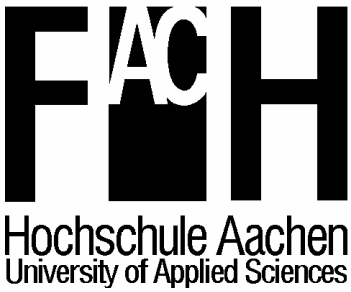


Fachhochschule Aachen

Fachbereich
Maschinenbau und Mechatronik

Mechatronik
Master of Sciences

Adaptive Pick & Place Operations with Sensor - controlled Robot

Rafael Omar Velazquez Lopez

Matr.-Nr.: 346074

Referent : **Prof. Dr.-Ing. Günther Starke**

Korreferent : **M. Sc. Thomas Kunkel**

August 2009

In collaboration with APS - European Centre for Mechatronics, Aachen, NRW.

DECLARATION

Hereby I declare that this thesis is my own work and that I did not use other sources than the literature and websites listed in the appendix.

Passages taken verbatim or analogously from published or non-published sources are denoted accordingly.

Pictures and drawings were created by myself or are referred to the used sources correspondingly.

This work was handed in neither in equivalent nor similar form at any other examination authority.

SECRECY

This thesis must not be copied, published or made available to a third party completely or in parts without permission of the author, the project supervisors or APS GmbH. Excepted from secrecy is the abstract.

ACKNOWLEDGMENTS

I would like to thank my family for all their invaluable support during the development of this Master Project.

I would like to thank Prof. Dr.-Ing. Günther Starke for all its trustful and M. Sc. Thomas Kunkel for its invaluable advices and help for the development of this project.

ABSTRACT

This thesis is part of the European research project “SOCRADES”, which was founded by the European Commission. The main topic is the collaboration of a robot and an integrated sensor by means of networking. The system to be developed has the objective to enable robots to perform handling operations like pick-up objects from batches and to place them on piles again without explicit programming of the Z-coordinate.

The objective of this thesis is to show the collaboration between the industrial robot SV3-X (from Motoman Company), a gripper tool and the laser sensor MQ-LA-S-AC240V (from MATSUSHITA Company) to perform handling operations. Therefore some handling procedures were developed, which describe the collaboration between the robots and the sensor in order to pick and place objects.

The software consists of several parts. The first is a graphical user interface (GUI) for the laser sensor control. This software is designed to allow external devices to communicate with the laser sensor by networking. To allow communication with the laser sensor an embedded system was designed that uses Microcontrollers and others electronic devices for configuration and data acquisition.

For the robot and the gripper there was no need to develop new software because it was already designed in a parallel project. It could be reused for the performance of the handling operations after some modifications. The modifications consist in adding some Object Classes with functions that allow the robot to reach different positions using point-to-point movement or following linear paths.

Finally to test the whole system with the purpose of show the correct collaboration between the robot, the gripper and the laser sensor a handling procedure will be made. The operation that will be performance is a piling sequence, which first will take the object A and then will place it over an object B.

Index

Index of tables

Index of figures

Abbreviations

PC	Personal Computer
I/O	Input/Output
LAN	Local Area Network
OO	Object Oriented
DC	Direct Current
GUI	Graphical User Interface
µC	Microcontroller
TCP	Transmission Control Protocol

1 Introduction

1.1 Purpose

Design and implement an embedded system which can pick and place objects using a robot (Motoman) and a sensor for object detection (Laser sensor).

Design and implement a PC graphical user interface (GUI), which can control the laser sensor and connect to other systems.

Design and implement a PC-GUI, which can control the robot movement allowing the system to develop the tasks of pick-up objects and place or piling them without explicit programming.

Figure 1.1 shows the relation between the different system that will be design and connected in order to fulfill the task of pick-up and place objects.

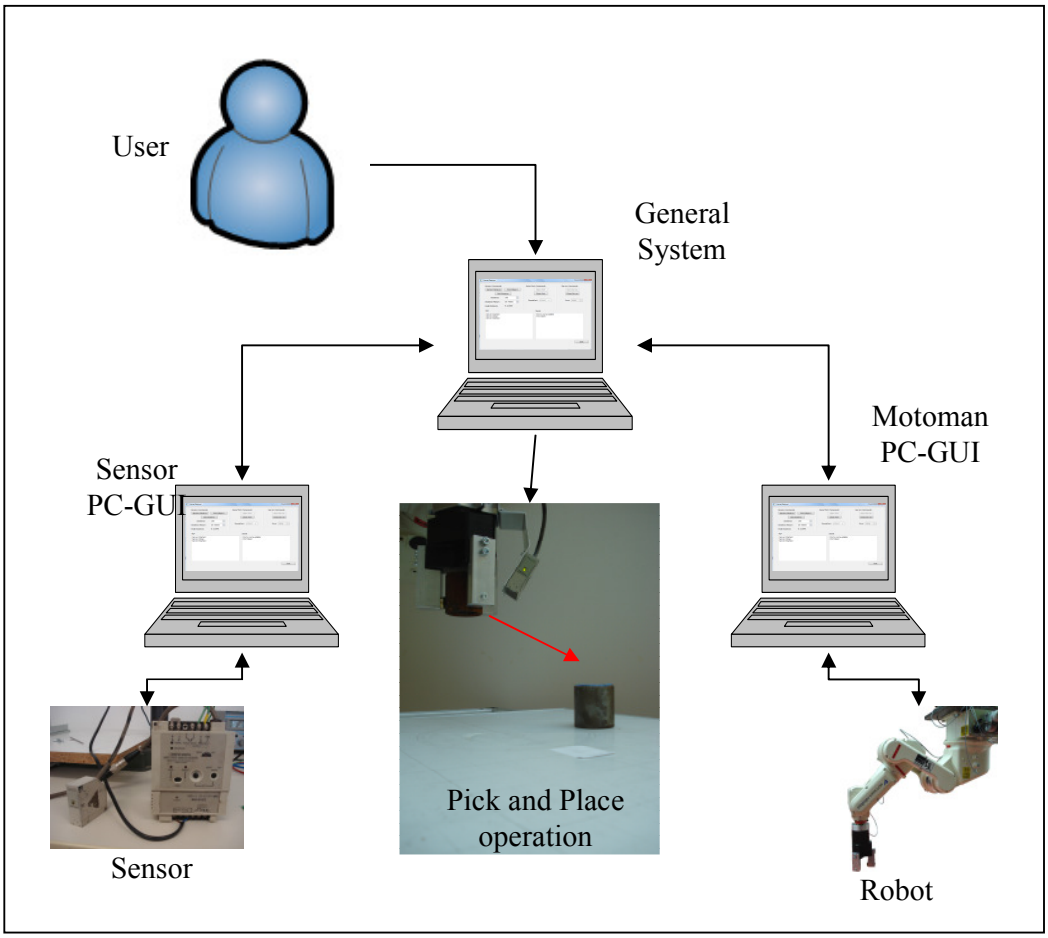


Figure 1.1 General relation of the System

1.2 Justification

Nowadays most of the industrial devices are develop in such way that they can make more than one task and also adapt to work with other devices. In that way the devices can be part of multiples systems with out have to change its configuration or design for a specific work. Because of this, more and more devices which can connect to networks are been designed.

The idea to develop this type of devices is to make easy the design of industrial systems because in that way the user only has to developed the application and not the control for each device.

At the same time the industrial systems are becoming more and more complicated, so the opportunity of have devices that can be controlled and accessed in an easy way becomes highly valuable. Because this allows the reduction of design time for such systems and as result the development becomes cheaper.

With the commonly use of robots in the industry a lot of different applications are been developed, one of this applications are the pick and place operations, however develop this operations normally use fixed programming. That means that the robot always will take and object from the same place and put it in another defined position. So, if the user wants to take an object from a different point the program for control the robot has to be changed and make that change means to spent money and time. Because of this a handling procedure that does not need explicit programming will be designed. The idea is to have a robot that can pick and place objects from different positions with out to have to program the position of the objects.

1.3 Organization of the thesis

Chapter two presents a brief introduction to sensors and measurement, microcontroller technology and robot programming process. Chapter three talks about the general pick-up and place object operations and the development of equations to move the robot. The chapter four talks about the PC Graphical User Interfaces develop. Chapter five covers the design of a microcontroller system for sensor data acquisition. Chapter six details the test used to check the performance of the whole system when all the pieces are put together. Finally, chapter seven presents conclusions and future possible extensions of the work.

2 Fundamentals

One of the most common tools used to find objects are the proximity sensors. And the way to know how much distance a tool must move to pick and object is to measure the distance between the object and the tool.

2.1 Proximity sensors

The first sensors used to detect nearby objects were the direct-contact tactile sensors. But, as these sensors only allow the detection when the object is touching the sensor, a lot of different tasks become impossible to solve. To solve this situation a new class of proximity sensors were developed. The objective of this new development was to extend the sensing range beyond that afforded by direct-contact sensors.

The proximity sensors are classified into several types in accordance with the specific properties used to initiate a switching action [MH]:

- Magnetic
- Inductive
- Ultrasonic
- Optical
- Capacitive

As result of constant research now a days is possible to find sensors that display high reliability characteristics, this make them well suited for operation in harsh or otherwise adverse environments, while providing high-speed response and long service lives. For all of these, proximity devices are valuable when detecting objects moving at high speed, when physical contact may cause damage, or when differentiation between metallic and nonmetallic items is required.

Magnetic, Inductive and Capacitive sensors are commonly used for short range detection. They relied on target presence to directly change some electrical characteristic or property (i.e., inductance, capacitance) associated with the sense circuitry itself.
As them depend in magnetic fields is no a good idea to use them for detect objects in a robot application. So the option is to use ultrasonic, optical o microwave sensors.

Ultrasonic proximity sensors

The ultrasonic proximity sensor is an example of a *reflective* sensor that responds to changes in the amount of emitted energy returned to a detector after interaction with the target of interest. Typical systems consist of two transducers (one to transmit and one to receive the returned energy), although the relatively slow speed of sound makes it possible to operate in the transceiver mode with a common transducer. The transmitter emits a longitudinal wave in the ultrasonic region of the acoustical spectrum (typically 20–200 kHz), above the normal limits of human hearing [MH].

This type of sensor is useful to detect objects over distances out to several feet. If an object enters the acoustical field, energy is reflected back to the receiver. The maximum detection range of ultrasonic sensors is dependent not only on emitted power levels, but also on the target cross-sectional area, reflectivity, and directivity.

The main advantage of these sensors over other type is the range detection, because for larger distances the power consumption is not too high. However, its main disadvantage is the way the ultrasonic wave is propagated. While the detection distance becomes larger, then the width of the detection zone becomes bigger. That makes troublesome try to locate small objects using ultrasonic sensors.

Optical proximity sensors

The next type of sensors is the optical ones. The work principle is to sense light levels.
Optical (photoelectric) sensors commonly employed in industrial applications can be broken down into three basic groups: (1) *opposed*, (2) *retroreflective*, and (3) *diffuse* (The first two of these categories are not really “proximity” sensors in the strictest sense of the terminology) [HM].

The work ranges vary from a few millimeters out to several hundred centimeters. Common robotic applications include floor sensing, navigational referencing, and collision avoidance. Industrial applications include sensing presence at a given maximum range (for counting, or to work on a part), sensing intrusion for safety systems, alignment, etc. As the optical sensors depend in light intensity levels, modulated near-infrared energy is typically employed to reduce the effects of ambient lighting, thus achieving the required signal-to-noise ratio for reliable operation. However as the infrared light is

invisible to the human eye visible-red wavelengths are sometimes used to assist in installation alignment and system diagnostics.

1) Opposed mode

The idea is to have two separate elements, a transmitter and a receptor. Each one is physically located on either side of the region of interest; the transmitter emits a beam of light, often supplied in more recent configurations by an LED that is focused onto a photosensitive receiver. Any object passing between the emitter and receiver breaks the beam, disrupting the circuit and creating a change in the sensor state.

Commonly called an “electric eye” at the time, the first of these categories was introduced into a variety of applications back in the early 1950s, to include parts counters, automatic door openers, annunciators, and security systems. Effective ranges of hundreds of feet or more are routinely possible and often employed in security applications [MH].

2) Retroreflective mode

These sensors evolved from the *opposed* variety through the use of a mirror to reflect the emitted energy back to a detector located directly alongside the transmitter. But as the mirrors normally need critical alignment, then Corner-cube retroreflectors were used instead of mirrors (see Figure 2.1), cutting down the critical alignment.

Corner-cube prisms have three mutually perpendicular reflective surfaces and a hypotenuse face; light entering through the hypotenuse face is reflected by each of the surfaces and returned back through the face to its source. A good retroreflective target will return about 3000 times as much energy to the sensor as would be reflected from a sheet of white typing paper (Banner, 1993). In most factory automation scenarios, the object of interest is detected when it breaks the beam, although some applications call for placing the retroreflector on the item itself [HM].

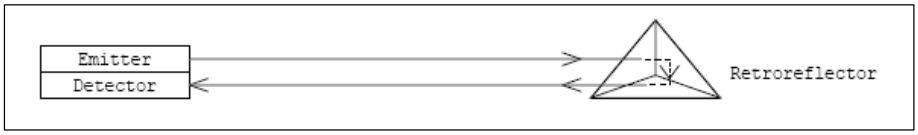


Figure 2.1 Corner-cube retroreflectors are employed to increase effective range and simplify alignment [HM].

3) Diffuse mode

Sensors in the diffuse category operate in similar fashion to retroreflective types. The difference is that energy is returned from the surface of the object of interest, instead of from a *co-operative reflector* (see Figure 2.2). This principle makes the random object detection an easy task to solve.

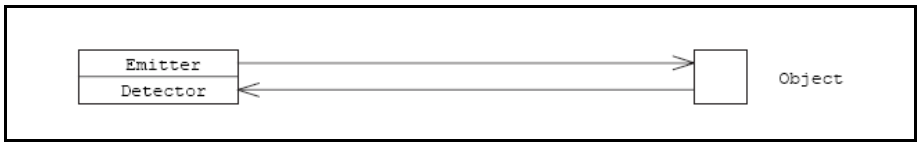


Figure 2.2 *Diffuse-mode proximity sensors* rely on energy reflected directly from the target surface [HM].

This type of sensor has several advantages over ultrasonic ranging for close-proximity object detection. There is no appreciable time lag since optical energy propagates at the speed of light, whereas up to a full second can be required to update a sequentially fired ultrasonic array of only 12 sensors. In addition, optical energy can be easily focused to eliminate adjacent sensor interaction, thereby allowing multiple sensors to be fired simultaneously. Finally, the shorter wavelengths involved greatly reduce problems due to specular reflection, resulting in more effective detection of off-normal surfaces. However, as happens with all the sensors, there are also some disadvantages, of course, is that no direct range measurement is provided, and variations in target reflectivity can sometimes create erratic results. Some variants that affect the reflectivity are the object color and the surface's texture.

Convergent Mode, diffuse proximity sensors can employ a special geometry in the configuration of the transmitter with respect to the receiver to ensure more precise positioning information. The optical axis of the transmitting LED is angled with respect to that of the detector, so the two intersect only over a narrowly defined region as illustrated in figure 2.3. It is only at this specified distance from the device that a target can be in position to reflect energy back to the detector. Consequently, most targets beyond this range are not detected. This feature decouples the proximity sensor from dependence on the reflectivity of the target surface and is useful where targets are not well displaced from background objects [HM].

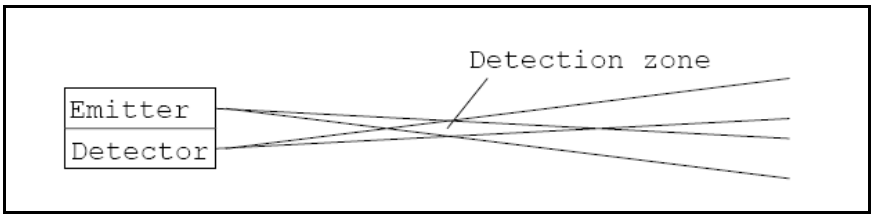


Figure 2.3 Diffuse proximity sensors configured in the convergent mode can be used to ascertain approximate distance to an object [HM].

However the problem with the convergent mode is that becomes impossible to know when the object is too far or too close to the object.

2.2 Distance measurement

For some applications (industrial, robotics, etc.) is not enough just to detect objects, also the distance to the object is important. To measure the distance from a reference point range sensors are used. A number of technologies have been applied to develop these sensors, the most prominent being light/optics, computer vision, microwave, and ultrasonic. As proximity sensors, range sensors may be of contact or noncontact types.
For pick-up and place operations the best sensors are the ones based in noncontact principles.

Time-of-Flight, Triangulation, or Field Based

There are many different classes and instances of noncontact ranging devices, but with very few exceptions they are based on one of the following three basic principles:
1. Energy propagates at a known, finite, speed (e.g., the speed of light, the speed of sound in air)
2. Energy propagates in straight lines through a homogeneous medium
3. Energy fields change in a continuous, monotonically decreasing, and predictable manner with distance from their source

The techniques associated with these basic phenomena are referred to as time-of-flight, triangulation, and field based, respectively [MISH].

Time-of-flight

(TOF) systems may be of the “round-trip” (i.e., echo, reflection) type or the “one-way” (i.e., cooperative target, active target) type. Round-trip systems measure the time taken for an emitted energy pattern to travel from a reference source to

a partially reflective target and back again. Depending on whether radio frequencies, light frequencies, or sound energy is used, these devices go by names such as radar, lidar, and sonar.

The One-way systems transmit a signal from a reference point and receive it at the target receptor or vice versa. In order to establish the time of flight some form of synchronizing both devices has to be present a both points.
TOF systems has the characteristic of its range resolution capability is based solely on the shortest time interval they can resolve, and not the absolute range being measured. So, if the object is near or far, the measurement error is basically constant.

Triangulation

Triangulation is based upon an important premise of plane trigonometry, which states that given the length of a side and two angles of a triangle, it is possible to determine the length of the other sides and the remaining angle.

Triangulation ranging systems are classified as either *passive* (use only the ambient light of the scene) or *active* (use an energy source to illuminate the target). Passive stereoscopic ranging systems position directional detectors (video cameras, solid-state imaging arrays, or position sensitive detectors) at positions corresponding to locations *P*1 and *P*2. Both imaging sensors are arranged to view the same object point, *P*3, forming an imaginary triangle. The measurement of angles θ and ϕ in conjunction with the known orientation and lateral separation of the cameras allows the calculation of range to the object of interest.

Active triangulation systems, on the other hand, position a controlled light source (such as a laser) at either point *P*1 or *P*2, directed at the observed point *P*3. A directional imaging sensor is placed at the remaining triangle vertex and is also aimed at *P*3. Illumination from the source will be reflected by the target, with a portion of the returned energy falling on the detector. The lateral position of the spot as seen by the detector provides a quantitative measure of the unknown angle ϕ, permitting range determination by the *Law of Sines*.

The performance characteristics of triangulation systems are to some extent dependent on whether the system is active or passive. Passive triangulation systems using conventional video cameras require special ambient lighting conditions that must be artificially provided if the environment is too dark.

Furthermore, these systems suffer from a correspondence problem resulting from the difficulty in matching points viewed by one image sensor with those viewed by the other. On the other hand, active triangulation techniques employing only a single detector do not require special ambient lighting, nor do they suffer from the correspondence problem. Active systems, however, can encounter instances of no recorded strike because of specular reflectance or surface absorption of the light.

Limiting factors common to all triangulation sensors include reduced accuracy with increasing range, angular measurement errors, and a *missing parts* (also known as *shadowing*) problem. *Missing parts* refers to the scenario where particular portions of a scene can be observed by only one viewing location (*P*1 or *P*2). This situation arises because of the offset distance between *P*1 and *P*2, causing partial occlusion of the target (i.e., a point of interest is seen in one view but otherwise occluded or not present in the other). The design of triangulation systems must include a tradeoff analysis of the offset: as this baseline measurement increases, the range accuracy increases, but problems due to directional occlusion worsen [MISH].

Field-Based Approaches

Whereas TOF and active triangulation techniques employ the wave propagation phenomena of a particular energy form, *field-based* approaches make use of the spatially distributed nature of an energy form.

The intensity of any energy field changes as a function of distance from its source. Moreover, fields often exhibit vector characteristics (i.e., directionality). Therefore, if the location of a field generator is known and the spatial characteristics of the field that it produces are predictable, remote field measurements contain information that may be used to infer distance from the source [MISH].

Laser-Based Active Triangulation Ranging and Range Imaging Sensors
Active Triangulation Basics

Figure 2.4 illustrates the basic active triangulation geometry. In this so-called “pinhole camera” model, practical aspects like lenses for projection and detection and mirrors for scanning are eliminated for clarity. It can be shown by means of similar triangles that the range is inversely proportional to the deflection of the imaged spot.

Where:

R= distance to object
b= baseline distance
f= lens to detector distance
u= detected spot position in the image plane

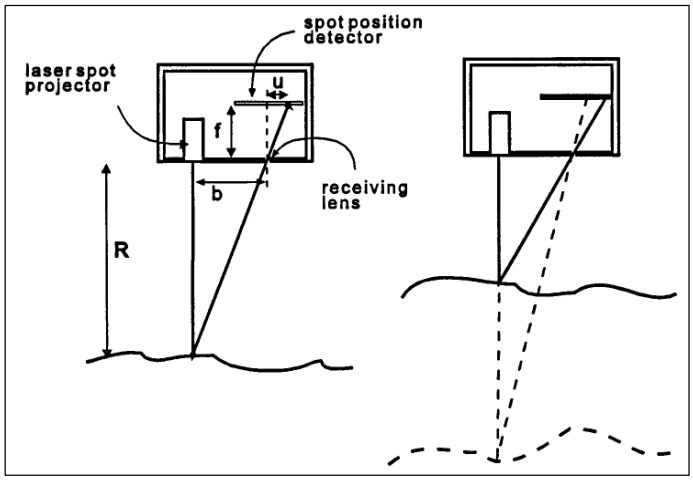


Figure 2.4 A simple pinhole camera model illustrates the basic active triangulation principle. As the distance *R* to the target surface changes, the spot position *U* on the detector changes, maintaining similarity between the large triangle outside the camera and the small triangle inside. There is an inverse relationship between *R* and *u [MISH]*

.

Based in the previews principles, for the object detection part, in the pick-up and place process, two different noncontact sensors were chosen. The first one was a light sensor (Sensor PZ-V31P,see Figure 2.5) and the second one was a laser sensor(Sensor MATSUSHITA MQ-LA-S-AC240V , see Figure 2.6).

Figure 2.5 Light Sensor

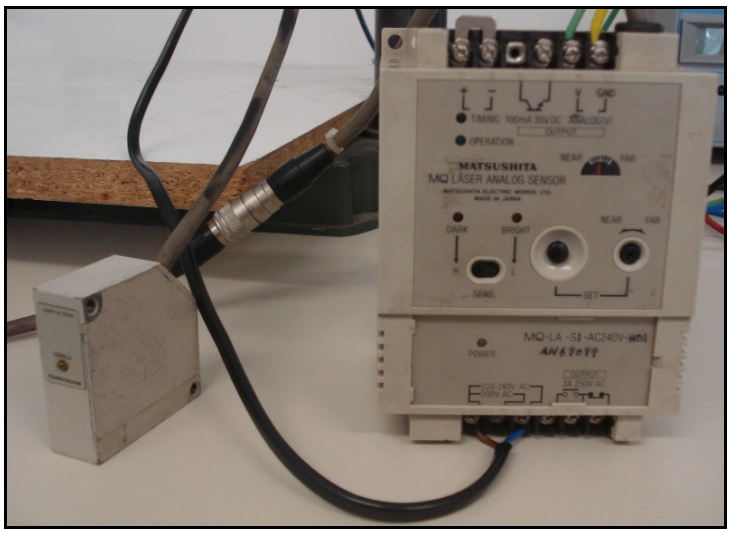


Figure 2.6 Sensor Laser with control and power unit.

The advantage of the light sensor is its low power consumption and its larger detection range (30cm), however its main disadvantage is the susceptibility to the external noise. So, working with the sensor in zones where the intensity of ambient light changes constantly causes to have errors in the object detection.

Another disadvantage is that this sensor only can detect objects, this is important because when the robot searches for any object, the robot is moving, so its difficult to predict the distance that will up to the object after the robot stops. The laser sensor has the advantage of can measure distances, however has to main disadvantages, the power consumption is higher than the power consumption of the light sensor and is detection range is shorter, only 14cm.

The laser sensor is no designed for object detection, however working with other devices is possible use it to detect objects. Also the laser sensor doesn't have problems when is measuring if there are light noise. Because of the versatility of the laser sensor, it was the option chosen for the project.

In chapter five is explained how the laser sensor is used to detect objects.

2.3 Microcontroller technology

The main advantage of the microcontrollers is its versatility. Because is possible to solve a lot of different control tasks just loading a different program, tasks as read sensors for data acquisition or convert a distance measurement sensor in a proximity sensor, which are the applications that will be used for this project.

2.3.1 Microcontroller characteristics

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhance RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed [DA16].

The microcontroller works with 5V as with 20mA power consumption other technical features of the ATmega16 are found in appendix A.5

Microcontroller Pinout:

The microcontroller ATmega 16 has 40 Pins. 32 of them are In/Out pins, and are divided in 4 ports.

- Port A: The 8 pins that build this port can be used as Analog inputs for Digital conversion, and also can be used as digital inputs or outputs.
- Port B: All pins can be set as Input or Output. This port has special pins for SPI-Bus (Serial Peripheral Interface) and one pin for external Interrupt INT2.
- Port C: All pins can be set as Input or Output. This port has special pins for JTAG.
- Port D: All pins can be set as Input or Output. Has 2 pins for external interrupts and 2 pins for USART communication.

The other 8 pins are:

- Pin 9. Reset. Is activated with 0V.
- Pin 10. VCC. Power Supply between 4.5 V and 5V.
- Pin 11. Ground.
- Pins 12 and 13. Pins for connect the Cristal Oscillator.

- Pins 32, 31 and 30. AREF, GND and AVCC. This pins are used for provide an analog voltage reference for the Analog Digital Converter (ADC).

The advantage of use a microcontroller over other devices is its special features. Features as ADC, USART, Interrupts, SPI-bus.
The ADC reads analog signals and converts them to a digital value, it is commonly use for data acquisition.
The USART and SPi-bus are used for communication with other devices, for example PCs.
The Interrupts are events that can be triggered by external voltage signals, or internal events as timers.

Figure 2.7 shows how is the pinout of the ATmega16.

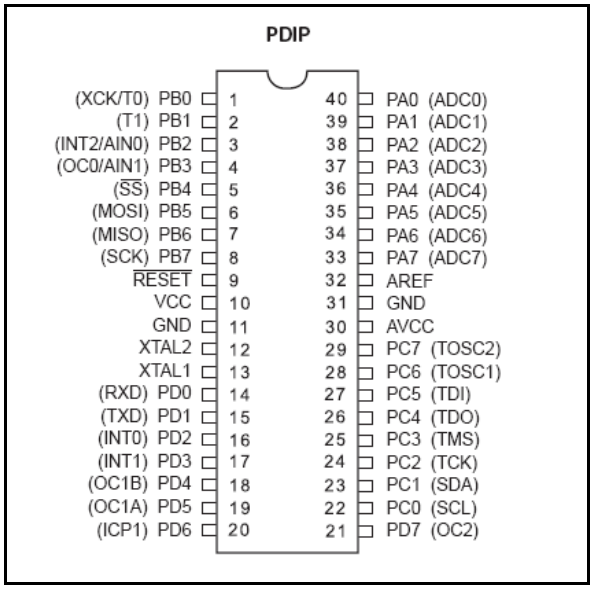


Figure 2.7 Pinout of ATmega 16 [DA16].

2.3.2 AVR Studio

The Software used to program the microcontrollers is the program AVR Studio from the ATMEL Company. The software allows the user to program using Assembler language or General C Code (GCC) language. Also with the help off some tools as the STK500 board the software allows to load the programs in the Microcontroller.

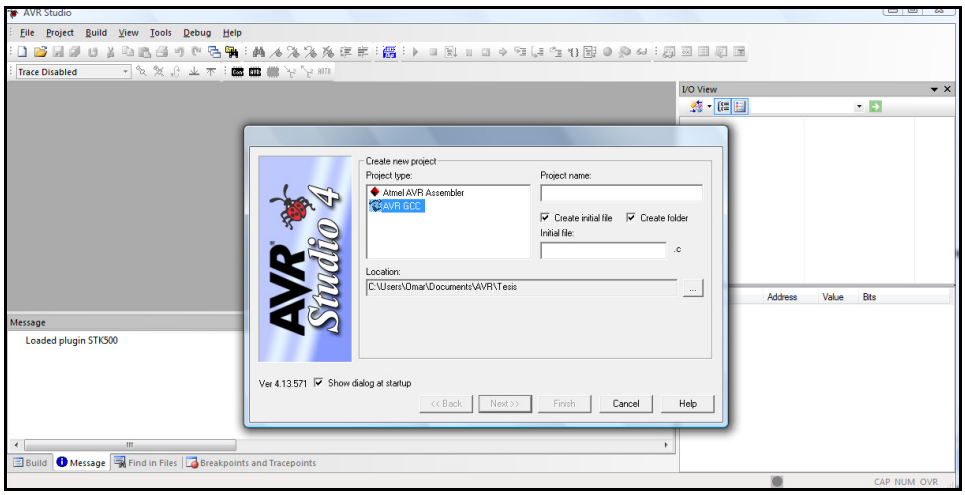


Figure 2.8 AVR Studio 4 user interface

2.4 Motoman

Nowadays there a lot of different robots in the industry, one of them is the Motoman (see figure 2.9). It has six rotational axes which kinematical structure is based on the principles of the Stanford Arm, which was designed by Victor Scheinman in 1969 in the Stanford Artificial Intelligence Laboratory.

One of the problems that had the commercial application of robots was the limited computing performance of PCs in the past, which made real time calculations of the inverse kinematical problem nearly impossible. Unlike the forward kinematical problem, the main different in these two calculations is that the forward calculation gives the tool pose from given joint angles, and the inverse kinematics calculation is non-unique giving more than one possible solutions of joint angles for one pose.

By Scheinman's arm solution this problem could be avoided to a large extend, since the positioning and orientating task was distributed of to upper and lower arm respectively. This could be achieved by a configuration of the kinematical chain in such a way that the axis of the lower arm intersect in one point, the so-called wrist centre point (WCP). For pose adjustment of the tool, the joint angles of the upper arm were set to position the WCP suitably, i.e. at the desired position of the tool centre point reduced by the vector describing the tool pose with respect to the WCP. The orientation was then adjusted by setting the joint angles of the lower arm. Thereby the inverse kinematical problem could be solved in closed form for the first time [MTT].

Figure 2.9 Motoman [MTT].

The most significant difference between the Stanford Arm and the industrial robots nowadays is the realization of the upper arm by three rotational joints. Scheinman's arm also included a translational axis. The upper arm of the Motoman features three rotational axes, one orientated vertically building the connection between the base and the first link and two horizontally orientated axes connecting the first with the second link and the latter with the third link respectively. The fourth axis is implemented in the third link, while the fifth axis is located at the WCP and is oriented orthogonally to the fourth axis. Tool flange and penultimate link are connected by the sixth axis, which is oriented orthogonally with respect to the fifth axis. The axes of the lower arm intersect in the WCP and thus build a so-called "in-line-wrist" [MTT].

Others characteristics of the Motoman are a total weight of 30 kg, a payload of 3 kg and an approximately spherical operating range of about 1.4 m in diameter. In appendix A.1 the Motoman's data sheet lists some technical data such dimensions, movement range, electrical data. The Motoman is controlled by an XRC control unit that can be fed with commands by a control panel or via a server.

The control unit features many functions, like for example definition of user specific reference systems, different motion types such as linear, joint or incremental movements as well as programming, just to mention some. For creation of programmes, the points lying on the path have to be taught in and the desired movement type and velocity have to be taught in. From the

programmes, the points on the path cannot be seen later on. Only current line, movement type and velocity are displayed. An example of a programme is given in figure 2.10 [MTT].

```
0000    NOP
0001    MOVJ VJ=50.00
0002    MOVJ VJ=12.50
0003    MOVL V=123
0004    END
```

Figure 2.10 Example program [MTT]

When the control unit is accessed via the server the number of features is reduced since not all functions were implemented yet. There are two main classes of functions defined by the manufacturer. “Status Read Functions” can be used for reading the robot status (current position, errors, servo status, etc.). “System Control Functions” are used for controlling the robot (running and interrupting programmes, load jobs, move the robot, etc.).

In figure 2.11 some examples for command lines are given. The first command line causes joint motion. The command structure differs from the commands used for automated jobs. Besides movement type and velocity the pose parameters, tool number and reference coordinate system have to be specified when the robot shall be moved. Movements are not defined by angles, but by pulses. The number of pulses causing rotations differs for each axis. Table 2.1 shows the number of pulses resulting in a 180-degree rotation for the six axes of the Motoman[MTT].

```
Pulse movement   Joint motion   Velocity: 10 deg/sec   Tool no.   Pulses for Motoman axes   Pulses for external axes

BscPMov          MOVJ           V 10                   7          100 200 300 400 500 600   0 0 0 0 0 0
BscIsLoc         1              (Get Pulses)
BscHoldOn                       (Set hold on; used for interupting programs)
BscHoldOff                      (Set hold off)
BscContinueJob                  (Start Job; execution starts from the current line of current job)
```

Figure 2.11 Some commands for operating the Motoman [MTT]

Axis	Pulses per 180 degrees
1st	240000
2nd	322000
3rd	242000
4th	155556
5th	162000
6th	102000

Table 2.1 Pulses causing rotations of 180 degrees for different axis [MTT]

For the current control application jus three commands are going to be used. The first command corresponds to the pulse movement, the second command reads the TCP position and a third one for read the robot's joint angles.

The commands are the follow ones:

BscPMov MOVJ V 1 0 0 0 0 0 0 0 0 0 0 0 0 0	Command for moving
BscIsLoc 1	TCP position
BscIsLoc 0	Joint angles value

In chapter four the use of these commands in the control task will be showed.

2.4 Gripper

The gripper is a tool designed to allow the Motoman to take objects. It has three fingers to take objects. Each one of the fingers is controlled by small electric coils. Then the gripper has only two states open and close.

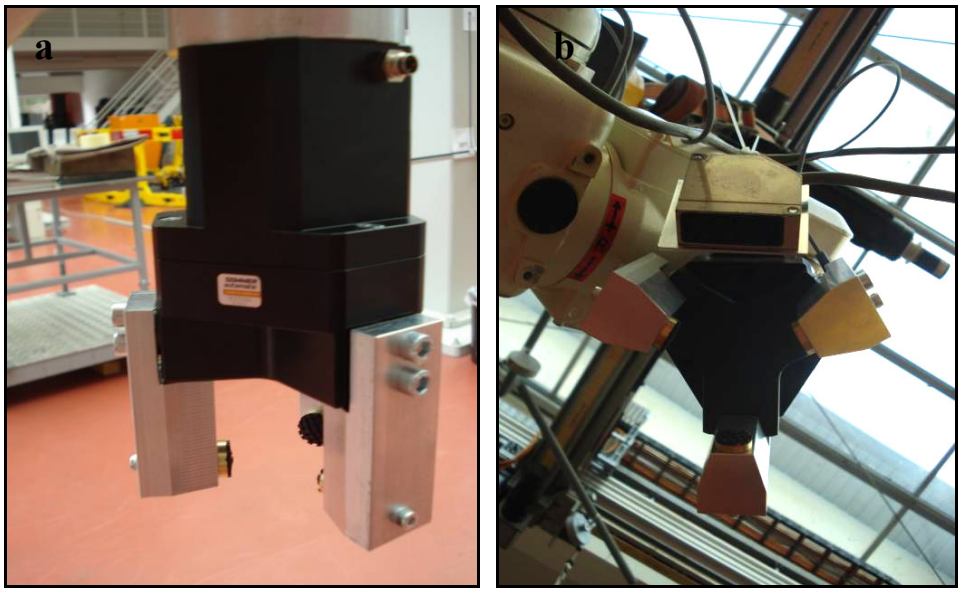


Figure 2.12 Gripper, a) Frontal view, b) bottom view.

3 Pick & Place operations

A smart robot is the one which can adjust its own behavior for solve different tasks. However in order to make a smart robot, first is necessary to develop a robot-program which can adapt to different situations.

3.1 Development of a robot program with adaptive functionality to run pick & place operations remotely controlled

The main purpose of the robot application is to pick an object from a place indicate by a user, and then place the object in another place also indicate by the user. This task has to be made using adaptive functionality. The adaptive functionality consists in implement a process which can adapt to different situations.

For this reason 2 main process sequences were developed. The first one is a procedure for pick up objects; the second one is a procedure for place objects. Also as result of the development of these two procedures a third process was develop, a procedure for piling objects.

3.1.1 Procedure for pick up and place objects

In order to pick an object a system between three devices (Robot, Distance Sensor and Gripper) has to be develop and also a sequence of steps has to be implemented.

The procedure implemented to pick up objects consists in a sequence of 8 steps, which are:

A) First the gripper's fingers are opened.

B) Then the robot moves to the position where the object is located. For move the robot the coordinates X and Y of the object are used. In that way the gripper is above the object (see Figure 3.1.A).

C) Search for the object. The search is made using a combination between the robot and the distance sensor. So, for find the object the robot moves down following a straight line. The movement stops when the distance sensor finds the object. In order for the sensor to find the object, the

sensor is continually measuring the distance between the gripper and the object, so when the distance between the object and the gripper becomes the distance value assigned by the user the object has been found (see Figure 3.1.B)

D) The distance between the gripper and the object is measured, that is because there is a delay when the sensor sends to the robot the stop signal, so when the robot stops, the distance between the robot and the object is different from the one that there were when the object was found. Then with the new distance value is possible to calculate the distance that the robot has to move to pick the object.

E) The robot calculates the distance to move.

F) The robot moves that value calculated.

G) The gripper closes around the object (see Figure 3.1.C).

H) Final step. With the object already picked the robot moves in a straight line to the position were the search movement starts (see Figure 3.1.D).

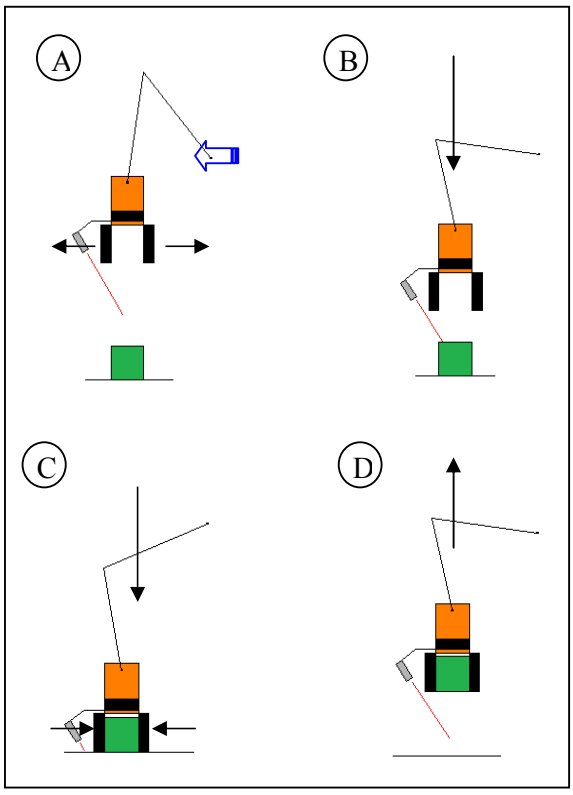


Figure 3.1 Pick up sequence.

The procedure for place an object is similar to the one used for pick the object. The sequence consists in seven steps, which are:

A) The robot moves to the coordinates X and Y where the object is going to be placed (see Figure 3.2.A).

B) Then the robot searches for the surface where the object is going to be placed. As in the pick sequence the robot moves down in a straight line. The movement stops when the sensor finds the surface (see Figure 3.2.B).

C) After found the surface the distance between the gripper and the object is measured.

D) The robot calculates the distance that has to move.

E) The robot moves the distance calculated.

F) Now the gripper's fingers are opened setting free the object.

G) Finally the robot moves back to the top position.

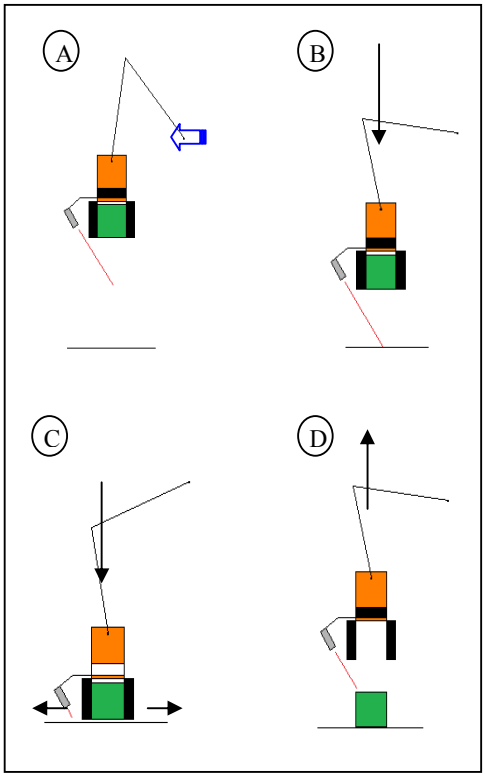


Figure 3.2 Place sequence

3.1.2 Procedure for Object Piling

The object piling procedure is a special case of the place procedure. The piling process is to put one object above another one, creating in that way levels of objects. The main difference between the place process and the piling process is, that in the place sequence the sensor searches for the surface where the object will be placed, and in the piling procedure the sensor searches for the object above the new object will be put.

Figure 3.3 shows the piling process.

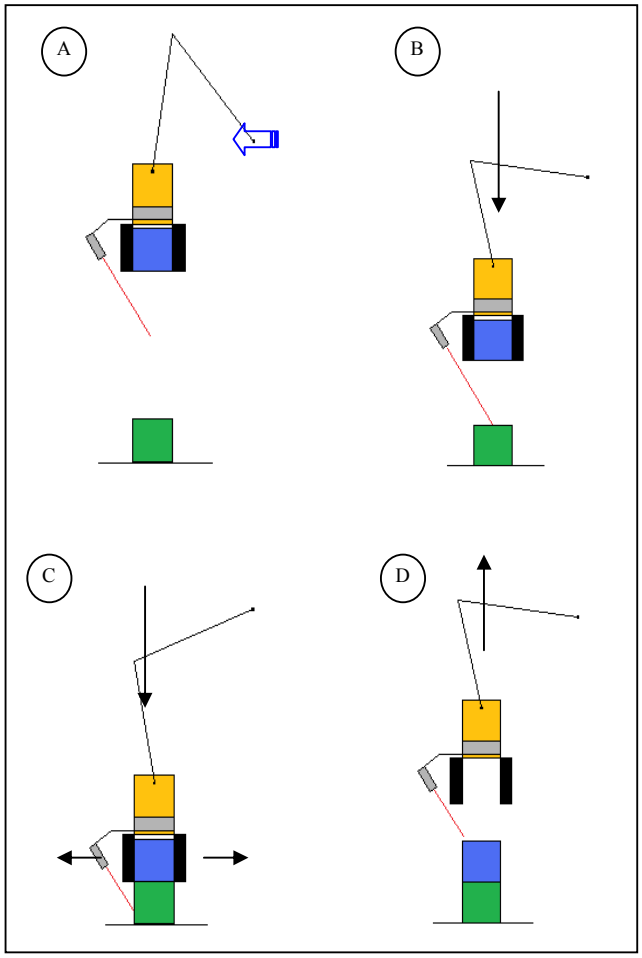


Figure 3.3 Piling Sequence

The seven steps for this process are the following:

A) The Robot moves to the coordinates X and Y, where the object, which will be the base, is located (see Figure 3.3.A)

B) The Robot searches for the object where the object picked is going to be placed. The robot moves down in a straight line. The movement stops when the sensor finds the base object (see Figure 3.2.B).

C) After found the object the distance between the gripper and the object is measured.

D) The robot calculates the distance that has to move.

E) The robot moves the distance calculated.

F) The gripper's fingers are opened setting free the object picked above the other object.

G) Finally the robot moves back to the top position.

Then, using combinations of these three different processes is possible to fulfill different task. For example, moving a pile of two objects (A and B) to another position, will need first to pick the object A from the original pile, then place that objet in the target zone where the pile is going to be. Now, the object B has to be picked and pile above the object A.

Figure 3.4 shows the complete sequence needed to pile the object B above the object A (next page).

To make possible the collaboration between the devices in order to fulfill the pick and place operation some software and hardware capable of control the devices has to be develop. The software that will be designed consist in one program capable to read the laser sensor in order to measure the distance to the objects and find them, also consist in a program capable of control the robot movements allowing it to take and place objects (see Chapter four). The hardware consists in an electronic circuit capable of read the sensor and sends the data to the PC. This circuit will be made using microcontrollers (see Chapter five).

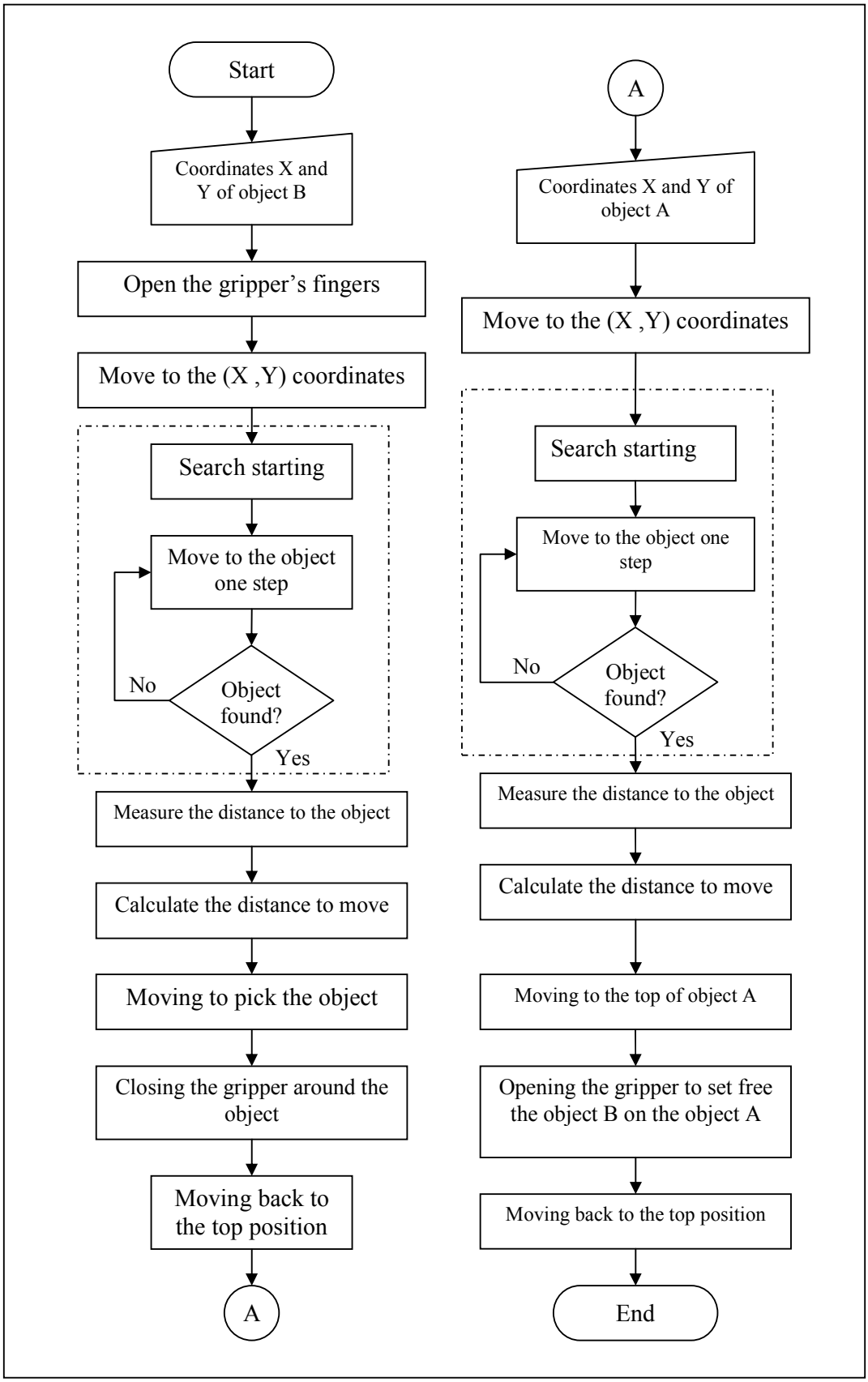


Figure 3.4 Example of a complete piling process

3.2 Development of robot movement equations

When the robot takes an object and then places it in a different position it has to move from one position to another position. There are two ways the robot can reach a position, point to point and following a continuous path. In point to point the only thing that matters is move from one position A to another position B, so the robot does not matter about how the movement is made. However when a continuous path is followed from position A to position B, the robot has to think how to make the movement.

The procedure that were developed for pick, place and piling objects makes the robot change from one position to another, and in order to have success the robot has to combine the point to point movement and the continuous path. When the robot moves to the coordinates X and Y of the object, it moves in point to point because the robot only needs to reach the position. But, when the robot moves searching an object, it has to move following a path, in this case a linear path.

However move following a path is the same that move from point to point, because a path is made following intermediate points between the positions A and B. The first thing the robot needs to know is how to move from one point to another point. To solve this, first a kinematical model has to be designed. The kinematical model was designed having two main considerations:

- Range of points that can be reached by the robot,
- Any possible object that can block the robot movement.

With these two considerations, two kinematical approaches were designed.

Figure 3.5 shows the first approach and figure 3.6 shows the second approach (next page).

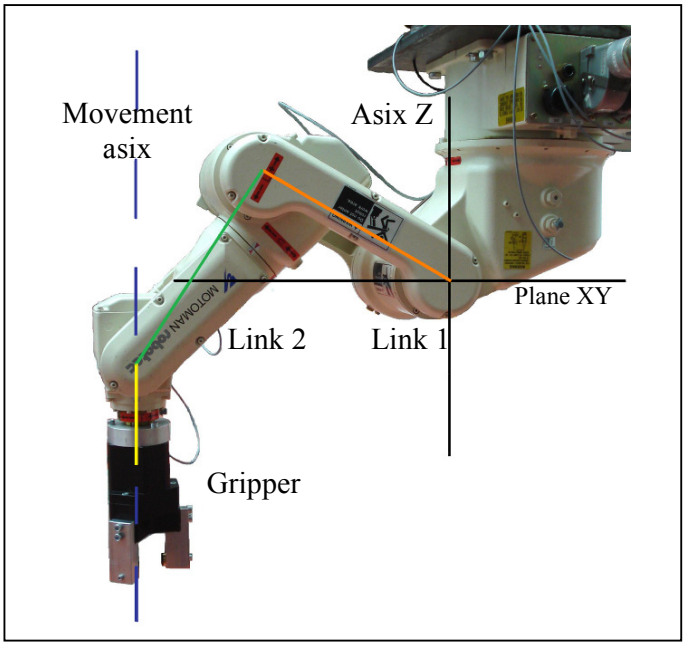


Figure 3.5 Kinematical model A

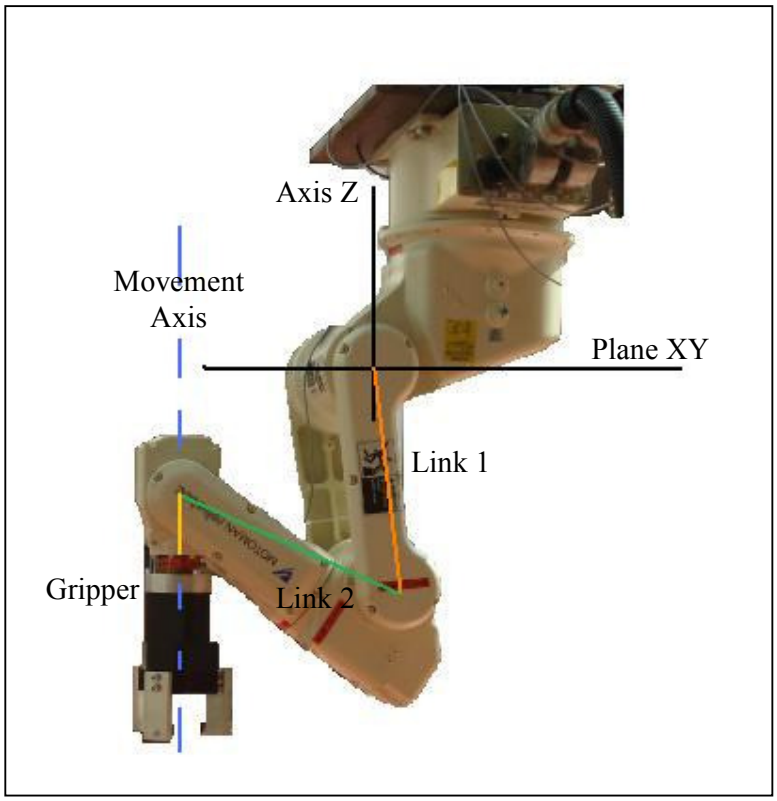


Figure 3.6 Kinematical model B

The advantage of use the approach A is that the robot can reach higher positions. The disadvantage is the robot needs more space to move, so working with other devices in the same work space becomes problematic.

The disadvantage of the approach B is the leak to reach higher positions. However its main advantage is that the robot does not need too much space to move as approach A needs, so working with other devices in the same work space is easier.

As one of the objectives of the project is work with other devices and as the difference between the points that can be reached with one or other model is not too critical, the approach B was chosen.

To reach a point a combination of six joint angles has to be calculated.

Table 3.1 shows the six joint angles.

Joint Angle	Name	Robot Axis
First	Alpha(α)	S (Turning/sweep)
Second	Beta(β)	L (Lower Arm)
Third	Epsilon(ε)	U (Upper Arm)
Fourth	Gamma(γ)	R (Wrist Roll)
Fifht	Theta(θ)	B (Bend/Pitch/Yaw)
Sixth	Zeta(ζ)	T (Wrist Twist)

Table 3.1 Robot Joint Angles

Changing the value of each of these angles allows the robot to reach any point in different ways. However that means that one point can be reached for different kinematical approaches. But as one kinematical model has been chosen, then there is just one combination of joint angles per each point that the robot wants to reach.

The kinematical model B has a special configuration where the angle Gamma always has a value of 0°. Then, the angles Theta and Zeta give the orientation to the tool (gripper). So, just three angles are used to reach the positions. These angles are Alpha, Beta and Epsilon.

The best way to explain the range of points the robot can reach is using cylindrical coordinates instead of Cartesian coordinates. That is because the robot work area has the shape of a sphere, however in order to calculate the movement along the Z axis is better to think in the work area as it was a cylinder. Cylindrical coordinates uses vector to describe any point. The vector has three components “radius”, “height” and “direction angle”. The radius indicates how distant is the point from the axis Z. The angle indicates the direction of the vector and the height indicates how far is the point from the X, Y plane.

From the Motoman’s datasheet is possible to watch that the Robot work area is more like a donut than a cylinder. A donut with height Z and where just the points between a radius=300 mm and a radius=677mm can be reached (see figure 3.7).

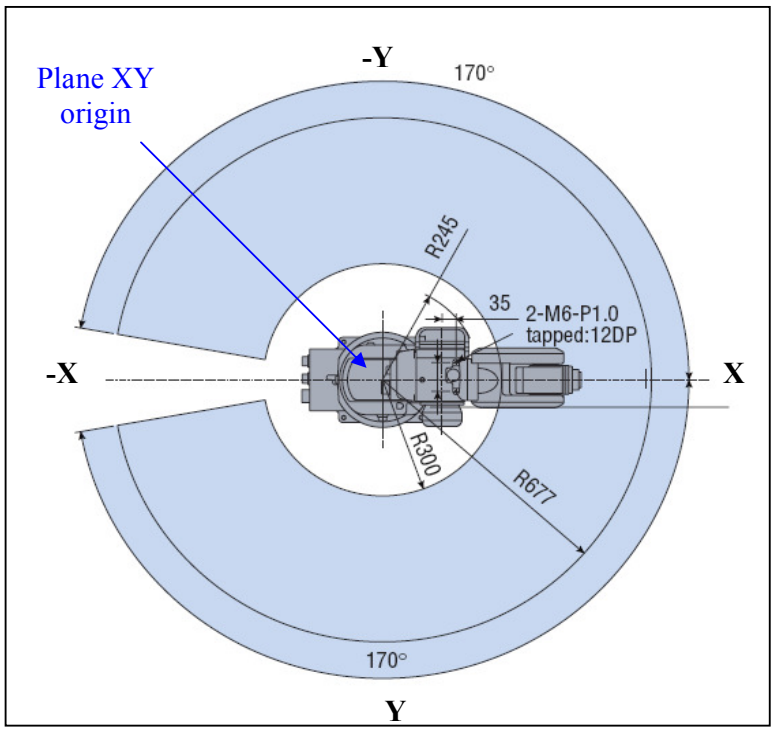


Figure 3.7 Motoman’s work area, plane XY

For the robot the way to understand these limits is converting them to joint angles values, this means that there is a maximum motion range for each joint angle (see Table 3.2). Figure 3.8 shows the reference systems for measure each of the main angles (Alpha, Beta, Epsilon and Theta).

Name	Maximum Motion Range	Robot Axis
Alpha	±170°	S (Turning/sweep)
Beta	+150°/-45°	L (Lower Arm)
Epsilon	+190°/-70°	U (Upper Arm)
Gamma	±180°	R (Wrist Roll)
Theta	±135°	B (Bend/Pitch/Yaw)
Zeta	±350°	T (Wrist Twist)

Table 3.2 Joint angles maximum motion range

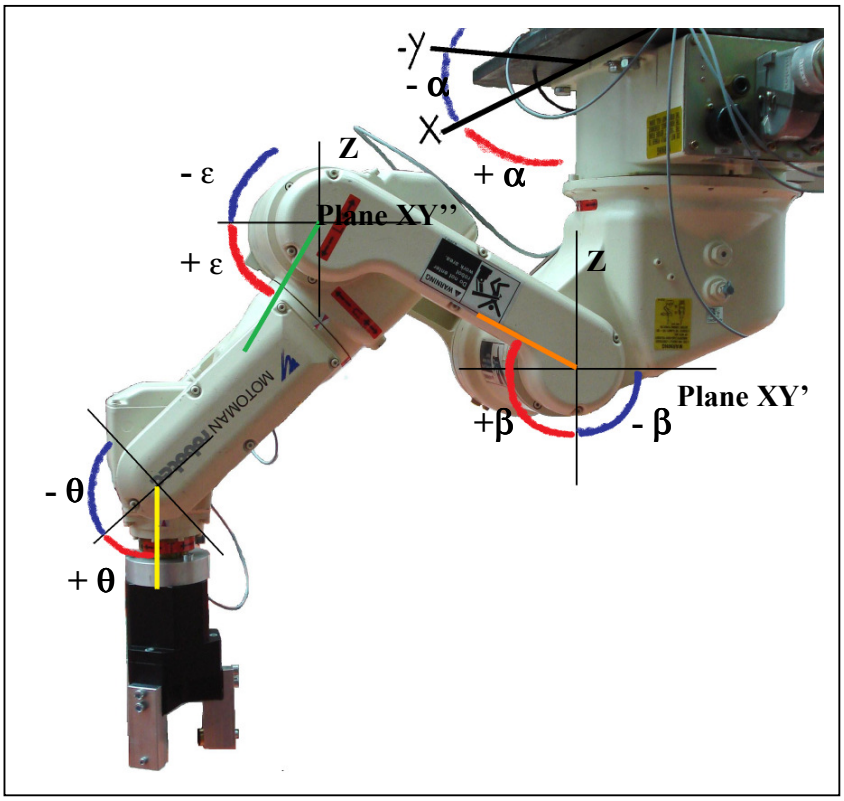


Figure 3.8 Reference systems to measure the joint angles

Using the kinematical model B six sets of equations were derived. The equations allows to predict which points are inside the motion range of the robot. The first two sets of equations calculate the joint angles for the points that are above the limit of a radius of 416.8 mm. The other four sets of equations calculate the joint angles of the points that are inside an area limited by a small radius of 300 mm and a big radius of 416.8 mm.

Each of these two groups follows the next conditions. In the two first sets of equations the conditions are:

a) Epsilon is smaller than 0° and the object distance is bigger than the length of the Link 2(assign in Figure 3.6) (266.8 mm) (see Figure 3.9).

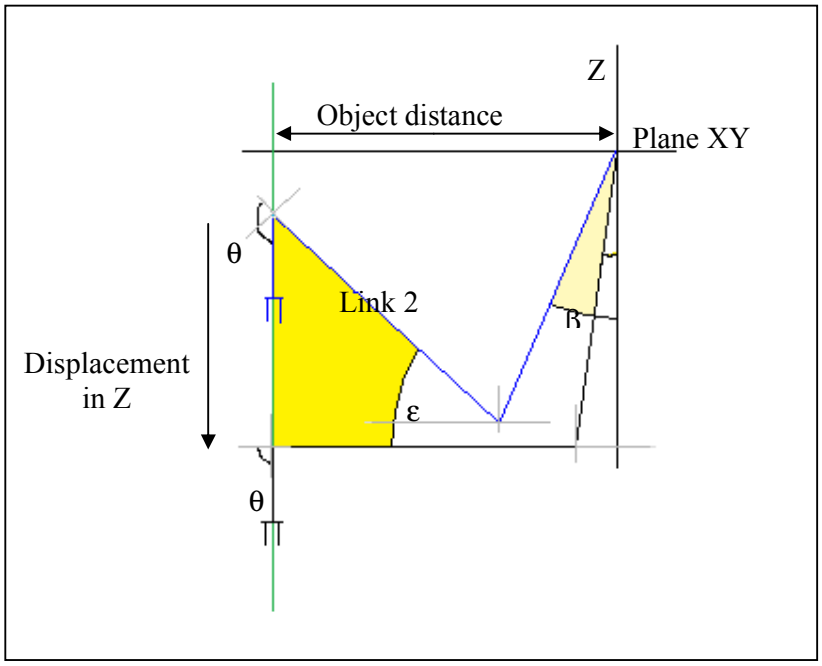


Figure 3.9

b) Epsilon is bigger than 0° and the object distance is bigger than the length of the Link 2 (266.8 mm) (see Figure 3.10).

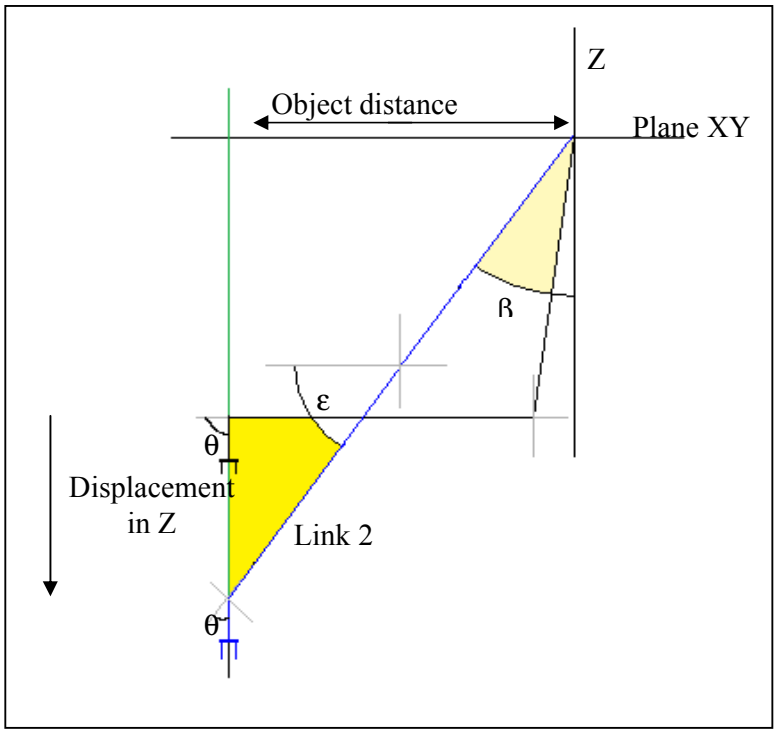


Figure 3.10

For the remaining fourth sets of equations the conditions are:

c) Beta is bigger than 0°, Epsilon is smaller than 0° and the object distance is smaller than the length of the Link 2 (266.8 mm) (see Figure 3.11).

.

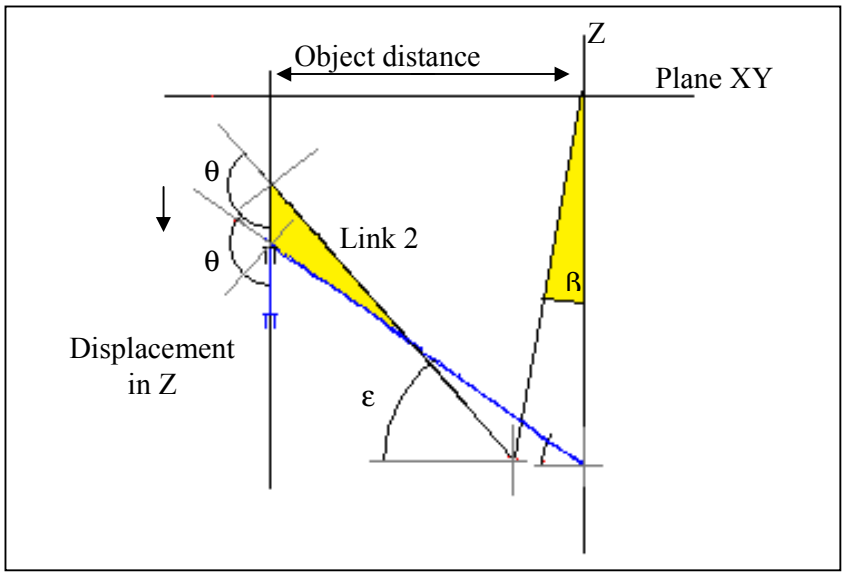


Figure 3.11

d) Beta is smaller than 0°, Epsilon is smaller than 0° and the object distance is smaller than the length of the Arm2 (266.8 mm) (see Figure 3.12).

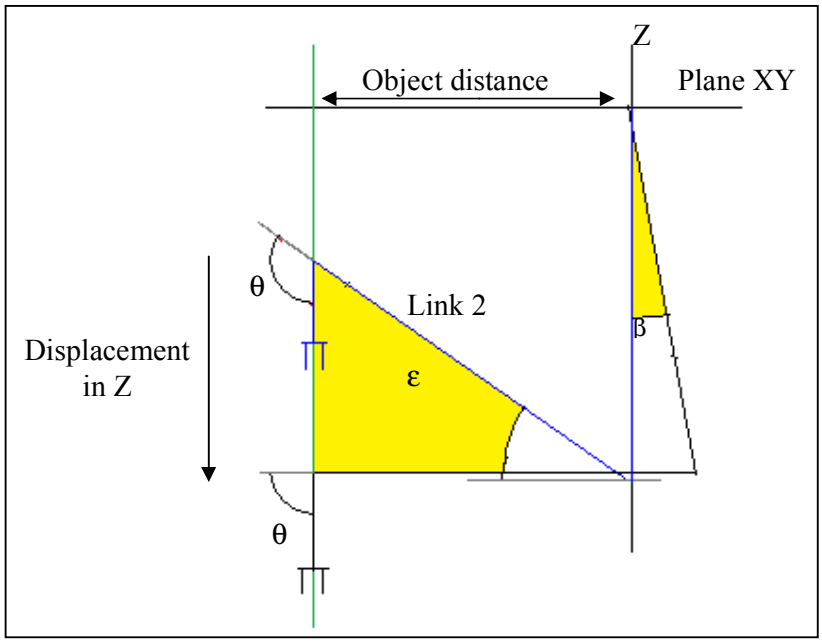


Figure 3.12

e) Beta is smaller than 0°, Epsilon is bigger than 0° and the object distance is smaller than Link 2 (266.8mm) (see Figure 3.13).

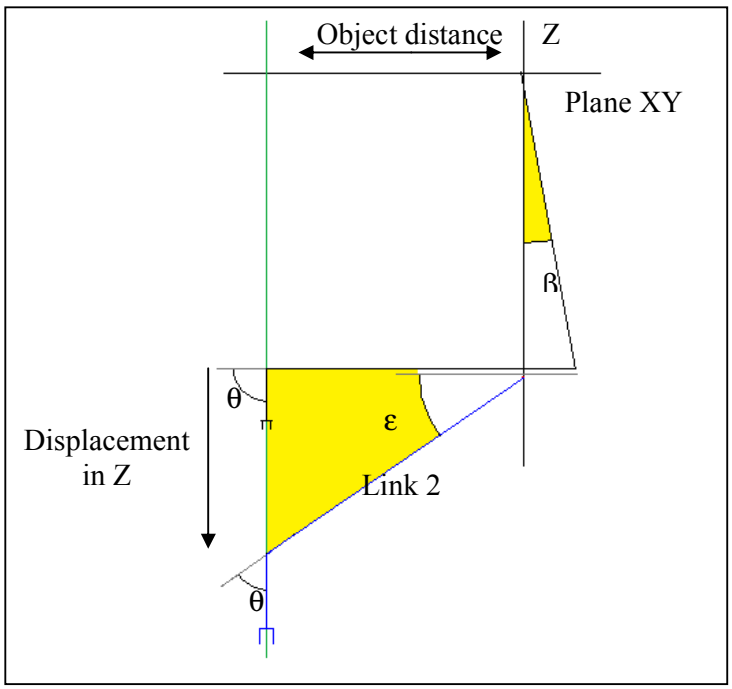


Figure 3.13

f) Beta is bigger than 0°, Epsilon is bigger than 0° and the object distance is smaller than Link 2 (266.8 mm) (see Figure 3.14).

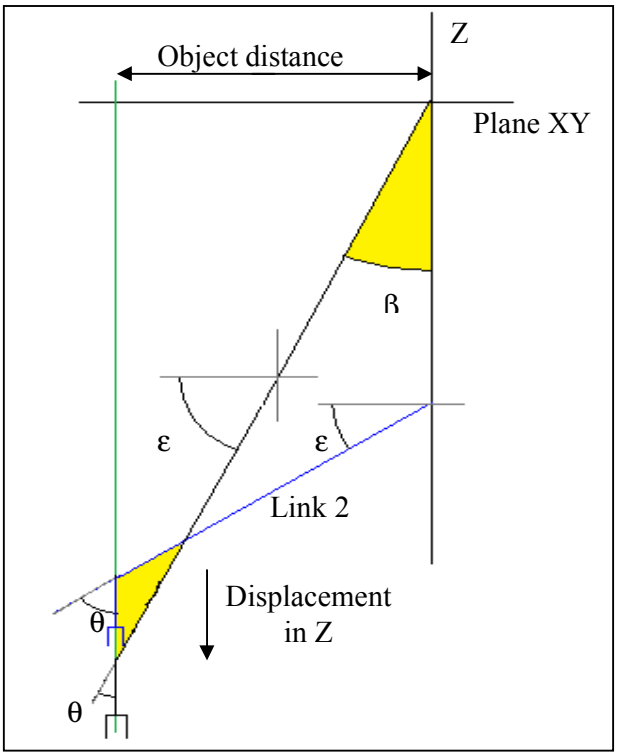


Figure 3.14

The sets of equations were used to solve the problem of find the robot joint angles for any X , Y and Z value gave, where X and Y define the distance between the point to reach and the robot's center (see Figure 3.7). The equations were deduced from a trigonometric model of the kinematic model (see Figure 3.15).

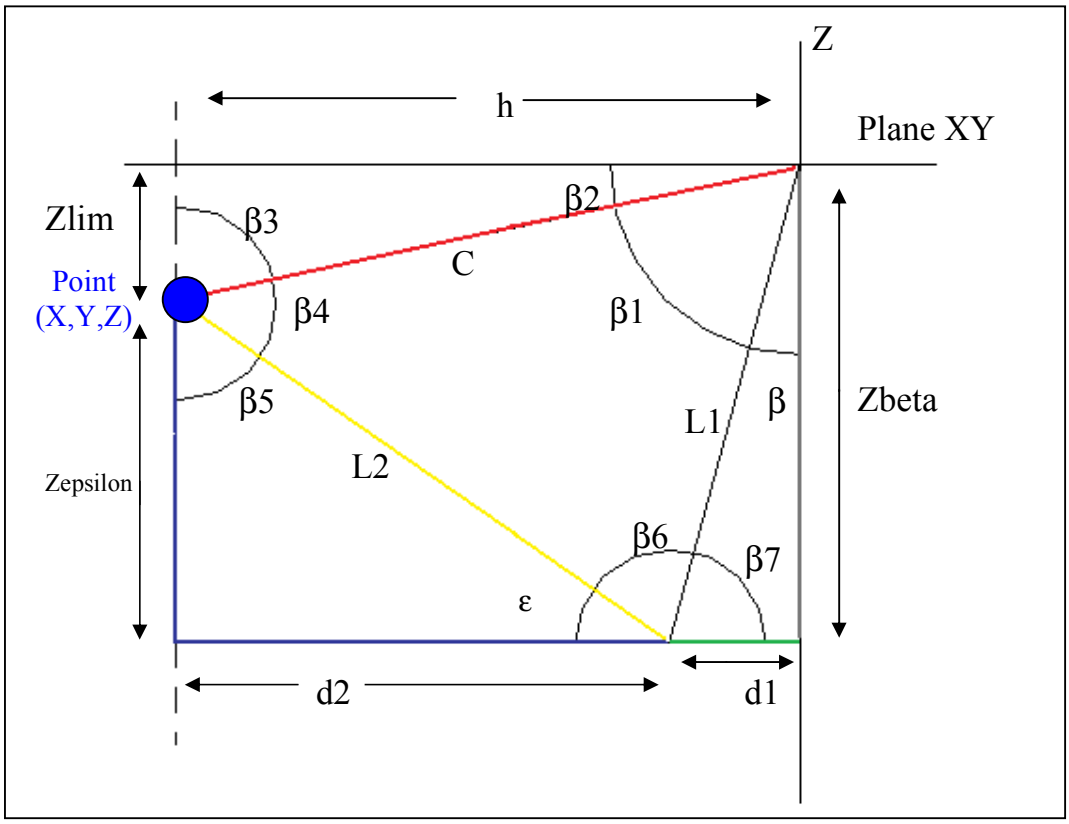


Figure 3.15 Trigonometric model

Where:

L1	length of the Link 1
L2	simplify length of the Link 2
zlim	limit of the minimum value of Z in order to avoid collisions
h	distance to the robot's center point
β	joint angle Beta
ε	joint angle Epsilon
Z	coordinate Z of the point (X, Y, Z)

from X and Y coordinates the value of h is calculated (see 1):

$$h = \sqrt{(x^2 + y^2)} \qquad (1)$$

from Z and h the value of angle β1 and β2 are found:

$$c = \sqrt{(h^2 + Z^2)} \qquad (2)$$

$$\beta 2 = \tan^{-1}(Z / h) \qquad (3)$$

$$\beta 1 = \cos^{-1} ((L1^2 + c^2 - L2^2)/(2 * L1 * c)) \qquad (4)$$

Then, the set of equations that allows to calculate the joint angles for each part of our kinematic model are:

a) First set

$\beta = 90 - \beta 1 - \beta 2$ (5)

$\beta 6 = \cos^{-1}((L1^2 + L2^2 - c^2)/(2 * L1 * L2))$ (6)

$\beta 7 = 180 - 90 - \beta$ (7)

$\varepsilon = 180 - \beta 6 - \beta 7$ (8)

$\varepsilon = -\varepsilon$ (9)

$\theta = -\varepsilon + 90 + 12.994$ (10)

$\varepsilon = \varepsilon - 12.994$ (11)

b) Second set

$\beta = 90 - \beta 1 - \beta 2$ (12)

$\beta 6 = \cos^{-1}((L1^2 + L2^2 - c^2)/(2 * L1 * L2))$ (13)

$\beta 7 = 180 - 90 - \beta$ (14)

$\varepsilon = 180 - (360 - \beta 6 - \beta 7)$ (15)

$\varepsilon 1 = 180 - \varepsilon - 90$ (16)

$\varepsilon = \varepsilon - 12.994$ (17)

$\theta = \varepsilon 1 + 12.994$ (18)

c) Third set

$\beta = 90 - \beta 1 - \beta 2$ (19)

$\beta 6 = \cos^{-1}((L1^2 + L2^2 - c^2)/(2 * L1 * L2))$ (20)

$\beta 7 = 180 - 90 - \beta$ (21)

$\varepsilon = 180 - \beta 6 - \beta 7$ (22)

$\varepsilon = -\varepsilon$ (23)

$\theta = -\varepsilon + 90 + 12.994$ (24)

$\varepsilon = \varepsilon - 12.994$ (25)

d) Fourth set

$\beta = 90 - (180 - \beta 1 - \beta 2)$ (26)

$d1 = \sin(\beta) * L1$ (27)

$d2 = h + d1$ (28)

$\varepsilon = \cos^{-1}(d2 / L2)$ (29)

$\theta = \varepsilon + 90 + 12.994$ (30)

$\varepsilon = -\varepsilon - 12.994$ (31)

e) Fifth Set

$\beta = 90 - (180 - \beta 1 - \beta 2)$ (32)

$d1 = \sin(\beta) * L1$ (33)

$d2 = h + d1$ (34)

$\varepsilon = \cos^{-1}(d2 / L2)$ (35)

$\varepsilon 1 = 180 - \varepsilon - 90$ (36)

$\theta = \varepsilon 1 + 12.994$ (37)

$\varepsilon = \varepsilon - 12.994$ (38)

f) Sixth set

$\beta = 90 - \beta 1 - \beta 2$ (39)

$\beta 6 = \cos^{-1}((L1^2 + L2^2 - c^2)/(2 * L1 * L2))$ (40)

$\beta 7 = 180 - 90 - \beta$ (41)

$\varepsilon = 180 - (360 - \beta 6 - \beta 7)$ (42)

$\varepsilon 1 = 180 - 90 - \varepsilon$ (43)

$\varepsilon = \varepsilon - 12.994$ (44)

$\theta = \varepsilon 1 + 12.994$ (45)

Note, the equations used a calculated length value for the L2 different from the real value of Link 2. That is because the reference axis for Link 2 is displaced 60mm from the reference axis of Link 1 (see Figure 3.16, next page).

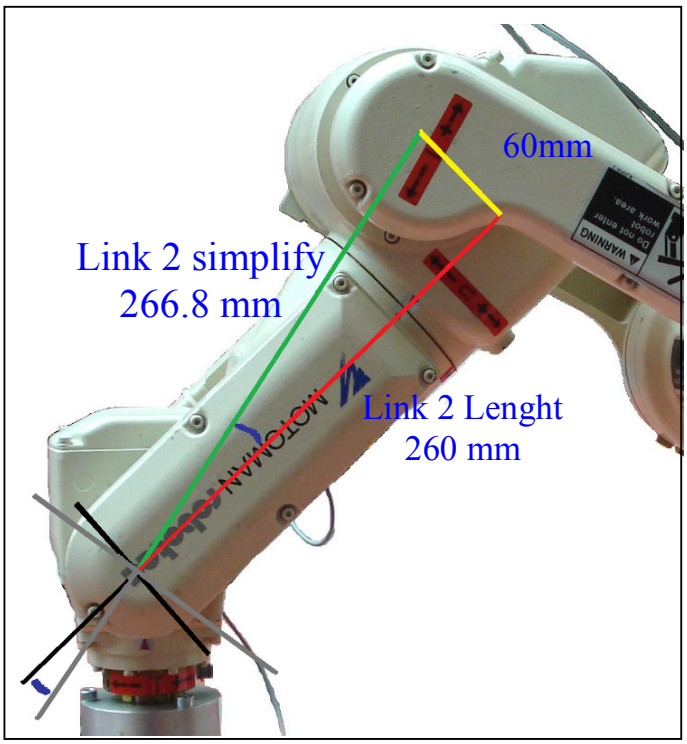


Figure 3.16

Then in order to simplify the equations of the kinematical model, the original reference axis system was displaced 12.99°, that displacement gave origin to a new value for Link 2 length.

$$L2 = \sqrt{((60mm)^2 + (260mm)^2)} = 266.8mm \quad (46)$$

With that displacement the functions to calculate the joint angles become easier to solve. So, finally the only thing to make is add ob subtract the displacement of 12.99° to the Epsilon angle calculated. This step is also included in each of the different equations sets.

To calculate the joint angle Alpha and the joint angle Zeta the next equations are used:

when Y is bigger than 0

$$\alpha = \cos^{-1}(X/h) \qquad (47)$$

otherwise

$$\alpha = -(\cos^{-1}(X/h) \qquad (48)$$

and

$$\zeta = \alpha \qquad (49)$$

As was explained the functions calculate the joint angles form the X, Y and Z values gave, however from any point gave try to find the correct value of Epsilon is not to easy, because in a initial calculation the value or Beta can falls in any of the six models. Then to determine which set of equations has to be applied to find the joint angles the next conditions are used:

- $h > L2$?

If h is bigger than L2 then the first and second sets are applied, otherwise the other four sets are used.

- $Z > L1$?

If Z is bigger than L1 then the first, third and fourth sets are applied, otherwise second, fifth and sixth sets are used.

- $\beta1 + \beta2 > 90$?

If $\beta1$ plus $\beta2$ is bigger than 90 fourth and fifth sets are applied, if not, then the remaining four sets are used.

So, if in the start calculation process h is bigger than L2, Z is bigger than L1 and $\beta1 + \beta2 < 90$ then the first set will be used to calculate the joint angles.

With the six equations seems that any point can be calculated, however there are some points that can not be calculated and other points that can be calculated but can not be reached by the robot, because of its design. To avoid the calculation of such points the next conditions have to be fulfilled:

- c is lower than L1+L2
- joint angles values must be inside their motions ranges (see Table 3.2)

The process to calculate the joint angles for a point (X, Y, Z) is as follows (see Figure 3.17).

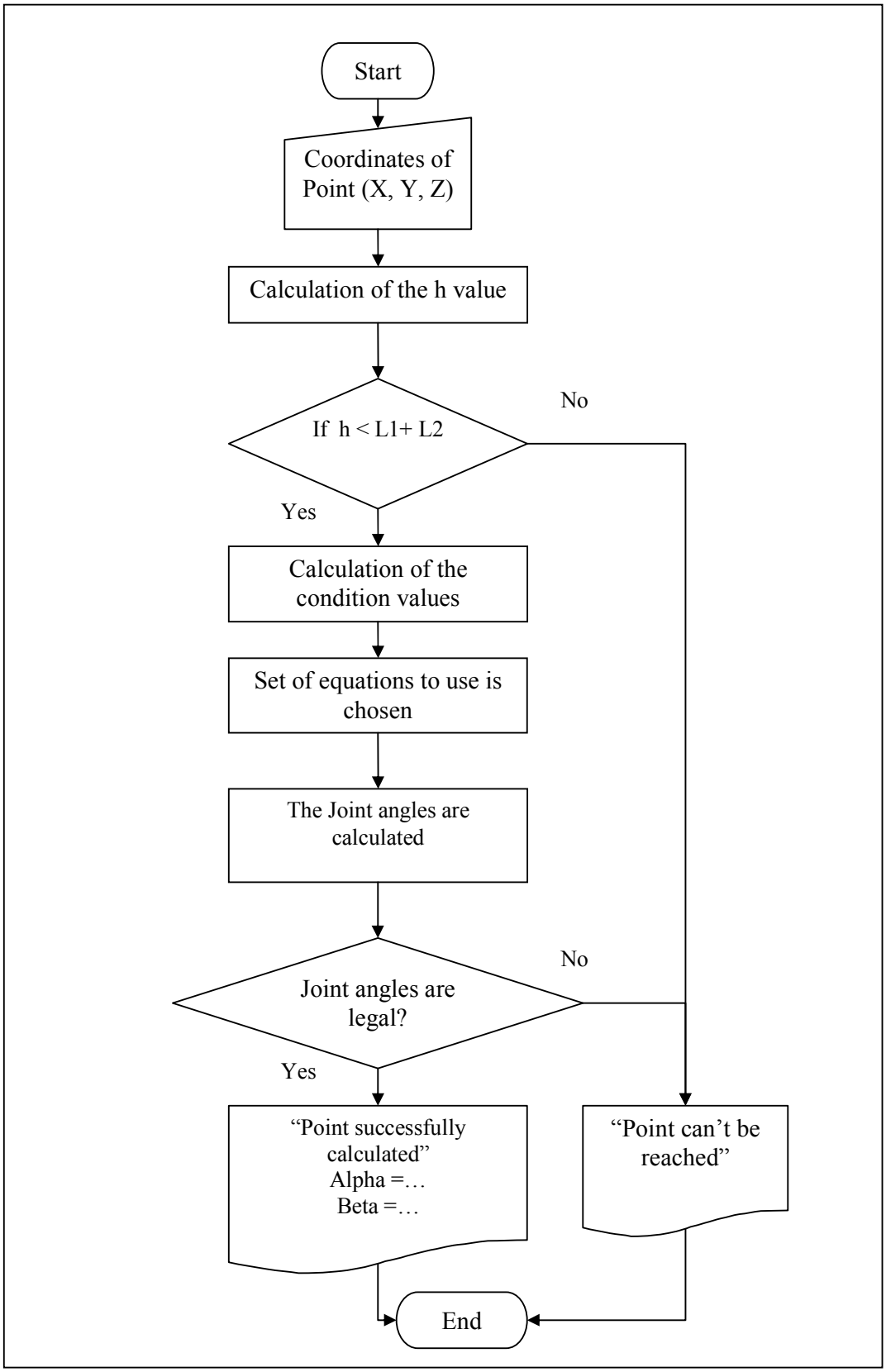


Figure 3.17 Joint angles calculation process

With the equations of joint angles for any point already deduced the last step to make the robot moves for one object to another object is to choose how the robot will move between the objects. The easiest way is move point-to-point, this means that only the joint angles for the final position has to be calculated and this can be made with the equations already deduced. However there are some times when the robot has to move following a path between one point and another point. This is the case when the robot wants to find, take or place and object. For these operations the robot must move in a linear path as was explained in chapter 3.1.

The way to make a lineal path is using linear interpolation between the origin point P1 and the end point P2. Therefore the line is divided in any number of points (P3,P4,P5,P6,P7,..,Pn). Then the robot instead of move from P1 to P2 will move from P1 to P3, then from P3 to P4, from P4 to P5, P5 to P6, and so on until finally move from Pn to P2. In this way the robot has moved from P1 to P2 following a linear path (see Figure 3.18).

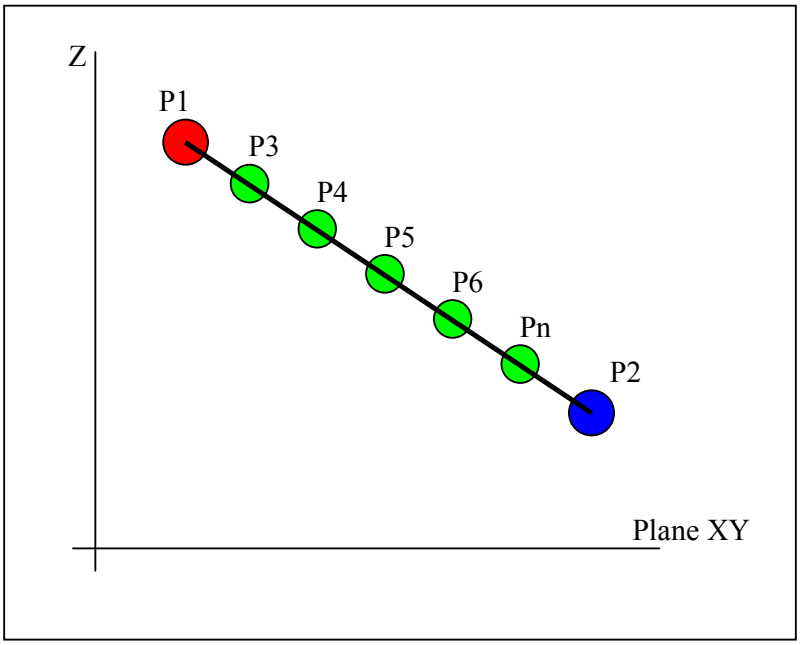


Figure 3.18 Linear path

The procedure use to calculate the points to follow a linear path is the next one:

- From P1(x1, y1 ,z1) and P2(x2, y2, z2) a V1 is calculated (see formula 50), then the size of the vector is determined (see formula 51)

$$V1 = P1 - P2 \quad (50)$$

$$d=\sqrt{(v1^2 + v2^2 + v3^2)} \quad (51)$$

- With d the number of points needed to reach P2 is calculated. As the coordinates of the points are given in millimeters, the value of d is also in millimeters. The resolution chose for determine the amount of points is the 5mm per point. Then the amount of points is calculated with formula 46.

$$ap = (d / 5) \qquad (52)$$

- Now the coordinates of the intermediate points are determined (see formula 53).

$$P_n = P_{n-1} + (V1 / ap) \qquad (53)$$

- the joint angles for each point are calculated.

- finally the P2 joint angles are calculated

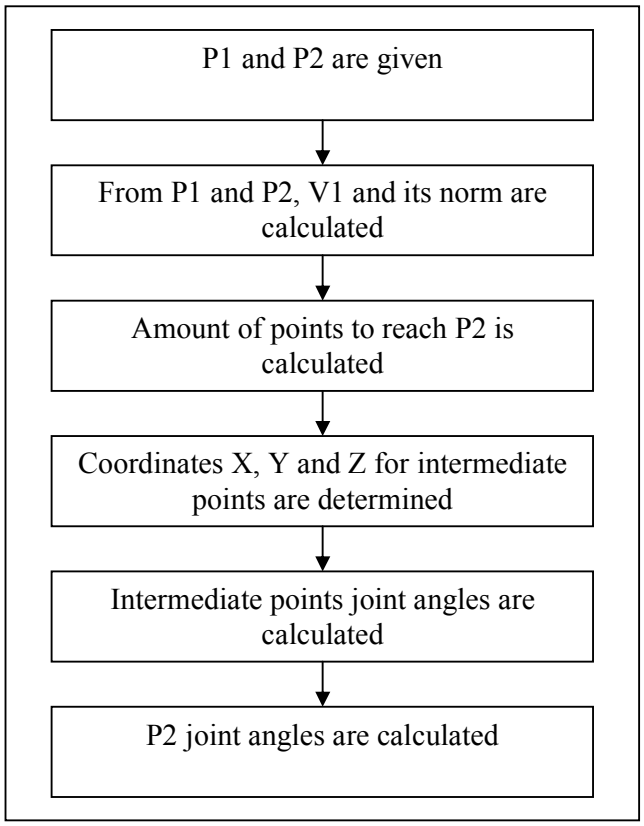


Figure 3.19 Linear path calculation process

4 Development of Graphical User Interfaces

To calculate the robot's joint angles for multiple points by hand normally will take some minutes or even some hours. So, the use of a PC to calculate these values is a good idea because in that way the calculation of points is made automatically and fast. To do this and to do the pick up and place operations some PC-GUIs were developed. These PC-GUIs are one PC-GUI to interact with the laser sensor and another one to interact with the Motoman.

4.1 Laser sensor control PC-GUI

The laser sensor control PC-GUI is designed to fulfill three tasks. The first one is establish a connection with a network in order that other devices or remotely users can access to the laser sensor functions. The second one is to provide local users with enough control to configure the sensor data acquisition system. And the third one is allows the communication between the PC-GUI and the laser sensor device(Figure 4.1).

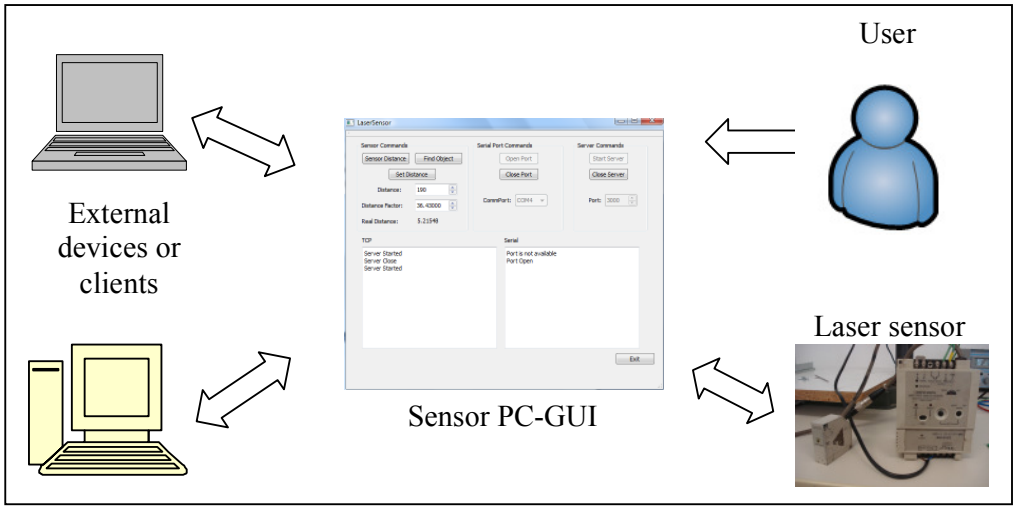


Figure 4.1 Communication diagram

a) Network Connection

The communication with other devices is made by a server, which use Transmission Control Protocol (TCP), which is a protocol for internet communication. When a device (client) connects to the laser sensor program, the client can access to the laser functions sending some commands that are listening by the server. These commands are the following:

- Sensor_Distance
- Find_Object
- Open_port
- Close_port

The first and second commands are used to get direct information from the laser sensor. The first command asks for the actual value of the sensor, so when this command is received, the PC-GUI sends a signal to the sensor to measure the distance to the object. When the distance measurement value is received from the laser sensor, this value is transmitted to the client whom asks for it.

The second command asks the sensor to starts a search procedure. This command has two possible answers from the sensor, the first is a confirmation telling that the object was found and the second is an error message telling that the laser sensor is off.
The third and four commands are used to control the state of the serial port of the PC where the laser sensor is connected.

b) User control interface

The user interface provides the user with control over the laser sensor software and the laser sensor functions. Also shows to the user the information that is received and transmitted to other clients and the information that is received and transmitted to the laser sensor (see Figure 4.2, next page).

The user can control three things. The first is the server. The user can start or close the server connection and change the server port number. The second is the serial CommPort. As with the server case the CommPort can be closed or opened and also selected from a list of CommPorts. The third thing is the laser sensor. From the user interface the sensor distance can be asked and a search procedure can be started. There are two things that can be configured from here, the minimum distance that has to be between the sensor and objects before the sensor can say that an object was found, and a distance factor that helps to find the real value of the distance measurement. This factor is important because the sensor gives a digital value between 0 and 255 and that value has to be converted to distance.

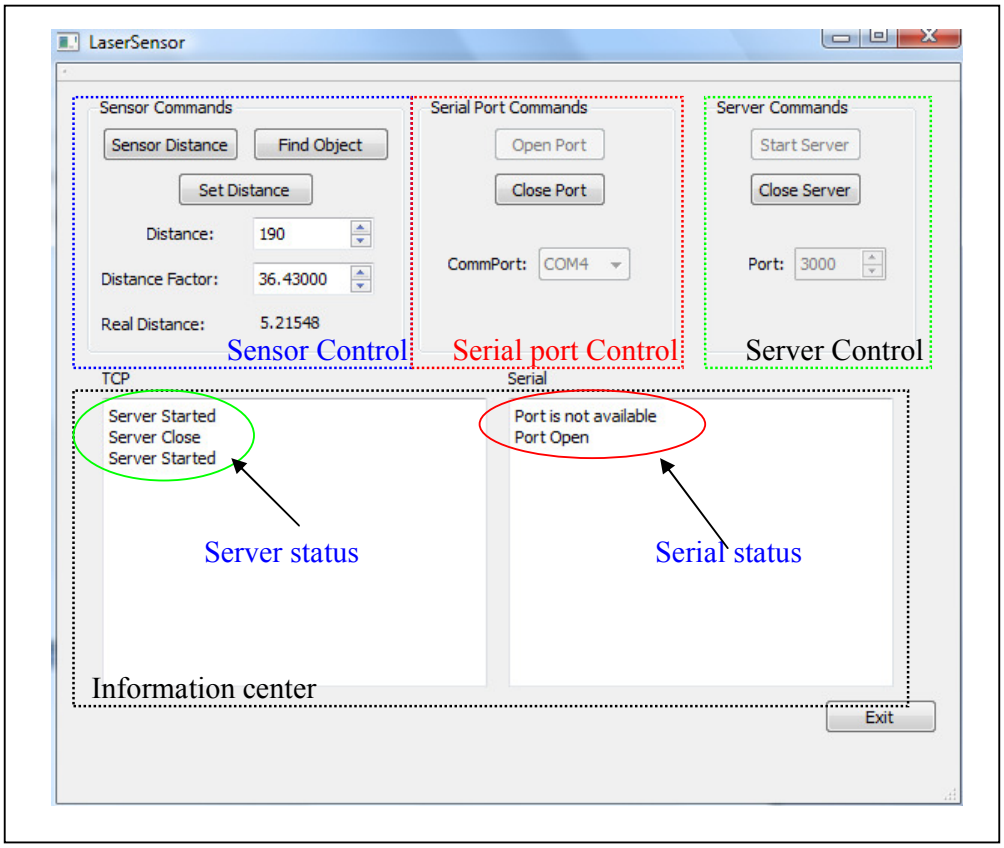


Figure 4.2 Laser Sensor Graphical User Interface

c) PC-Sensor communication

The communication between the laser sensor and the PC is made by serial communication which uses the protocol RS232 and the following configuration:

- 9600 bps
- 8 data bits
- 1 stop bits
- no parity.

As external clients communicate with the sensor software with commands, the program communicates with the laser sensor also with commands. However the commands are sent as hexadecimal values and not as strings. There are three commands that the program can send to the sensor.

- Set distance value
- Find object
- Distance to the object

The first command is used to set the distance for search procedures. The second one starts a search procedure. And the last one asks for the distance between the sensor and any object. When the sensor answers to any of these commands the program receives hexadecimal data which has to be interpreted before the information can be displayed to the user or send to any client connected to the sensor program.

Code level view

The PC-GUI was developed with Object Orientated Programming and the software used to write the program code is Qt Creator.

For an easy develop and understanding the program is divided in three object classes:

- main class
- serialthread class
- lasersensor class

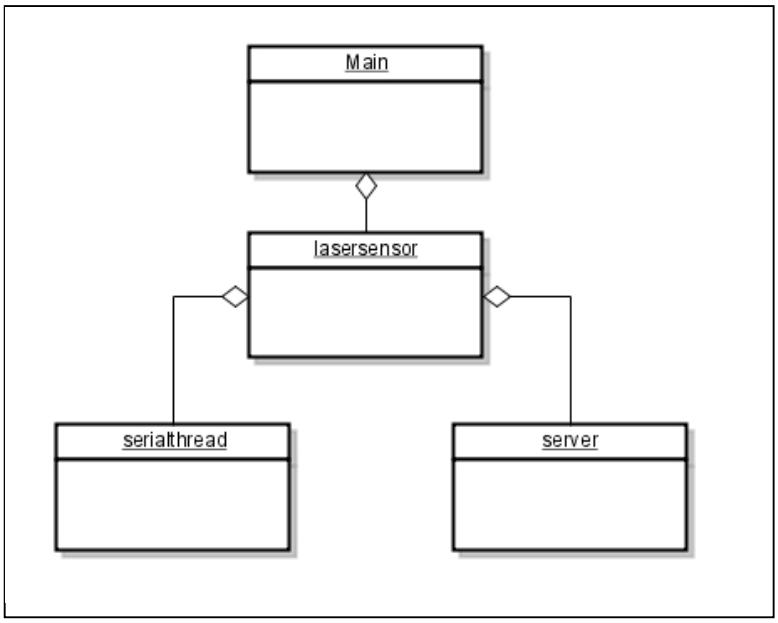


Figure 4.3 Program structure

The main class is used to start the laser sensor class and to show the graphical interface (see Figure 4.3).

The serialthread is a compilation or functions used to gain access of the serial port. The purpose of this class is to give to the serial port the properties and functionality of an object. This class has six attributes, which define the

configuration of the serial port comm, and twelve methods, which allow the user to use the serial port (see Figure 4.4).

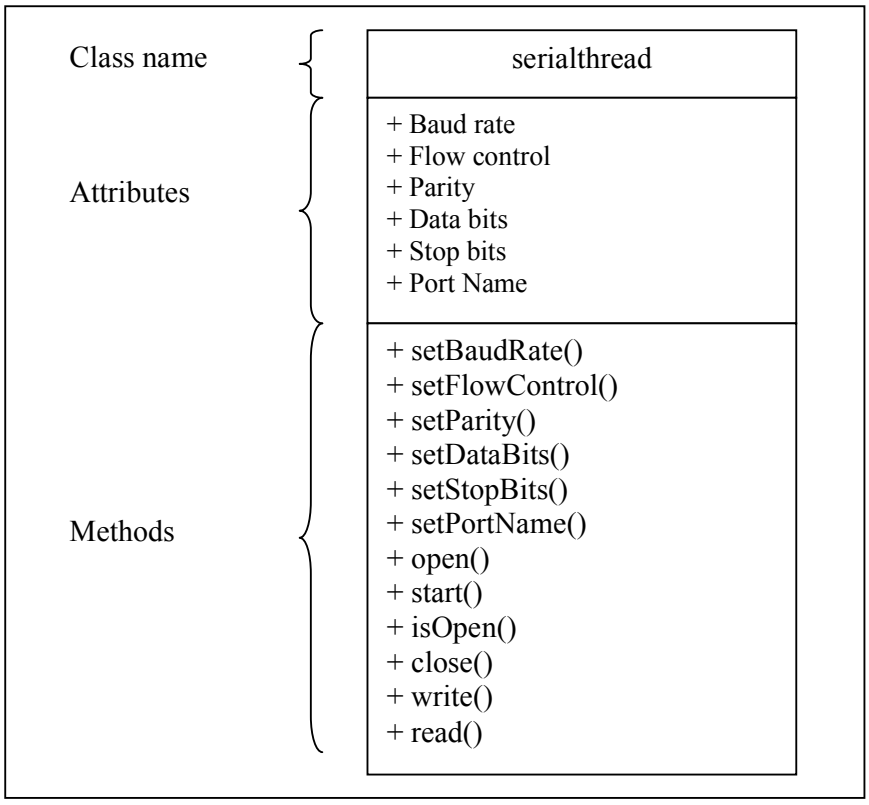


Figure 4.4 Serialthread class

The laser sensor class is designed to allow the communication with the user, sensor and external clients. It has five attributes and fifteen methods (see Figure 4.5, next page). To do that four methods are designed to manage the serial port comm. (sendSerial, serialReadyRead, OpenPort_clicked, ClosePort_clicked), five to manage the server (sendTcp, tcpReadyRead, serverNewConnection, StartServer_clicked, CloseServer_clicked), one two manage the numerical values that will send to external clients (ChangeRealDistance), three to manage the sensor (SensorDistance_clicked, FindObject_clicked, SetDistance_clicked()), one to create the laser sensor object (LaserSensor) and one to destroy it when the applications is ended (~LaserSensor).

To access to the serial port the sensor class create an internal object using the serialthread class. To communicate with external clients the laser sensor class creates a server object. The communication with the user is made by buttons that calls some methods of the class, and displays that show to the user the information received or transmitted by the server or by the PortComm.

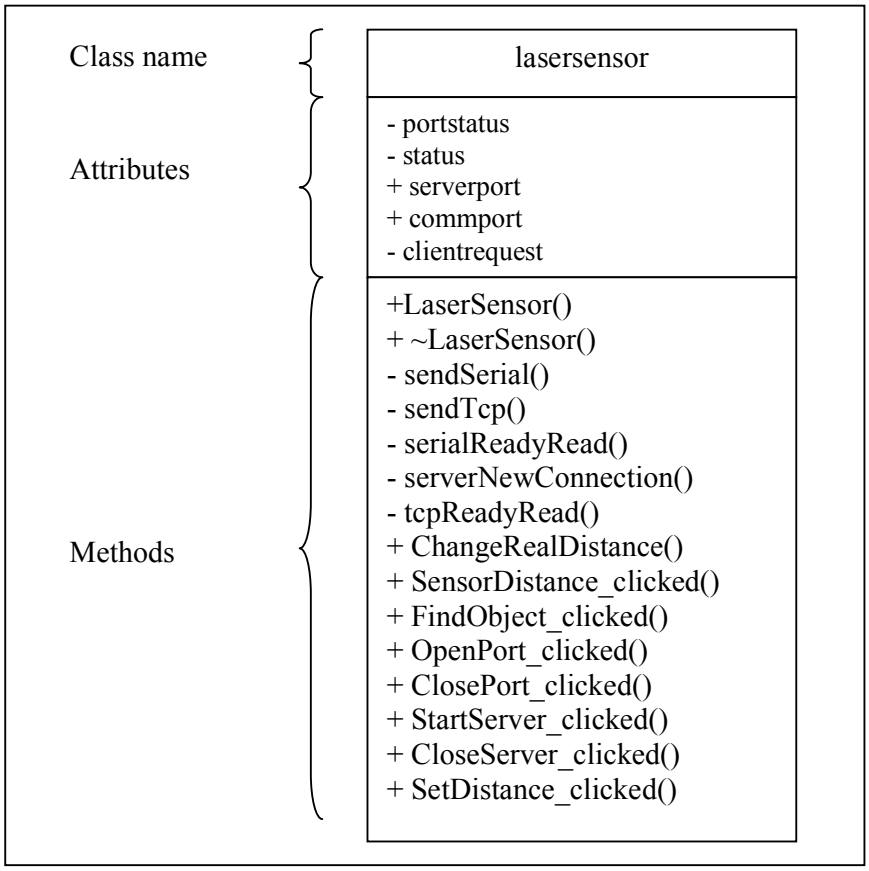


Figure 4.5 lasersensor class

4.2 Robot control PC-GUI classes

The robot PC-GUI is a program developed in a previous project also part of the Socrades project. Because of that in this chapter just the changes made to the original program are going to be explained. The purpose of the program is to allow external clients to communicate with the Motoman robot (see Figure 4.6, next page). As in the laser sensor program, the communication between the robot PC-GUI and other devices is made by Transmission Control Protocol (TCP).

The PC-GUI consists in two parts. The first part is a graphical user interface (see figure 4.7, next page) which shows to the user the information received and transmitted by the PC-GUI to the Motoman and to the clients connected to it. The second part is a collection of classes which manage all the information received y transmitted by the program and which also manage the behavior of the Motoman robot. In this part is where the Motoman class is located, which is the interest object of this chapter.

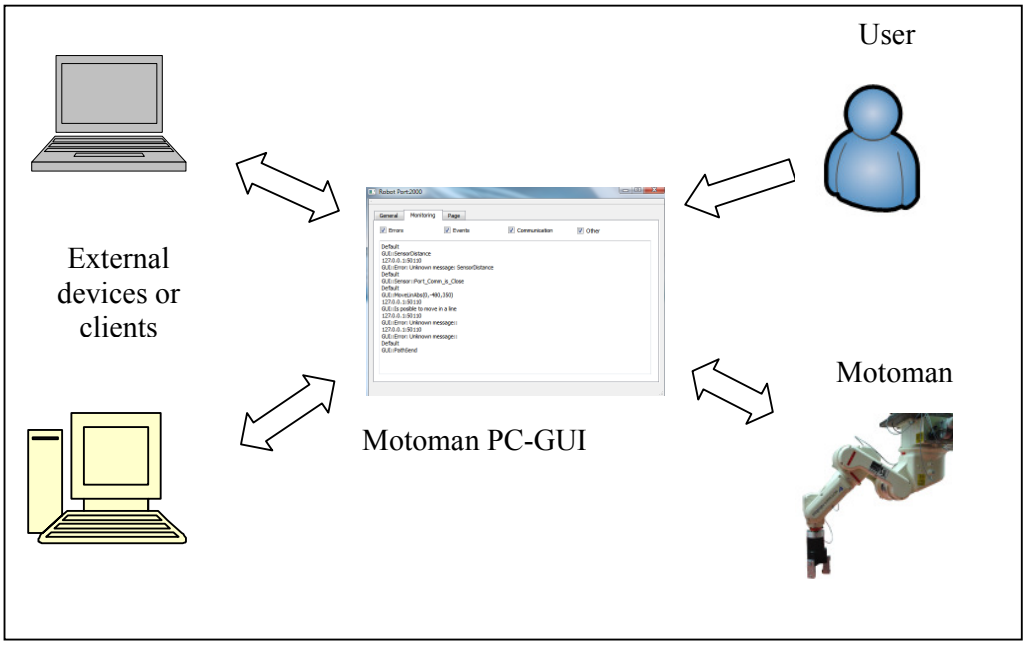


Figure 4.6 Communication diagram

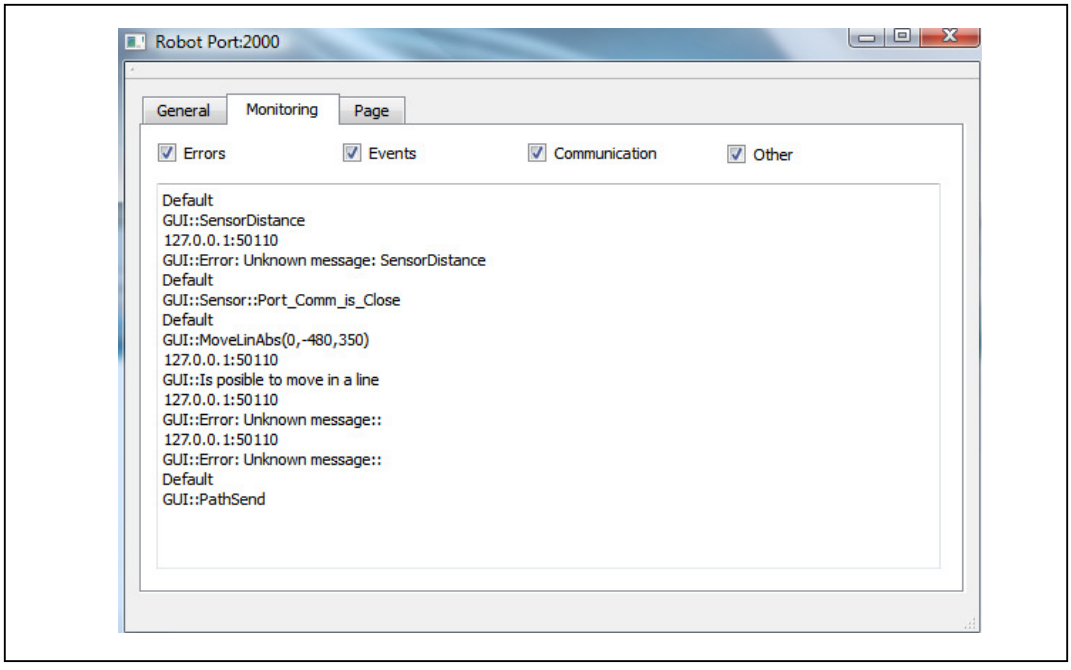


Figure 4.7 Motoman Graphical User Interface

In the original version the Motoman class fulfills the tasks of create a connection between the PC-GUI and the Motoman Server, and the task of calculate the joint angles for any point (X ,Y, Z).

The new version of the Motoman class also has the purpose of create a connection with the Motoman server and the calculation of joint angles. However in order to allow the Motoman to develop pick and place operations

some changes were make to the original class. These changes were the creation of two more classes and a restructuration of the functions of the Motoman class. The new classes are:

- motomanpickdrop
- motomanlinecurve

The first class has the functions needed to develop the pick and place operations in a modular and smart way. These means that these functions can be accessed for an external client with out have to give any type of especial information to the robot. The second class fulfills the task of calculate the join angles for point-to-point movement and for linear paths.

With the new restructuration of the Motoman class its main functions change from calculation tasks to management tasks. These management tasks are to determine what have to do the robot, move point-to-point, move following a path, search for an object of surface or just ask for the position of the Motoman's tool.

The new object relation for the Motoman class is showed in figure 4.8.

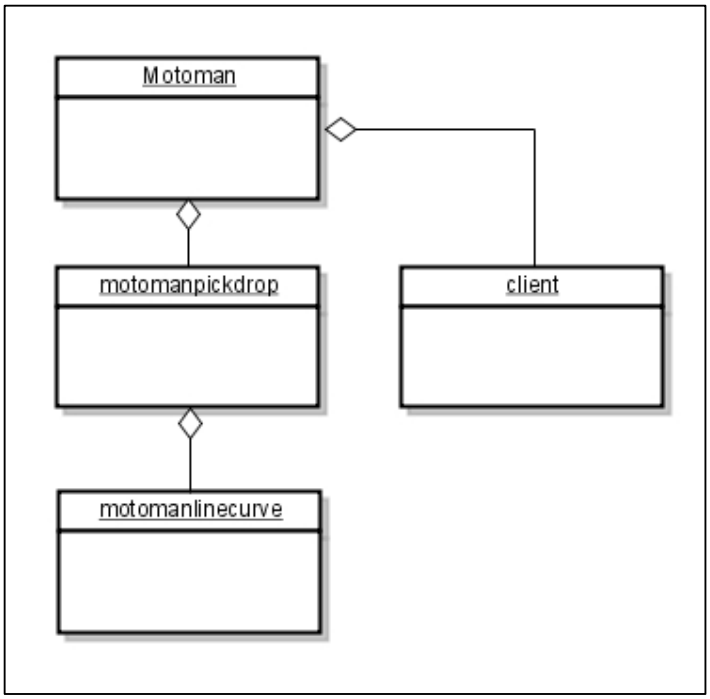


Figure 4.8 Motoman object structure

The new design of the motoman class has nine attributes and seven methods (see figure 4.9).

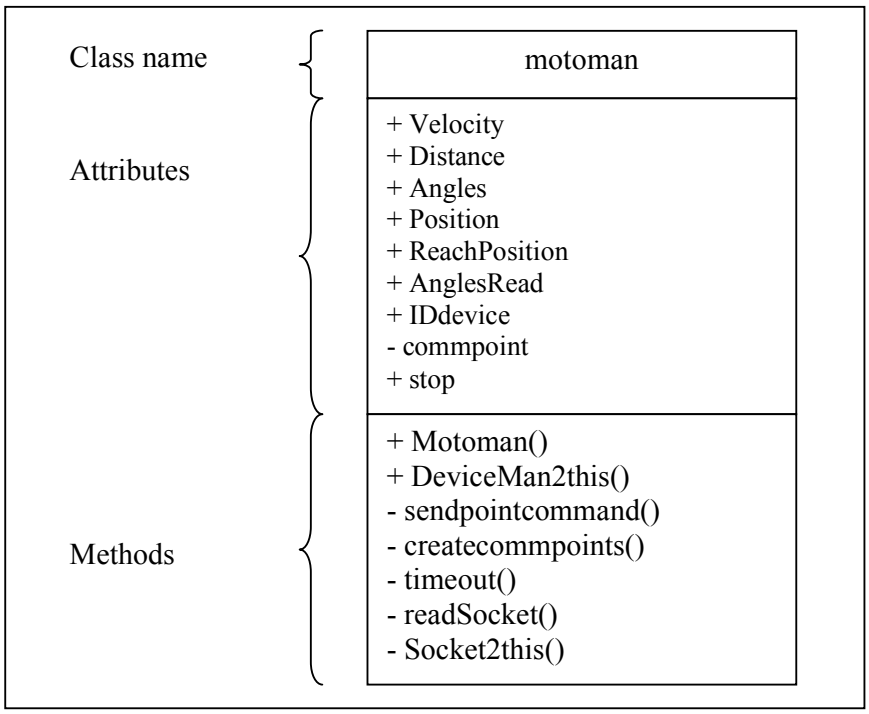


Figure 4.9 Motoman class

Figure 4.10 and Figure 4.11 shows the attributes and the methods of the new classes motomanpickdrop and motomanlinecurve.

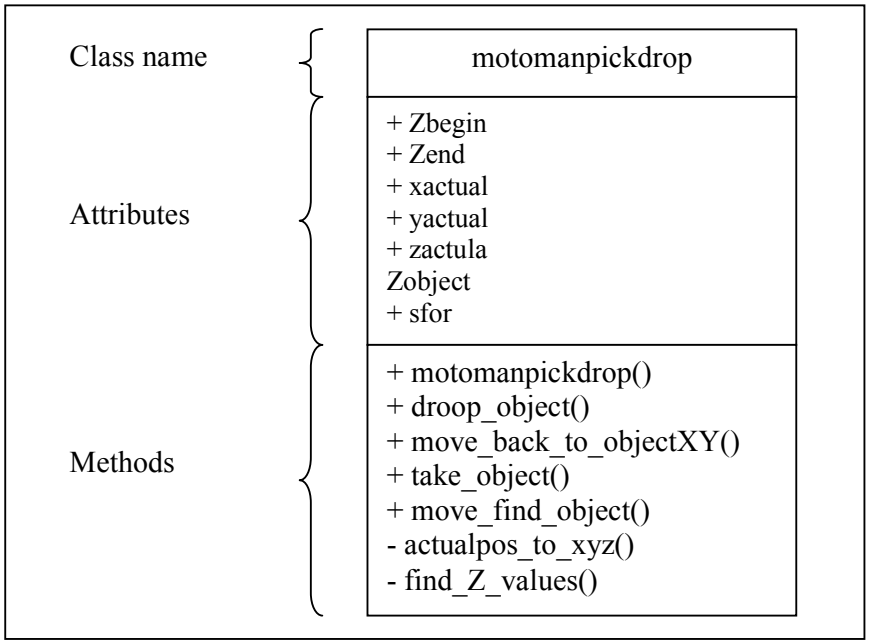


Figure 4.10 motomanpickdrop class

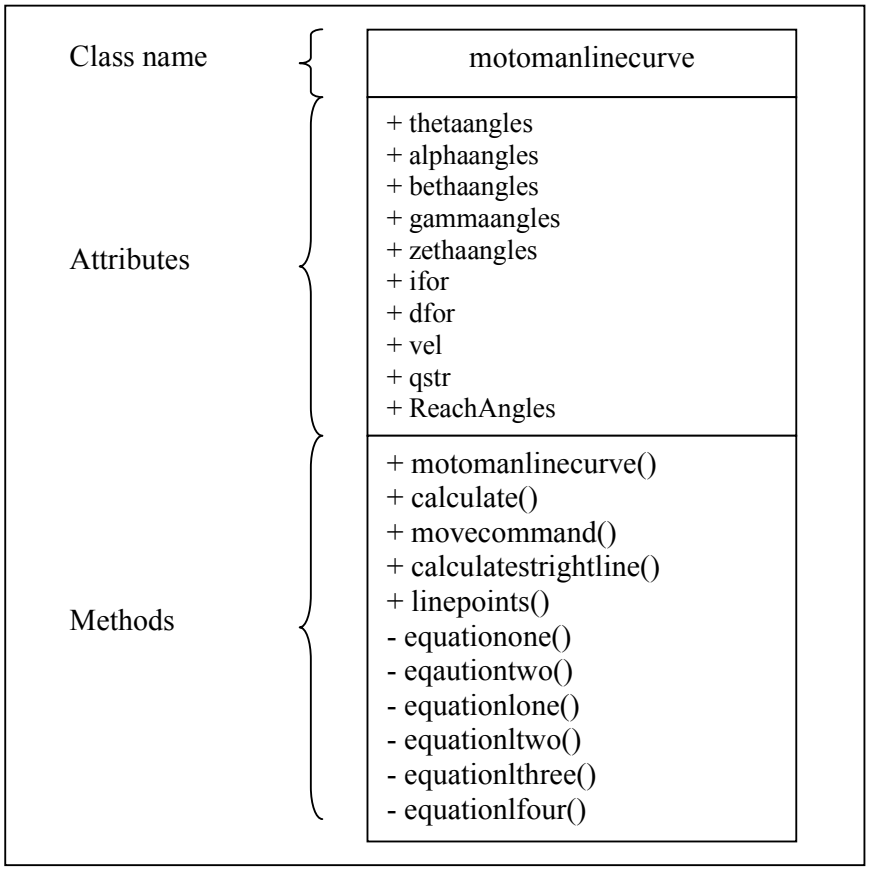


Figure 4.11 motomanlinecurve class

As with the laser sensor program the Motoman PC-GUI also communicates with other devices using commands. The commands related to the Motoman class are divided in three categories, robot status, move to point, and pick & place. The commands are the following ones:

Robot status:

- GetAngles()
- GetPosition()
- SetVelocity(velocity)
- GetVelocity()

Move to point:

- MoveStart
- RobotStop
- MoveAbs(X,Y,Z)
- MoveLinAbs(X,Y,Z)
- MoveRel(X',Y',Z')
- MoveLinRel(X',Y',Z')

Pick & place:

- MoveZFIND
- MoveZTAKE
- MoveZDROP
- MoveZBACK

In the first group of commands, the first command transmits to the client the value of the robot joint angles (α, β, ε, γ, θ, ζ). The second one transmits the coordinates X, Y and Z of the tool's position. The third command transmits the velocity that the robot uses to move. And the fourth one change the value of the velocity with the value sent by the client, i.e. "SetVelocity(5)" where "5" is the need robot's velocity.

For the second group, the first command moves the robot to an initial position. The second one allows stopping the robot when it is moving following a path. The third moves the robot to Point two (X_2, Y_2 ,Z_2) using point to point. The fourth one moves the robot from the Point one (X_1, Y_1, Z_1) to the Point two (X_2, Y_2, Z_2) following a linear path. The fifth command moves the robot from the Point one (X1, Y1, Z1) to the relative coordinates (X', Y', Z') using point-to-point, i.e. the actual position of the robot is P1(500, 200, 300) when the command "MoveRel(100,-50,100)" arrives the robot will move to the new position P2(600, 150, 400). The sixth command fulfills the same task as the fifth command however instead of moving point-to-point the robot will move following a linear path.

The third group are commands for pick and place operations. All the commands are designed to make the robot moves following linear paths. The first command makes the robot moves down until the maximal range of displacement is reach or an object or surface is found. This command fulfills the step "search for an object" of the pick and place procedures. The second command moves the robot the correct distance to pick an object. The third command moves the robot the correct distance to place an object. The fourth one moves the robot up until the top position is reached, this position is predefined as Z = 200mm. The distance that will move the robot to take an object is the distance between the robot's tool and the object plus the height of the object (50mm) and the distance the robot will move to place and object is just the distance between the robot's tool and the object or surface where the object already picked is going to be placed.

Note, to get the best result when the pick and place commands are used is highly recommended that the laser sensor is also working and connected to the

client that is managing the procedure, that is because the laser sensor is the device which will know when an object was found, and also is the device which measures the distance between the robot' tool and the objects.

5 Microcontroller system for sensor data acquisition, interpretation and remote commanding

The second part of the control system is the microcontroller system for sensor data acquisition. The microcontroller system is divided in two parts the sensor control program and the sensor mounting. The first part consist in develop a microcontroller program capable of control the sensor, read the sensor and communicate with the PC-GUI. The second part consists in mounting the sensor and to design the control circuits.

5.1 Sensor control program (microcontroller)

The microcontroller has three main tasks:

- Reads the sensor distance
- Watch over the sensor on/off state
- And communicate with the PC-Gui.

The communication between the PC-GUI and the microcontroller uses the protocol RS232. The communication purpose is transmits commands from the PC-GUI to the microcontroller and the sensor data from the microcontroller to the PC-GUI.
The commands send by the PC-GUI have the function of find objects, measure the distance between the sensor and the robot, and set the distance for find objects.

For solve this tasks using the microcontroller, three tools were use, USART (communication), ADC (distance measurement) and external interrupts (sensor state). Also two different programs were written, the first program fulfills the main tasks (control and communication) and the second program runs the watch over operation.

Main microcontroller program

The main program is divided in five parts:

- Microcontroller configuration
- PC-GUI commands reading
- Sensor reading (find object and distance measurement)
- Watch over the sensor state (only start the operation)
- And send data to the PC-GUI

Microcontroller configuration:

The microcontroller configuration part configures the USART, the ADC, the interrupts and the microcontroller PIN ports.

Port configuration:

The ATMEGA16 has 4 output/input ports, the project application uses different pins from the 4 ports however just 3 of the ports needs to be configure. These ports are PINA, PINB and PINC. And the configuration for these ports is the following:

PORTA: All the pins are set as inputs

PORTB: All the pins are set as inputs

PORTC: PINC4 is set as input and the rest are set as outputs

USART configuration:

The USART is set to 9600 bps, 8 data bits, 1 stop bits and no parity. The receiver, transmitter and receiver interrupts are enabled.

ADC configuration:

The ADC is set for run in one lecture mode, and the lecture is left adjust, this means that the 10 bits resolution only the 8 more significant are take in count for calculation effects. Only channel 0 is used.

Interrupts configuration:

There are two interrupts used in the program, the external interrupt INT2 and the USART receiver interrupt. The first one is configured as Rising-edge and is used just when the find object operation is running. The second interrupt is configured to listen all the time to the commands sends by the PC-GUI.

Code to configure the Microcontroller

```
void configure()
        {
        DDRA=0x00; // PortA as input
        PORTA=0xFF;
        DDRC=0b11101111; // PortC as output and PINC4 as input
        PORTC=0b00010000;
        DDRB=0X00; // PortB as input
        PORTB=0XFF;
        adcinic(0); // function for configure the ADC
        Init_USART(); // USART configuration
/*      GICR   |=(1<<5); // Enables the External Interrupt
```

```
MCUCSR |=(1<<6); // Rising edge */
SREG   |=(1<<7); // Enables Global Interrupts (bit7 is I) and start the USART interrupt
}
```

PC-GUI commands reading:

The program which is loaded in the microcontroller number 1 depends completely from the instructions send by the PC-GUI, in other words, the microcontroller process the sensor data only when the PC-GUI asks it. Because of that the command reading part is one of the central points of the program. The microcontroller has to be able to listen in any moment the information transmitted by the PC-GUI and also must understand any command send to it.

There are three main commands send by the PC-GUI, “set distance value”, “find object” and “distance to the object”, these commands are sent as hexadecimal values to the microcontroller, so when a command arrives the USART interruption triggers and save the value in a variable called “request”, then a special function “command()” deciphers the information and tells to the microcontroller what to do.

The function structure is the next one:

Funtion for read the commands

```
void command()
        {
        //Sets the distance value for find an object
        if((request!=4)&&(request!=1)&&(request!=3)&&(request!=6)&&(request!=0))
                {
                limit=request; //The limit value becomes the value send by the PC-GUI
                object=limit;
                sensoranswer();
                }
        //Reads the distance between and object and the sensor
        if((PINA & (1<<PINA6))||(request==4))
                {
                object=sensor();
                sensoranswer();
                }
        //Find Object
        if((PINA & (1<<PINA7))||(request==1))
                {
                find_object();
                }
        }
```

Sensor reading (find object and distance measurement):
The sensor reading is basically done with the microcontroller's ADC, the ADC reads the sensor voltage signal and converts this voltage value to a digital value. When the reading is done, the program determines the distance between the object and the sensor. This value can go from 0 to 255, 255 is the largest distance a 0 is the shorter one, however a value 0 can also means that the sensor is working out of its range.

Because of this a work range was set. This range goes from 255 to 21, then any value below 21 means that the sensor is working out of its range. After the distance is measured the digital value is sent to the PC-GUI.

For find and object multiple distance measurements are done. The process consists in a loop which compares the distance measurement with the value set for find the object.

The loop ends when both values are equal meaning that and object has been found. Finally a message of "Object find" is sent to the PC-GUI.

The code for the loop is the next one:

```
void find_object()
        {
        ..
        //Comparation loop between the distance measurement and the set value
        while((value!=limit))
                {
                //distance measurement
                ADCSRA|=(1<<ADSC);
                loop_until_bit_is_clear(ADCSRA,ADSC);
                value=ADCH;
                }
        ..
        }
```

Watch over the sensor state (only start the operation):

Know when the sensor is "on" or is "off" is critical when a "find object" operation is in process, when an object is found the microcontroller sends a signal which is going to stop the movement of the robot, then if for any possible reason the sensor is "off" when a "find object" operation is in process the microcontroller will never send the stop signal causing that the robot could result damage. To avoid this situation a special function was implemented using the

microcontroller. This function has the purpose of monitor the sensor's state when a "find object" process is running. Then if in any moment the sensor stops working the microcontroller sends a signal to stop the robot.

This special function was implemented using the ADC and consist in read all the time the sensor's signal. However as was explained when a "find object" operation is in process the ADC is running, so for avoid any problem between the sensor reading and the sensor monitoring, the task was divided in two microcontrollers, the first one just start the monitoring process and the second one executes the process. The interaction between these ones is made using the external interrupts. So, to start the process the first microcontroller sends a signal of 5 volts which triggers the external interrupt in the second microcontroller. When the search process ends with any problem and the object is found the microcontroller sends a signal of 0 V which also triggers the external interrupt and stops the monitoring function. But if there is any problem while the search process still running, the second microcontroller sends a signal of 5V which triggers the external interrupt in the first microcontroller, so the microcontroller ends the search process and sends a stop signal to the PC-GUI.

Sending data to the PC-GUI:
Send information from the Microcontroller to the PC-GUI is an important task because is how the sensor information is transmitted to the Robot. The information is sent using simple hexadecimal values. The code use to do it is the next:

```
//Function for send bytes
void SendByte(unsigned char u8Data)
{
// Wait if a byte is being transmitted
while((UCSRA&(1<<UDRE)) == 0);
// Transmit data
UDR = u8Data;
}
```

Auxiliary Microcontroller Program

The second program where the monitor action is made, consists in two parts

- Microcontroller configuration
- Watch over the sensor state

Microcontroller configuration:

The configuration part configures the ADC, the interrupts and the microcontroller PIN ports.

Port configuration:

The configuration for the microcontroller ports is the following:

PORTA: All the pins are set as inputs

PORTB: All the pins are set as inputs

PORTC: All the pins are set as outputs

ADC configuration:

The ADC is set for run in one lecture mode, the lecture is left adjust, this means that the 10 bits resolution only the 8 more significant are take in count for calculation effects. Two channels are used, channel 0 and channel 1.

Interrupts configuration:

Only the external interrupt INT2 is used. However this one is configure according to the search step, when is waiting for start the watch over process, the external interrupt is configure as rising edge, so any change from 0V to 5V will trigger the interrupt. Otherwise when the watch over process is running the INT2 is configure as falling edge, and then any change from 5V to 0V will trigger the interrupt.

Watch over the sensor state:

The part that the second microcontroller redeems is the direct watch over operation. Because the sensor signal goes between -10V and 10V, and the microcontroller can read only positive signals; the signal needs to be adequate. This problem was solved dividing the sensor signal in two parts, a positive part and a negative part. Then in order to can watch over the completed sensor signal two channels of the ADC were used.

Now, to know if the sensor is working or not, is to watch if in any moment the signal becomes 0V. To know that the microcontroller read both signal parts, when both of them have at the same time a value of 0V is possible to know that the sensor is no working.

5.2 Sensor Mounting

After design the control program a new task was found, integrate the sensor with the robot. This task was divided in two parts, the first part is the mechanical solution and the second part is the electronic solution.

The mechanical solution tells how to mount the sensor over the robot. And the electronic solution tells how to supply electrical power to the sensor and how to read the sensor's signal.

Mechanical mounting

For the mechanical problem the best position to mount the sensor is over the gripper, the reason is that the gripper is the critical robot's part that we want to control using the laser sensor, also is the nearest part to the objects that we want to pick-up or place. As the gripper is the nearest part to the objects we avoid one important problem, the sensor range. The laser sensor has a detecting range of 5cm to 14 cm so is very important that the object, that is going to be found, is inside this range, for this sake the sensor works with a maximal detecting distance of 10cm and a minimal detecting distance of 5.5cm. Looking at the gripper shape we found that there is no a simple way for mounting the sensor over the gripper.

Figure 5.1 Gripper

For this reason we have to design a new piece that fulfills the function of joining the sensor with the gripper. In order to design the new piece we have to think in the next restrains:

1. Sensor detection range
2. sensor's dimensions
3. angle of the sensor
4. Gripper's dimensions
5. gripper's shape
6. the sensor has to look for the object middle point
7. Object high
8. gripper and sensor safety
9. easy to fabricate

The first step was to take the dimensions and the shape of the gripper. Then the main obstructions between the sensor and the object were found. The mains obstacles are the gripper's fingers, so the position of the sensor has to avoid all the time any possible interference of the gripper's fingers.

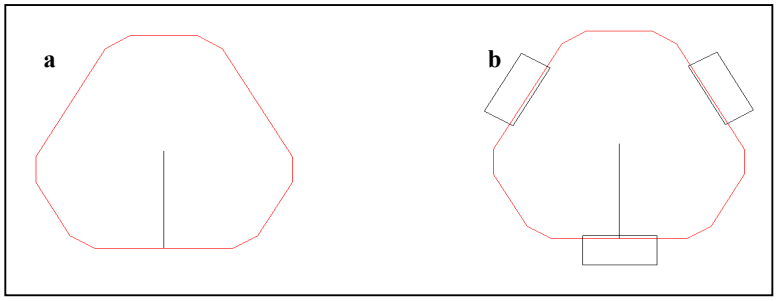


Figure 5.2 a) Shape of the upper part of the gripper,
b) Shape off the gripper with fingers.

Now we have to think a shape for our piece. First of all, it has to have the shape of the gripper. The reason is for guaranties that our piece won't move out of place when the robot is moving. However as the gripper shape is a complicate one we decide to find a simple shape.

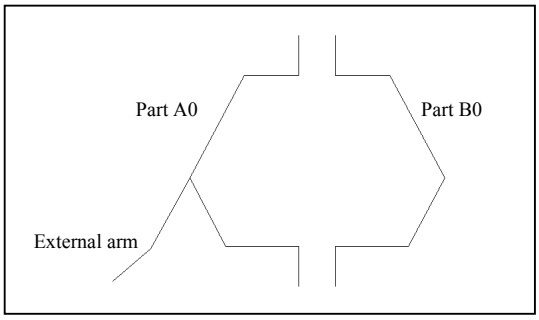


Figure 5.3 Sensor support, upper view

The design of the piece is made of two parts, which are going to be joined with screws. Also the sensor is going to be fixed in the external arm, which one is pointing to the middle point off the gripper (see Figure 5.3). However this design has two problems, is complicate to fabricate and due to the shape of the gripper is probably that with the time it presents a small movement when the robot moves, for this reason we choose to change the design.

For get a piece easier of fabricate the new design consists in a piece made of four parts. Of these four parts, three parts correspond to “Part A” and the other one corresponds to “Part B”.

After that the “Part B” will melt with a “Part A” making the “Part C”. In that way our final piece is made of two “Part A” and one “Part C” (see Figure 5.4).

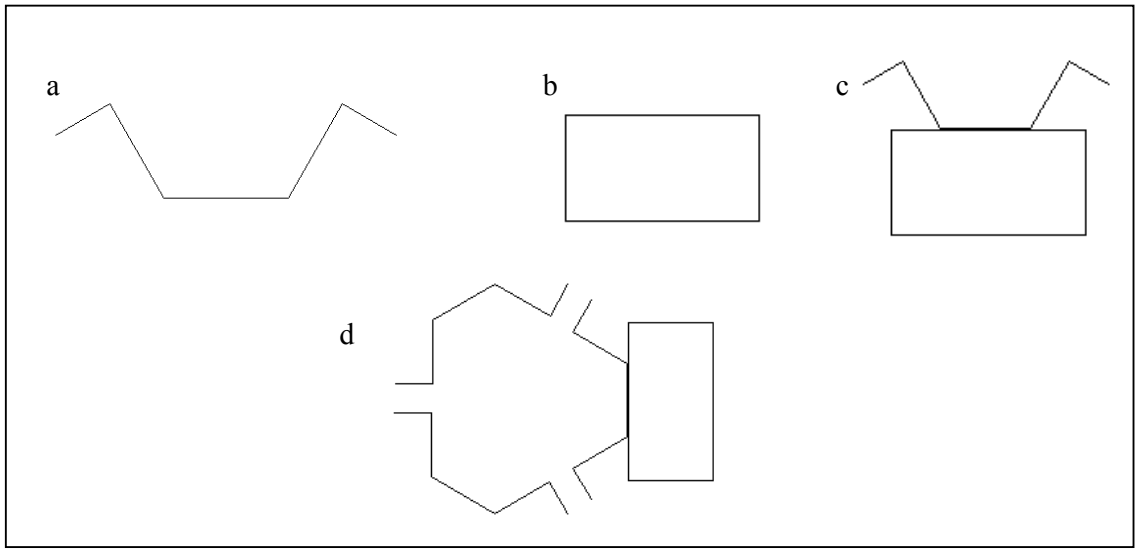


Figure 5.4 Superior view of: a) Part A, b)Part B, c)Part C,
d) Sensor Support final design Top View

As the preview design the three “Part A” are going to be joined with screws, however the difference between this new design and the old one is that the new design is made to fix perfectly with the gripper. This is because the screws help to adjust the piece to the gripper. The sensor is going to be mounted in the external arm. Until now the shape of the piece has been design taking care of not violate our restrains (easy to fabricate, gripper shape, the sensor looking for the object middle point and gripper dimensions), however our piece is not completed design yet. In order to complete the last design step, we are going to

base in four main restrains (sensor detection range, object high, gripper shape and dimensions).

Knowing the maximal sensor range (14 cm), the object high (5cm), gripper shape, the gripper high (6.11cm) and a maximal center distance between objects (12.5cm), we design a practical system that allows us to find the best inclination angle for the sensor. Also with this system the dimensions for the Part B were found. The system has the finality to show the work area of the sensor (see Figure 5.5).

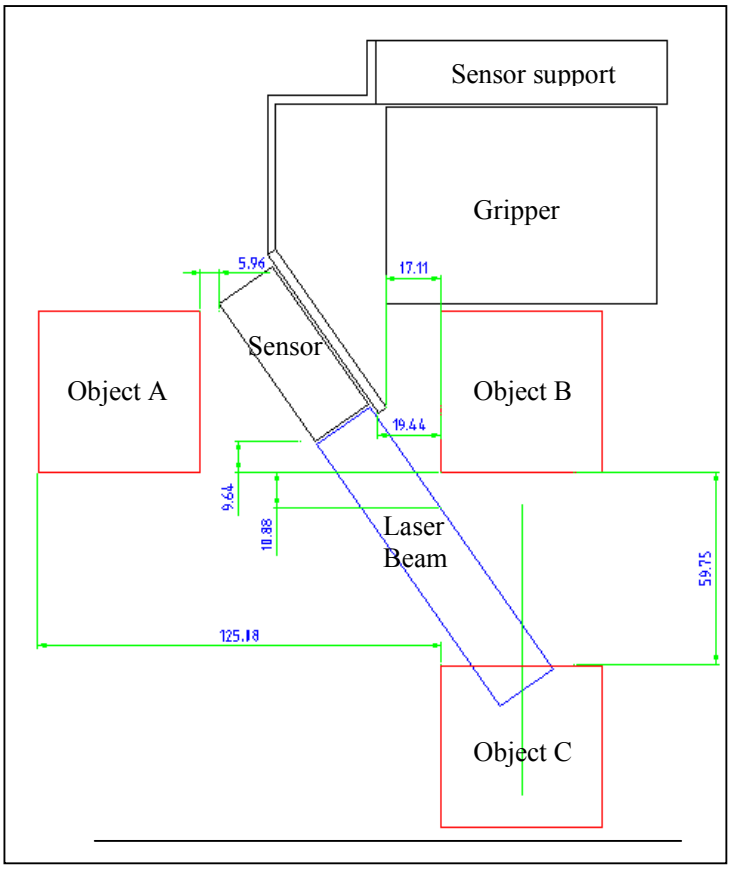


Figure 5.5 Sensor distance range.

Finally our pieces look like:

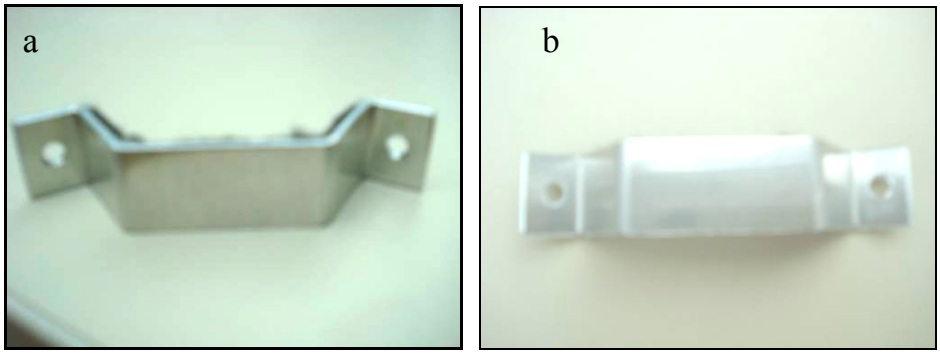


Figure 5.6a and 5.6b Part A

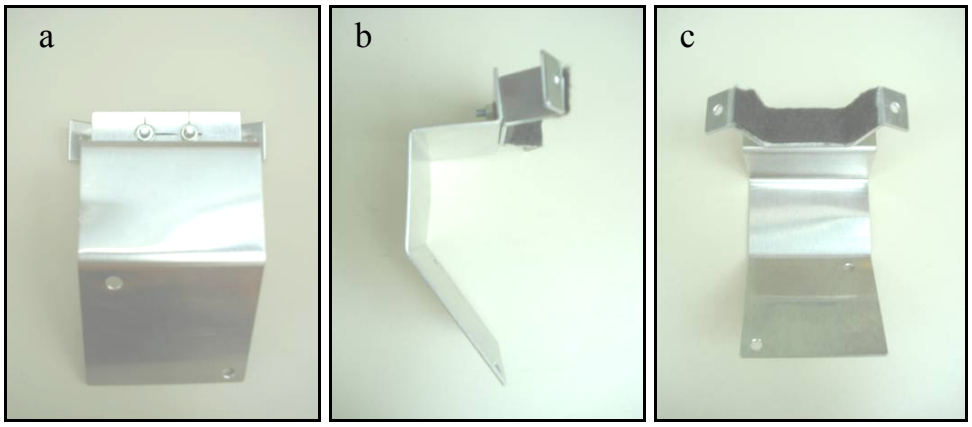


Figure 5.7 Part C, a) Behind view, b) lateral view and c) frontal view.

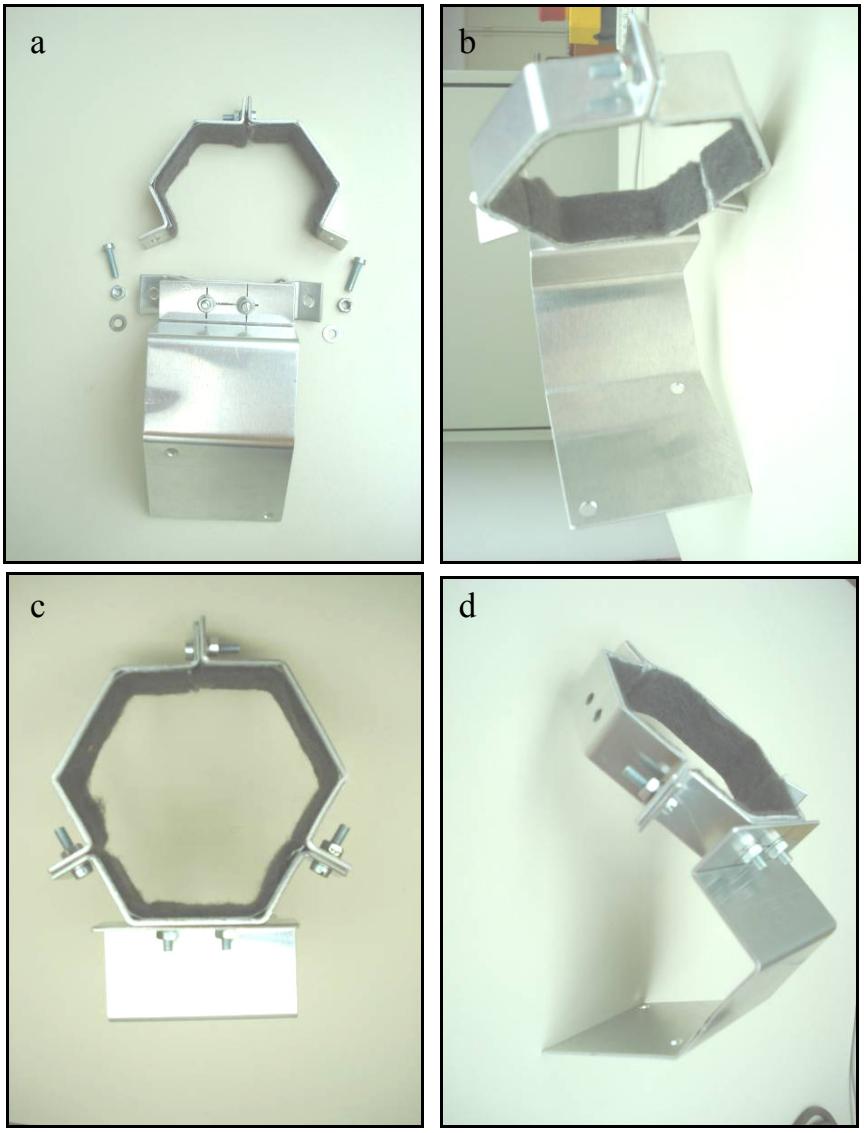


Figure 5.8 Final part, a) before final assembling, b) frontal view, c) top view and d) lateral view.

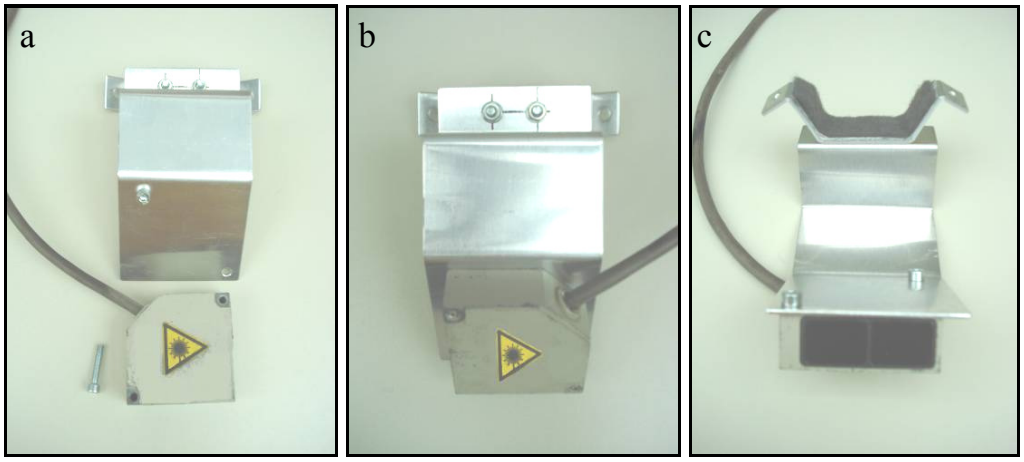


Figure 5.9 Sensor Mounting, a) part C and laser sensor before mounting, b)sensor mounted, behind view and c) sensor mounted, frontal view.

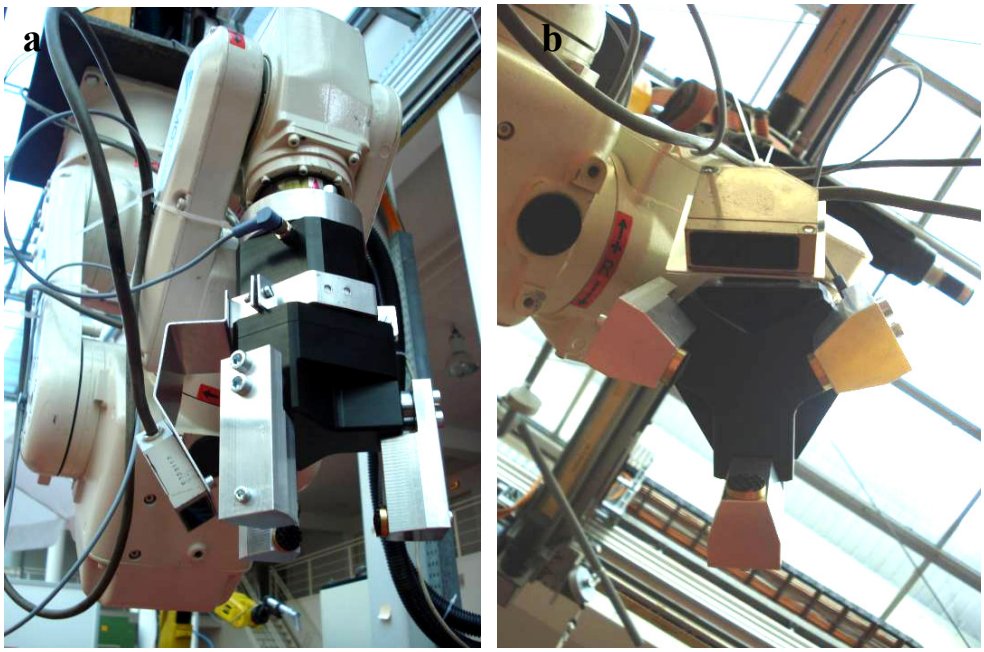


Figure 5.10 Final mounting of the sensor with the gripper, a) lateral view and b) bottom view.

Electronic design

The second part of using the sensor is “How to read it?” and “How to watch over it?”, for this action a embedded system was developed. The embedded system has two basic parts, a microcontroller and the laser sensor.

The purpose of the laser sensor is to measure the distance between the gripper and the objects.

The purpose of the microcontroller is to interpret the laser sensor measures, send them to a Pc-GUI and watch over the laser sensor.

Then the electronic design is centered in create an electronic circuit that joins the microcontroller and the laser sensor. The circuit needs to provide power supply to all the parts and a way to the microcontroller for control the laser sensor.

The main point of or designs is how to couple the laser sensor’s signal with the microcontroller. The laser sensor gives an analog signal that goes between 10V and -10V (see Figure 5.11). The microcontroller can read analog signals between 0V and 5V.

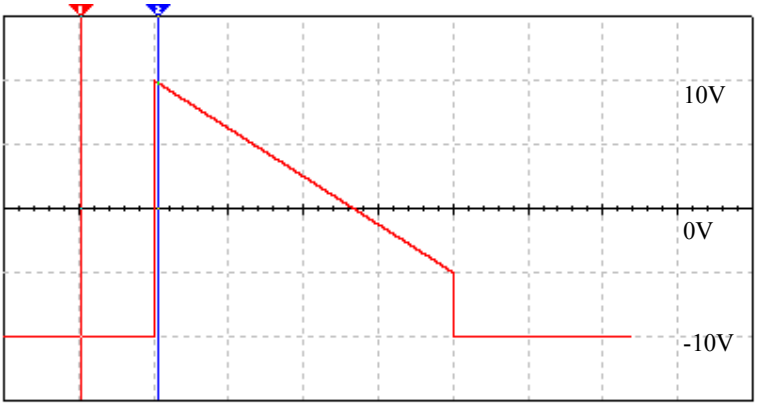


Figure 5.11 Laser Signal

The best way of coupling the laser sensor and the microcontroller is modifying the signal to read. We have two ways of modify it. The first option is to add a level of 10V to the laser’s signal, so the signal is always positive and then just attenuate it.

The second option is to cut out the negative part of the laser signal and then attenuate the new signal. For both options the final signal always has to be between 5V and 0V, so our range of 5V is going to give us how many times the laser's signal has to be attenuated.

For select and option the advantages and disadvantages of each one are going to be analyze. The main advantage of the first option is that we can read the laser sensor's signal in its complete shape and range but the main disadvantage is the way to add the 10V to the signal, this operation uses OPAMs, and the OPAMs for an adding operation of 10V to a signal of 10V need a power supply of -24V and 24V, so the circuit becomes more complicate. The second option has the advantage of be simple to apply. The only thing need to cut the signal is a DIODE, however the negative part of the signal is going to be lost, so is impossible to read the Laser Sensor during all its range.

As we want a circuit simple of make the option would be the second, however losing control of the Laser Sensor in any moment isn't an option, so the first option would be the option to choose. Now there is another factor that can be considered, when the laser's signal provides the system with important control information. In figure 5.11 is possible to watch the laser sensor range and understand that the most critical points for the measurement are in the far point to the Sensor. This point also corresponds to the higher value in the laser sensor's signal.

So, for or control operations just the higher values of the signal are important, and as this part of the curve doesn't lost when we apply the second option, then is possible to choose the second way to couple the signal.

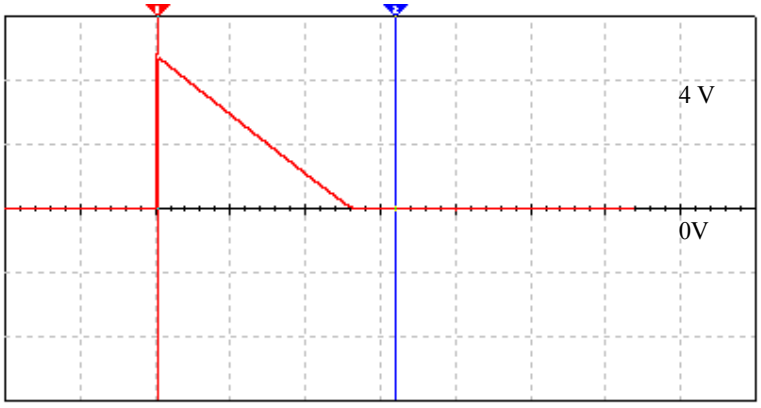


Figure 5.12 Signal to read with the Microcontroller

Then, applying the second option, the signal that is going to be read with the microcontroller is the one showed in Figure 5.12 This signal is created using a DIODE and attenuate using a voltage divider. The voltage divider consist in two resistors one of 3KΩ and another one of 2KΩ. So with these values we have a final signal between 0V and 4V.

Now, the signal is ready to be interpreted by the microcontroller. This task is fulfilled with the ADC in the microcontroller. So the signal is connected to the microcontroller's "PINA0".

Once the laser sensor's signal is coupled with the microcontroller, the third task of the microcontroller "How to watch over the laser sensor" needs to be fulfilled. The only way to watch over the laser sensor is watching over the laser sensor's signal, so the laser sensor signal needs to be read all the time or at least when a critical procedure is in process.

This task is also solved using the ADC in the microcontroller; however this time the complete shape of the signal needs to be read. The positive part of the signal is already available so just the negative part of the signal is needed for get the complete signal of the laser sensor, to get it another DIODE is used, this one cuts out the laser sensor's signal positive part. Now the signal is attenuate in the range of 0V and -5V.

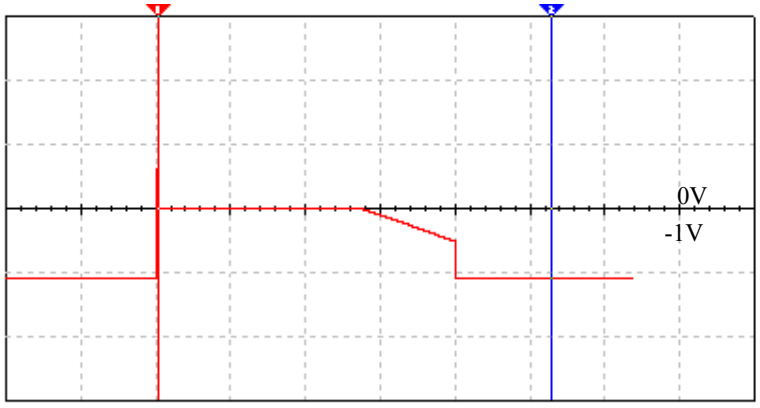


Figure 5.13 Negative part of the laser's signal after attenuation

The microcontroller just can read positive signals so the negative signal is inverted using an OPAM with negative gain. As the signal to multiply with the OPAM is a small one is possible to use a voltage supply of 5V and -5V.

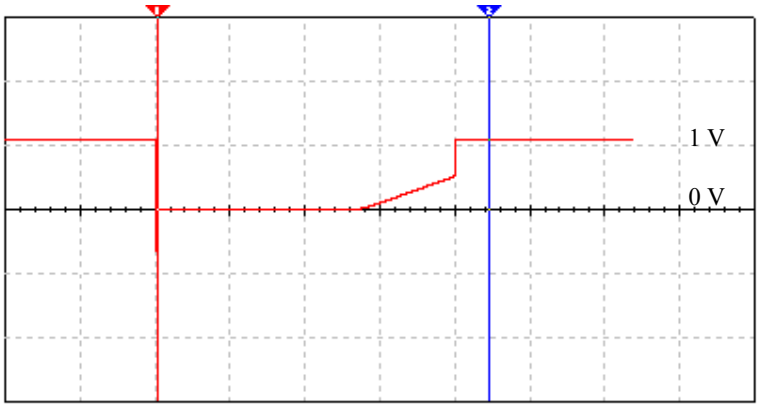


Figure 5.14 Final Signal

The new final signal is the one showed in Figure 5.14. With these two signals is possible to have a total control of the laser sensor. In Figure 5.15 and 5.16 are showed the circuits used for coupling the signals. The circuit showed in figure 5.15 split the signal in positive and negative part and the circuit showed in figure 5.16 rectifies the negative part.

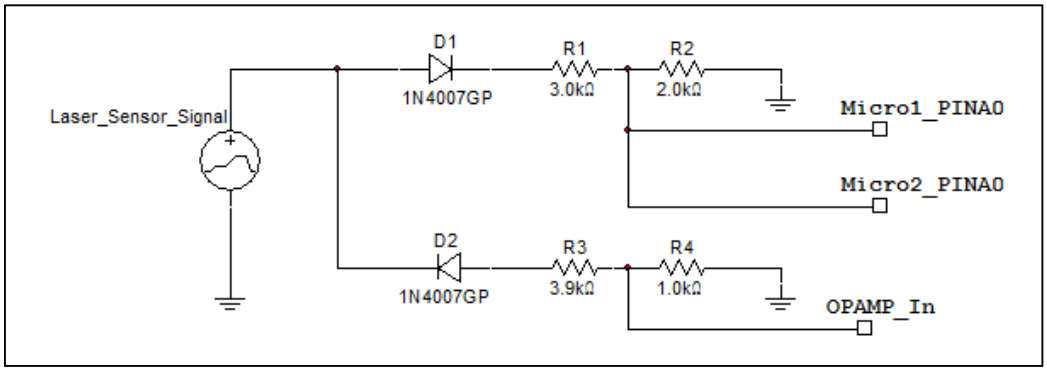


Figure 5.15 Circuit for cut and attenuate the signal of the laser sensor

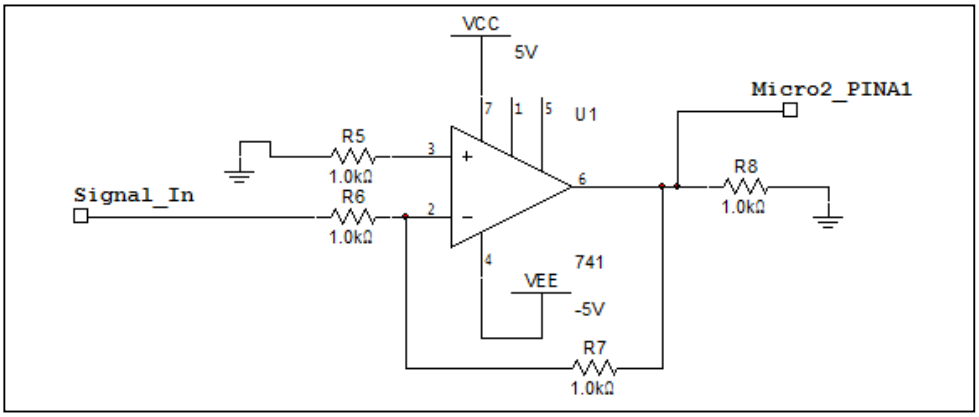


Figure 5.16 Inverter circuit

Now, the figure 5.17 shows the control circuit.

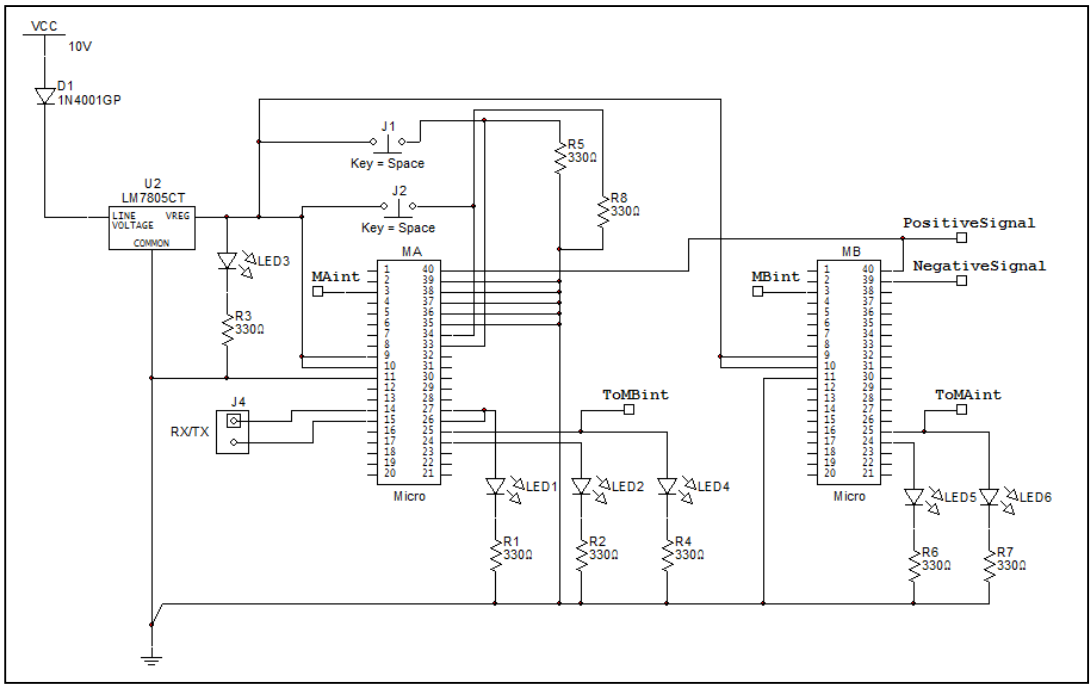


Figure 5.17 Control Circuit

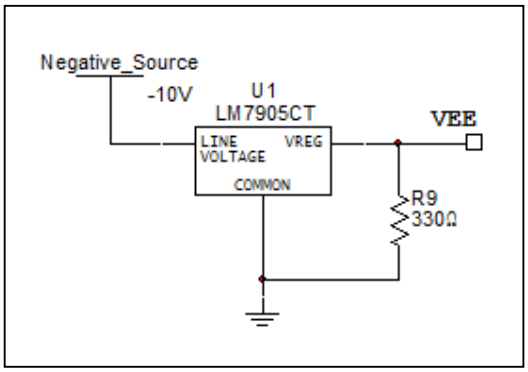


Figure 5.18 Negative source circuit

Finally there is one circuit more needed, one circuit for coupling the computer serial port with the microcontroller serial port (USART), for this task the circuit MAX232 is used, the typical connection is showed in figure 5.19 (next page).

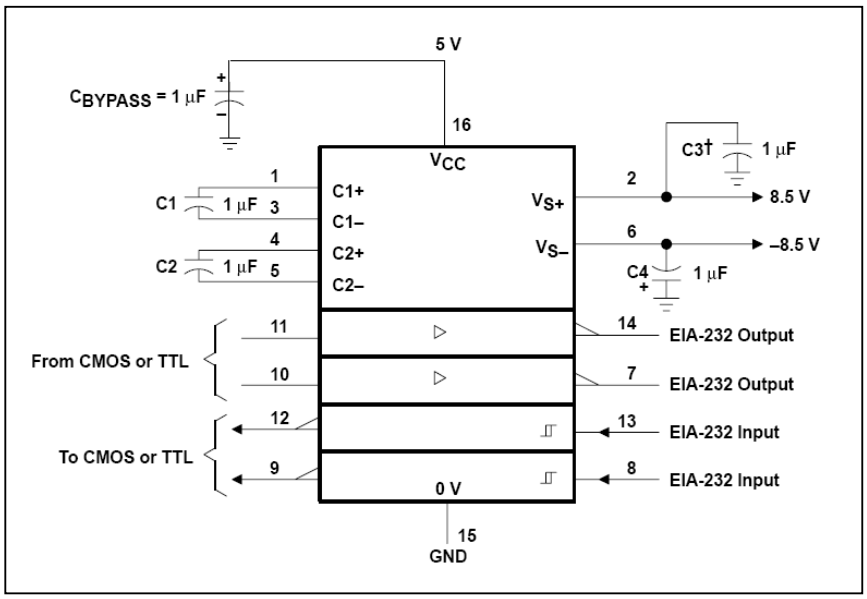


Figure 5.19 RS232 coupling circuit

The microcontroller's pin TX is connected to MAX232 pin 11 and the microcontroller's pin RX is connected to MAX232 pin 12. The computer's TX is connected to MAX232 pin 13 and the computer's RX is connected to MAX232 pin 14.

The components needed for make all the circuits are:

Control circuit

2 Microcontroller ATMEGA16
6 Leds
9 Resistors 330Ω
1 Voltage Regulator LM7805CT
1 Voltage RegulatorLM7905CT
1 Diode 1N4001
2 Push Button

Signal circuit

1 OPAM LM471
2 Diode 1N4007
1 Resistor 3KΩ
1 Resistor 2KΩ
4 Resistor 1KΩ

RS232 coupling circuit

5 Capacitor 1µF
1 MAX232

6 Functional tests and evaluation of the system performance

6.1 Tests

Until this moment everything has been a theoretical approach with out validation, so to demonstrate that the approach develop during the previous chapters to this one fulfills the objective of “pick and place objects with out explicit programming by networking collaboration of a robot and interpreted sensors” some test were made. The test has the purpose of check the correct performance of the laser sensor, the correct performance of the Motoman and finally the correct performance of the complete system when both devices collaborate between them to pick and place objects.

6.1.1 Laser Sensor

To verify the correct performance of the system relate to the laser sensor the devices shows in Table 6.1 were tested in the points also listed in Table 6.1.

Device	*Point to test*
Microcontroller Atmega16	Program
	Fuse bits Configuration
	Serial Communication
	Port Read
	External Interrupts
Laser Sensor	Output Signal
	Range of measurement
PC-GUI	Server control
	PortComm control
	Interface control

Table 6.1 Laser sensor tests

The Program and Fuse bits configuration of the microcontroller first were tested with the AVR Studio software, when a program is been uploaded to any microcontroller the software allows to check if the program was correctly upload and check if the fuse bits configuration is correct. In that test the two point did not have any problem.

To test the Port Read, External Interrupts and the self program, an interface with buttons to simulate signals and LEDs to show the behavior of the microcontroller were designed. The serial communication were tested sending

some hexadecimals values from a PC to the microcontroller and then reading in the PC the data arrived from the microcontroller. This test was made with the laser sensor PC-GUI.

All of these test showed a correct performance, which can be appreciate in the final test of this chapter (see point 6.2)

The test related to the PC-GUI are all test of communication, the test consisted in connect external clients with the PC-GUI and then send commands in order to test the communication, to do this the PC-GUI was used as a mirror, so any message that were received was sent back to the device whom sent it. The interface test consisted in verify that when the user interacted with the program the commands gave by him were done correctly, this part was tested using the debugger of the QT Creator software, because the debugger allows to follow the changes that happen in the program when this is running. The port Comm were tested sending hexadecimal values to the port and then reading these values with an oscilloscope. When the communication was correct the test was to connect the PC with the microcontroller. At this point the first problem of the system happens. The problem was that the PC-GUI was not responding to the values sending by the microcontroller. That was because the PC-GUI needs to wait few milliseconds between sending and receiving data, and the microcontroller was transmitting information before the wait time ended. To solve this a delay of 250miliseconds was added to the microcontroller program before send any answer to the PC-GUI.

The laser sensor tests are divided in two parts. The first part consisted in get the work range of the sensor, this range was between 4cm and 14cm. Which is the range of distance where the sensor gives a voltage value different from -10V. The second part was to check the precision and exactness of the sensor when is measuring a distance. To check this some readings were taken with the sensor suspended in different heights. Table 6.2 shows the result of this test.

Point	*Measure 1*	*Measure 2*	*Measure 3*	*Measure 4*
P1	179	177	178	181
P2	145	146	144	143
P3	147	147	148	146
P4	180	181	179	182
P5	146	149	147	147

Table 6.2 Sensor measurement repeatability

From table 6.2 is possible to appreciate that the precision has a variation of 4 digital units when a distance is measured

6.1.2 Motoman

To verify the correct performance of the system relate to the Motoman robot the devices shows in Table 6.3 were tested in the points also listed in Table 6.3.

Device	*Point to test*
Motoman	Server connection
	Motoman commands
PC-GUI	Interface
	Server and client connection
Motoman movement	Calculation of joint angles
	point-to-point movement
	linear path

Table 6.3 Software Tests

The Motoman was the tested with a software client. The test consisted in send some commands and verifies that the Motoman did what has asked. The commands sent as part of the test are the following:

- BscPMov MOVJ V 1 0 0 80500 60500 0 40500 0 0 0 0 0 0 0
- BscIsLoc 0
- BscIsLoc 1

The first command is used to move the robot given the value of the joint angles. The second command asks for the value of the tool coordinates and the third command asks for the value of the robot joint angles.

The PC-GUI was tested connecting some clients to the program and sending some command to it. This was another mirror test were the PC-GUI has to send back the commands received.

The Motoman movement was tested just after the classes developed to move the robot where finished and included in the PC-GUI program. To test this point, three test were made. The first consist is ask the PC-GUI to calculate the joint angles for any point and then just display the value. This test gives answers as the next line:

-119969 31953 9804 0 74436 -50939

where the value of the joint angles is expressed in pulses.

The second was to connect the PC-GUI to the Motoman server and send the values calculated in the test one. The purpose of this test was to verify that the

functions deduce in chapter three, calculates the joint angles for any point correctly. The next sequence of pictures (see Figure 6.1) shows the robot moving trough seven different points, from point P1 (0, -480, 224) to point P7 (0, -480, 494).

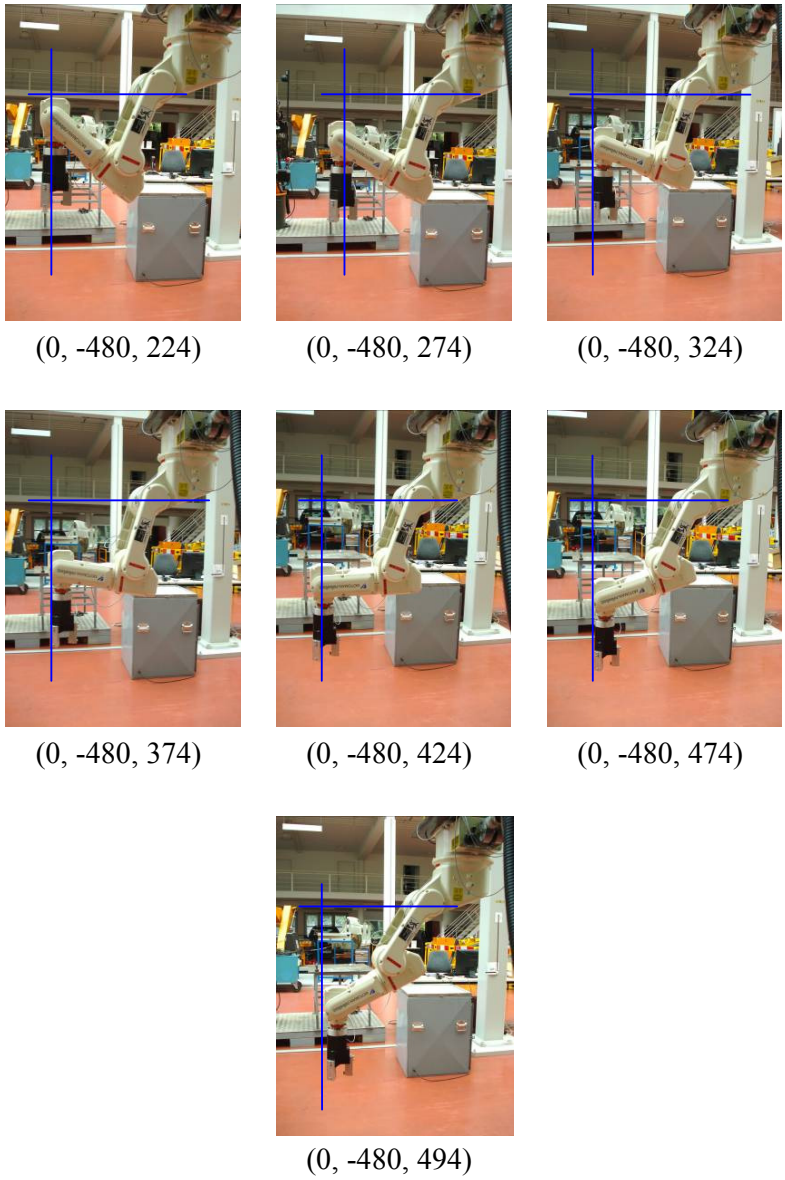


Figure 6.1 Robot points sequence

From the sequence is possible to see that the behavior of the robot corresponds with the behavior proposed in chapter three (see Figures 3.6 and 3.9 to 3.14).

The third test was to check the calculation of linear paths and also the performance of the robot while moving in straight line. To do this test a pen was

attached to the robot's tool (see Figure 6.2). The objective of this test is to use the robot to draw straight lines (see Figure 6.3).

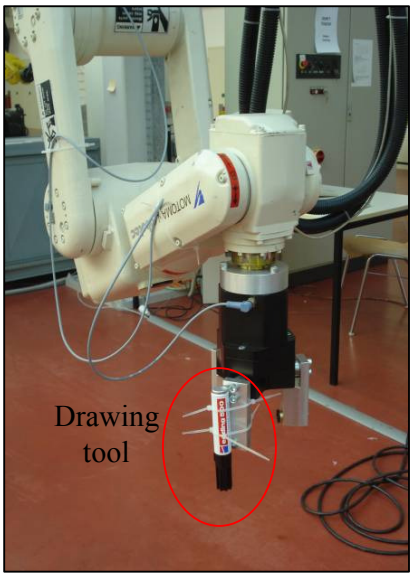


Figure 6.2 Drawing tool for the third test

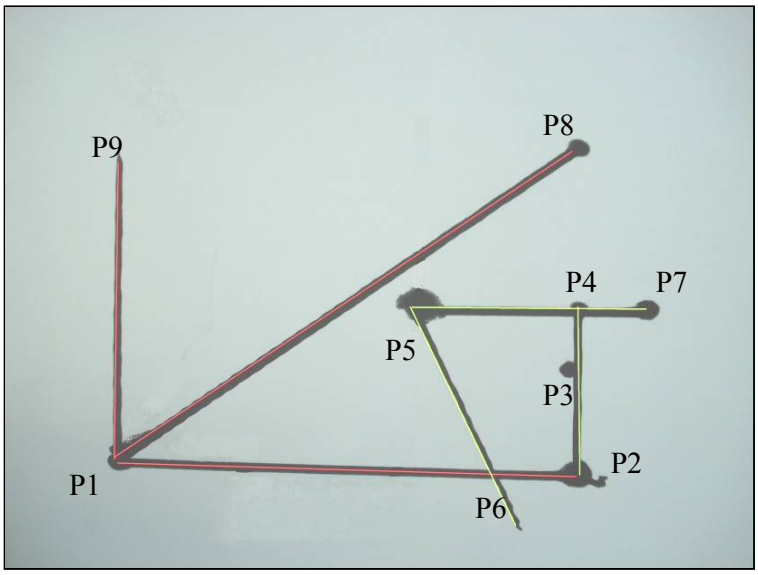


Figure 6.3 Lines drown by the robot

Figure 6.3 shows a favorable result of the test, because is possible to appreciate straight lines between the points, this means that the procedure propose in chapter 3.2 to calculate linear paths works okay. Also means that the robot does not have problems moving following linear paths.

6.2 Evaluation of the system Performance

The final test is a complete test of the whole system. This test has the purpose of shows the performance of the collaboration between the laser sensor and the Motoman to do a pick and place operation following the procedures developed in chapter 3.1.

To run this test an external client program will be used. The client has the function of manage the pick and place operation. Also to have a faster network the client is connected to the Motoman PC-GUI, and the Motoman PC-GUI is connected to the Laser sensor PC-GUI. In that way the client can control the whole system only sending commands to the Motoman PC-GUI (see Figure 6.4)

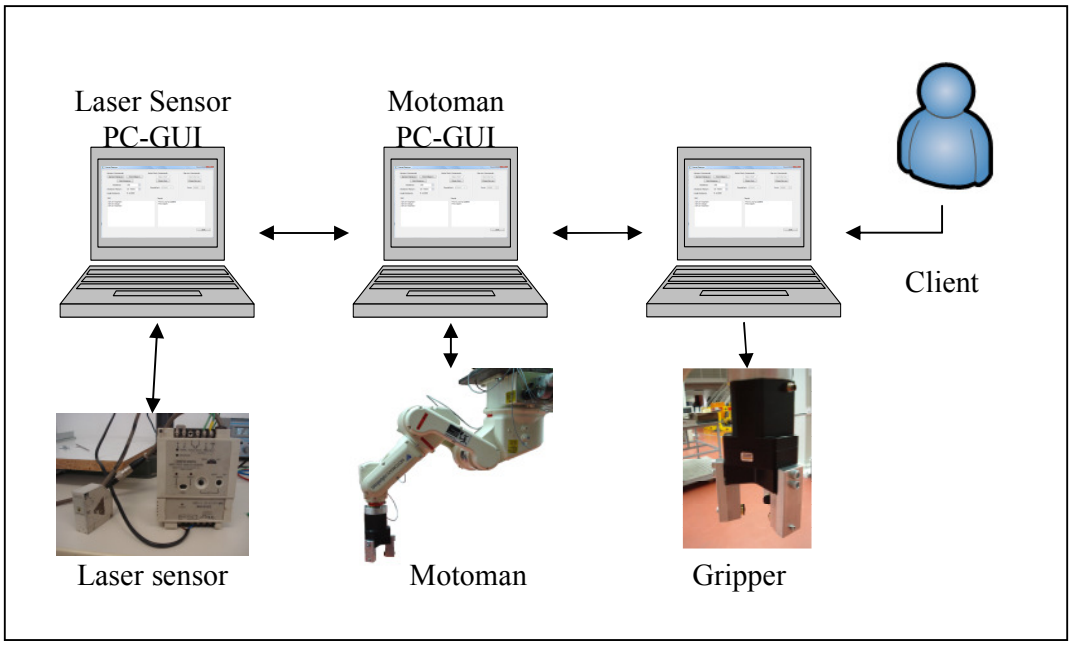


Figure 6.4 Structure of the test system

In order to connect the Motoman PC-GUI with the laser sensor PC-GUI the program was modified one more time. The modification consists in the addition of a new class that allows the communication between both programs. The class is sensortrial and runs a client to connect with the laser sensor PC-GUI (see Figure 6.5 and 6.6).

With this change the Motoman PC-GUI adds two new commands to control the laser sensor. Te commands are:

- SensorFind, puts the laser sensor system in search mode.
- SensorDistance, ask to the laser sensor to measure the distance.

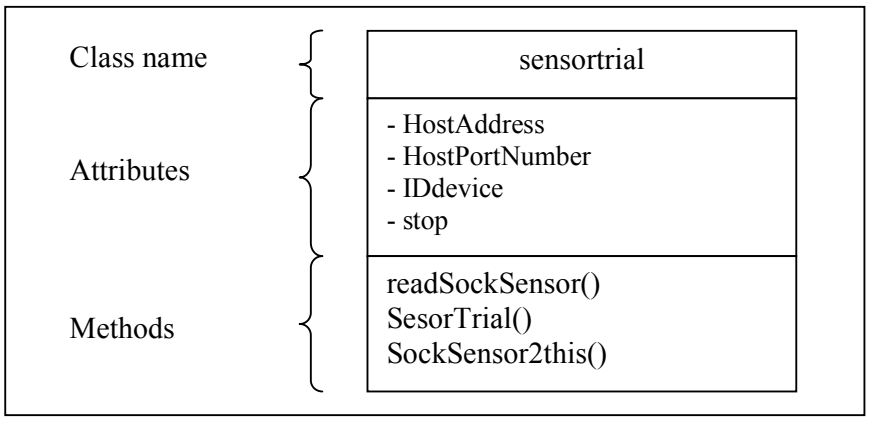


Figure 6.5 sensortrial class

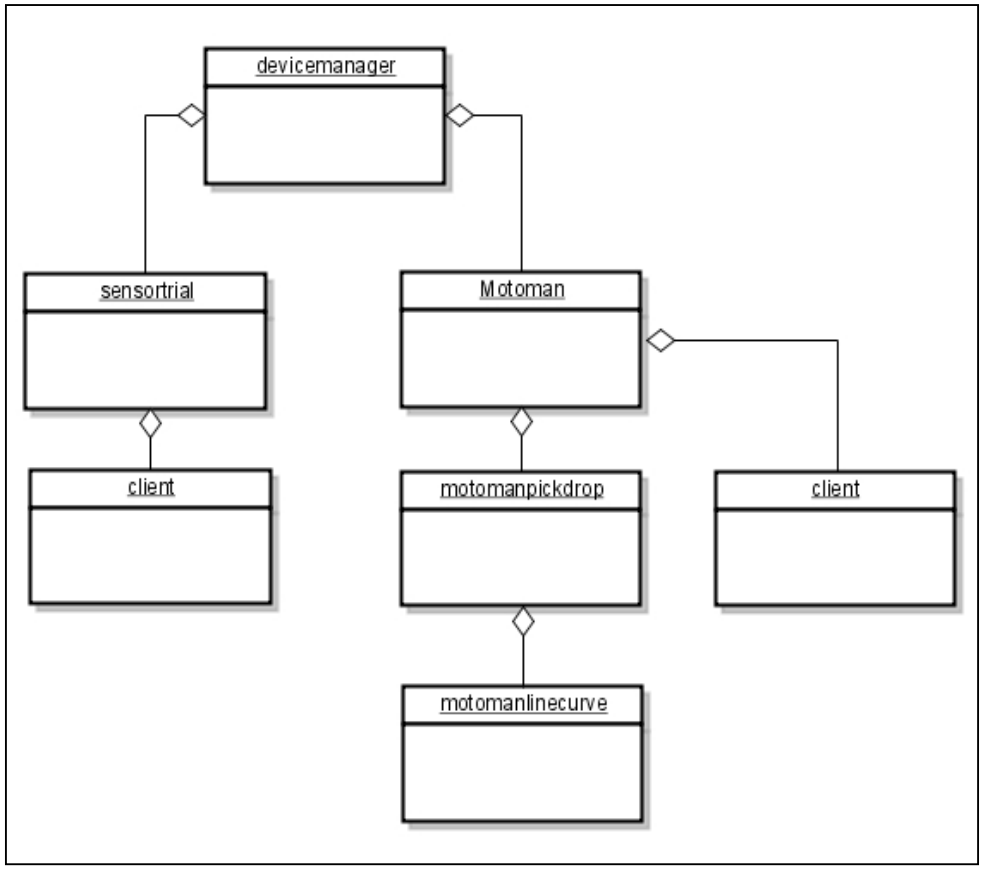


Figure 6.6 Relation between the Motoman object and the sensor object

The test were run in two levels, software level and hardware level. The software level shows the performance of the PC-GUIs when they have to communicate with other devices. The hardware level shows the collaboration of the Motoman, the laser sensor and the gripper to move the objects. The test will be a piling operation, where the Object A will be placed on the object B.

Software level

Before start the test the IP address and port of communication for each program has to be configured. The configuration for the programs is the following one:

IP address: 127.0.0.1 , as all the PC-GUIs runs at the same PC the IP address is the same for all the devices.
Motoman PC-GUI server address, port 2000
Motoman server address, port 4000
Laser sensor PC-GUI server address, port 3000
Laser sensor PC-GUI serial port, port Comm1

To know if the performance of the software communication is the correct one the information display on each PC-GUI is used (see Figures 6.7 , 6.8 and 6.9).

And to follow the change of information (commands) between the net the following color code is used:

Commands send by the Client:,	blue
Commands send by the Laser sensor,	green
Commands send by the Laser sensor PC-GUI,	red
Commands send by the Motoman,	green
Commands send by the Motoman PC-GUI,	yellow

```
system: Looking up the host
system: Waiting for the connection
system: Connection established

system:SEND:SensorFind
Server:Sensor::SensorFind

system:SEND:MoveZFIND
system: READ
Server:Motoman::Command Received MoveZFIND
Server:Motoman::Is posible move to find
system: READ
Server:Sensor::ObjectFind
Server:Motoman::Command Received RobotStop
system: READ
Server:Motoman::Lineal Path Sent
system: READ
Server:Motoman::Position Reached
system: READ
Server:Motoman::0.19,-470.06,334.71,0.00,0.00,179.92

system:SEND:SensorDistance
system: READ
Server:Sensor::SensorDistance
system: READ
Server:Sensor::Distance:6.99972550096075
```

Figure 6.7 Client communication data

```
Default
GUI::Conection OK
GUI::Verifying the initial position
GUI::Error: Unknown message:: -ERROR! puts -1 !
GUI::Error: Trying to write to an unconnected device:Comm::Send::Motoman::Position Reached
GUI::Position ask?
GUI::Motoman::0.00,-470.03,199.99,0.00,0.00,180.00
GUI::Error: Trying to write to an unconnected device:Comm::Send::Motoman::0.00,-
470.03,199.99,0.00,0.00,180.00
GUI::SensorFind
127.0.0.1:1104
GUI::Error: Unknown message: SensorFind
127.0.0.1:1104
GUI::Error: Unknown message: SensorFind
Default
GUI::MoveZFIND
127.0.0.1:1104
GUI::Is posible move to find
Default
GUI::Sensor::Object_find
Default
GUI::RobotStop
127.0.0.1:1104
GUI::Error: Unknown message: RobotStop
Default
GUI::PathSend
Default
GUI::Position ask?
127.0.0.1:1104
GUI::Motoman::0.19,-470.06,334.71,0.00,0.00,179.92
Default
GUI::SensorDistance
127.0.0.1:1104
GUI::Error: Unknown message: SensorDistance
Default
GUI::Sensor::Distance: 6.99972550096075
```

Figure 6.8 Motoman PC-GUI communication data

```
----------TCP----------
Server Started
Server Close
Server Started
Client Connected
IN : Find_Object
OUT: Object_find
IN : Sensor_Distance
OUT: Distance: 6.99972550096075

----------Serial----------
Port Open
Port Closed
Port Open
OUT:
Object_Find
OUT: ┘
Distance: 255
Distance calculation: 6.99972550096075
```

Figure 6.9 Laser sensor PC-GUI communication data

As is appreciate in Figure 6.7 the client stars sending the command “SensorFind” which triggers the begging of the pick process which is a interchange of commands between all the programs. Also the figures show that the flux of information is made following a logical way and that there are not confusion between the systems when the information is transmitted. From this

result is possible to conclude that the PC-GUIs have a correct performance related to the network behavior.

About the pick and place procedure performance the commands sent by the client were:

SensorFind
MoveZFIND
SensorDistance
MoveZTAKE
CloseGripper
MoveZBACK

MoveRel(120, 0, 0)
SensorFind
MoveZFIND
SensorDistance
MoveZDROP
OpenGripper
MoveZBACK

The first six commands fulfills the task or pick the object A, the remaining seven commands place the object A above the object B. From this part is possible to see that the only thing that the system needs to know for pick or place an object are the coordinates X and Y of the position where the object will be took or placed.

Hardware level

The hardware level is explained with the sequence showed in Figure 6.10(next page) and Figure 6.11 (page 90).

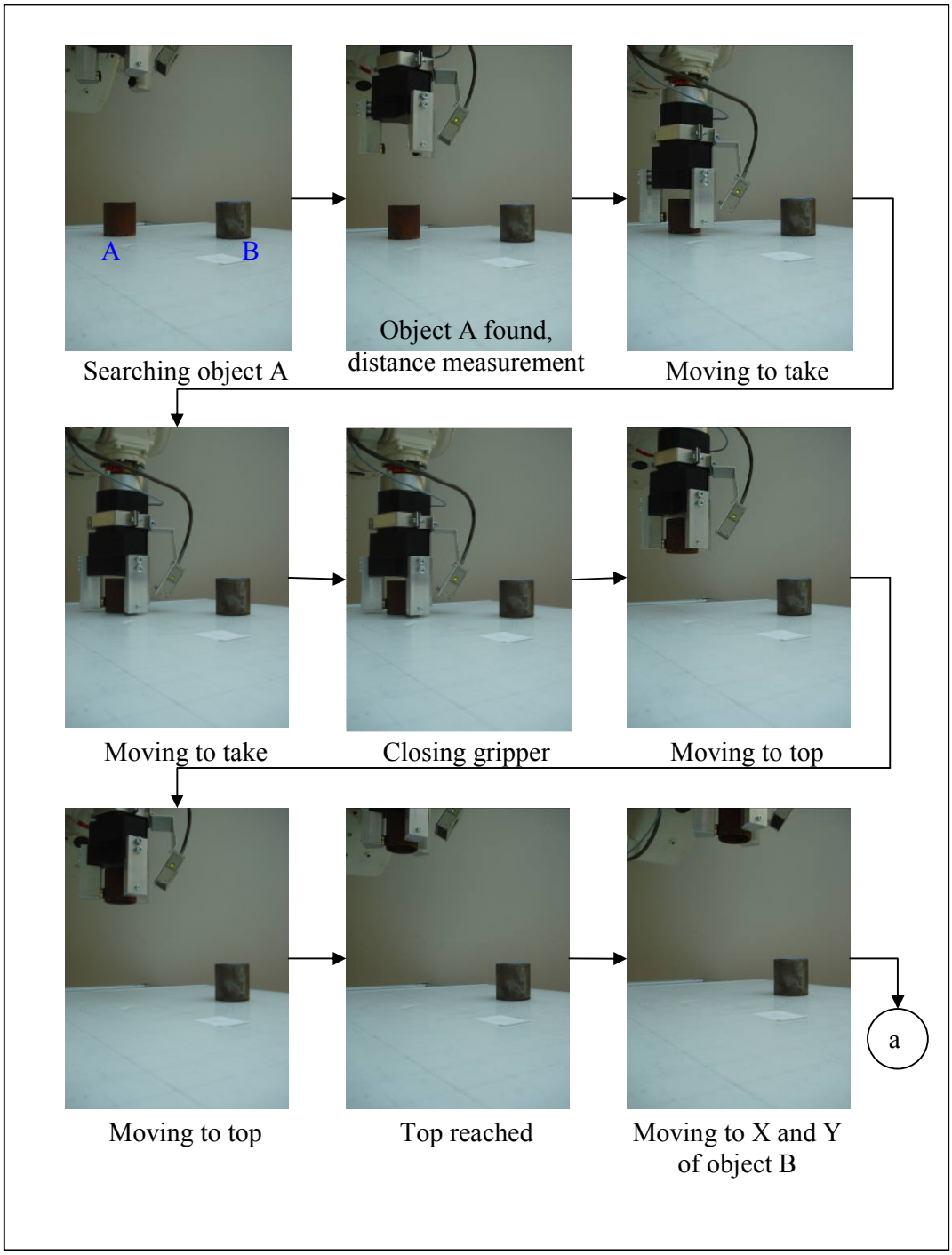


Figure 6.10 Piling process part A

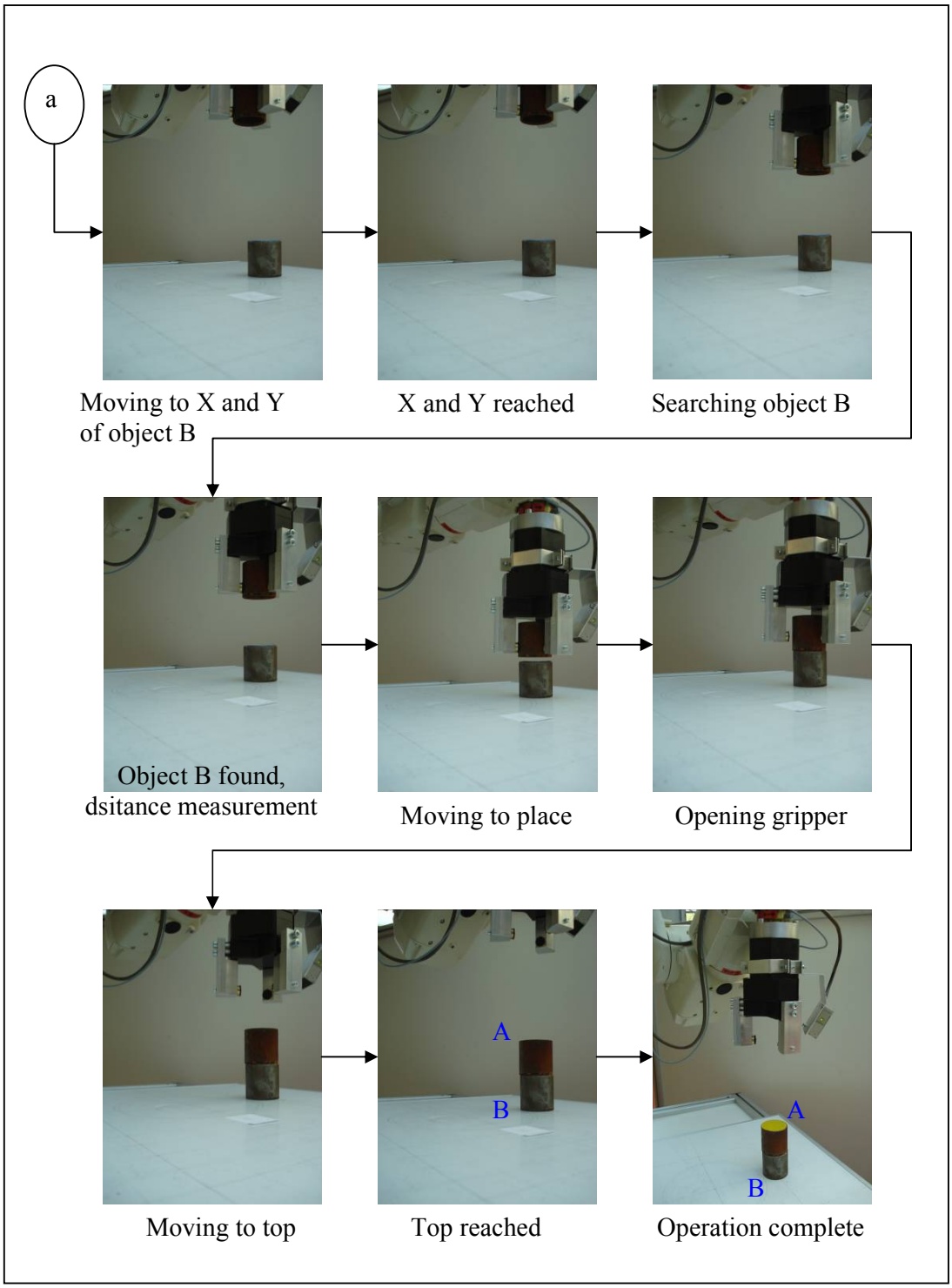


Figure 6.11Piling process part B

In the complete sequence can be appreciate the different steps needed to pick and place and object. These steps are the same steps explained in chapter 3.1. The sequence is the direct result of the commands sent by the client device to

the Motoman PC-GUI and the collaboration between the Motoman, the laser sensor and the gripper. From the sequence is possible to say that the pick procedure and the collaboration of device at hardware level have a correct performance.

So, considering the result of the test at software and hardware level the result is that the system has a correct performance when the Motoman system and laser sensor system collaborates to pick, place or pile objects.

7 Conclusions and Future Work

7.1 Conclusions

Within this project it could be shown that is possible to develop handling procedures that do not need explicit programming. Also was showed that is possible to have different devices working together to do handling procedures with out have to develop specific software for each device in order to do the required task. This means that the software developed to control each device allows the device to be part of multiply systems with any need of change one single line of code. By example the laser sensor system with its two main functionalities “Detect object” and “Measure distance” can be used to avoid collisions using the “Detect object” function, or to reproduce surfaces using the “Measure distance” function. In the same way the Motoman with the functionality or move point-to-point and move following linear paths can be used to develop task like welding and filming. This is possible because the PC-GUIs related to the devices were designed to make the devices be part of a net and be access by TCP, which makes very easy the communication between devices, because there is not need to develop any type of communication protocol for the system. The second thing that makes easy the communication is the use of commands that allows the access to the services provide by each device in an easy way.

The thing that makes the handling procedure developed really versatile is that the only data that has to be provided to the system which is managing the procedure are the coordinates X and Y where the objects to be taken are located, because the system will search the object using the Z axis. And that allows the robot to pick objects even from batches.

Also was demonstrated that is possible to use ranging sensors as proximity sensors. That was the case of the laser sensor which originally was a ranging sensor and now works as proximity and ranging sensor. One more thing that was demonstrated is that the robot can be used to draw planes just changing the gripper tool for a drawing tool, because when the robot is moving following paths it can move over the plane XY , YZ or XZ without vary the value of the remaining axis.

With the final test was showed that the software developed to control the Motoman and the laser sensor, was vital to fulfil the objective of show the collaboration of both devices to pick and place objects with any explicit programming.

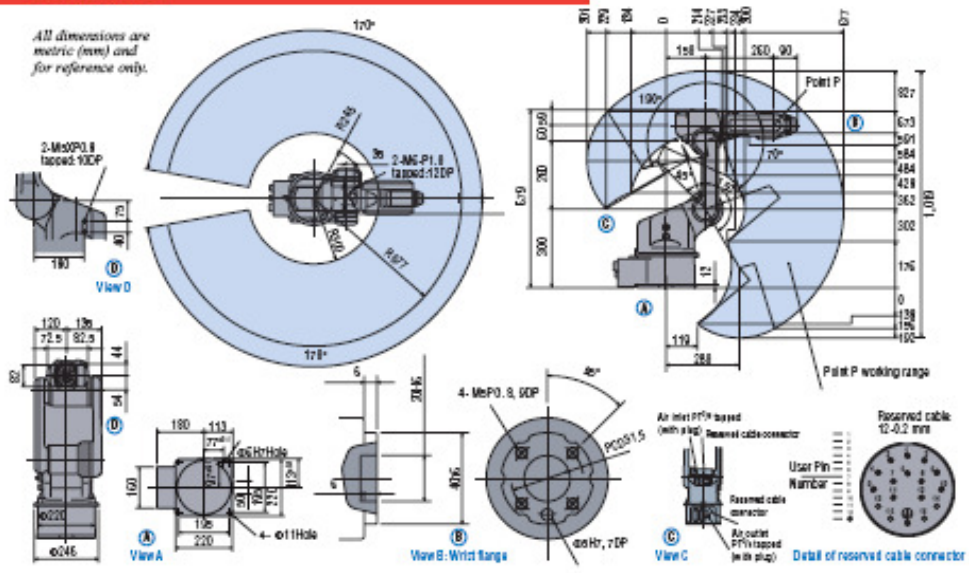
7.2 Future Work

As future work lets the task of improve the speed of the system when a handling procedure is done, because now takes between one minute and two minutes move one object from one point to another point. Also there is the idea of includes a camera in the system with the purpose to indicate to the manager device the coordinates X and Y of the objects to be picked and the coordinates X and Y of the position where the objects will be place. The camera also will have the task of check if the new position for the object is not to near to other objects that can damage the robot during the handling procedure.

Appendix A

A.1 Motoman KNICKARMROBOTER SV3X Data Sheet (SV3X Robot)

SV3X Robot



SV3X SPECIFICATIONS

Structure		Vertical jointed-arm type
Controlled Axes		6
Payload		3 kg (6.6 lbs.)
Vertical Reach		1,010 mm (48.1")
Horizontal Reach		677 mm (26.7")
Repeatability		±0.08 mm (±0.001")
Maximum Motion Range	S-Axis (Turning/sweep)	±170°
	L-Axis (Lower Arm)	+150°/-45°
	U-Axis (Upper Arm)	+190°/-70
	R-Axis (Wrist Roll)	±180°
	B-Axis (Bend/Pitch/Yaw)	±135°
	T-Axis (Wrist Twist)	±350°
Maximum Speed	S-Axis	210°/s
	L-Axis	170°/s
	U-Axis	225°/s
	R-Axis	380°/s
	B-Axis	380°/s
	T-Axis	420°/s
Approximate Mass		38 kg (66.2 lbs.)
Brakes		All axes
Power Consumption		1 kVA
Allowable Moment	R-Axis	5.39 N • m
	B-Axis	5.39 N • m
	T-Axis	2.94 N • m
Allowable Moment of Inertia	R-Axis	0.1 kg • m²
	B-Axis	0.1 kg • m²
	T-Axis	0.05 kg • m²

XRC 2001 CONTROLLER SPECIFICATIONS

Structure	Free-standing, enclosed type
Dimensions (mm)	750 (w) x 860 (h) x 550 (d) (29.5" x 33.9" x 21.7")
Approximate Mass	70 kg (154.4 lbs.)
Cooling System	Indirect cooling
Ambient Temperature	During operation: 0° C (32° F) to +45° C (113° F) During transmit and storage: -10° C (14° F) to +60° C (140° F)
Relative Humidity	90% max. non-condensing
Primary Power Requirements	3-phase, 200/220 VAC (+10% to -15%) at 50/60 Hz
Grounding	Grounding resistance: ≤100 ohms Separate ground required
Digital I/O	Specialized signals (hardware): 12 inputs/3 outputs General signals (standard max): 40 inputs/40 outputs Expandable to 256 inputs/256 outputs
Position Feedback	By absolute encoder
Drive Units	Servo packs for AC servomotors
Axes/Detail	Software servo control
Program Memory	5,000 steps and 3,000 instructions
Pendant Dim. (mm)	200 (w) x 325 (h) x 77 (d) (7.9" x 12.8" x 3.0")
Pendant Buttons Provided	Teach Play, Remote, Servo On, Start, Hold, Emergency Stop, Edit Lock
Safety	Emergency Stop Pushbuttons, 3-position Deadman, Brake release switches Meets ANSI/RIA R15.06-1999 standard

MOTOMAN
a YASKAWA company

MOTOMAN CORPORATE HEADQUARTERS
805 LIBERTY LANE, WEST CARROLLTON, OHIO 45449
TEL: 937.847.6200 • FAX: 937.847.6277
WEB SITE: www.motoman.com

A.2 ATmega 16 Features

Features
• High-performance, Low-power AVR® 8-bit Microcontroller

• Advanced RISC Architecture
– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 x 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16 MHz
– On-chip 2-cycle Multiplier

• Nonvolatile Program and Data Memories
– 16K Bytes of In-System Self-Programmable Flash
Endurance: 10,000 Write/Erase Cycles
– Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
– 512 Bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
– 1K Byte Internal SRAM
– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface
– Boundary-scan Capabilities According to the JTAG Standard
– Extensive On-chip Debug Support
– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features
– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture
Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC
8 Single-ended Channels
7 Differential Channels in TQFP Package Only
2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
– Byte-oriented Two-wire Serial Interface

– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator

• Special Microcontroller Features
– Power-on Reset and Programmable Brown-out Detection
– Internal Calibrated RC Oscillator
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby
and Extended Standby

• I/O and Packages
– 32 Programmable I/O Lines
– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Operating Voltages
– 2.7 - 5.5V for ATmega16L
– 4.5 - 5.5V for ATmega16

• Speed Grades
– 0 - 8 MHz for ATmega16L
– 0 - 16 MHz for ATmega16

• Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
– Active: 1.1 mA
– Idle Mode: 0.35 mA
– Power-down Mode: < 1 µA

References

[MISH] *Measurement, Instrumentation, and Sensors Handbook*
CRCnetBASE 1999
© 1999 by CRC Press LLC
Page 285 to 295

[MH] *THE MECHATRONICS H A N D B O O K*
The University of Texas at Austin
Austin, Texas
Page 460

[DA16] Datasheet: Atmel-ATmega16: 8 bit AVR Microcontroller with 16kBytes In-System Programmable Flash, Rev 2466N–AVR, October 2006

[MTT] Development of a Joystick Control for Telenavigation of a Hybrid Robot System
M. Sc. Thomas Kunkel
Master Thesis

[ZAC] Kooperation von Roboter und Sensoren auf der basis von “Web Services” in einem draftlosen Sensor / Aktor Netzwerk
Zeuxis A. Conde de Felipe
Diplomarbeit, Aachen 2007

[MD] MOTOMAN KNICKARMROBOTER SV3X data sheet
YASKAWA company
October 2003