angle α around axis $\boldsymbol{z}_G$ and β around axis $\boldsymbol{y}_G$, so the mirror plane normal $\boldsymbol{x}_M$ is calculated as:

$$\boldsymbol{x}_M^G = \boldsymbol{R_z}(\alpha) \cdot \boldsymbol{R_y}(\beta) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\alpha)\cos(\beta) \\ \sin(\alpha)\cos(\beta) \\ -\sin(\beta) \end{bmatrix} \tag{5.12}$$

where $\boldsymbol{R_z}$ and $\boldsymbol{R_y}$ are 3×3 rotation matrices.

To find the point $\boldsymbol{m}$ of intersection of the ray with the mirror plane the following equation is solved with respect to $\overline{\lambda}$:

$$(\boldsymbol{x}_M^G)^{\mathrm{T}} \cdot (\overline{\lambda}\boldsymbol{p}^G - \boldsymbol{o}_M^G) = 0 \tag{5.13}$$

hence:

$$\boldsymbol{m}^G = \overline{\lambda}\boldsymbol{p}^G \tag{5.14}$$

The direction of the reflected ray is calculated using a local frame of reference A with $\boldsymbol{x}_A$ directed as the mirror normal and $\boldsymbol{z}_A$ orthogonal to both the incident ray and the mirror normal. The direction of the incident ray is expressed in this frame as:

$$\hat{\boldsymbol{r}}^A = \boldsymbol{R}_G^A \cdot \frac{\boldsymbol{p}^G}{|\boldsymbol{p}^G|} \tag{5.15}$$

To obtain the direction of the reflected ray $\hat{\boldsymbol{r}}'^A$ is sufficient to change sign to the y component of versor $\hat{\boldsymbol{r}}^A$. The line representing the reflected ray is written as:

$$\boldsymbol{r}'^G = \boldsymbol{m}^G + k \cdot \boldsymbol{R}_A^G \cdot \hat{\boldsymbol{r}}'^A \qquad \forall k \in \mathbb{R} \tag{5.16}$$

The position of target point $\boldsymbol{t}$ on the plane π_{FOV} is calculated by intersecting $\boldsymbol{r}'$ with the plane and solving with respect to k. The target point $\boldsymbol{t}$ corresponding to the generic point $\boldsymbol{p}$ on the CCD sensor is then written as:

$$\boldsymbol{t}^G = \boldsymbol{m}^G + \overline{k}(\alpha,\beta) \cdot \boldsymbol{R}_A^G \cdot \hat{\boldsymbol{r}}'^A \tag{5.17}$$

with these relations it is possible to solve the inverse kinematics of the system, so that angles α and β of the mirror can be calculated for a desired position $\boldsymbol{t}_{\mathrm{c}}$ of the center of the field of view on plane π_{FOV}. These is achieved by solving the following:

$$\boldsymbol{R}_A^G \cdot \boldsymbol{t}_{\mathrm{c}} = \boldsymbol{R}_A^G \cdot \boldsymbol{m}_{\mathrm{c}} + \overline{k}(\alpha,\beta) \cdot \hat{\boldsymbol{r}}'^A \tag{5.18}$$

where $\boldsymbol{m}_{\mathrm{c}}$ is the intersection of the ray passing through the center of the sensor and parallel to axis y_G with the mirror plane. Vectorial equation (5.18) is a set of three equations from which α and β can be evaluated as functions of $\boldsymbol{t}_{\mathrm{c}}$.

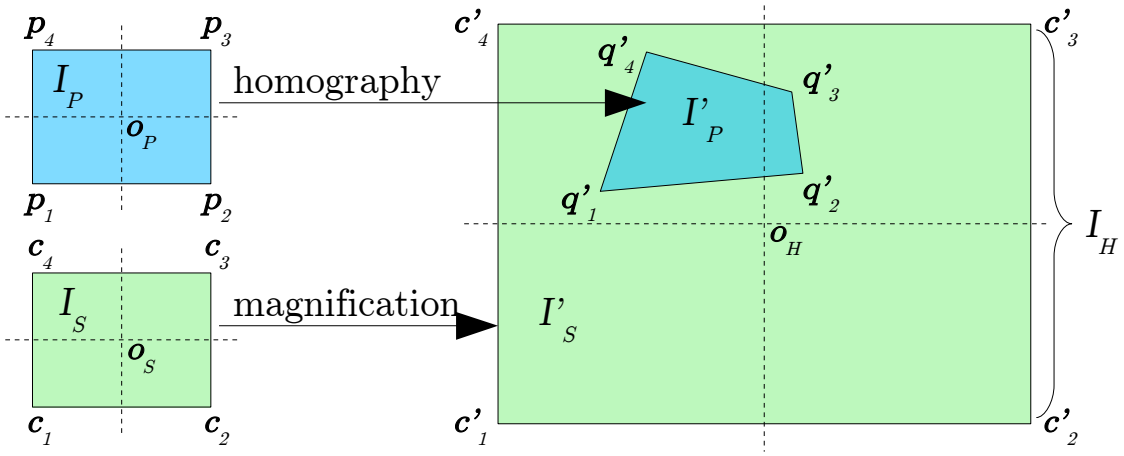


Figure 5.21. Graphical representation of the formation of the high-resolution image I_H by patching I'_P, a homographically corrected image of the primary eye I_P, over I'_S, the image of the secondary eye I_S magnified by a factor ζ.

5.7 Image processing

Real cameras are not geometrically exact, since lenses introduce minor irregularities into images, typically radial distortions. Several camera calibration techniques are available for compensation of these distortions [49, 61]. In this system, the *camera_calibration* function of the Halcon library has been used to compute the calibration parameters. This needs to be done only once at system setup. During the experiments, the frame grabber then applies the corresponding transformation to each captured frame. A telecentric camera model has been used with radial distortions for the CCD camera (non-zero focal length) and a pinhole camera model with radial distortions (zero focal length) for the CMOS camera.

As explained in Section 5.4, the wide-FOV, high-resolution image I_H is obtained by patching a magnified, low-resolution, secondary-eye image I'_S with high-resolution images I_P captured with the primary eye. The magnification factor is $\zeta = \Phi/\varphi$, the ratio of the FOVs. This requires to calculate the exact intersection of the FOVs of the primary and secondary eye, in order to apply the patch at the correct location. To make reasoning easier, the same plane π_{FOV} as in Fig. 5.20 will be referred. Both sensors collect rectangular images. However, when directed toward a generic point $\boldsymbol{t}_c$ the optical axis of the primary eye is not perpendicular to π_{FOV}, hence the intersection of the FOV of the primary eye with π_{FOV} is not a rectangle. It is therefore necessary to compute the projective transformation of the FOV: by applying this transformation to the primary eye image, the corrected image I'_P is obtained and can be used as a patch in I_H (see Fig. 5.21). This is obtained by computing the four points $\boldsymbol{t}_i$ ($i \in \{1,2,3,4\}$), on the plane π_{FOV} corresponding to the corners of I_P. By knowing the orientation of the mirror, and the horizontal and vertical FOV of the CCD camera, φ_h and φ_v, it is possible to calculate these points by writing the equations of the four straight lines that pass through $\boldsymbol{o}_G$ and, with respect to $\boldsymbol{y}_G$

make an angle of $\delta_h = \pm\varphi_h/2$ with $\boldsymbol{x}_G$ and $\delta_v = \pm\varphi_v/2$ with axis $\boldsymbol{z}_G$. In formulas:

$$\begin{aligned}\boldsymbol{r}_i = \boldsymbol{o}_G + \lambda\,[&\sin(\delta_v)\,\boldsymbol{x}_G + \cos(\delta_v)\,\cos(\delta_h)\,\boldsymbol{y}_G + \\ &+\,\cos(\delta_v)\,\sin(\delta_h)\,\boldsymbol{z}_G]\,,\ \forall\lambda\in\mathbb{R}\end{aligned} \tag{5.19}$$

By using (5.15) and (5.16), $\boldsymbol{r}'_i$ is computed, i.e. the rays reflected by the mirror; substituting them in (5.17) yields the intersection points $\boldsymbol{t}_i$ of the reflected rays with the plane π_{FOV}. The following step is to compute the pixel coordinates $\boldsymbol{q}_i$ in I_S corresponding to the four points $\boldsymbol{p}_i$, which is done by:

$$\begin{bmatrix} q'_{ix} \\ q'_{iy} \end{bmatrix} = \frac{\zeta}{2(d-f)} \begin{bmatrix} D_h(t_{ix} - o_{Hx})\tan\frac{\Phi_h}{2} \\ D_v(t_{iy} - o_{Hy})\tan\frac{\Phi_v}{2} \end{bmatrix} \tag{5.20}$$

where D_h and D_v are the horizontal and vertical size of the CMOS image in pixel, $\boldsymbol{o}$ are the pixel coordinates of the image center and f is the focal length of the CMOS camera. Note that the transformed CCD image I'_P is patched on the high-resolution image I_H, therefore, in (5.21) the relation $\boldsymbol{q}'_i = \zeta\boldsymbol{q}_i$ is introduced.

By using homogeneous coordinates for the points ($\tilde{\boldsymbol{q}}_i^{\mathrm{T}} = [\boldsymbol{q}_i^{\mathrm{T}}\ 1]^{\mathrm{T}}$), the 2D projective transformation (also called homography) $\tilde{\boldsymbol{H}}$ that ensures point correspondence between the coordinates of the four corners $\boldsymbol{p}_i$ of I_P (in pixels), and the destination points $\boldsymbol{q}'_i$ can be computed from the equations:

$$\tilde{\boldsymbol{Q}}' = \tilde{\boldsymbol{H}} \cdot \tilde{\boldsymbol{P}} \tag{5.21}$$

where $\tilde{\boldsymbol{Q}}' = [\tilde{\boldsymbol{q}}'_1\ \tilde{\boldsymbol{q}}'_2\ \tilde{\boldsymbol{q}}'_3\ \tilde{\boldsymbol{q}}'_4]$ and

$$\tilde{\boldsymbol{P}} = \begin{bmatrix} -\frac{S_h}{2} & \frac{S_h}{2} & \frac{S_h}{2} & -\frac{S_h}{2} \\ -\frac{S_v}{2} & -\frac{S_v}{2} & \frac{S_v}{2} & \frac{S_v}{2} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{5.22}$$

with S_h and S_v, i.e. the horizontal and vertical size of the CCD image in pixel. A simple way to solve these equations is to use the pseudoinverse matrix:

$$\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{Q}}' \cdot \tilde{\boldsymbol{P}}^{\mathrm{T}} \cdot (\tilde{\boldsymbol{P}} \cdot \tilde{\boldsymbol{P}}^{\mathrm{T}})^{-1} \tag{5.23}$$

Numerically more robust solutions can be computed e.g. by means of constrained least-squares solution [25]. The homography transformation that maps points $\boldsymbol{p}_i$ to $\boldsymbol{q}'_i$ is computed by means of the Halcon function *vector_to_hom_mat2d*. By applying it to I_P, by means of the *projective_trans_image* function, I'_P is obtained. I'_P can now be applied to $I_{S'}$.

By first covering the whole low-resolution image I'_S with I'_P patches, a full high-resolution image I_H is obtained. Subsequently, the secondary eye can be used as motion detector and only update those parts of I_H where changes have been detected, by directing the primary eye toward those areas, computing I'_P and then patching it over the current I_H.

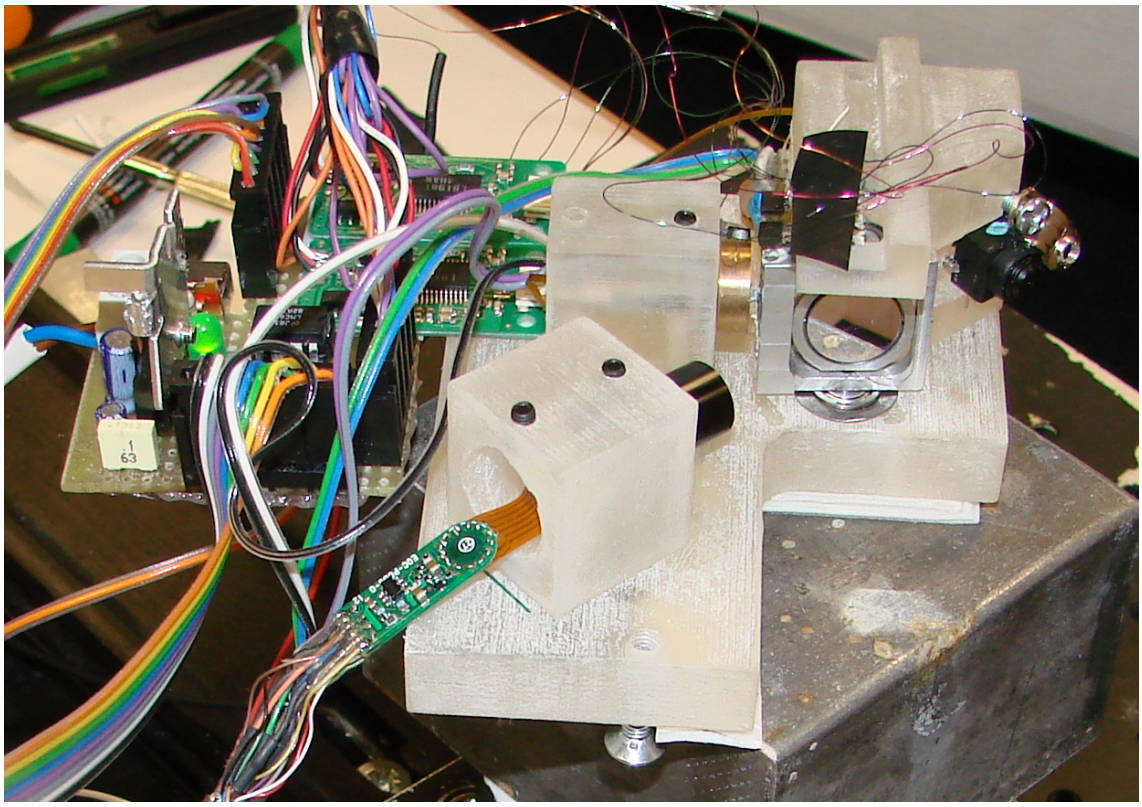


Figure 5.22. Picture of the assembled vision system. Part of the frame has been fabricated in acrylic polymer, while the precision components are in aluminum.

5.8 Experiments

Both the CMOS and CCD sensors used have 640 × 480 resolution. Therefore $D_h = S_h = 640$ and $D_v = S_v = 320$. After assembling the system (see Fig. 5.22), the camera calibration procedure of the Halcon library has been performed to automatically compensate for radial distortions during image acquisition. The computation of the calibration parameters requires manual positioning of a grid at various orientations in front of the camera. The procedure is repeated for both cameras. Then, distortion correction is performed automatically by the frame grabber board at every image acquisition. The calibration curves for the tilt and yaw angles of the mirror have been measured, i.e. the relation between the number of pulses generated by the motor drivers and the resulting α and β angles has been measured. As explained in Section 5.4, each pulse theoretically corresponds to a rotation angle of $360°/337 \simeq 1.07°$ of the motor shafts. Unfortunately, this was not true in practice, therefore the correspondence between pulses and angles has been manually measured. An origin-constrained least squares linear fit on the collected data results in a coefficient 1.27°/pulse with 95% confidence bounds of ±0.012°/pulse (±0.9%).

The system has been tested by manually setting combinations of α and β to frame some chosen points on a sheet of graph paper, in order to estimate the accuracy of

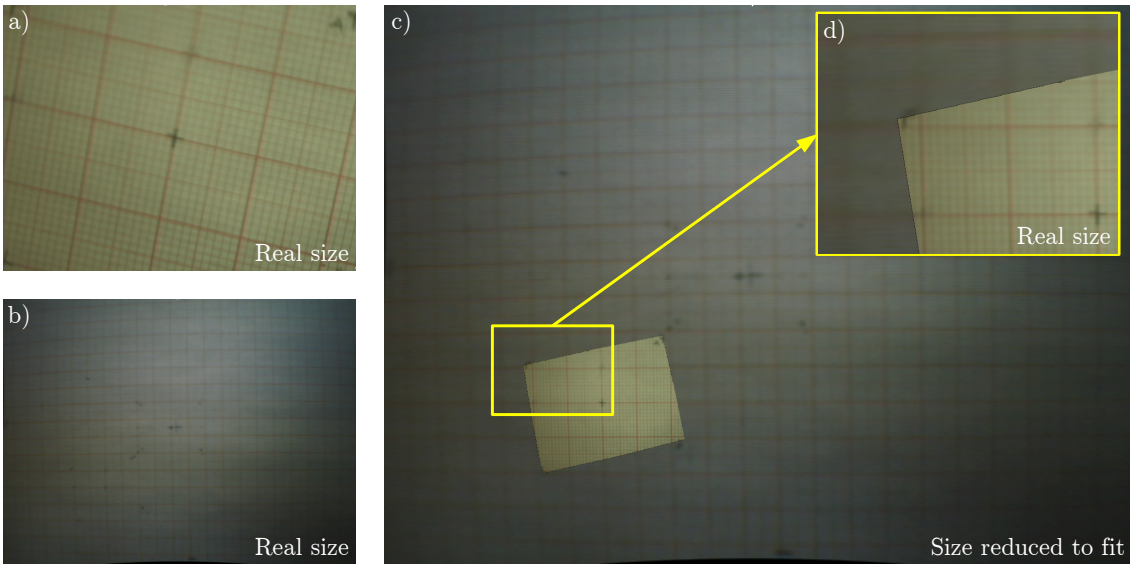


Figure 5.23. Example of the image fusion process. a) primary eye image I_P; b) secondary eye image I_S; c) combined high-resolution image I_H, with I'_P superimposed on I'_S. The image size has been reduced to fit in the figure; d) real size of the highlighted box in I_H. The whole image is at the resolution of the primary eye.

the image fusion algorithm. Figure 5.23 shows a typical result. The overlap of the patch I'_P on I'_S is not always perfect and I_H is not a continuous image. The accuracy and precision of the system have been calculated: a set of couples (α,β) has been selected and repeatedly positioned the system at those values. For each positioning, the real attained position of the image center on the sheet of graph paper has been measured. Accuracy has been computed as the average deviation between the real and desired position, obtaining 0.77°(2.6 mm on π_{FOV}) and precision as the standard deviation of repeated measures for the same position, obtaining a mean precision of 1.6°(5.5 mm on π_{FOV}).

The positioning errors are caused by several sources of errors in the computation. The principal are listed below, together with a possible solution or work-around:

- errors in camera calibration that cannot completely compensate for radial distortions, especially at the image corners; workaround: use cameras with larger FOVs and crop the external parts of I_P and I_S;

- errors in motor positioning caused by inappropriate position sensing: pulses generated by the counter-electromotive force are not repeatable and not accurate for precise positioning purposes; solution: use of external sensors like hall effect sensors to measure α and β;

- misalignment in the mechanical assembly of the system, e.g. camera axes not perfectly aligned with the mirror rotation axes; solution: mount the cameras

on precision sliders to allow manual alignment; workaround: correct misalignments by means of image processing algorithms;

- errors due to the non-real-time calculation in the motor positioning algorithm. The operating system used for this prototype has not real-time features; solution: use of different and separated motion controller, for motor position calculations, leaving the central unit to image processing.

To identify the major contribution to the total error, the precision of the image fusion method (i.e. image calibration and projective transformation) has been calculated, by manually aligning the centers of the images I'_P on I'_S and by measuring the mean deviation of the corner points from the real position. The resulting error was only $0.36°$(1.2 mm on π_{FOV}).

The largest source of error is the incorrect positioning of the mirror, due to the absence of encoders on the motors. The used positioning method, based solely on pulse count, is not sufficient for precise positioning.

Errors can never be completely removed. Even in the best case, the position of the patch, as computed in (5.18) and (5.23) will probably not be optimal. An additional method to further reduce the misalignment is to use automatic image registration techniques, such as maximising mutual information by means of a gradient descent [51, 9] using the computed patch position as starting point.

5.9 Motor sensorization

As already mentioned in section 5.8 one of the error sources is the incorrect motor position sensing. This is a difficult task because the system components have been selected for future miniaturization: Namiki motors do not have encoders for incompatible sizes. Moreover the position sensing furnished by the Namiki drivers is not accurate because it is based on the measure of the counter-electromotive force (CEMF) and the zero crossing detection. This method is not suitable for very accurate positioning tasks such as the mirror orientation. CEMF is the voltage induced into an inductor due to an alternating or pulsating current. For electric motors the generated electromotive force is called back-electromotive force (BEMF). When a brushless DC motor rotates, each winding generates a voltage related to BEMF, which opposes the main voltage supplied to the windings according to Lenz's Law. The polarity of this BEMF is in opposite direction of the main voltage. BEMF depends mainly on three factors:

- Angular velocity of the rotor;
- Magnetic field generated by rotor magnets;
- The number of turns in the stator windings.

Once the motor is designed, the rotor magnetic field and the number of turns in the stator windings remain constant. The only factor that governs BEMF is the angular velocity of the rotor and as the speed increases, BEMF also increases. The motor technical specification gives a parameter called, BEMF constant that can be used to estimate BEMF for a given speed. The potential difference across a winding can be calculated by subtracting the BEMF value from the supply voltage. The motors are designed with a BEMF constant in such a way that when the motor is running at the rated speed, the potential difference between the BEMF and the supply voltage will be sufficient for the motor to draw the rated current and deliver the rated torque. If the motor is driven beyond the rated speed, BEMF may increase substantially, thus decreasing the potential difference across the winding, reducing the current drawn which results in a drooping torque curve. The last point on the speed curve would be when the supply voltage is equal to the sum of the BEMF and the losses in the motor, where the current and torque are equal to zero. A comparison between waveforms generated by brushless DC motors provided with hall sensors could help in the problem understanding. In this case hall sensors are integrated by the manufacturer inside the motor case. The relationship between the Hall sensors and BEMF, with respect to the phase voltage, is depicted in Fig. 5.25. Every commutation sequence has one of the windings energized positive, the second negative and the third left open. As shown in Figure 5.25, the Hall sensor signal changes the state when the voltage polarity of back EMF crosses from a positive to negative or from negative to positive. In ideal cases, this happens on zero-crossing of back EMF, but practically, there will be a delay due to the winding characteristics. This delay should be compensated by the microcontroller that performs the control calculations. Another aspect to be considered is related to low speeds. Because BEMF is proportional to the speed of rotation, at a very low speed, the BEMF would be at low amplitudes to detect zero-crossing. The motor has to be started in open loop, from standstill and when sufficient BEMF is built to detect the zero-cross point, the control should be shifted to the BEMF sensing. The minimum speed at which BEMF can be sensed is calculated from the BEMF constant of the motor. With this method of commutation, the Hall sensors can be eliminated and in some motors, the magnets for Hall sensors also can be eliminated. This simplifies the motor construction and reduces the cost as well [58]. Namiki motors used for this project are not equipped with internal hall sensors and at low speed the microcontroller has to calculate the voltage representing the counter-electromotive force in open loop. This is not suitable for very precise positioning applications.

These considerations explain why the Namiki motor driver gives in output pulses even if the motor is stopped and the not-repeatable pulses generation. Additional sensors have been added to solve the problem of positioning, for α angle a two analog hall sensors unit has been used while for β angle an infrared proximity sensor has been used. A magnet is attached to the frame (α) and the two hall sensors give two output that are proportional to the magnetic field intensity. This two signals are acquired with the 6624 DAQ board, subtracted and the result is on-line filtered with a third order Butterworth low-pass filter with cut-off frequency of 6 Hz (see

Fig. 5.26). The calibration of the system needs to be performed only one time before the use in order to acquire values and to find the relationship between the mirror α angles and the sensor unit values (eq. 5.24). Figure 5.28 shows the polynomial curve that fits data (9th degree).

$$\begin{aligned} Pa(x) = &-4.261e-009*x^9 + 6.959e-009*x^8 + 4.644e-007*x^7 + \\ &- 3.203e-007*x^6 - 1.82e-005*x^5 + 5.696e-007*x^4 + \\ &+ 0.0001045*x^3 - 8.521e-005*x^2 - 0.01764*x + -0.03059 \end{aligned} \quad (5.24)$$

The infrared proximity sensor is used for the second DOF of the mirror (frame β) and the response is proportional to the distance of the frame β from the sensor. Data read from the sensor are on-line filtered with the same third order Butteworth low-pass filter with cut-off frequency of 6 Hz (see Fig. 5.27). Data are fitted with a polynomial curve (6th degree eq. 5.25) and the results are shown in Fig. 5.28.

$$\begin{aligned} Pb(x) = &-5.803e-007*x^6 - 8.175e-007*x^5 + 6.039e-005*x^4 + \\ &- 0.0001645*x^3 + 0.001756*x^2 + -0.04223*x + 1.927 \end{aligned} \quad (5.25)$$

Listing 5.1. α motor position control, Kp=1, Ki=10

```
do{
 qApp->processEvents();
 DAQmxReadAnalogScalarF64 (taskHandle4B, 10.0,
 &valueB, NULL);
 error =  Pa - valueB;
 errorTot = errorPrev + error;
 Va = (Kp * error) + (Ki * (errorTot));
 errorPrev = error;
 if(Va > 0.0){
  data[0]=1;
  if(Va > 4.0) Va = 4.0;
  if(Va<2.0) Va = 2.0;
 }
 else if(Va < 0.0) {
  data[0]=0;
  if(Va < -4.0) Va = -4.0;
  if(Va > -2.0) Va = -2.0;
 }
 DAQmxWriteDigitalLines(taskHandle2a,1,1,10.0,
 DAQmx_Val_GroupByChannel,data,NULL,NULL);
 DAQmxWriteAnalogScalarF64 (taskHandle3A, 1,
 10.0, abs(Va), NULL);
} while( (error > 0.01) || (error < -0.01));
```

Listing 5.2. β motor position control, Kp=1, Ki=10

```
do{
 qApp->processEvents();
 DAQmxReadAnalogScalarF64 (taskHandle4A, 10.0,
 &valueA, NULL);
 DAQmxReadAnalogScalarF64 (taskHandle4C, 10.0,
```

```
&valueC, NULL);
diff=valueA-valueC;
error =  diff-Pb;
errorTot = errorPrev + error;
Vb = (Kp * error) + (Ki * (errorTot));
errorPrev = error;
if(Vb > 0.0){
        data[1]=1;
 if(Vb > 4.0) Vb = 4.0;
 if(Vb < 2.0) Vb = 2.0;
}
else if(Vb < 0.0) {
 data[1]=0;
 if(Vb < -4.0) Vb = -4.0;
 if(Vb > -2.0) Vb = -2.0;
}
DAQmxWriteDigitalLines(taskHandle2a,1,1,10.0,
DAQmx_Val_GroupByChannel,data,NULL,NULL);
DAQmxWriteAnalogScalarF64 (taskHandle3B, 1,
10.0, abs(Vb), NULL);
} while( (error > 0.01) || (error < -0.01) );
```

The non-real-time platform used for the endoscope prototype constitutes another source of errors for motor positioning calculations. The central unit is able to process data acquired from motor sensors every 15ms that is a sufficient timing constraint for this prototype, in which instrument tracking feature has not been yet implemented. More stringent timing constraints have to be considered and a different and real-time operating system has to be implemented. Two different approaches are now investigated: one is related to the use of a real-time PC which is delegated to the motor control and the other is related to the use of a more embedded solution with a DSP.

The first solution concerns the use of a different PC with a real-time operating system like Linux RTAI or S.Ha.R.K., that support the 6025 DAQ board. The main advantage is the possibility to use the data acquisition board 6624 also with this configuration. The reading of sensors can be performed in real-time mode, the OS ensures the respect of the process time constraints. The algorithm for motor position control can be implemented in C programming language and particularly a Proportional Integral (PI) controller can be used. Two real-time tasks can be implemented for the motor position controllers. The position commands are sent from the control unit, that performs the image processing, through the network connection using the UDP protocol. The task aimed at the network data receive is a non-real-time process since the asynchronous nature of the communication. This approach require two separate PCs, one with the frame-grabber board and non-real-time operating system in which image processing is implemented and a second PC that receives the two desired positions relative to the mirror configuration and performs the motors control.

The second solution involves the use of a DSP that performs the calculations relative to the control of the motor position. The PC, that acquires the two cameras, send out the two desired positions through serial communication to the DSP in which can be implemented the PI control algorithm.

5.10 Conclusions

The aim of this project was to study, design and develop a robotic vision system for endoscopy that allows to overcome some of the limitations of the traditional endoscopes like the reduced image resolution. The fabricated prototype is composed by two video cameras that collect images and, subsequently, the central processing unit merge the images together. The first video camera has a narrow field of vision with high resolution, it focuses on surgical detail of interest which it is moved to. The second video camera has a wide field of vision and it is fixed. The entire operating field is visualized without moving the endoscope. The "local" vision field is moved inside the "global" field. It should be possible to track surgical instruments with the more resolute camera giving to the surgeon in every moment the fixed global view and the detailed view of the zone in which there are the instruments. The introduced vision system is considered *bioinspired* because its operation principle derives and imitates the visual system of the *Portia* fimbriata family spiders. Furthermore it is a *robotic* system since, contrary to classical optical systems it has an actuating mechanism that allows to orient the visual field in order to visualize the target of interest which can be, as example, the tip of the surgical instrument instrument.

The components have been selected with further miniaturization in mind. The encountered problems have been outlined and the proposed solutions can be implemented thanks to the wide flexibility of the global system. Many approaches can be investigated, from the more "prototyped" solution with the use of two PCs, one for the two cameras and the other, a real-time PC, for the motor control, to the more "embedded" solution with the use of a PC, for the image acquisition and processing, and a DSP for motor control. In both cases the necessary time is only relative to the practical implementation because all these systems are ready to operate with simple developing instruments. Both the real-time PC and the DSP are programmable using free C programming languages, furthermore the DSP is programmable through the serial port of the PC, without the proprietary programmer. Sensors can be added or changed simply adding more connections and modifying the software applications relative to sensors processing.

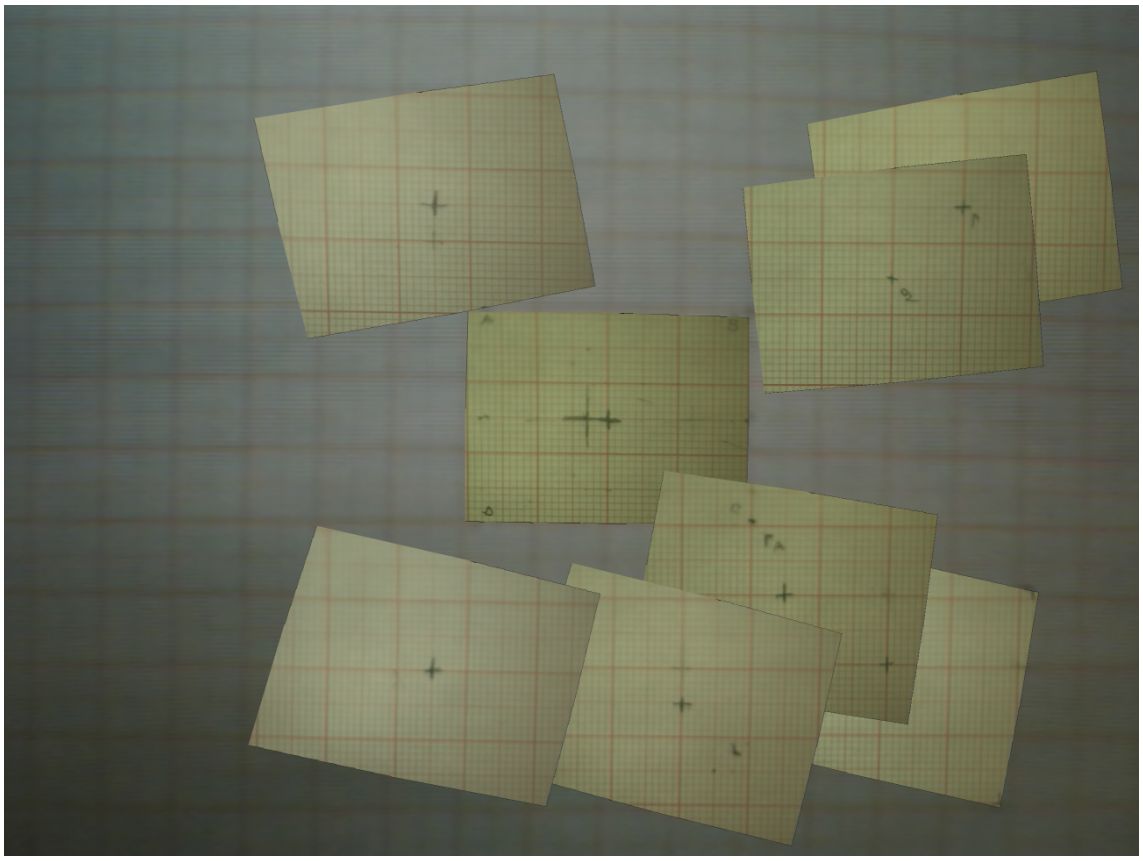


Figure 5.24. High-resolution image I_H obtained after applying several patches to the magnified low-resolution image.

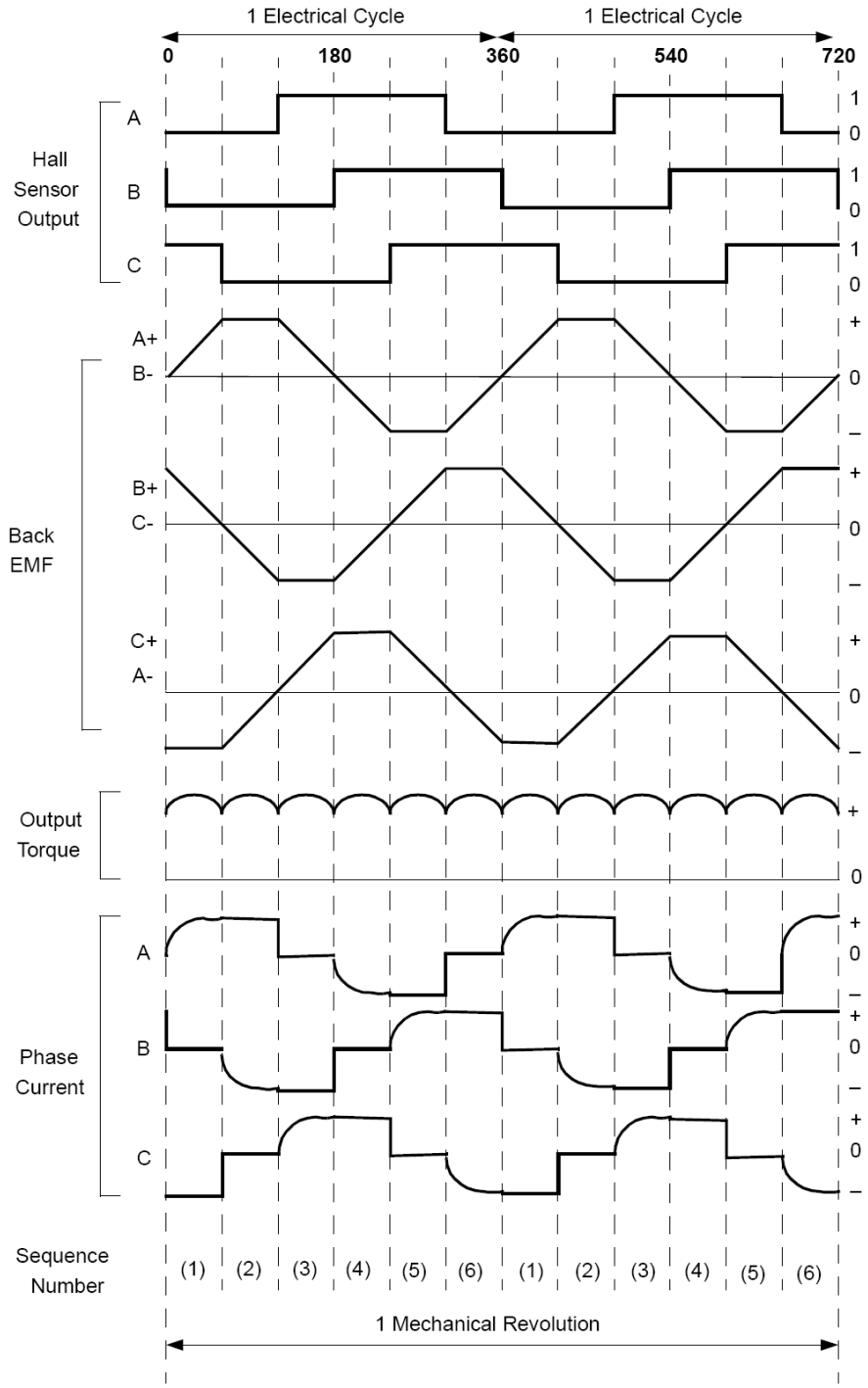


Figure 5.25. Hall sensor signal, back-electromotive force (BEMF), torque and phase current [58].

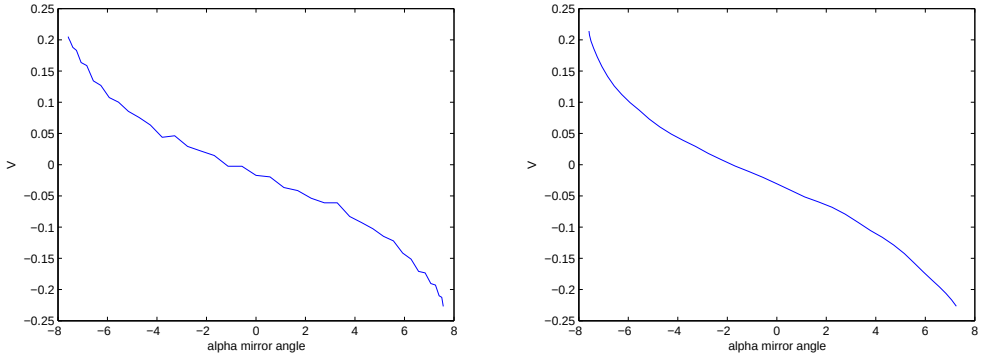


Figure 5.26. Acquired data for α angle (left) and filtered data for α angle (right).

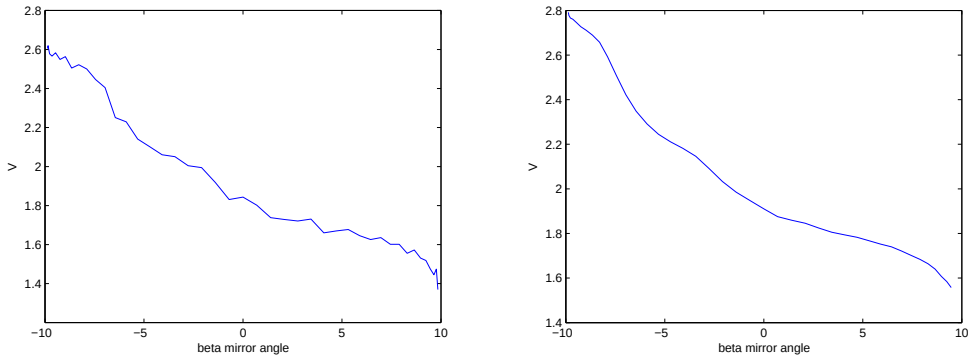


Figure 5.27. Acquired data for β angle (left) and filtered data for β angle (right).

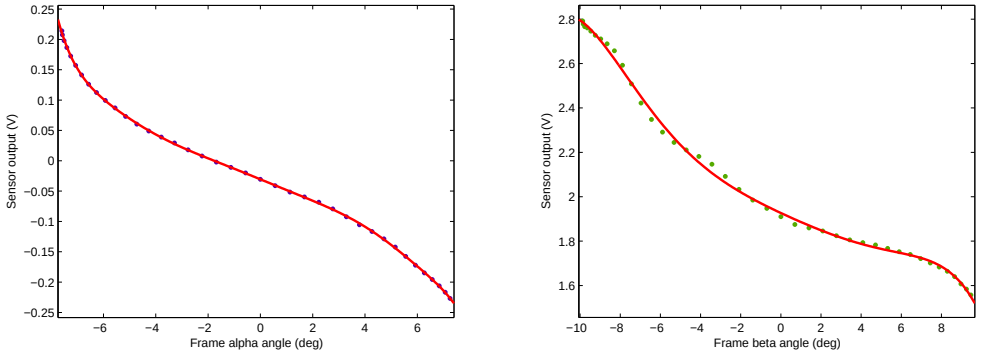


Figure 5.28. Fitting of the sensor real values for α angle (left) and for β angle (right).

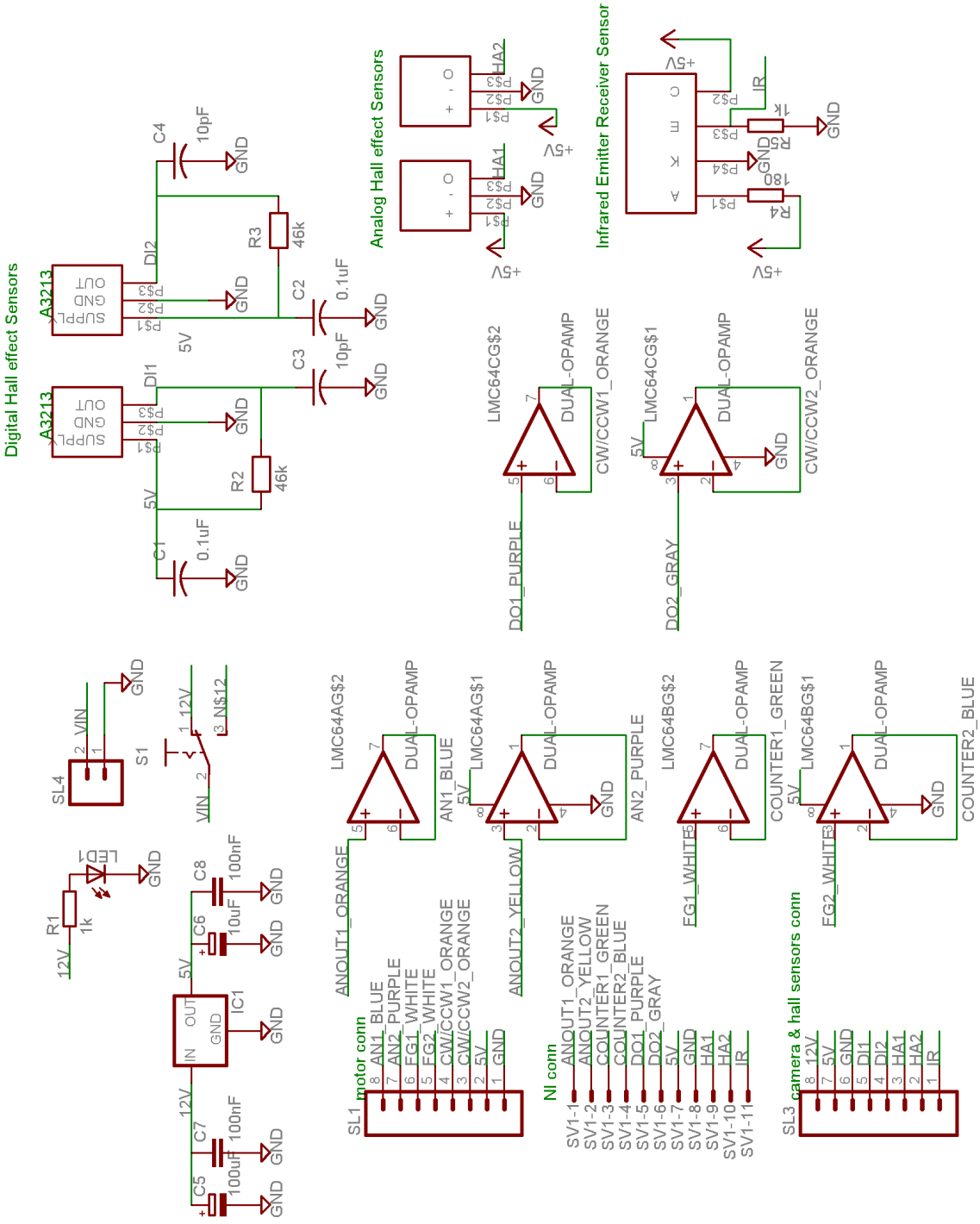


Figure 5.29. Electronic board for the endoscope.

Figure 5.30. Graphical user interface for controlling the mirror position.

Chapter 6

Respiro. a simple breathing monitor

6.1 Introduction

The evaluation of medical performance is very important because can furnish indication about the ergonomic of the instruments that he uses. Commercial devices are more related to baby breathing monitoring and consists of a sensorized pad placed just under the mattress. Little movements are revealed and breathing related movements are measured, an alarm can be started if the baby is not moving like for instance the Babysense II Infant Respiratory Baby. A similar solution is not suitable for a surgical application because the surgeon has to stand near the operating table and the monitor system has not to disturb him. The breathing frequency can be used to measure the degree of fatigue of a subject and if it is uses in combination with other physiological parameters a better evaluation can be carried out. The developed prototype has implemented only the breathing monitoring with the revelation of the expiration and inspiration phases.

6.2 System description

The aim is to monitor the breathing related movements through the measurement of the thorax size difference from the maximum inspiration phase and the maximum expiration phase. A linear slide, mounted on a solid support, is used in order to be compliant with the breathing motion and an array of three analog hall effect sensors is employed to measure the cursor displacement. A permanent magnet is mounted on the slider and it moves under the sensors (see fig. 6.1). An adjustable belt is used to fit different thorax sizes, one belt side is fixed to the support while the other side is fixed[!]to an elastic strip that is fixed also to one side of the cursor (see fig. 6.1). The elastic is used like a spring link for the expiration phase to call back the slider in rest position. The global system is compliant with thorax extension. Three analog sensors have been used to measure the movements of the magnet mounted on the

cursor. Every sensor gives in output a voltage proportional to the intensity of the magnetic field produced by the permanent magnet. The combination of the three sensor outputs produces a single value that can be related to the slide position. A calibration procedure has to be performed only one time when the system is assembled. The core is a microcontroller PIC16F876 by Microchip that samples the three analog signals with a sampling time of $T_{s=1ms}$ with 10bit precision. The sampled data are sent to the host unit connected using the RS-232 protocol. Data are collected in the host PC where a GUI (graphical user interface) on line shows the values of the hall sensors, the breathing frequency and the position of the slider. Data are also written in log files for off-line elaborations. Software for the host PC is written with C++ programming language using QT 3.2.1 non commercial version for the GUI. The functional scheme of the system is represented in fig. 6.2. Respiro is connected at the serial port of the host PC that send in output a particular string used for waking up Respiro. A connection LED is activated, the sensors sampling is started and acquired data are sent to the host PC.

This configuration needs a PC with a serial port that stores data and this could be done in a laboratory environment but not for self measuring. An additional module can be implemented for this purpose with the use of an extra memory like SD (Secure Digital) or MMC (Multi Media Card) because the microcontroller has not enough memory to store data for minutes or hours. MMC accepts data with the SPI protocol (Serial Peripheral Interface) that can be easily implemented with the used microcontroller. Respiro could be transformed in a portable system with autonomous power. In fact the system is powered with a single battery (9V transistor type) with brown-out detection for stopping the system with low battery. 6.1 6.2.

Listing 6.1. SPI communication

```
char SPI(char d) // send character over SPI
{
 SSPBUF = d; // send character
 while (!BF); // wait until sent
 return SSPBUF; // and return the received character
}

char Command(char befF,uns16 AdrH,uns16 AdrL,char befH )
{ // sends a command to the MMC
 char a;
 SPI(0xFF);
 SPI(befF);
 SPI(AdrH.high8);
 SPI(AdrH.low8);
 SPI(AdrL.high8);
 SPI(AdrL.low8);
 SPI(befH);
 SPI(0xFF);
 return SPI(0xFF); // return the last received character
}

bit MMC_Init() // init SPI
{
 SMP = 0; // input is valid in the middle of clock
 CKE = 0;        // rising edge is data capture
```

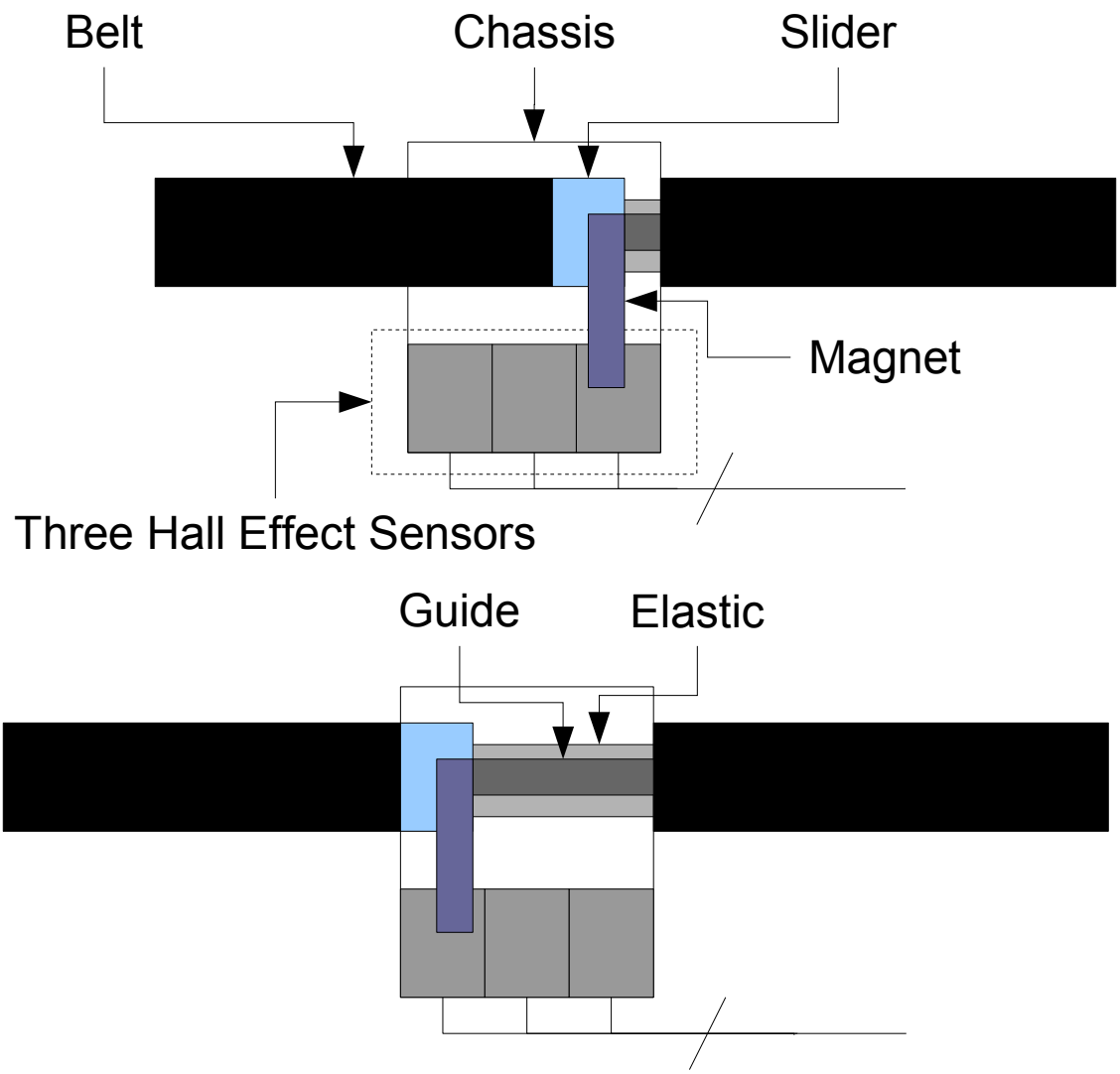


Figure 6.1. Respiro schematic view: maximum expiration phase (top) and maximum inspiration phase (bottom)

```
CKP = 1;          // high value is passive state
SSPM1 = 1; // speed f/64(312kHz), Master
SSPEN = 1; // enable SPI
CS = 1; // disable MMC

char i;
// start MMC in SPI mode
for(i=0; i < 10; i++)SPI(0xFF); // send 10*8=80 c
                                // lock pulses
CS=0; // MMC-Enabled

if (Command(0x40,0,0,0x95) !=1) goto mmcerror;
// Reset MMC
```

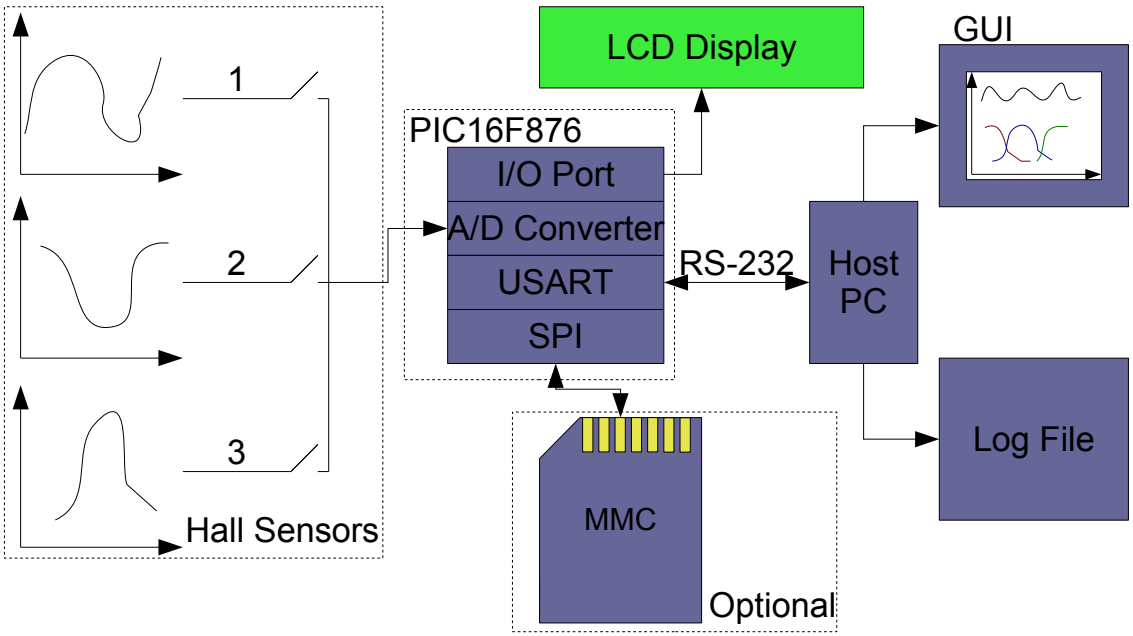


Figure 6.2. Respiro: functional scheme

```
st: // if there is no MMC, prg. loops here
 if (Command(0x41,0,0,0xFF) !=0) goto st;

 return 1;
mmcerror:
 return 0;
}

bit writeramtommc(uns16 AdrH,uns16 AdrL)
// write RAM (sector1..4) to MMC
{
 // 512 byte-write-mode
 if(!i){
  if (Command(0x58,AdrH,AdrL,0xFF) !=0) {
         SerString(''MMC: write error 1 '');
         return 1;
  }
  SPI(0xFF);
  SPI(0xFF);
  SPI(0xFE);
 }
 // write ram sectors to MMC

 SPI(AD_CONV1_HIGH); // sampled values from hall sensors
 SPI(AD_CONV1_LOW);  // are written on mmc
 SPI(AD_CONV2_HIGH);
 SPI(AD_CONV2_LOW);
 SPI(AD_CONV3_HIGH);
 SPI(AD_CONV3_LOW);
 i=i+6; // index used to monitor the 512 bytes
 if(i>509){
  SPI(255); // at the end, send 2 dummy bytes
  SPI(255);
```

```
  j = SPI(0xFF);
  j &= 0b.0001.1111;
  if (j != 0b.0000.0101) {
   SerString(``MMC: write error 2 '');
   return 1;
  }
  while(SPI(0xFF) != 0xFF); // wait until MMC is not
                            // busy anymore
  i=0;
  return 2;
 }
 return 0;
}
```

Listing 6.2. LCD communication

```
void write4(uns8  nyble)
{
 uns8 temp;
 nyble &= 0xf; // Prepare to send the nibble
               // (4 bits) by getting
               // rid of the top four
 temp = PORTC & 0xf0; // Store the current
                      // state of bits 7-4 of
                      // PORTC in temp
 temp |= nyble; // OR the PORTC state and
                // nibble to be
                // sent together
 PORTC = temp; // Write 7-4 of PORTC and 3-0
               // of nyble to PORTC

 EN=1; // Bring enable high
 nop(); // Wait a teensy bit
 EN=0; // Bring Enable Low

 return; // Return nothing
}

void write8(uns8 byte)
{
 uns8 nibble;
 nibble = (byte & 0xf0) >> 4; // Rotate the high
 // 4 bits (7-4) of byte into bits (3-0) of nibble
 write4(nibble); // Write the high 4 bits to the LCD
 nibble = byte & 0xf; // Copy the low four bits of byte
 // into the low four bits of nibble
 write4(nibble); // Write the low 4 bits to the LCD
}

void initlcd(void)
{
 delayms(20);    // Wait for LCD to power up ( >15ms )
 RS=0; // Set RS low for instruction
 write4(3);      // Set interface to 8 bits
 delayms(5); // Wait for LCD execute instruction ( >4.1ms )
 write4(3);      // Set interface to 8 bits
 delayms(1); // Wait for LCD execute instruction ( >100us )
 write4(3); // Set interface to 8 bits
 delayms(5); // Wait for LCD execute instruction
 // (At this point we could actually start using
 // the busy flag)
 write4(2);      // Set the display to 4 bit interface
 delayms(5); // Wait for LCD execute instruction
 write8(0x28); // Set the display to two line
 delayms(5); // Wait for LCD execute instruction
```

```
 write8(6);
 delayms(5); // Wait for LCD execute instruction
 write8(1);     // Clear the LCD
 delayms(5); // Wait for LCD execute instruction
 write8(0xf);
 delayms(5); // Wait for LCD execute instruction
 return;
}
```

The global system is depicted in fig. 6.3 where the assembled hall effect sensors are visible together with the linear slider used for the prototype. An old floppy disk (3.5in) is employed for the support structure and the cursor is the mobile window that protects the inner disk, sensors are attached in the fixed part while the magnet is integral with the mobile window.

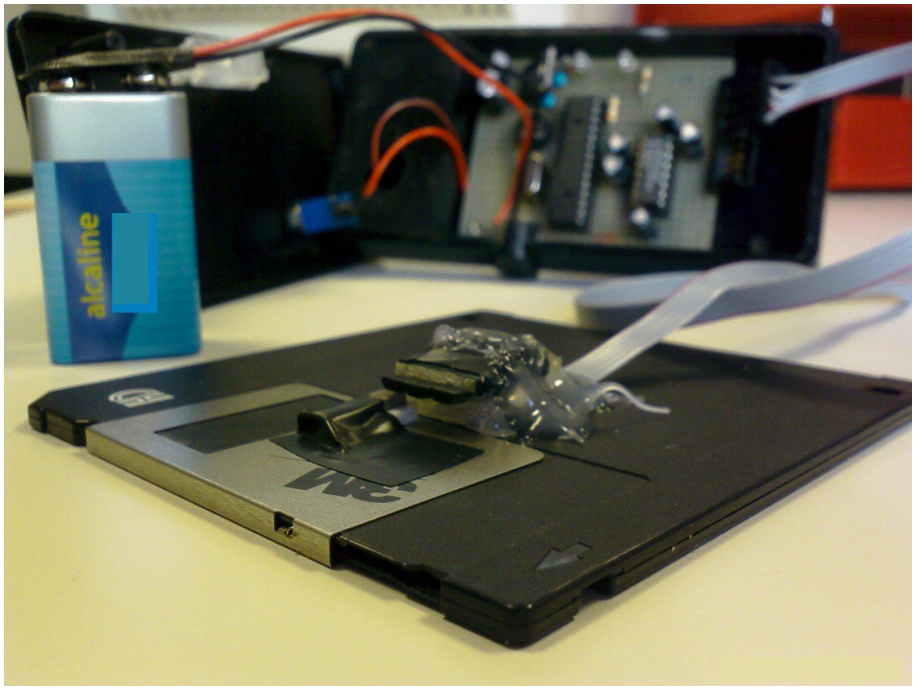


Figure 6.3. Respiro: detail on the sensors

Respiro is very modular because other types of sensors can be attached and read, like for instance accelerometers. Moreover part of data elaboration could be implemented directly on the microcontroller for real-time calculations if a PC is not available. A display could be used for showing the on line breathing frequency.

6.3 Experimental results

Respiro has been tested in a laboratory environment for a study on subjective time perception [38], for this purpose the memory extension and the display have not

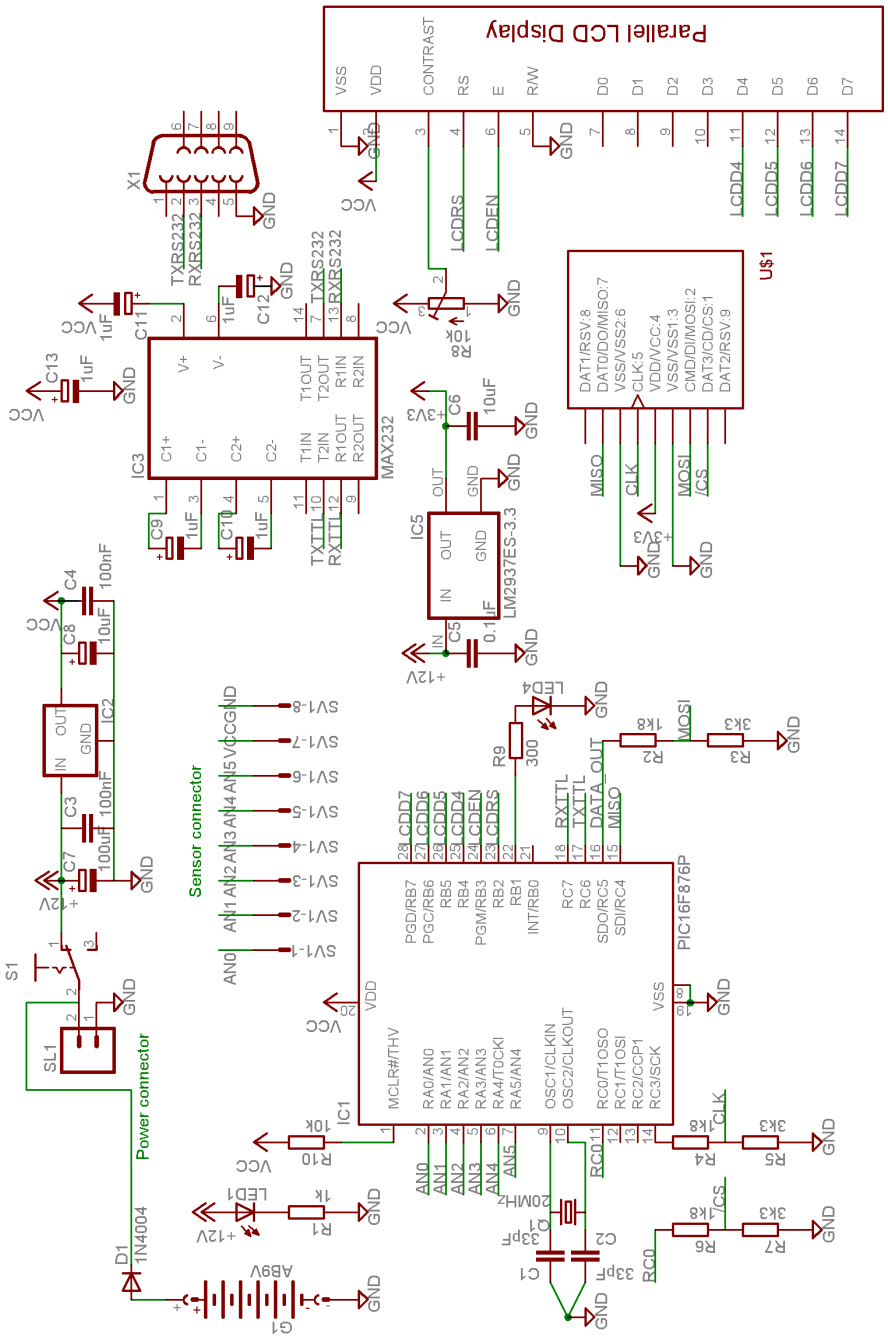


Figure 6.4. Respiro: electronic board schematic

been yet implemented. The study is related to the relationship between time perception and blinking and breathing frequencies in healthy subjects. The aim concerns the demonstration that the spontaneous blink is a sort of a servomechanism of the neuro-psychological processes, responsible for the correct functioning of the

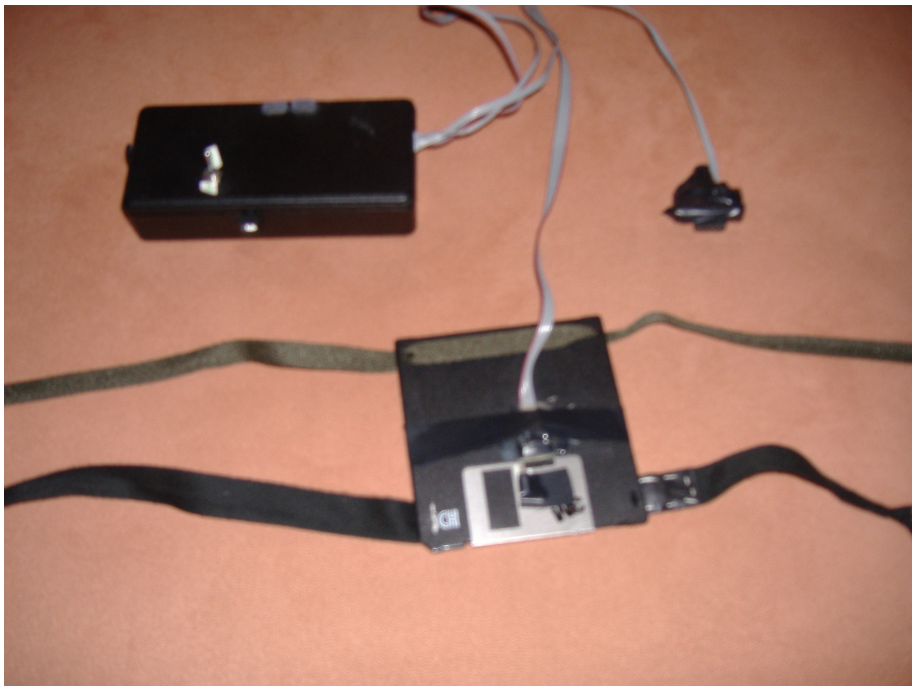


Figure 6.5. Respiro global view with belt

internal biological watch that allows the correct perception of passing time. The main measured parameter was the blinking frequency but this is related to breathing frequency so both the measures have been performed. The more the subject is concentrated the more the blinking frequency is high and the breathing frequency is low.

Chapter 7

Conclusions

In this thesis three projects have been described together with software and hardware tools used for prototyping the instruments. Two of these projects are focused on minimally invasive surgery and particularly on laparoscopy. *Leonardo* is a hand-held robot with additional DOFs than a traditional instrument, designed to be controlled in a "close teleoperation" concept. The surgeon can control it with one hand, while he/she is standing at the operating table and with the other hand he/she controls a traditional laparoscopic instrument. This approach can help the surgeon to perform complex tasks during the intervention only when he/she really needs more dexterity, like in sutures. Many works in this field have led to hand-held robotic instruments that have an end-effector actuated by one or more motors that are located in the handle of the instrument. This results in heavy and not very ergonomic instruments, moreover they don't allow changing the end-effector. *Leonardo* has an ergonomic handle and in the first version it has a commercial dexterous end-effector, but the motors are dislocated from the handle using flexible transmissions. We created the interface between the commercial end effector and the handle without motors. The master part has a joystick that allows to the surgeon to control the DOFs of the end-effector. The system has four motors and allows yaw, pitch, grasp and roll movements. This project started with the study of the best interfaces to handle the instrument using a virtual environment for implementing the end-effector visualization and its kinematics. Four different control modes have been implemented to compare the effectiveness of different interfaces and the experiments have been carried out in few days. The alternative approach would have been to fabricate four instruments, requiring much more time and money. The first prototype allows to mount different surgical tool tips (scissors, graspers and others) without modifying the handle part of the instrument. The development of the instrument has been carried out together with the evaluation of the best interface to control it and as a result of this study the second version has a different and more ergonomic handle that allows direct mapping between the orientation of the handle and the tip. The definitive version has a different end-effector designed to have 4 DOFs that substitutes the commercial one.

In the second project a bioinspired robotic system is presented. It is based on

a wide-angle camera and a telephoto camera that allows, by means of scanning movements, to produce an image with the field of vision of the wide-angle camera, but at the resolution of the telephoto system. The underlying functioning has been derived from the vision system of jumping spiders. Theoretically, the resolution of the system is limited only by the optical limits of the telephoto system and by the time required to complete the scan that forms the image. The system is suitable for applications in which the camera system remains static and the environment changes progressively, as for instance in minimally invasive surgical interventions: the background remains quite static, while surgical instruments move in the foreground. Results of first experiments are encouraging showing that, by opportunely (controllare) combining the use of two traditional cameras, it is possible to achieve a higher image resolution with respect to traditional laparoscopes using two traditional cameras and in particular the achieved resolution is about 7 times higher than commercial instruments. The prototype has not been designed to work inside the body but system components are selected with further miniaturization in mind: the Namiki motors have very small sizes, together with the CMOS camera that has typical field of view of commercial laparoscopes. To hold the whole camera still, the endoscope could be held by a robot assistant, or mounted on a fixed or mobile miniature robot for more accurate positioning in the surgical field.

The third project concerns the development and the experimental validation of *Respiro*, a breathing detector system. This instrument is easy to use, cheap, because the cost of the prototype is below 15 euros, and easy to fabricate because commercial components have been used. Respiro can be employed in combination with other sensors in order to evaluate surgeon performance, e.g. by correlating the breathing frequency with the fatigue degree of the surgeon. The system has been tested with good functionalities in a medical study related related to the relationship between time perception and blinking and breathing frequencies in healthy subjects. The development of this measurement instrument has been carried out using a microcontroller platform, creating a portable embedded system with autonomous power.

Prototyping platforms are very important to increase the speed of the development process. The three presented projects have been developed using a PC equipped with data acquisition boards (DAQ boards) for the implementation of high level algorithms and more embedded hardware like microcontrollers (outside the PC) for real-time tasks. The common strategy is to use the PC with the DAQ boards for all the prototypes and to customize only the microcontroller-based electronics outside the PC for reading sensors or for driving actuators. In fact the initial stage of a project presents difficulties in terms of choice of sensors and actuators and a high system flexibility is required in order to overcome problems without changing the core of the development platform. The microcontroller platform is totally independent from the proprietary programmer since a *bootloader* has been implemented to program the device through the serial port of a PC. The bootloader receives a user program, developed for the microcontroller, from the PC and writes it in the flash memory, then launches this program in execution. This results in a very simple

interface where high level programs are written using C++ programming languages with all the facilities of QT for creating graphical user interfaces and the libraries for using the DAQ boards. Instead the microcontroller programs are written in C programming language, compiled and then transferred using a developed program called *downloader* that dialogs with the bootloader present in the microcontroller. Only electrical connections have to be changed for switching from a project to another. Simulation is a very important issue, this can be useful when the mechanical system is not yet fabricated. Virtual reality environment can help to simulate the theoretical models created during the design stage.

All projects have been developed using platforms studied and assembled during the design stage. The flexibility and the possibility to have acquisition channels or operating systems that allow the respect of timing constraints can help to develop new prototypes and controlling algorithms without worrying about the used sensors or actuators. When the feasibility of the instrument is proved a more embedded solution has to be studied in order to approach the definitive prototype version. All the development platforms used in this work will be used in future for prototyping new robotic instruments.

Appendix A

Power considerations for endoscopic capsules

Endoscopic capsules are autonomous robots and they need to have the power module embedded since the power supply powers all the robot components. It is in the centre of the schematic depicted in Fig. 2.2. Autonomous robots are typically employed in non-structurated environments and often electrical wires can not be used. The use of battery packs can solve this problem where size constraints do not represent a limit. In capsular endoscopy very narrow size constraints exist since the capsule dimensions should be about 10mm in diameter by 20mm in length. Moreover the use of electrical wires is not allowed because the capsule path passes from the esophagus to the intestine passing through the stomach. Commercial capsules uses button batteries to power the camera and the RF data transmission modules. Legged capsules represent an important part in research works because they allow to move inside the cavity in a deterministic way. Unfortunately the actuation of mechanical parts needs the use of motors that are one of the most power consuming part in a robotic system. They have a high current absorption especially in the transient in order to start the motion. Traditional batteries can suffer this high power density requirements since chemical reactions need a specific time to be carried out.

The suited motors for endoscopic capsules need to be small and they have to be strong enough to produce forces in the order of 1N. Namiki SBL04-0829 PG04-337 are DC brushless motors and they are suitable for these kind of applications. A study on current absorption has been carried out in order to understand the power requirements of these motors. A test-bed has been created fixing a pulley to the motor and lifting up a variable load from 10g to 50g, respectively resulting in torque values from 0.49mNm to 2.45mNm. The last one corresponds to the stall torque and the vlues are obtained by muliplying the pulley radius (5mm) with the gravity acceleration and with the load weight. Results are depicted in Fig. A.2 where both the transient and the steady state are considered.

Rechargeable batteries are not suitable for capsule applications since they need special safety solutions for the charge phase. Furthermore lithium-ion (Li-ion) batteries as well as nickel metal hydride battery (NiMH) need time for charging and in

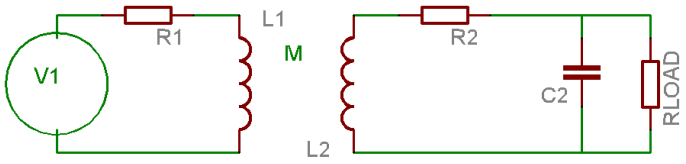


Figure A.1. Wireless powering receiver module schematic.

case of the capsule a recharge mechanism is too complicated and can take much size. The charge of lithium-ion batteries needs long time to be completed and the optimal discharge is performed with loads that absorb constant current. Wireless power transmission modules can be integrated in a pill as showed in [31]. The aim of the study was to demonstrate the feasibility of substituting two button batteries with one module ables to receive energy through an extern magnetic field. The obtained rate is about 150mW, able to power a white LED. Size can be decreased while power can be increased. Also wireless powering can suffer the high power density required by a motor and the use of a component able to store energy can help to furnish the motor request during the transient. The last component of the receiver part in the wireless system is a capacitor, as can be seen in Fig. A.1 schema wireless, and the use of a supercapacitor instead of a normal capacitor can be an interesting solution.

While electrostatic capacitors have been used as energy storage elements for nearly a century, low capacitance values have traditionally limited them to low-power applications as components in analogue circuits, or at most as short-term memory backup supplies. Recent developments in manufacturing techniques have changed this, however, and with the ability to construct materials of high surface-area and electrodes of low resistance has come the ability to store more energy in the form of electric charge. Supercapacitors have been investigating for their high power density capability. The energy storing capacity of supercapacitors and batteries is frequently compared in relation to their weight, by comparing their energy density in watt-hours per kilogram (Wh/kg) or joules per kilogram (J/kg). Similarly, the term power density (in watts per kilogram, W/kg) is used to compare how fast batteries and supercapacitors can deliver their energy, i. e. how much power (Watts) they can give off in relation to their weight. All supercapacitors have much higher power densities than batteries, but power densities can vary appreciably between different supercapacitors. The energy stored in a capacitor depends on its capacitance and the voltage applied to it, according to the formula $W = 1/2CU^2$ (where W is energy in Joules, C capacitance in Farads and U voltage in Volts). Because of the extremely high capacitances (C) of supercapacitors, the amount of energy stored is enough to give a practically usable energy storage device.

When the power requirement in a practical application shows large variations over time, e g in electric vehicles (during acceleration, going uphill, etc), supercapacitors can deliver the extra power needed. Otherwise, the batteries of the vehicle

must be given an over-capacity of many times the average capacity needed, leading to unnecessary extra weight and cost (www.skeleton-technologies.com). The function of batteries depends on chemical reactions in the electrolyte and on the electrodes, which cause the materials to deteriorate with each charge-discharge cycle and limit the lifetime of the battery (500-1000 cycles). In a supercapacitor electrical charges are only moved back and forth during every cycle. This does not affect the materials appreciably and a supercapacitor has a lifetime of many more charge-discharge cycles than a battery (up to 1000000 cycles). Chemical reactions are temperature dependent. The chemical reactions in a battery are dramatically slowed down at low temperatures. In supercapacitors no complicated chemical reactions take place. Supercapacitors are hence much less affected by temperature changes, provided that the changes stay within the operating range of the supercapacitor. A brief comparison between batteries, traditional capacitors and supercapacitors is reported in Table A.1.

In the capsule the high power density of the supercapacitor can be used for managing the transient of the motors. The charge phase and the discharge can be performed in very short time because chemical reactions are not present. A supercapacitor can deliver all of its energy almost instantly, while a battery only delivers a low trickle of power and is damaged if overtaxed.

Nowadays automotive is one of the major application field and sizes do not represent a problem. In the endoscopic capsule size constraints are very important and smaller components have to be used, especially for supercapacitors. The idea is to couple the battery with a supercapacitor for powering a Namiki DC brushless motor. The results are depicted in Fig A.3 where the different behavior between the two components can be highlighted since the supercapacitor gives instantaneously the power for the transient while the battery attends slowly. The Lithium battery has a voltage of 3.8V and two supercapacitors of value 3F have been employed. These results seem to be very interesting because the battery consumption can be more regular avoiding fast discharging phenomenon. The main problem of this approach is represented by the size of the supercapacitors. The capacitance depends on the surface area of the electrodes, the more the electrodes are big the more the capacitance will be high. The sizes of the two supercapacitors are about 10mm in diameter by 20mm in length. In the capsule new supercapacitor technology will have to be investigated maybe pointing on the use of structured carbon nanotubes for fabricating supercapacitors.

Property	**Batteries**	**Traditional Cap**	**SuperCap**
Power Density	*low*	*extremely high*	*high*
Energy Density	*high*	*extremely low*	*moderately high*
Lifetime	*low*	*high*	*high*
Temperature Dependence	*high*	*low*	*low*

Table A.1. Comparison between batteries, traditional capacitors and supercapacitors

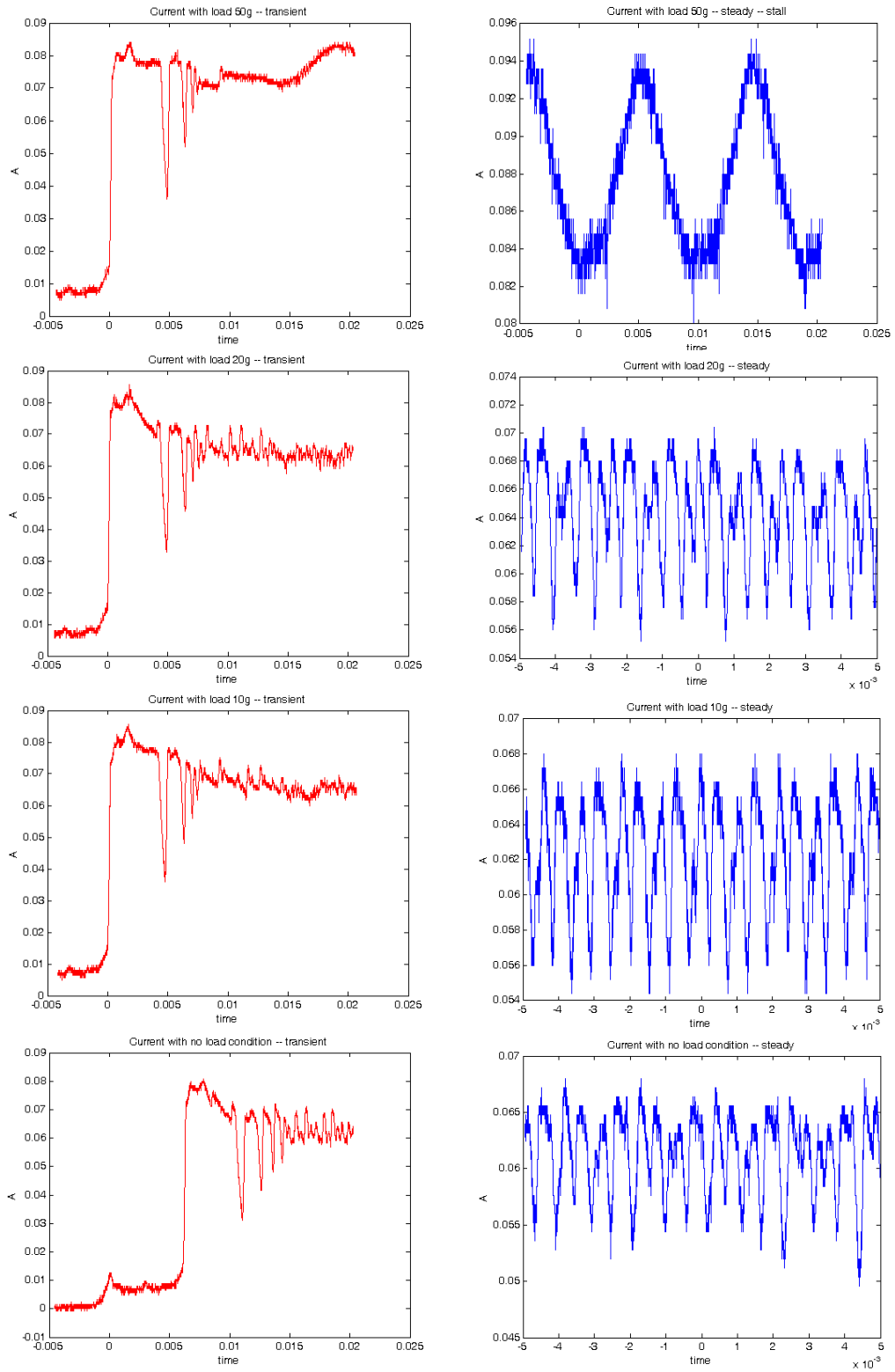


Figure A.2. Current absorption in Namiki DC brushless motor.

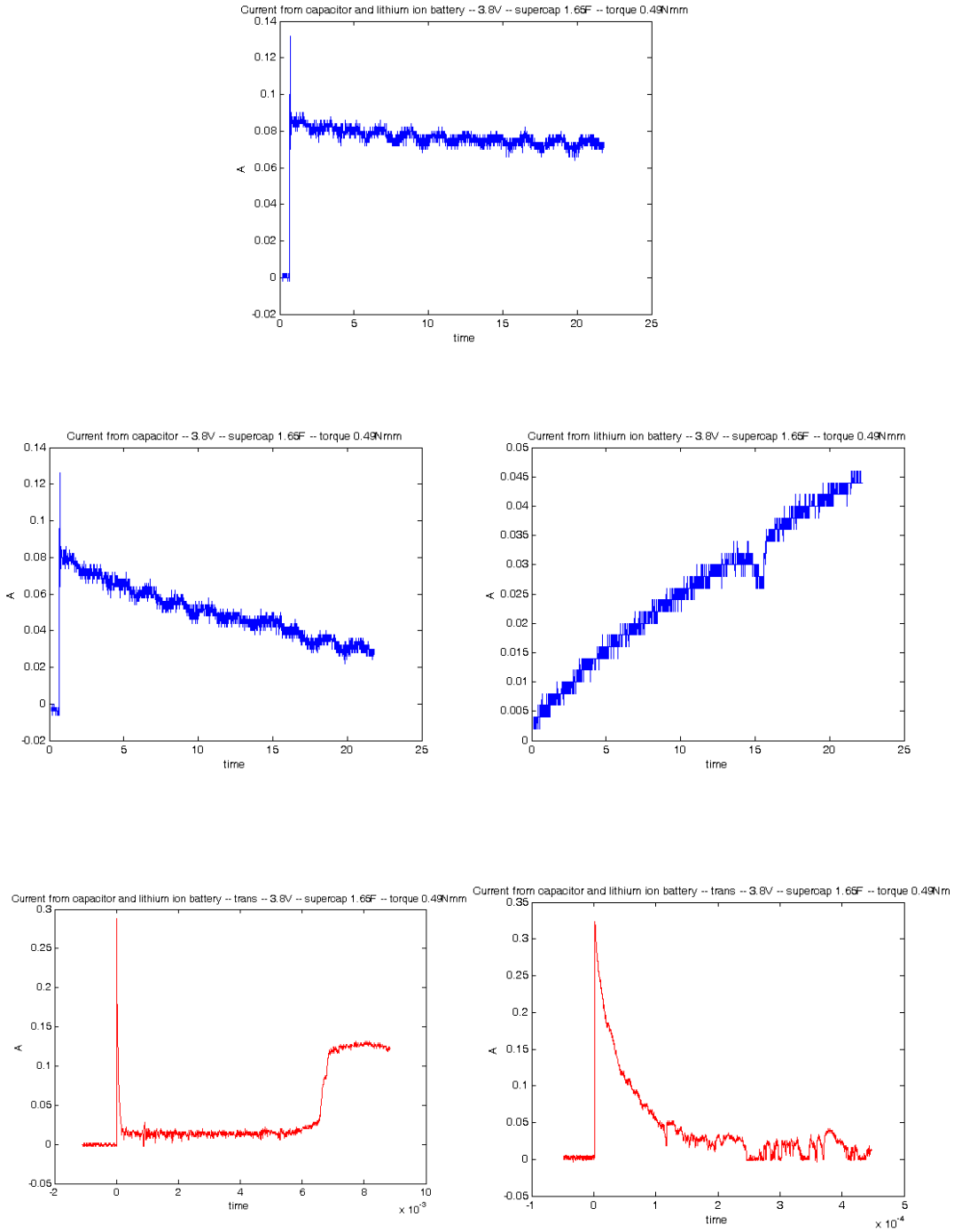


Figure A.3. Current absorption in Namiki DC brushless motor using battery in combination with supercapacior.

Appendix B

RTAI Linux

The term "real time" can have significantly different meanings, depending on the audience and the nature of the application in question. However, computer science literature generally defines only two types: soft and hard real-time systems.

A "soft real-time system" is characterized by the ability to perform a task, which on average, is executed according to the desired schedule. A video display is a good example, where the loss of an occasional frame will not cause any perceived system degradation, providing the average case performance remains acceptable. Although techniques such as interpolation can be used to compensate for missing frames, the system remains a soft real-time system, because the actual data was missed, and the interpolated frame represents derived rather than actual data.

"Hard real-time systems" embody guaranteed timing, can not miss timing deadlines and must have bounded latencies. Since deadlines can never be missed, a hard real-time system can not use average case performance to compensate for worst-case performance. One example of a hard real-time task would be monitoring transducers in a nuclear reactor, which must use a complex digital control system in order to avoid disaster. RTAI provides these necessary hard real-time extensions to Linux, thus enabling it to be targeted at applications where the timing deadlines can not be missed.

B.1 Introduction to Linux real time systems

B.1.1 Introduction

RTAI provides hard real-time extensions to Linux, yet standard Linux has support for the POSIX 1003.13 real-time extensions, so how does the "real-time" capability of standard Linux fall short?

The POSIX 1003.13 standard defines a "Multi-User Real-Time System Profile" which allows for "real-time" processes to be locked into memory to prevent the process from being paged to hard disk, and a special scheduler which ensures these processes are always executed in a predictable order.

Linux meets this standard by providing POSIX-compliant memory lock (mlock), special schedule (sched_setsched) system calls and POSIX RT signals. While those features do provide improved deterministic performance, the resultant tasks can not be defined as hard real-time tasks, since the soft real-time process can be blocked by kernel activity.

Listing B.1. POSIX approach

```
lock process into memory
// This will prevent it being out swapped to disk
set the process scheduling policy to real-time
// high priority
for{i=1;i<n;i++)
{
        read start time
        Sleep(100);      // 100 millisconds
        read end time
        calculate deviation and store result
}
output average and worst-case deviations
```

The standard Linux real-time POSIX API and the real-time kernel offer dramatically distinct services, especially within a multi-user and multi-tasking application. A simple program in pseudo-code, shown in Listing B.1, can be used to demonstrate the system performance and issues surrounding the POSIX approach.

This program reads the absolute time before entering a system call which is used to suspend the process for a predetermined interval. Next, the program reads the absolute time after the system call returns. If the system is heavily loaded with a high level of kernel activity, the return from the system call will be delayed by that activity, thus the magnitude of this delay is defined by the difference between the expected sleep time (i.e., 100 milliseconds) and the actual sleep time.

For a standard Linux system that is idle, the above task performance is extremely stable and the timing deviation is very low, yielding a worst-case deviation from an average of approximately 100 microseconds on a Pentium II 300MHz system.

However, if the program is run on a standard Linux system, using the same POSIX approach but with simultaneous I/O activity, the performance dramatically deteriorates, since the facilities of standard Linux do not allow any way for the test program to preempt the I/O-intensive application. The average execution quickly becomes unacceptable when, for example, one edits a very large file and causes hard disk activity at the same time the test program is running. This inability to preempt causes the test program to suffer typical deviations of 30 milliseconds, and for the worst case, in excess of hundreds of milliseconds. Consequently, standard Linux processes using the POSIX approach cannot be used for reliable hard real-time performance in multi-tasking environments.

Although members of the Linux community have discussed introducing a low-latency scheduler in future kernels to enhance soft real-time performance, standard Linux will still fall well short of the guaranteed response time required by a hard real-time task.

The performance advantage of real-time Linux is easily seen by running this same test program as a real-time task, yielding worst case deviations of less than 12.5 microseconds regardless of I/O activity.

Along with the scheduling benefits of the real-time kernel, the efficiency of the real-time architecture allows Linux to run under aggressive task iteration rates. For example, when running this program at 500-microsecond intervals as a POSIX real-time task, Linux stops completely. Running this program as a real-time task leaves enough resources for Linux to continue to run.

The operating requirements of a general-purpose operating system such as Linux are different from those of a hard real-time system. This difference leads to other issues, summarized below, which limit the potential for standard Linux to act as a real-time operating system.

- The Linux kernel uses coarse-grained synchronization, which allows a kernel task exclusive access to some data for long periods. This delays the execution of any POSIX real-time task that needs access to that same data.

- Linux does not preempt the execution of any task during system calls. If a low-priority process is in the middle of a fork system call and a message is received for the video display process, then the message will unfortunately be held in the queue until the call completes, despite its low priority. The solution is to add preemption points in the kernel, having the deleterious effect of slowing down all system calls.

- Linux makes high-priority tasks wait for low-priority tasks to release resources. For example, if any process allocates the last network buffer and a higher-priority process needs a network buffer to send a message, the higher-priority process must wait until some other process releases a network buffer before it can send its message.

- The Linux scheduling algorithm will sometimes give the most unimportant and nicest process a time slice, even in circumstances where a higher-priority process is executable. This is an artifact of a general-purpose operating system that ensures a background maintenance process; e.g., one that cleans up log files and runs even if a higher-priority process were able to use all the available processor time.

- Linux reorders requests from multiple processes to use the hardware more efficiently. For example, hard-disk block reads from a lower-priority process may be given precedence over read requests from a higher-priority process in order to minimize disk-head movement or improve the chances of error recovery.

- Linux will batch operations to use the hardware more efficiently. For example, instead of freeing one page at a time when memory gets tight, Linux will run through the list of pages, clearing out as many as possible, which will delay the execution of all processes. This is clearly desirable for a general-purpose operating system, but is undesirable for real-time processes.

Real-time and general-purpose operating systems have contradictory design requirements. A desirable effect in one system is often detrimental in the other.

Unfortunately, any attempt to satisfy both requirements in the same kernel often results in a system that does neither very well. That is not to say general-purpose functionality and real-time determinism cannot be achieved simultaneously. In fact, operating systems that combine these requirements exist now, and they are indeed deterministic, preemptive and contain bounded latencies (thus meeting the requirement for a hard real-time system). However, the worst-case behavior for those latencies can be unacceptably slow.

B.1.2 Introduction to RTAI

In order to make Linux usable for hard real-time applications, members of the Department of Aerospace Engineering, Politecnico di Milano (DIAPM) envisioned a real-time hardware abstraction layer (RTHAL) onto which a real-time applications interface (RTAI) could be mounted. Unfortunately, further investigation revealed that the Linux kernel available in late 1996, 2.0.25, was not yet mature enough for the concept.

Around the same time, a group headed by Victor Yodaiken at the New Mexico Institute of Mining and Technology (NMT) in Soccorro, NM introduced its real-time Linux (RTLinux), which provided the DIAPM team with the opportunity to learn further about the Linux kernel, the hardware and the modifications necessary to provide preemptive and deterministic scheduling. The turning point came in 1998 when the Linux 2.2.x kernel featured key improvements, including much-needed architectural changes to the Linux/hardware interface. These changes, combined with the experience gained by the DIAPM team while working with their own evolution of the NMT-RTLinux kernel, and the concepts of 1996, resulted in RTAI.

RTAI provides guaranteed, hard real-time scheduling, yet retains all of the features and services of standard Linux. Additionally, RTAI provides support for multiprocessor architecture with the ability to assign both tasks and IRQs to specific CPUs, x486 and Pentiums, simultaneous one-shot and periodic schedulers, both inter-Linux and intra-Linux shared memory, POSIX compatibility, FPU support, inter-task synchronization, semaphores, mutexes, message queues, RPCs, mailboxes, the ability to use RTAI system calls from within standard user space and more.

B.1.3 RTAI Architecture

The underlying architecture for RTLinux and RTAI is quite similar. For each implementation, the Linux operating system is run as the lowest priority task of a small real-time operating system. Thus, Linux undergoes no changes to its operation from the standpoint of the user or the Linux kernel, except that it is permitted to execute only when there are no real-time tasks executing. Functionally, both architectures provide the capability of running special real-time tasks and interrupt handlers that execute whenever needed, regardless of what other tasks Linux may be performing. Both implementations extend the standard Linux programming environment to real-time problems by allowing real-time tasks and interrupt handlers to communicate with ordinary Linux processes through a device interface or shared memory.

The primary architectural difference between the two implementations is in how these real-time features are added to Linux. Both RTLinux and RTAI take advantage of Linux's loadable kernel modules when implementing real-time services. However, one of the key differences between the two is how these changes, to add the real-time extensions, are applied to the standard Linux kernel.

RTLinux applies most changes directly to the kernel source files, resulting in modifications and additions to numerous Linux kernel source files. Hence, it increases the intrusion on the Linux kernel source files, which can then result in increased code maintenance. It also makes tracking kernel updates/changes and finding bugs far more difficult.

RTAI limits the changes to the standard Linux kernel by adding a hardware abstraction layer (HAL) comprised of a structure of pointers to the interrupt vectors, and the interrupt enable/disable functions. The HAL is implemented by modifying fewer than 20 lines of existing code, and by adding about 50 lines of new code. This approach minimizes the intrusion on the standard Linux kernel and localizes the interrupt handling and emulation code, which is a far more elegant approach. Another advantage of the HAL technique is that it is possible to revert Linux to standard operation by changing the pointers in the RTHAL structure back to the original ones. This has proven quite useful when real-time operation is inactive or when trying to isolate obscure bugs.

Many have surmised that the HAL could cause unacceptable delays and latencies through the real-time tasking path. In fact, the HAL's impact on the kernel's performance is negligible, reflecting highly on the maturity and design of the Linux kernel and on those who contributed to its development [1].

B.1.4 Real-Time Application Interface

The HAL supports five core loadable modules which provide the desired on-demand, real-time capability. These modules include rtai, which provides the basic rtai framework; rtai_sched, which provides periodic or one-shot scheduling; rtai_mups, which provides simultaneous one-shot and periodic schedulers or two periodic schedulers,

each having different base clocks; rtai_shm, which allows memory sharing inter-Linux, between real-time tasks and Linux processes, and also intra-Linux as a replacement for the UNIX V IPC; and rtai_fifos, which is an adaptation of the NMT RTLinux FIFOs (file in, file out).

Like all kernel modules, these can be loaded and unloaded (using the standard Linux insmod and rmmod commands) as their respective capabilities are required or released. For example, if you only interrupt handlers need to be installed, only the rtai module has to be loaded. If also the standard Linux processes for communication are needed, using FIFOs, then the rtai and rtai_fifos modules will be loaded. This modular and non-intrusive architecture allows FIFOs to run on any queue and use immediate wake-ups as necessary.

B.1.5 The Real-Time Task

The real-time task is implemented similarly to RTAI; i.e., it is written and compiled as a kernel module which is loaded into the kernel after the required RTAI core module(s) has been loaded. This architecture yields a simple and easily maintained system that allows dynamic insertion of the desired real-time capabilities and tasks. The example below shows all that is required for a task to be scheduled in real time (Listing B.2):

Listing B.2. insmod

```
insmod /home/rtai/rtai
insmod /home/rtai/modules/rtai_fifo
insmod /home/rtai/modules/rtai_sched
insmod /path/rt_process
```

To stop the application and remove the RTAI (Listing B.3):

Listing B.3. rmmod

```
rmmod rt_process
rmmod rtai_sched
rmmod rtai_fifo
rmmod rtai
```

The facilities ldmod and remod may be used to load and unload the core modules.

B.1.6 Task Scheduler

RTAI's task scheduler allows hard real-time, fully preemptive scheduling based on a fixed-priority scheme. All schedules can be managed by timing functions and real-time events such as semaphore acquisition, clocks and timing functions, asynchronous event handlers, and include inter-task synchronization.

B.1.7 One-Shot and Periodic Scheduling

RTAI supports both one-shot and periodic timers on Pentium and 486-class CPUs. Although both periodic and one-shot timers are supported, they may not be instantiated simultaneously; i.e., one-shot and periodic tasks may not be loaded into the kernel as modules at the same time.

Using these timers (instantiated by rtai_sched), periodic rates in excess of 90KHz are supported, depending on the CPU, bus speed and chip set performance. On Pentium processors, one-shot task rates in excess of 30KHz are supported (Pentium II, 233 MHz), and on 486 machines, the one-shot implementation provides rates of about 10KHz, all while retaining enough spare CPU time to service the standard Linux kernel.

The limitation of rtai_sched to support simultaneously one-shot and periodic timers is mitigated by the MultiUniProcessor (MUP) real-time scheduler (rtai_mups), which provides the capability to use, simultaneously, both a periodic and a one-shot timer or two periodic timers with different periods, at a performance equivalent to that noted above under rtai_sched. Note that, since the MUP uses the APIC (advanced programmable interrupt controller) timer, it cannot operate under multiprocessor architecture and requires each MUP-scheduled task to be locked to a specific CPU (thus the MultiUniProcessor designation); however, the MUP retains all coordination and IPC services, so that no other capabilities are lost.

B.1.8 Floating-Point Operations

Floating-point operations within real-time tasks/ISRs (interrupt service routines) are possible, provided these tasks are marked, upon loading, as tasks which require the FPU. This method provides real-time task access to the FPU while still allowing FPU access to standard Linux tasks.

B.1.9 Interrupt Handling

RTAI provides efficient and immediate access to the hardware by allowing, if one chooses, interaction directly with the low-level PC hardware, without first passing through the interrupt management layers of the standard Linux kernel.

The ability to individually assign specific IRQs to specific CPUs, as described in further detail below, allows immediate, responsive and guaranteed interface times to the hardware.

B.1.10 Inter-Process Communication

The term inter-process communication (IPC) describes different ways of message passing between active processes or tasks, and also describes numerous forms of synchronization for the data transfer.

Linux provides standard system V IPC in the form of shared memory, FIFOs, semaphores, mutexes, conditional variables and pipes which can be used by standard user processes to transfer and share data. Although these Linux IPC mechanisms are not available to do real-time tasks, RTAI provides an additional set of IPC mechanisms which include shared memory, message queues, real-time FIFOs, mutexes, semaphores and conditional variables. These are used to transfer and share data between tasks and processes in both the real-time and Linux user-space domains.

RTAI's remote procedure call (RPC) mechanism is similar in operation to QNX messages available to real-time tasks, and transfers either an unsigned integer or a pointer to the destination task(s).

The RTAI mailbox implementation provides the ability to send any message from user space to real-time tasks, real-time tasks to real-time tasks, and user tasks to user tasks (using LXRT) via any definable mailbox buffer size. Multiple senders and receivers are allowed, where each is serviced according to its priority.

B.1.11 proc Interface

From version 0.9, RTAI includes a proc interface which gives useful information on the current status of RTAI, including schedulers loaded; real-time tasks activity, priority and period; and FIFOs in use and their associated buffer sizes. The development of even more features is currently underway.

B.1.12 SMP Support

RTAI provides true support for symmetric multiprocessing (SMP) architectures through its task and IRQ management. By default, all tasks are assigned to run on any CPU (of a SMP platform). Each task, however, can be individually assigned to any CPU subset, or even to a single CPU. Additionally, it is possible to assign any real-time interrupt service to any specific CPU. Because the ability to force an interrupt to a specific CPU is not related to the SMP scheduler, RTAI retains the flexibility to perform these two operations independently.

These capabilities provide a method of statically optimizing the real-time application, if manual task distribution handles the task more efficiently than the automatic SMP load-distribution services of Linux.

B.1.13 Linux-RT (LXRT)

Since real time Linux tasks are implemented as loadable modules, they are, for all practical purposes, an integral part of the kernel. As such, these tasks are not bounded by the memory protection services of Linux, and they have the ability to overwrite system-critical areas of memory, bringing the system to an untimely halt. This limitation has been a large frustration to those of us who have erred during real-time task development.

RTAI's LXRT solves this problem by allowing the development of real-time tasks, using all of RTAI's hard real-time system calls from within the memory-protected space of Linux and under a 'firm' real-time service. When the developer is satisfied with a task's performance within LXRT, the task is simply recompiled as a module and inserted into the kernel (along with the associated modules which provide RTAI's real-time features) to transition from firm to hard real-time.

LXRT's firm real-time service, similar to that offered by the Kansas University Real-Time (KURT) patch, provides soft real-time combined with fine-grained task scheduling. Performance under LXRT is quite good, yielding latencies not much greater than for a standard Linux system call leading to a task switch. Although this is very valuable as a development tool, we should not lose sight of the fact that RTAI's firm real-time implementation can prove especially useful for those tasks which don't require hard real-time, but yet are not quite satisfied with the scheduling performance of standard Linux.

B.1.14 POSIX Compatibility API

RTAI implements a compliant subset of POSIX 1003.1.c through the use of a single loadable module. These calls support creation, deletion, attribute control and environment control for threads, mutexes and condition variables. The resultant POSIX support is similar to standard Linux threads, except that parent-child functions (which are not appropriate for real-time tasks, since all threads are considered to be part of a single process) and signal handling (which is currently in development) are not supported.

B.1.15 Typical Performance

RTAI is now competitive from both a cost and performance perspective with the commercial RTOS currently available. Since the performance of any RTOS system is determined by the performance of the RTOS itself, the performance of the hardware on which it is running, and the test procedure used to acquire the data, absolute performance figures are very difficult to quantify, often making comparisons difficult between fundamentally similar RTOSes. However, the data below was measured on, and is representative of, typical Pentium II 233MHz and 486 platforms. For these performance characterizations, an early version of the RTHAL module was demonstrated running a timer at 125KHz (Pentium II, 233MHz) while simultaneously servicing Linux, which was working under a heavy load. During this demonstration, the average and maximum jitters about the periodic timer were $0\mu s$ and $13\mu s$, respectively. This performance, combined with additional tests, can be summarized in this way:

- Maximum periodic task iteration rate: 125KHz
- Typical sampling task rate: 10KHz (Pentium 100)

- Jitter at maximum task iteration rate: 0-13μs UP, 0-30μs SMP
- One-shot interrupt integration rate: 30KHz (for Pentium-class CPU), 10KHz (for 486-class CPU)
- Context switching time: approximately 4μs

B.2 Installing RTAI Linux

The RTAI installing is not a standard procedure since it depends from the PC hardware and the used Linux distribution that has to be patched. Many guides can be found and the procedure listed below is based on all these works. The components used for the installation are:

- Notebook with Pentium III 900MHz;
- Suse 9.3 Professional Linux distribution;
- Linux kernel-2.6.10;
- RTAI 3.2 test3 version;

The Linux kernel "linux-2.6.10.tar.bz2" can be downloaded from the website http://www.kernel.org/ while RTAI Linux "rtai-3.2.tar.bz2" can be downloaded from the website http://www.aero.polimi.it/RTAI/. The files can be extracted, as user, in the folder /opt/rtai in order to separate the real-time kernel from the original one.

```
tar xjf linux-2.6.10.tar.bz2
tar xjf rtai-3.2-test3.tar.bz2
```

Than kernel can be patched and configured as user:

```
cd linux-2.6.10
patch -p1 <../rtai-3.2-test3/base/arch/i386/
        patches/hal-linux-2.6.10-i386-r9.patch
cp arch/i386/defconfig .config
make xconfig
```

The *xconfig* command opens a graphical interface in which it is possible to configure the kernel. Some options have to be changed:

- Select *Device drivers* → *Block devices* → *Loopback device support.*
- Select *Loadable module support* → *Enable loadable module support.*
- Select *Loadable module support* → *Forced module unloading.*
- Deselect *Kernel hacking* → *Compile the kernel with frame pointers.*

- Select *Processor type and features → Preemptible kernel.*

Then the kernel has to be compiled and installed:

```
make
su
# make
# make modules_install
# make install
```

Then only "lilo" or "grub" have to be fitted. The option "lapic" has to be added in these files.

```
# shutdown -r now
```

Restart the system with the new kernel and then it is possible to launch test applications in order to verify if the installation succeeded.

```
su
# cd /usr/realtime/testsuite/kern/switches
# ./run
```

press ctrl-c to stop.

B.3 Application example

The example listed below is an application that reads a single joystick value every 0.1ms and generates 8 PWM signals with duty cycle variable and period of 10ms (100Hz). The duty cycle depends on the joystick values and varies from 20% and 80%. In this application the 8 PWM signals have the same duty cycle but the granularity of the task allows to customize each of the 8 signals according to the 8 input values. The joystick acquisition is performed through the PC serial port. A microcontroller samples continuosly the joystick values and than the PC with RTAI Linux asks these values by sending a particular character on the serial port. The microcontroller gives back the sampled values and a great number of data acquisition channels can be used. 8 different inputs can be acquired changing each of the 8 PWM channels according to the input values. The PWM signals with frequency of 100Hz can be used for the DC motor driver MCDC2805 by Faulhaber that accepts in input pulse width modulated signals (position or speed commands) with frequency from 100Hz to 1kHZ.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <stdio.h>
#include <errno.h>
#include <sys/mman.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
```

```
#include <rtai_lxrt.h>
#include <rtai_sem.h>
#include <rtai_usi.h>
#include <sys/io.h>

#define PARPORT_IRQ  7
#define BASEPORT 0x378
#define BAUDRATE B19200
#define MODEMDEVICE "/dev/ttyS0"
#define _POSIX_SOURCE 1 /* POSIX compliant source */
#define FALSE 0
#define TRUE 1
#define PERIOD 100000

static SEM *dspsem;
static volatile int endjoystick = 1;
static volatile int endsquare = 1;
int duty = 50;
char buf[5];

static void *read_joystick(void *args)
{
        RT_TASK *handler;
        printf("readjoy\n");
        if (!(handler = rt_task_init_schmod(nam2num("RDJDLR"),
         0, 0, 0, SCHED_FIFO, 0xF))) {
                printf("CANNOT INIT HANDLER TASK -> RDJDLR <\n");
                exit(1);
        }
        rt_allow_nonroot_hrt();
        mlockall(MCL_CURRENT | MCL_FUTURE);
        rt_make_hard_real_time();
        endjoystick = 0;
        int fd,c, res;
        struct termios oldtio,newtio;
        char key='k';
        fd = open(MODEMDEVICE, O_RDWR | O_NOCTTY );
        if (fd <0) {perror(MODEMDEVICE); exit(-1); }
        tcgetattr(fd,&oldtio);/* save current port settings */
        bzero(&newtio, sizeof(newtio));
        newtio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
        newtio.c_iflag = IGNPAR;
        newtio.c_oflag = 0;
        /* set input mode (non-canonical, no echo,...) */
        newtio.c_lflag = 0;
        newtio.c_cc[VTIME] = 0;/* inter-character timer unused */
        newtio.c_cc[VMIN] = 5;/* blocking read until 5 chars received */
        tcflush(fd, TCIFLUSH);
        tcsetattr(fd,TCSANOW,&newtio);
        printf("ok\n");
        while( !endjoystick ){/* loop for input */
                write(fd,&key,1);/* send one character */
                res = read(fd,buf,5);/* returns after 5 chars have been input */
                /* so we can printf... */
                printf(":%s:%d\n", buf, res);
                duty=buf[1]*0.392;/* duty cycle based on joy value */
                rt_sem_signal(dspsem);
                //usleep( 100000 );
        }
        tcsetattr(fd,TCSANOW,&oldtio);
        rt_make_soft_real_time();
        rt_task_delete(handler);
        endjoystick = 1;
        return 0;
}
```

```
static void *send_pwm(void *args)
{
        RT_TASK *handler;
        RTIME period;
        int t;
        if (!(handler = rt_task_init_schmod(nam2num("SQHDLR"),
         0, 0, 0, SCHED_FIFO, 0xF))) {
                printf("CANNOT INIT HANDLER TASK > SQHDLR <\n");
                exit(1);
        }
        rt_allow_nonroot_hrt();
        mlockall(MCL_CURRENT | MCL_FUTURE);
        rt_set_oneshot_mode();
        start_rt_timer(0);
        period = nano2count(PERIOD);
        rt_make_hard_real_time();
        endsquare = 0;
        rt_task_make_periodic(handler, rt_get_time() + period, period);
        // pwm period of 100Hz
        while ( !endsquare ) {
                outb_p(255, BASEPORT);
                for(t=1;t<=duty;t++){
                        rt_task_wait_period();
                }
                outb_p(0, BASEPORT);
                for(t=1;t<=(100-duty);t++){
                        rt_task_wait_period();
                }
        }
        stop_rt_timer();
        rt_make_soft_real_time();
        rt_task_delete(handler);
        endsquare = 1;
        return 0;
}

int main(void)
{
        RT_TASK *maint; //, *squaretask;
        int thread, joystickthread;
        if (!(maint = rt_task_init(nam2num("MAIN"), 1, 0, 0))) {
                printf("CANNOT INIT MAIN TASK > MAIN <\n");
                exit(1);
        }
        // create semaphore to notify main() when interrupt occurs
        if (!(dspsem = rt_sem_init(nam2num("DSPSEM"), 0))) {
                printf("CANNOT INIT SEMAPHORE > DSPSEM <\n");
                exit(1);
        }
        // ask for permission to access the parallel port from user-space
        if (iopl(3)) {
                printf("iopl err\n");
                rt_task_delete(maint);
                rt_sem_delete(dspsem);
                exit(1);
        }
        outb_p(0x10, BASEPORT + 2); //set port to interrupt mode; pins are output
        thread = rt_thread_create(send_pwm, NULL, 10000);  // create thread
        // thread period of 10kHz
        while (endsquare) {// wait until thread went to hard real time
                usleep(100000);
        }
        joystickthread = rt_thread_create(read_joystick, NULL, 10000);
        // create thread
        // thread period of 10kHz
        while (endjoystick) {// wait until thread went to hard real time
```

```
                usleep(100000);
        }
        while (!endsquare && !endjoystick ) {
                rt_sem_wait(dspsem);
                printf("ok [use CTRL-C to end]\n");
        }
        endjoystick = 1;
        endsquare = 1;
        printf("TEST ENDS\n");
        outb_p(0, BASEPORT);
        rt_thread_join(thread);
        rt_thread_join(joystickthread);
        rt_task_delete(maint);
        rt_sem_delete(dspsem);
        return 0;
}
```

Appendix C

Bootloader for microcontroller and DSP

C.1 Microchip PIC16F876 and dsPIC30F2010

The *Bootloader* is a program that resides inside the microcontroller. The used code is based on the Tiny Bootloader that can be employed for both the PIC16F876 and the dsPIC30F2010. The main advantage is represented by the possibility to use the PC serial port for programming the device instead of the official programmer. Obviously the proprietary programmer needs to be used only one time, when the bootloader program has to be flashed inside the device. The bootloader receives a user program from the PC and writes it in the flash memory, then launches this program in execution. Bootloaders can only be used with those microcontrollers that can write their flash memory through software. The bootloader itself must be written into the flash memory with an external programmer. In order for the bootloader to be launched after each reset, a "goto bootloader" instruction must exist somewhere in the first 4 instructions; There are two types of bootloaders, some that require that the user reallocate his code and others that by themselves reallocate the first 4 instructions of the user program to another location and execute them when the bootloader exits. The *Downloader* is a program that runs on the PC and its aim is to transfer the HEX file, produced by the CC5X C compiler, into the program memory of the microcontroller PIC16F876 through the PC serial port. It is based on the original bootloader made by Petr Kolomaznik. The graphical user interface has been designed with Qt 3.2.1 non-commercial edition (see Fig. C.1). The downloader used for the dsPIC30F2010 is the original version made by Petr Kolomaznik.

```
void FormBootLoader::init()
{
  bool fSuccess;
  DCB dcbCommPort;
  char* SerLinka;
  DWORD BaudRateTransmission;
  a=comboBoxCOM->currentText();
  SerLinka=(char*)a.ascii();
```

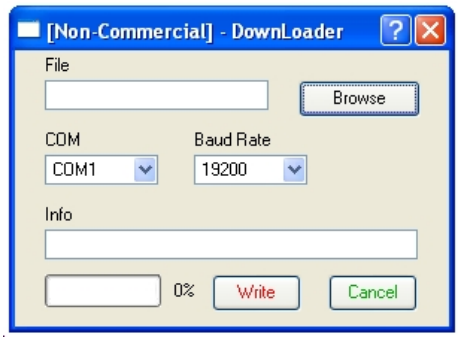


Figure C.1. Downloader GUI.

```
COMMTIMEOUTS TimeOuts;
hComm = CreateFileA(SerLinka,          //open port
  GENERIC_READ | GENERIC_WRITE,
  0,                                    //exclusive access
  0,                                   //no security attrs
  OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL,
  0
  );
if(hComm == INVALID_HANDLE_VALUE)
{
  lineEditInfo->setText("Open port error!");
  return;
}
//set of parameters
fSuccess = GetCommState(hComm, &dcbCommPort);
if(!fSuccess)
{
  lineEditInfo->setText("Read port parameters error!");
  return;
}
a=comboBoxBaud->currentText();
dcbCommPort.BaudRate = (a.toDouble());
dcbCommPort.ByteSize = 8;
dcbCommPort.Parity = NOPARITY;
dcbCommPort.StopBits = ONESTOPBIT;
dcbCommPort.fBinary = 1;
fSuccess = SetCommState(hComm, &dcbCommPort);     //write of parameters back
if(!fSuccess)
{
  lineEditInfo->setText("Write port parameters error!");
  return;
}
// Set timeouts of communication port short. Setting high timeout
// introduces problems for console mode.
// ReadFile seems to block outgoing communication
TimeOuts.ReadIntervalTimeout = 50;
TimeOuts.ReadTotalTimeoutMultiplier = 0;
TimeOuts.ReadTotalTimeoutConstant = 10;
TimeOuts.WriteTotalTimeoutMultiplier = 20;
TimeOuts.WriteTotalTimeoutConstant = 1000;
fSuccess = SetCommTimeouts(hComm,&TimeOuts);
if(!fSuccess)
{
  lineEditInfo->setText("Set communication timeout error!");
  return;
}
```

```
  //clear buffers
  fSuccess = PurgeComm(hComm,
  PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
  if(!fSuccess)
  {
    lineEditInfo->setText("Clear buffers error !");
    return;
  }
}


void FormBootLoader::Browse()
{
  filename = QFileDialog::getOpenFileName(
                              QString::null, "Hex file (*.hex)", this,
                              "file open", "BootLoader -- File Open" );
  if ( !filename.isEmpty() ) {
    lineEditFile->setText( filename );
    QFile HexFile(filename);
    HexFile.open(IO_ReadOnly );
    QTextStream stream( &HexFile );
    NumLines=0;
    while(!stream.atEnd())
    {
      stream.readLine();
      NumLines++;
    }
    }
    else
        lineEditInfo->setText( "File Open abandoned" );
}

/*=============================================================================
   Hex string -> byte conversion
=============================================================================*/
unsigned char FormBootLoader::GetHex(unsigned char* cPtr)
{
   unsigned char ucHulp;
   if(cPtr[0]<='9') ucHulp = cPtr[0] - '0';
   else ucHulp = cPtr[0] - 'A' + 10;
   ucHulp = ucHulp << 4;
   if(cPtr[1]<='9') ucHulp = ucHulp + cPtr[1] - '0';
   else ucHulp = ucHulp + cPtr[1] - 'A' + 10;
   return ucHulp;
}

bool FormBootLoader::Communication(unsigned char Instr, unsigned char* VysBuff)
{
  DWORD Sended;
  DWORD Received;
  unsigned char CheckSum;
  unsigned char NumberOfData, N, Pointer;
  unsigned char RecBuff[41];
  unsigned char SendBuff[41];
  unsigned char SendLength;
  unsigned char RecLength;
  int Code, I, J;
  bool bRetVal;
  bool fSuccess;
  unsigned int Address;
  fSuccess = true;
  SendBuff[0] = Instr;
  SendLength = 1;
  RecLength = 1;
  if (Instr == WRITE)
  {
```

```
  Address = GetHex(&VysBuff[3]);
  Address = Address << 8;
  Address = Address + GetHex(&VysBuff[5]);
  Address = Address >> 1;
  if ((Address >= 0x2000) && (Address < 0x2100))
  {
  //don't send address from 0x2000 to 0x20FF
  return(true);
  }
  if ((Address >= 0x2100))
  {
  return(true);
  }
  SendBuff[1] = (unsigned char)(Address >> 8);       //high byte of address
  SendBuff[2] = (unsigned char)(Address & 0x00FF);  //low byte of address
  NumberOfData = GetHex(&VysBuff[1]);
  SendBuff[3] = NumberOfData;                        //number of data
  CheckSum = 0;
  for (N=1;N<=NumberOfData/2;N++)
  {
  Pointer = (N-1) * 4;
  I = GetHex(&VysBuff[11+Pointer]);
  SendBuff [5 + ((N-1)*2)] = I;                //high byte of instruction
  CheckSum = CheckSum + I;
  I = GetHex(&VysBuff[9+Pointer]);
  SendBuff [6 + ((N-1)*2)] = I;                //low byte of instruction
  CheckSum += I;
  }
  SendBuff[4] = CheckSum;                          //checksum
  SendLength = 5 + NumberOfData;
  RecLength = 2;                                   //wait for 2 bytes
  }
  qApp->processEvents();
  PurgeComm(hComm,PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
  WriteFile(hComm, SendBuff, SendLength, &Sended, NULL);   //send
  ReadFile(hComm, RecBuff, RecLength, &Received, NULL);    //receive
  if(Received > 0)
  {
    switch(Instr)
    {
    case IDENT:
      if (RecBuff[0] == IDACK) bRetVal = true;
      else bRetVal = false;
      break;

    case WRITE:
      if ((RecBuff[0] == DATA_OK) && (RecBuff[1] == WOK)) bRetVal = true;
      else bRetVal = false;
      break;

    case DONE:
      if (RecBuff[0] == WOK) bRetVal = true;
      else bRetVal = false;
      break;
    }
  }
  else fSuccess = false;
  PurgeComm(hComm,PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
  if(!fSuccess)
  {
    lineEditInfo->setText( "Timeout of communication!" );
    bRetVal = false;
  }
  return bRetVal;
}
```

```
/*=================================================================
    Change baudrate of already open COM port
=================================================================*/
bool FormBootLoader::ChangeComSpeed(int iSpeed)
{
    bool fSuccess;
    DCB dcbCommPort;
    // Set of parameters
    fSuccess = GetCommState(hComm, &dcbCommPort);
    if(!fSuccess)
    {
        return(false);
    }
    dcbCommPort.BaudRate = iSpeed;
    fSuccess = SetCommState(hComm, &dcbCommPort);     //write of parameters back
    if(!fSuccess)
    {
        return(false);
    }
    fSuccess = PurgeComm(hComm,
    PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
    if(!fSuccess)
    {
        return(false);
    }
    return(true);
}

void FormBootLoader::Write()
{
  char* Data;
  bool ComOK, EndOfRecord;
  int NumberOfLines;
  float LineNumber;
  DWORD Received;
  DWORD Sended;
  unsigned char RecBuff[2];
  unsigned char SendBuff[2];
  bool AutoStart;
  COMMTIMEOUTS TimeOuts;
  QString line;
  // Disable all except Cancel button
  pushButtonBrowse->setEnabled(0);
  pushButtonWrite->setEnabled(0);
  comboBoxCOM->setEnabled(0);
  EndOfRecord = false;
  if (ChangeComSpeed(19200))
  {
    EscapeCommFunction(hComm, SETDTR);          // trigger pin = 0
    lineEditInfo->setText( "Reset" );
    EscapeCommFunction(hComm, SETRTS);          // Reset = 0
    EscapeCommFunction(hComm, CLRRTS);          // Reset = 1
    GetCommTimeouts(hComm,&TimeOuts);
    TimeOuts.ReadIntervalTimeout = 0;
    TimeOuts.ReadTotalTimeoutMultiplier = 1;
    TimeOuts.ReadTotalTimeoutConstant = 100;
    SetCommTimeouts(hComm,&TimeOuts);

    lineEditInfo->setText( "Searching for bootloader." );

    AutoStart = false;
    CancelStart = 0;
    ComOK = false;
    while ((AutoStart == false) && (CancelStart == 0))
    {
      qApp->processEvents();
```

```
  PurgeComm(hComm,PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
  SendBuff[0] = IDENT;
  WriteFile(hComm, SendBuff, 1, &Sended, NULL);    //send IDENT
  ReadFile(hComm, RecBuff, 1, &Received, NULL);    //receive IDACK
  if ((Received == 1) && (RecBuff[0] == IDACK)) AutoStart = true;
}
TimeOuts.ReadIntervalTimeout = 50;
TimeOuts.ReadTotalTimeoutMultiplier = 100;
TimeOuts.ReadTotalTimeoutConstant = 10;
SetCommTimeouts(hComm,&TimeOuts);
if (AutoStart && (!CancelStart)) ComOK = true;
if (ComOK)
{
  lineEditInfo->setText( "Writing, please wait!" );
  QFile HexFile(filename);
  HexFile.open(IO_ReadOnly );
  QTextStream stream1( &HexFile );
  if (stream1.atEnd()) ComOK = false;
  LineNumber = 0;

  while (ComOK && !EndOfRecord)
  {
    if (CancelStart) ComOK = false;

    line = stream1.readLine();
    LineNumber++;
    progressBarWrite->setProgress( (LineNumber*100) / NumLines );
    qApp->processEvents(10);
    if (line.length() != 0)
    {
      Data = (char *)line.ascii();

      if (Data[0] == ':')
      {
        if ((Data [7] == '0') && (Data [8] == '0'))
        {
          if (!Communication (WRITE, (unsigned char *)Data)) ComOK = false;
        }
        else
        {
          if ((Data [7] == '0') && (Data [8] == '1'))
          {
            EndOfRecord = true;           // End of File Record
          }
        }
      }
      else
      {
        ComOK = false;
        lineEditInfo->setText( "Hex file error!" );
      }
    }
    else
    {
      ComOK = false;
      lineEditInfo->setText( "Hex file error: Empty line" );
    }
  }
}
if (ComOK == true)
{
  if (Communication (DONE, (unsigned char *)Data))
  {
    progressBarWrite->setProgress(100);

    lineEditInfo->setText( "All OK!" );
```

```
        EscapeCommFunction(hComm, CLRDTR);          // trigger pin = 1
        EscapeCommFunction(hComm, SETRTS);          // Reset = 0
        qApp->processEvents();
        EscapeCommFunction(hComm, CLRRTS);          // Reset = 1
      }
      else
      {
        ComOK = false;
      }
      if (CancelStart)
      {
        lineEditInfo->setText( "Cancel of writing !" );
      }
      else
      {
        if (!ComOK)
        {
          lineEditInfo->setText( "Wrong writing !" );
        }
      }
      EscapeCommFunction(hComm, CLRDTR);          // trigger pin = 1
    }
    else
    {
      if (!CancelStart)
      {
        lineEditInfo->setText( "Timeout of communication !" );
      }
      else
      {
        lineEditInfo->setText( "Cancel of searching for bootloader." );
      }
    }
  }
  TimeOuts.ReadIntervalTimeout = 50;
  TimeOuts.ReadTotalTimeoutMultiplier = 0;
  TimeOuts.ReadTotalTimeoutConstant = 10;
  SetCommTimeouts(hComm,&TimeOuts);

  if(CancelStart!=2)  // not closing form
  {
    pushButtonBrowse->setEnabled(1);
    pushButtonWrite->setEnabled(1);
    comboBoxCOM->setEnabled(1);
  }
  return;
}

void FormBootLoader::Cancel()
{
  CancelStart=1;
}

void FormBootLoader::changePort()
{
  bool fSuccess;
  DCB dcbCommPort;
  char* SerLinka;
  DWORD BaudRateTransmission;
  COMMTIMEOUTS TimeOuts;

  CloseHandle(hComm);
  a=comboBoxCOM->currentText();
  SerLinka=(char*)a.ascii();
  hComm = CreateFileA(SerLinka,          //open port
    GENERIC_READ | GENERIC_WRITE,
```

```
    0,                                      //exclusive access
    0,                                     //no security attrs
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    0
    );
  if(hComm == INVALID_HANDLE_VALUE)
  {
    lineEditInfo->setText("Open port error!");
    return;
  }
  //set of parameters
  fSuccess = GetCommState(hComm, &dcbCommPort);
  if(!fSuccess)
  {
    lineEditInfo->setText("Read port parameters error!");
    return;
  }
  a=comboBoxBaud->currentText();
  dcbCommPort.BaudRate = (a.toDouble());
  dcbCommPort.ByteSize = 8;
  dcbCommPort.Parity = NOPARITY;
  dcbCommPort.StopBits = ONESTOPBIT;
  dcbCommPort.fBinary = 1;
  fSuccess = SetCommState(hComm, &dcbCommPort);     //write of parameters back
  if(!fSuccess)
  {
    lineEditInfo->setText("Write port parameters error!");
    return;
  }
  // Set timeouts of communication port short. Setting high timeout
  // introduces problems for console mode.
  // ReadFile seems to block outgoing communication
  TimeOuts.ReadIntervalTimeout = 50;
  TimeOuts.ReadTotalTimeoutMultiplier = 0;
  TimeOuts.ReadTotalTimeoutConstant = 10;
  TimeOuts.WriteTotalTimeoutMultiplier = 20;
  TimeOuts.WriteTotalTimeoutConstant = 1000;
  fSuccess = SetCommTimeouts(hComm,&TimeOuts);
  if(!fSuccess)
  {
    lineEditInfo->setText("Set communication timeout error!");
    return;
  }
  //clear buffers
  fSuccess = PurgeComm(hComm,
 PURGE_TXABORT+PURGE_RXABORT+PURGE_TXCLEAR+PURGE_RXCLEAR);
  if(!fSuccess)
  {
    lineEditInfo->setText("Clear buffers error!");
    return;
  }
}
```

Bibliography

[1] *RTAI Programming Guide 1.0*, September 2000.

[2] W. T. Ang, N. Riviere, and P. K. Khosla, "Design and implementation of active error canceling in handheld microsurgical instrument." in *Proc. IEEE/RSJ International Conference of Intelligent Robots and Systems*, 2001, pp. 1106–1111.

[3] P. J. Ashenden, *The VHDL Cookbook*, 1990.

[4] P. Berkelman, E. Boidard, P. Cinquin, and J. Troccaz, "Ler: the light endoscope robot," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems – IROS*, vol. 3, 27-31 Oct. 2003, pp. 2835–2840vol.3.

[5] D. Birkett, "Laparoscopy, what will the future hold?" *Minimally Invasive Therapy & Allied Technology*, vol. 10(6), pp. 259–260, 2001.

[6] M. Braun, "Surgical instrument," Patent EP1 486 172.

[7] J. D. Brown, J. Rosen, L. Chang, M. Sinanan, and B. Hannaford, "Quantifying surgeon grasping mechanics in laparoscopy using the blue dragon system," *Medicine Meets Virtual Reality*, vol. 13, pp. 34–36, 2004.

[8] G. Burdea, *Force and touch feedback for virtual reality*, I. John Wiley & Sons, Ed. Wiley Interscience Publication, July 1996.

[9] T. Butz and J.-P. Thiran, "Affine registration with feature space mutual information," *Lecture Notes in Computer Science*, vol. 2208, pp. 549–556, 2001.

[10] K. Byungkyu, L. Hun-Young, K. Kyoung-Dae, Y. Jeong, and J. Park, "A locomotive mechanism for a robotic colonscope." in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 1373–1378.

[11] F. Cepolina and R. Michelini, "Review of robotic fixtures for minimally invasive surgery," *International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 1, no. 1, pp. 43–63, 2004.

[12] S. Cotin, N. Stylopoulos, M. Ottensmeyer, P. Neumann, D. R. D., and S. Dawson, "Metrics for laparoscopic skills trainers: the weakest link!" in *Proc. MICCAI - LNCS 2488*, T. Dohi and R. Kikinis, Eds., MICCAI 2002. Berlin Heidelberg: Springer Verlag, 2002, pp. 35–43.

[13] P. Dario, M. C. Carrozza, M. Marcacci, S. D'Attanasio, B. Magnami, O. Tonet, and G. Megali, "A novel mechatronic tool for computer-assisted arthroscopy." *IEEE Trans Inf Technol Biomed*, vol. 4, no. 1, pp. 15–29, Mar 2000.

[14] P. Dario, M. C. Carrozza, and A. Pietrabissa, "Development and in vitro tests of a miniature robotic system for computer-assisted colonoscopy," *Computer Aided Surgery*, vol. 4, no. 1, pp. 1–14, January-February 1999.

[15] P. Dario, M. Carrozza, A. Pietrabissa, A. Magnani, and B. Lencioni, "Endoscopic robot," Patent US19 995 906 591, 1999-05-25.

[16] P. Dario, B. Hannaford, and A. Menciassi, "Smart surgical tools and augmenting devices," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 5, pp. 782–792, October 2003.

[17] S. D'Attanasio, O. Tonet, G. Megali, M. Carrozza, and P. Dario, "A semiautomatic handheld mechatronic endoscope with collision avoidance capabilities." in *Proc. IEEE International Conference of Robotics and Automation*, 2000, pp. 1586–1591.

[18] B. Davies, "Robotic devices in surgery," *Min Invas Ther & Allied Technol*, vol. 12, pp. 5–13, 2003.

[19] F. Focacci, M. Piccigallo, O. Tonet, G. Megali, and P. Dario, "Design of a hand-held robotic instrument for dexterous laparoscopic surgery," *Minim Invas Ther*, vol. 14, no. 4-5, pp. 308–9, 2005, abstract in Proc. 17th Conference of the Society for Medical Innovation and Technology - SMIT 2005.

[20] F. Focacci, M. Piccigallo, O. Tonet, G. Megali, A. Pietrabissa, and P. Dario, "Lightweight hand-held robot for laparoscopic surgery," in *IEEE International Conference on Robotics and Automation*, 2007.

[21] D. Gossot and G. Lange, "Deflectable and rotatable endoscopic instrument with intuitive control," *Minimally Invasive Therapy & Allied Technologies*, vol. 10, no. 6, pp. 295–299, November 2001.

[22] S. Gotoh, "Endoscopes with latest technology and concept," *Minimally Invasive Therapy & Allied Technologies*, vol. 12, pp. 222–226, 2003.

[23] G. S. Guthart and J. K. Salisbury, "The Intuitive^{TM} telesurgery system: overview and application," in *Proc. ICRA 2000*, 2000, pp. 618–621, daVinci.

[24] B. Hannaford, "Experimental measurements for specification of surgical mechanisms and understanding of surgical skill," Lecture notes of the European summer school on surgical robotics, Montpellier, France, Lecture Notes, September 2003.

[25] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[26] J. E. N. Jaspers, M. Bentala, J. L. Herder, B. A. de Mol, and C. A. Grimbergen, "Mechanical manipulator for intuitive control of endoscopic instruments with seven degrees of freedom," *Minimally Invasive Therapy & Allied Technologies*, vol. Volume 13, Number 3, pp. 191 – 198, June 2004.

[27] M. Kaneko, W. Paetsch, and H. Tolle, "Input-dependent stability of joint torque control of tendon-drivenrobot hands," in *Industrial Electronics, IEEE Transactions on*, vol. 39, no. 2, April 1992, pp. 96–104.

[28] K. Kim, D. Kim, K. Matsumiya, E. Kobayashi, and T. Dohi, "Wide fov wedge prism endoscope," in *27th Annual International Conference of the Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005.*, 01-04 Sept. 2005, pp. 5758–5761.

[29] G. S. Kubota, "Matching embedded pc boards to medical applications," *Medical Electronics Manufacturing*, 2001.

[30] F. Lai and R. D. Howe, "Evaluating control modes for constrained robotic surgery," in *Proc. ICRA 2000*, 2000, pp. 603–609.

[31] B. Lenaerts and R. Puers, "Inductive powering of a freely moving system," *Sensors and Actuators*, vol. 123–124, pp. 522–530, 2005.

[32] A. J. Madhani and J. K. Salisbury, "Wrist mechanism for surgical instrument for performing minimally invasive surgery with enhanced dexterity and sensitivity," Patent US5 797 900.

[33] N. Matsuhira, M. Jinno, T. Miyagawa, T. Sunaoshi, T. Hato, Y. Morikawa, T. Furukawa, S. Ozawa, M. Kitajima, and K. Nakazawa, "Development of a functional model for a master slave combined manipulator for laparoscopic surgery," *Advanced Robotics*, vol. 17, p. 523, 2003.

[34] A. Miller, P. Allen, and D. Fowler, "In-vivo stereoscopic imaging system with 5 degrees-of-freedom for minimal access surgery." *Stud Health Technol Inform*, vol. 98, pp. 234–240, 2004.

[35] R. Nakamura, T. Oura, E. Kobayashi, I. Sakuma, T. Dohi, N. Yahagi, T. Tsuji, M. Shimada, and M. Hashizume, "Multi-dof forceps manipulator system for laparoscopic surgery - mechanism miniaturized & evaluation of new interface," *Lecture Notes in Computer Science*, vol. 2208, p. 606, 2001.

[36] T. Nakao and A. Kashitani, "Panoramic camera using a mirror rotation mechanism and a fast image mosaicing," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 2, 7-10 Oct. 2001, pp. 1045–1048vol.2.

[37] D. Oleynikov, M. E. Rentschler, J. Dumpert, S. R. Platt, and S. M. Farritor, "In vivo robotic laparoscopy." *Surg Innov*, vol. 12, no. 2, pp. 177–181, Jun 2005.

[38] M. Palmieri, "Percezione del tempo e ammiccamento," Master's thesis, Università di Pisa, F.M.C., 2006.

[39] L. Phee, D. Accoto, A. Menciassi, C. Stefanini, M. Carrozza, and P. Dario, "Analysis and development of locomotion devices for the gastrointestinal tract." *IEEE Transaction Biomedical Engineering*, vol. 49, pp. 613–616, june 2002.

[40] C. Richards, J. Rosen, B. Hannaford, C. Pellegrini, and M. Sinanan, "Skills evaluation in minimally invasive surgery using force/torque signatures," *Surgical endoscopy*, vol. 14, pp. 791–798, 2000.

[41] J. K. Salisbury, W. T. Townsend, D. M. Dipietro, and B. S. Eberman, "Compact cable transmission with cable differential," Patent US4 903 536.

[42] M. Schurr, G. Buess, and K. Schwarz., "Robotics in endoscopic surgery: Can mechanical manipulators provide a more simple solution for the problem of limited degrees of freedom?" *Minimally Invasive Therapy & Allied Technology*, vol. 10, pp. 289–293, 2001.

[43] C. Stefanini, A. Manciassi, and P. Dario, "Modeling and experiments on a legged microrobot locomoting in a tubular, compliant and slippery environment," *International Journal of Robotics Research*, vol. 25, pp. 551 – 560, 2006.

[44] S. Takashi, N. Kazuo, F. Toshiharu, O. Soji, M. Yasuhide, K. Masaki, J. Makoto, and M. Nobuto, "Development of configuration differing master slave combined forceps," *Advanced Motion Control, 2004. AMC '04. The 8th IEEE International Workshop on*, pp. 227–232, March 2004.

[45] R. H. Taylor and D. Stoianovici, "Medical robotics in computer-integrated surgery," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 5, pp. 765–781, 2003.

[46] M. J. Tierney, T. G. Cooper, C. A. Julian, S. J. Blumerkranz, G. S. Guthart, and R. G. Younge, "Mechanical actuator interface system for robotic surgical tools," Patent US2 003 083 673.

[47] O. Tonet, T. U. Ramesh, G. Megali, and P. Dario, "Image analysis-based approach for localization of endoscopic tools," in *Prooceedings of Surgetica*, 2005.

[48] O. Tonet, F. Focacci, M. Piccigallo, F. Cavallo, M. Uematsu, G. Megali, and P. Dario, "Comparison of control modes of a hand-held robot for laparoscopic surgery." in *MICCAI (1)*, 2006, pp. 429–436.

[49] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 323–344, August 1987.

[50] G. J. M. Tuijthof, S. J. M. P. van Engelen, J. L. Herder, R. H. M. Goossens, C. J. Snijders, and C. N. van Dijk, "Ergonomic handle for an arthroscopic cutter," *Minimally Invasive Therapy & Allied Technologies*, vol. Volume 12, Numbers 1-2, pp. 82 – 90, March 2003.

[51] P. Viola and W. M. Wells III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.

[52] S. Voros, E. Orvain, P. Cinquin, and J.-A. Long, "Automatic detection of instruments in laparoscopic images: a first step towards high level command of robotized endoscopic holders," in *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, February 20-22, 2006, pp. 1107–1112.

[53] D. T. Wallace, C. A. Julian, T. A. Morley, and D. S. Baron, "Surgical tools for use in minimally invasive telesurgical applications," Patent US2 002 111 621.

[54] Y. Wang, D. R. Uecker, and Y. Wang, "A new framework for vision enabled and robotically assisted minimally invasive surgery," *Computerized Med. Imaging and Graphics*, vol. 22, pp. 429–437, 1998.

[55] G. Q. Wei, K. Arbter, and G. Hirzinger, "Real-time visual servoing for laparoscopic surgery. controlling robot motion with color image segmentation," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 16, no. 1, pp. 40–45, Jan.-Feb. 1997.

[56] H. Yamashita, N. Hata, M. Hashizume, and T. Dohi, "Handheld laparoscopic forceps manipulator using multi-slider linkage mechanisms," *Lecture Notes in Computer Science*, vol. 3217, 2004.

[57] Y. Yamauchi, J. Yamashita, Y. Fukui, K. Yokoyama, T. Sekiya, E. Ito, M. Kanai, T. Fukuyo, D. Hashimoto, H. Iseki, and K. Takakura, "A dual-view endoscope with image shift," in *Computer Assisted Radiology and Surgery*, 2002.

[58] P. Yedamale, "Brushless dc (bldc) motor fundamentals," Microchip Technology Inc., Tech. Rep., 2003.

[59] D. Yelin, I. Rizvi, W. M. White, J. T. Motz, T. Hasan, B. E. Bouma, and G. J. Tearney, "Three-dimensional miniature endoscopy," *Nature*, vol. 443, p. 765, October 2006.

[60] M. L. Young, L. Jinhee, P. Jisang, K. Byungkyu, P. Jong-Oh, K. S. Hyun, and H. Yeh-Sun, "A self-propelling endoscopic system." in *Proc. IEEE/RSJ International Conference on Intelligent Robot and Systems*, 2001, pp. 1117–1122.

[61] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.