YILDIZ TECHNICAL UNIVERSITY
FACULTY OF ELECTRIC AND ELECTRONICS
DEPARTMENT OF COMPUTER ENGINEERING

SENIOR PROJECT

DEVELOPMENT OF THE SIMULATION AND THE GRAPHICAL CONTROL INTERFACE OF A MAP BUILDING ROBOT

Project Supervisor: Assist. Prof. Sırma YAVUZ

Project Group

İbrahim ÖK, 03011044

İstanbul, 2007

TABLE OF CONTENT

ABBREVIATION LIST

GUI	Graphical User Interface
RF	Radio Frequency
IR	Infra Red
IDE	Integrated Development Environment
SWT	Standard Widgets Toolkit
UML	Unified Modeling Language
UART	Universal Asynchronous Receiver Transceiver
USART	Universal Serial Asynchronous Receiver Transmitter
SFD	Start Frame Delimiter
SYNCH	Synchronization
PC	Personal Computer
(M)Hz	(Mega) Hertz
MMC	Main Microcontroller
NACK	Negative Acknowledge
I/O	Input / Output
EMI	Electromagnetic Interference
ESC	Electronic Speed Control
A/D	Analog to Digital Converter
ACK	Acknowledge
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
D/A	Digital to Analog Converter
DC	Direct Current
DIP	Dual In-Line Package

FIGURE LIST

TABLE LIST

PREFACE

I take it is a must to express my gratitude to my elder brother Mehmet Ök, who supported me all through my work and Assistant Professor Sırma Yavuz, who didn't denied her very precious help and guidance from me in my work for the completion of the graduation project in Department of Computer Sciences, Electrical and Electronic Engineering Faculty, Yıldız Technical University.

ÖZET

Robotların insan hayatındaki yeri azımsanamayacak ölçüde artmaktadır. Bu artışla beraber robotun insanlarla ve içinde bulunduğu ortamla etkileşime girmesi de kaçınılmaz hale gelmektedir.

Bu çalışmanın amacı çevrenin ön topolojik bilgilerine ya da bir takım referans nesnelerinin yer bilgisine sahip olmadan ve ihtiyaç duymadan robotun yörüngesinin ve konumunun belirlenmesi işleminin simülasyonunu ikinci bir bilgisayardan gelen komutlara göre gerçek zamanlı olarak yapmaktır. Simüle edilen robot, bilinmeyen bir ortamda bilinmeyen bir noktadan harekete başlayarak bir taraftan bu ortamı algılamak, bir taraftan da kendi yerini tahmin edebilmek de ve başlangıç noktasına döndüğünü de algılayabilmektedir. Proje eş zamanlı konum belirleme ve harita çıkarma işinde kullanılan bir robotun simülasyonu ve kumanda arayüzünün oluşturulmasını kapsamaktadır. Projede kullanılmakta olan robotun donanım aksamı bölümdeki hocalarımız tarafından gerçekleştirilmiştir. Sistem iki bilgisayar üzerinde çalışmakda olup bu bilgisayarlar seri porttan haberleşmektedirler. Programlama için Java dili kullanımıştır.

1. bilgisayarda, araca iletilecek olan kumanda işaretleri 2. bilgisayardan gelen verilere gore oluşturulmakta ve 2. bilgisayardan gelen veriler izlenmektedir (sensör bilgileri, motor bilgileri, yön bilgisi ve kamera görüntülerinin izlenebileceği bir pencere içeren bir arayüz).

2. bilgisayarda ise bir mekânın planını oluşturmak üzere otonom hareket etmekle görevli araba simülasyonu çalışmaktadır. 1. bilgisayardan komut geldikçe (örn. sağa dön, hızlan vs.) bu arabanın komutlara uygun hareket etmesi ve kendi yörüngesini çizdirmesi izlenmektedir.

ABSTRACT

The place of the Robots in human life is on the increase. With this increase the interaction of robots and people and the environment they act in is becoming more and more indispensable. One of the pillar stones of this interaction is the ability to define the environment and locate self in it.

The aim of this study is to perform simultaneous trajectory and location determination simulation based on commands issued by a second computer. This is done by without having any topological information of the environment or any reference object location. The simulated Robot, starting its movement from an arbitrary point in an unknown environment, is able to map the environment meanwhile also able to locate itself and determine whether it has returned to the point it started. The project is on the simulation of a locating and mapping robot and creating the control interface of it. The hardware part of the robot has been built by our faculty professors. The system works on two computers which communicate through a serial port. Java has been used for programming.

In the first computer command signals to be sent to the robot has been created based on the data from second computer and also data from the second computer has been monitored (sensor data, engine data, directional data and a window interface to view the camera on the robot). On the second computer, a car simulation program is running to create a plan of the environment. If a command from the first computer comes (e.g. turn right, accelerate etc.) the environment plan is created as the car follows these commands the process can also be observed.

1. INTRODUCTION

Before we move to the steps of the project let's see something about structures which are going to be used. The main structure here is a robotic system.

1.1. Robotic Systems

When people think of a robot, they dream of a machine that is able to walk like people, has humanlike behaviors, and more importantly a machine which is able to think and decide like a human is considered. The sci-fi films people watch has a great impact on this thought. C3PO can be given as a typical example from the Star Wars movies we have seen years ago and still delightfully follow the new episodes.

As people watch these movies, robots were being used may be not in daily life but in factories. The best example for this is the Robotic arms that carry parts, paint or weld. These robots, which are totally different than the robots in our minds, but are like human arms, are used in many parts of the busy and sensitive factories doing the monotonous work in patience instead of humans.

With the developing technology, robots in time evolved from arms into machines that can percept the environment, react to it and able to travel between two points. These kind of robots are called "the mobile/itinerant robots". One of the best examples for such robots is the "Sojourner" designed by NASA to do researches on Mars.
Even though the word "robot" was first used in 1920, the first concepts and robot-like machines go back as early as 3000 BC. It is known that Ancient Egyptians, Greek and Anatolian civilizations developed automatic water clocks. In Ilyada of Homeros manmade female servants are mentioned. It is written in the old books that in 100 BC an Alexandrian scientist has made automatic doors, fountains etc that run on water or steam power.

In the closer eras it is told that Leonardo Da Vinci had a walking mechanical lion.

In this progress of robot technology, El Cezeri (12th century AD), not very well known in western cultures, has many advanced theories and applications on robotic technology considering his era.

When we consider all samples called robot, it is clear that to define and categorize them in terms of humanlike functions and appearance is not correct. Therefore the following definition can be used instead: “Machines that have functions and behaviors that are alike the live”.

And as the basic properties of robots:
It can be said that they are “self-sufficient in terms of functions and programmable”.

Marked properties of the robotic systems on process level are:

1-Robots can perceive their situation and location, and the environment that they exist in.
2- They can make decisions based on comparing predefined duties and perceived environment, their location and situation.
3- By applying the decisions they can change the environment, their situation and location.
By realizing these functions by machines and in a system, a kind of intelligence is given to manmade machines and systems. Here, the intelligent behaviors of the machines and systems are the marked behaviors perceived by man.

It has been accepted that there are important differences between human intelligence and machine’s and system’s intelligent behaviors in terms of basic concept and structures. It is expected that these differences will diminish with the current and near future developments.

The expectations on the Robotics Technology applications in near future will develop based on the results obtained in robot technology research centers and university research studies. Industrial expectations are building lighter and faster robots. The most

important factor in robotics technology is to decrease the production cost for the manufacturing engineers. Similarly industrial mobile robots will decrease the production cost and increase the output.

1.2. Definition Of The Robot For Which The Interface Will Be Built

The robot used in the project, as can be seen in the picture, is in the form of a car.
IR sensors are widely used in robotic sciences and automation, process control, remote sensors and security systems in the world. They are generally used in basic object detection as convergent sensor, counting, ground sensing, position control, depth and distance monitoring, obstacle sensor, and computer visual systems [24,25,26].
The robot in the project finds out whether there are obstacles around and their distance thanks to IR sensors and transfers the data to the computer through RF module. As a map of the environment is created based on these signals, also control commands are created and sent to the robot by the computer.

Robot and the Computer have created an autonomous system in the project.

The basic movements that the robot perform:

Turning: turning to the degree in the command signal Finding a wall: When the robot is left in an environment the robot advances until it reaches an obstacle or wall. When it finds a wall it places itself parallel to it. As in figure 2 the robot senses the wall in position 1 and turns around until side sensors have the same values. In positions 2 and 3 the differences between sensor values has been displayed in light colors. In position 4 the values of two sensors are the same.
Now that the wall and the robot are in parallel the wall following step can start.
Wall Following: The robot will continue along the parallel wall as long as the difference in values between two side sensors is below a certain threshold.

Deceleration: It is expected that the robot will decelerate before starting the turn.
Acceleration: Following the turn process, if there are no obstacles in front, the robot will accelerate.

Above is the introduction of robot and its interface to be used in the project.

Below mapping and interface creation processes, which makes up the core of the project, will be explained.

1.3. Project Scope

The aim of the project, as mentioned above, is to create an interface for the robot and ability to create a 2D graphical representation of the environment that the robot moves in, based on the data received from the robot.

The robot's environmental data is gathered by the switches and IR sensors and transferred to the computer through serial communication ports. Switches provide crash data, and 6 IR sensors provide the distance data to the surrounding objects.
The first computer gathering the data used in this study is called the server and also has the interface for the audience to follow the process. It also has a command interface through which users can send command signals to the robot. The second computer processes the data from the first computer (Server) and creates a 2D graphical representation of the environment.
The design and application of these programs in the computers defines the scope of the project.

2. FEASIBILITY STUDY

Feasibility studies are based on the following criteria. These are:

2.1. Technical feasibility study

Before we move to the steps of the project let's see something about hardware cost.

2.1.1. Establishing the hardware to be used

The cost of the robot in the study is about 4348.6 YTL as calculated by department staff.

Additional hardware required for the project are serial to serial cable, usb to serial converter, ,joystick for controlling, interface and 2 P4 computers.

2.1.2. Establishing the software to be used in the study

Before we move to the steps of the project let's see something about software cost.

2.1.2.1. Basic Software Tools

The programming of the robot has been completed in C programming language. However the interface software has also been programmed in Java. Netbeans 5.5 has been used for its easy to use GUI (graphical user interface) tools. Serial communication libraries (javax.comm) and joystick controller libraries (javax.lwjgl) have been used in addition to the software.

2.1.2.2. Why Java and Netbeans

Nowadays the cost decreases as much as the software development stage shortens. Therefore it is very important to use the IDE (integrated development environment)

which speeds up the process. Even though currently the programming language used in network or multi-user applications are C#, Java, Delphi or C++, the chosen programming language and IDE for this study is Netbeans. The cause of this choice is mentioned above.

Alternatively if C++ were used, run time performance would be faster; if C# were used, communication and graphical design processes would be faster. But the success of the Netbeans in developing applications has made us choose it.

Furthermore Java libraries provide most advantages and make easier communication and joystick control.

2.1.2.3. The Operating System

Windows XP Professional operating system is chosen based on the fact there are many programs for this system.

2.2. Economical feasibility study

A special attention has been paid to have the legal and licensed software in the system design and hosting. The cost is detailed below.

2.2.1. Software Cost

Before we move to the steps of the project let's see something about software cost.

Table 2.1 Software Cost

Windows XP Professional	$299.00 (microsoft.com)
Netbeans 5.5	0 $
TOTAL	$299.00

2.2.2. Hardware Cost

Before we move to the steps of the project let's see something about hardware cost.

Table 2.2 Hardware Cost

P4 PC computer (Simulator)	$500.00
Centrino Notebook (Controller)	$1000.00
Joystick	$15.00
Usb2Serial converter for notebook	$20.00
TOTAL	$1535.00

2.3. Legal Feasibility Study

The project will be realized in Department of Computer Sciences, Electrical and Electronic Engineering Faculty, T.C. Yıldız Technical University. All copyrights belong to the department. Without consent and allowance of the department, the program cannot be used or copied in any other program or separately in part or as a whole.

2.4. Alternative Feasibility Study

In the application developed it was also possible to create mapping functions only without the graphical command interface. But that way no intervention and interaction would be possible externally. This way both the data gathered is displayed (visuals, speed and direction data) and also the robot can be freely moved through the command interface.

3. SYSTEM ANALYSIS

Before we move to the steps of the project let's see something about system analysis.

3.1. System Analysis Steps

1. Obtaining the technical data from robot's micro-controller designer Hilmi Kemal Yıldız about the hardware parts.
2. Obtaining comments of our instructors of the project on the remote control of the Robot.
3. Feasibility studies of the project and determining the required software and hardware.
4. Designing and creating UML diagrams of the classes based on the communication protocol.
5. Definition of inter-class relations and implementation of the classes.
6. Designing and implementing the interface and classes for the robot simulation.
7. Creation of the command interface and definition of class relations.
8. After the creation the testing of the command interface and robot simulation interfaces and updates.
9. Comments of our instructors and updates
10. Handing in the project.

3.2. System analysis defining possible problems and solutions

During system analysis, defining possible problems will provide the chance to take necessary precautions and will also help to see future problems of following stages, system development for example. By creating the information system based on the possible problems the designer will have advantages in the future and also prevent unnecessary expenses. We can list possible problems and solutions as follows:

a. Users need to be informed on the command interface and its use. Through this we can prevent misuses and inability to use it effectively

b. In the system requirements command interface and communication parts should be developed as planned. If any of these have a small mistake whole system is affected.
c. Properties for software maintenance and all steps taken to develop the system should be documented. In other case, when we return for maintenance or for a problem we might pay a great price
d. Project should be tested by someone out of the design team. Thus we can find out any problem. The following questions answers should be positive by the testing personnel: can you control all functions of the robot through the remote control, Is there anything wrong with the communication of robot and the remote control, and Does the simulation have all properties of the robot?

3.3. Defining the hardware

There will be 2 computers in the project. The computers will use the serial port for communication. The computers and their uses are listed below.

The computer should be mobile since it will be used for development of command interface design and robot control. For this reason a Fujitsu Siemens Amilio 1451 Laptop will be used. Properties are as follows:

The simulation computer can just be a desktop computer. The project's simulation software will be able to disabled by connecting the real robot. For this reason the following computer with listed properties is used:

These two computers communicate through the serial port. For the notebook a USB2SERIAL converter has been used. In addition to this a 12 button 3 axis joystick has been used as part of the remote control in the Laptop.

A camera and receiver has been used to get pictures from the robot. Receiver has been linked to an external TV card for the Laptop.

3.4. Defining the Software

It is useful to analyze the project in two parts as we did in the hardware.

Eclipse 3.2 has been used for the control computer. To ease the interface design process Windows Builder Plug-in has been installed on Eclipse. The programming language used is Java and SWT as mentioned in the feasibility. Finally the Laptop has Windows XP installed.

Netbeans 5.5 has been used for the simulation program. Just like in control this program has been written in Java, too. The Desktop Pc has Windows XP installed.

4. SYSTEM ARCHITECTURE

The simulation in the project is an identical copy of the slam robot in our school. The responses to inputs, behaviors, hardware and all other aspects have been simulated through software. Robot communicates with the control Pc over RF signals. The simulation and control PC communicates through the same protocol over wire. To give information about the robot's parts will help us to understand what the simulation program does. The following is the robot's hardware properties.

The work flow of the designed program is as follows:

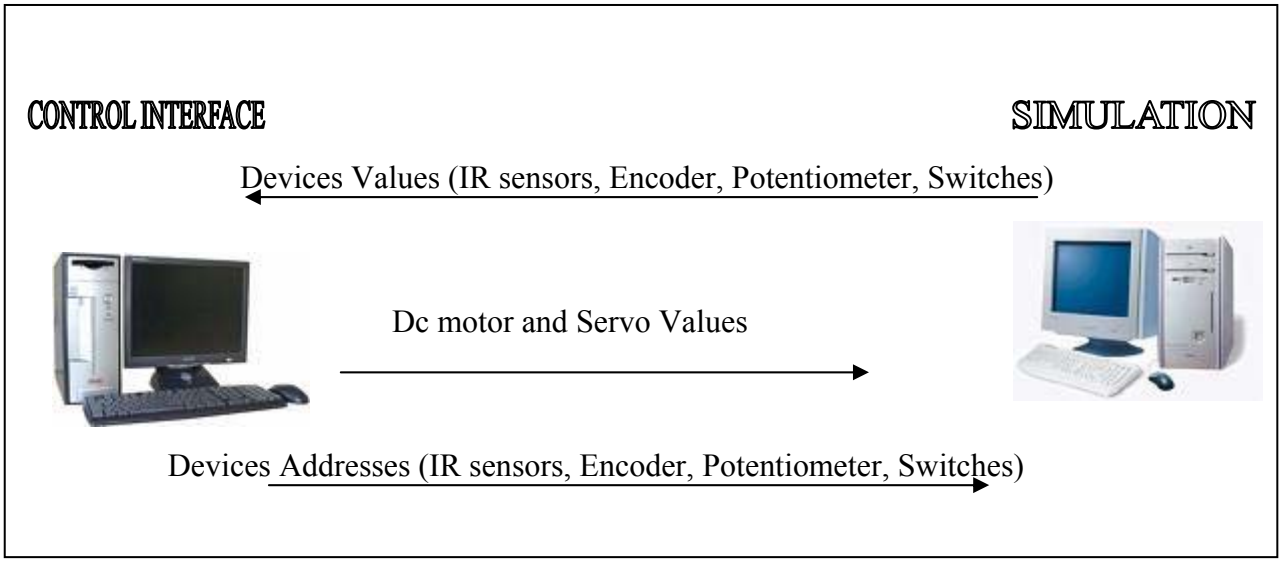

Figure 4.1. The work flow of System

Microcontroller unit of robot is responsible for acquiring the data of surrounding objects by means of infrared-sensors and switches. Besides this, it is responsible for acquiring the data of the direction and the speed of the robot by means of potentiometer and encoder located on the front wheel of the robot.

4.1. Microcontroller

A microcontroller could be likened to the “brains” of the robot. It can be programmed by the designer to accomplish the task at hand. It is responsible for sending commands to other individual systems in the robot, receiving data from external devices, and coordinating activities. For this specific project, a number of microcontrollers would have been appropriate, but two have been chosen for their ease of use, availability of support, and because they enclosed all of the requirements needed. The Microchip 18F452 PIC and 12F675 microcontrollers have been chosen. Two different microcontrollers have been chosen, and the table below shows the features of each microcontroller.

Table 4.1. Microcontrollers

Features	**Microchip 18F452**	**Microchip 12F675**
Max Clock Frequency (MHz)	40	8 (Internal)
RAM (Bytes)	1536	64
Flash Memory (Bytes)	32768	1792
EEPROM (Bytes)	256	128
PWM Outputs	2	-
Timers	5(1-8Bit, 3-16Bit, 1-WDT)	1(1-8bit)
A/D	8/10-Bit 8 Channel	10 bit 4 Channel
Serial Interfaces	USART, I2C, SPI	-
I/O Pins	33	6
Package	40-Pin DIP	8-Pin DIP

The need for two different microcontrollers becomes apparent when one investigates the matter. The 18F452 doesn’t provide enough counter and timer for the requirements of the encoder. Therefore the 12F675 was used for determining the direction of the front Wheel by way of encoder.

4.1.1. Microchip 18F452

The 18F452 is the main microcontroller, and from this point on will be referred to as the Main Microcontroller (MMC). MMC will be in control of all the functions of the robot. The MMC will make use of it's USART to communicate serially with the RF receiver, in order to receive the data packets from the vision system. It will also use two data pins to communicate with external devices such as the 12F675, and the eight analog to digital on chip converters can be used to interface to local sensors. The 18F452 also comes in a convenient and popular package, a 40-pin DIP.
The usual connections are made, such as power, clock, and ground, along with the specific functions that the robot needs to utilize, such as the USART for receiving data from the RF receiver and some ports to interface to other parts of the design.

4.1.2. Microchip 12F675 Encoder Data Retrieve

The 12F675 has not all of the necessary features to perform the function of the main microcontroller, but it can use to implement to determine the front wheel direction. The 12F675 solves this problem. It listens the encoder indexes when it change direction, it interrupts MMC and inform it about the new direction. Therefore MMC only counts the index and when direction changed it stores the results, clear it's counter and goes on counting.

4.2. RF Communication

The RF part of the project has been greatly simplified because both of side computer and pc use the same RF module. It is a 433.92 MHz RF receiver WIZ-SML-IA from Aurel. This is a transceivers for point-to-point data transfer in half-duplex mode and its card integrates a 100kbps XTR transceiver. This device will be used to receive the data in serially, and should integrate perfectly into our current system through the USART on the MMC. The MMC listens the RF Module for incoming data by using USART interrupts.

4.3. Servo

Figure 4.2. Futaba S3003

A servo motor is comprised of a DC motor, a gear train, a potentiometer connected to the output shaft and an integrated circuit for positional control. All this hardware is packaged in a black casing. A servo motor is used because it is a low power device that has precise control in the angular position of the front wheel. The servo used in the system is the Futaba S3003, as shown in Figure 3.1. It's specification is shown in Table 3.1.

Table 4.2. Specifications of Servo

Speed	*0.23 sec/60 degrees at 4.8 v*
Torque	*44.4 oz/in (3.2 kg/cm) at 4.8 v*
Size	*1.59"L x o.78"W x 1.42"H w/o output shaft*
Weight	*1.31 oz (37.2g)*
Connector	*"J" type with approx. 5"*

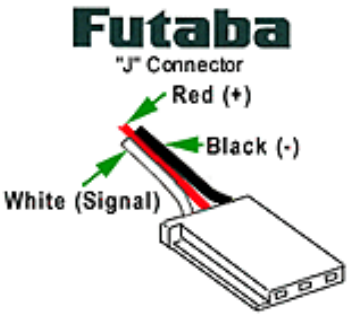

Figure 4.3. "J" type connector

As shown in Figure 6, this specific servo uses a “J” type connector and has three leads to operate the servo. One for 4V – 6V power supply, second for ground and the third is a control input signal that carries a pulse width signal. The control signal is used to control the angular position of the servo. According to the Futaba s3003 datasheet, the servo optimum operating frequency is 50Hz. The servo is tested using a microcontroller that generates the control signal from 1ms to 2 ms of a 20ms period (50Hz). In order to achieve 50Hz from the PIC18F452, software PWM implementation is needed in our case. Hardware implementation of PWM is faster than software implementation. But we can not run the servo motor by way of hardware PWM. Because the minimum frequency that can be achieved by hardware PWM is greater than the operating frequency of servo. Besides this, minimum frequency depends on the crystal frequency used in the circuit. Minimum frequencies that can be achieved by hardware PWM are shown in Table 3.3.2.

Table 4.3. Min. frequencies and Max. periods of Hardware PWM

Crystal Frequency	Minimum frequency of Hardware Pwm	Maximum period of Hardware Pwm
4 Mhz	250 Hz	4 ms
10 Mhz	625 Hz	1.6 ms
16 Mhz	1000 Hz	1 ms
20 Mhz	1250 Hz	0.8 ms
33 Mhz	2000 Hz	0.5 ms
40 Mhz	2441 Hz	0.4 ms

The pulse train results are shown in Figure 3.3 and Table 3.3. A pulse with 1.5ms puts the servo to the center. The pulses with 1.0ms and 2.0ms are used to produce 45 degrees anticlockwise and 45 degrees clockwise rotations respectively. It is also important that the servo shares the same ground as the microcontroller or the servo may run unpredictably.

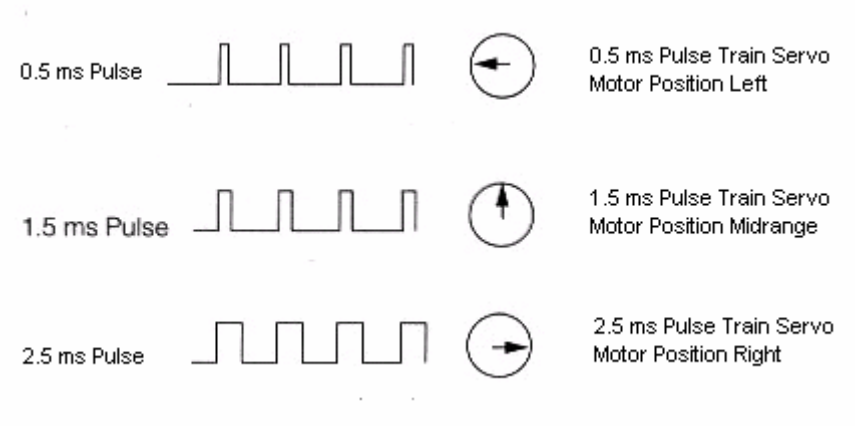

Figure 4.4 Pulses and Directions of Servo

Table 4.4. Pulse Duty and Angles of Servo

Pulse Width Duty / ms	Angle / degrees
0.5	0
1.0	45
1.5	90 (center)
2.0	135
2.5	180

4.4. Infrared Sensors

These units report the distance to a given target as an analog voltage. Using an analog to digital converter on the microcontroller, one can determine the range to a given target. Infrared sensors function by emitting a special wavelength of light invisible to the human eye unaided, infrared, and then calculating the angle of return of the light. This method uses simple trigonometry to determine the distance the sensor is from an object. However, if the light is blocked or reflected away from the sensor, it will not be able to

detect the object. The method of measuring distance is shown below in Figure 4.5. Infrared Distance Measurement.

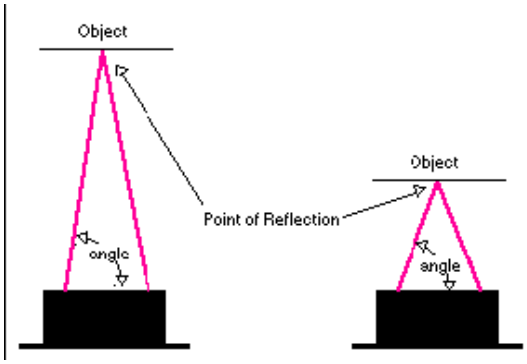

Figure 4.5. Infrared Distance Measurement

The infrared sensor, which would fit the project most appropriately, is the Sharp GP2D12 infrared sensor. It has a range of 10 cm to 80 cm, which should have sufficient range for our purposes, but it was mainly chosen for the cost. The minimum range of the sensor is incurred because the angle of return becomes too wide for the width of the sensor, causing the emitted infrared beam to be unable to be sensed.

Figure 4.6. Sharp GP2D12 Infrared Sensor

Most infrared sensors do not have a maximum range as far as the GP2D12, unless they are significantly more expensive.

4.5. Dc Motor and ESC

DC motors are more common and easily available. Dc motor was used to move the robot. They are generally more powerful, easier to interface to, and allow the robot to move faster. However, despite these advantages, they also have a serious disadvantage. They have no method for exact control of the distance to travel. They simply turn on and drive and slowly turn off. This is a problem for our project because the robot needs to have a method by which it can drive a specified distance and stop to survey its surroundings. It would be possible to implement our own control device to determine the number of rotations of the wheel by having an encoder get clock pulse each time the wheel has rotated. However, this measurement is not very precise and therefore the DC motor must be examined critically in order to interface it properly with the rest of the mobile platform and achieve the desired goals.

The control of motor was provided by an esc. The working principle is same as servo motor. A pulse with 1.5ms stops the dc motor. The pulses between 0.5ms and 1.5ms are used to move forward and the pulses between 1.5ms and 2.5ms are used to move backward.

4.6. Encoder

Incremental encoders are sensors capable of generating signals in response to rotary movement. In conjunction with mechanical conversion devices, such as measuring wheels or spindles, incremental shaft encoders can also be used to measure linear movement. The shaft encoder generates a signal for each incremental change in position. The encoder used on robot is Hengstler RI 32. Its resolution is 1024. It has 6 leads. Their descriptions are shown in table 3.6.1. Yellow leads generate one clock pulse in a cycle. Green and white leads generate 1024 pulses in a cycle. But between green and white there is 90 degrees phase difference. This difference make possible to realize the direction of the encoder. This is shown in the figure 3.6.1.

Table 4.5. Encoder lead descriptions

Lead	Description
Red	DC 5 V/ 10 - 30 V
White	Channel A
Green	Channel B
Yellow	Channel N
Black	GND
Yellow-Black	Alarm

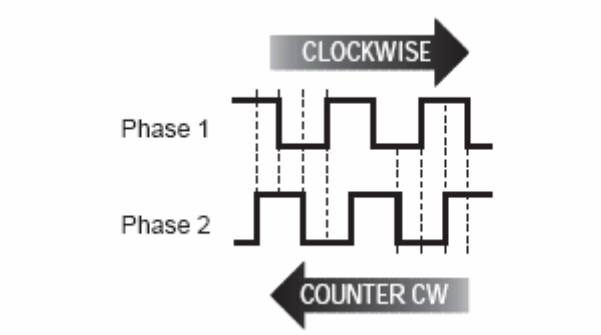

Figure 4.7. Channel A and Channel B

4.7. Potentiometer

Potentiometer is an instrument for measuring electrical potential, on the robot; a potentiometer was attached on the front wheel in order to obtain direction information. Because servo motors are reliable but some times they fail because of obstacles.

4.8. Switches

Switches are devices which open or break an electric current; there are eight switches on the robot to notice collision.

5. COMMUNICATION PROTOCOL PACKET SCRUCTURE

The protocol between slam robot and the control has been exactly used for the protocol used between simulation and the control. This protocol structure has been developed by our colleague who programmed the microcontrollers. Information on the protocols is given below.

In this Project the robot is responsible for performing commands sent by computer. That is to say robot is slave and computer is master. If computer sends a command, robot carries out this commands, and replies. But in some special circumstances robot acts on its own. Collision is one of the examples of these circumstances. When the robot collides with something, it stops moving and makes computer aware of collision state.

Integrating communication through the onboard UART required a significant amount of research in new areas. However, it was not overly difficult to implement for testing purposes. A loop can be created that travels along a series of bytes that have the values to be sent stored in them and send them one at a time.
The packet structure that must be sent to the robot or computer is formatted as follows.

Figure 5.1. Packet Structure

5.1. SYNCH

The synchronization byte comes first. This is the flag of the data, if it comes, next byte (SFD) is expected. The pattern of synch byte is 01010101 (0x55h) because this is the most difficult byte that can be send by RF due to the nature of RF communication.

5.2. SFD

The second byte is start frame delimiter; this byte is determiner of boundaries. The pattern of sfd byte is 01111110 (0x7Eh). If this byte is correct, receiver expects the length byte; else it goes back and waits for synch byte again.

5.3. LENGTH

The third byte is length of the bytes which follow this byte. In other words, this declares the sum of the length of address byte, control byte, payload bytes and crc bytes. It can be up to a maximum of 255. Specifying the packet length is necessary because each packet can contain a different amount of data. For example, a packet containing all devices information will have the maximum amount of payload. However, a simple HELLO packet has just 7 bytes. Therefore, specifying the packet length is essential to coordinating messages properly.

5.4. ADDRESS

The fourth byte shows the sender and receiver addresses. First fourth bits of this byte show sender address and the next four bits show receiver address. For instance, if sender address is 0xFh and the receiver address is 0x0h then address byte must be 0xF0h.

5.5. CONTROL

The fifth byte is control byte. This byte contains control data. Control bytes and their meanings are shown in the table 4.1.1.1. It is possible to add a new control byte because of new situations. But these control bytes are enough to handle robot control for now. These bytes make difference sense for robot and computer. For example collision packet can only be sent by robot to computer.

Table 5.1. Control Bytes

Control Byte	**Meaning**
ACK (0x00)	Last packet received successfully
NACK (0x01)	Time out while receiving last packet
COLLISION (0x02)	Robot collided with something
ERROR (0x03)	Error in data packet
HELLO (0x04)	To verify the establishment of the connection

ACK means the last packet has been received by computer. When computer sends a packet and receives its response on time, next packet's control byte will be ACK. NACK means that robot's response to the last packet has not been received on time.
COLLISION means robot collided with something, ERROR means data is received by one of the side but there is error on data. HELLO is a connection establishment byte. At the beginning of connection HELLO packet sends to robot, if robot takes this packet, it resends the same packet to the sender.

5.6. PAYLOAD

Payload part of packet is the main data that want to be transmitted. In this project a flexible packet structure was used. On the robot a unique address was assigned for every device, and devices are separated into main two groups. In the first group there are readable devices. Sensors are an example of these devices. Computer only sends readable devices' addresses and robot reads the value of these devices and sends to computer. In the second group there are writable devices. Servo is an example of these devices. If computer wants to set the value of servo, it must send the address of these devices and in the following byte value of these devices must be sent. Devices and their addresses are shown in the table 7. Readable and writeable devices can be distinguished from each other by checking the most significant bits of address byte. If it is a readable device, the most significant bit of address byte is 0, if it is a writable device the most significant bit of address byte is 1.

Table 5.2. Devices' Addresses

Device Address	Device Name
0x00	IR sensor on the front
0x01	IR sensor on left side front
0x02	IR sensor on left side back
0x03	IR sensor on the back
0x04	IR sensor on right side back
0x05	IR sensor on right side front
0x06	Potentiometer on the front wheel
0x07	Encoder
0x08	Switches (Every bit of this byte shows one of the switch respectively)
0x80	Dc motor
0x81	Servo

This packet structure also gives an opportunity to create different size of payload. We can only send one of the device information or all of the devices information. Devices' sequence in the payload is not important.

In some environment EMI can decrease the performance of RF communication. If packet size increases, at the same time the number of corrupted packet increases. In this circumstance we can reduce the packet size by demanding devices' value one by one, this can increase the performance.

5.7. CRC

CRC bytes are two bytes shows 16 bit CRC checksum. A CRC (cyclic redundancy check) is a type of hash function used to produce a checksum – a small, fixed number of bits – against a block of data, such as a packet of network traffic or a block of a computer file. A CRC "checksum" is the remainder of a binary division with no bit carry (XOR used instead of subtraction), of the message bit stream, by a predefined (short) bit stream of length n, which represents the coefficients of a polynomial. The checksum is used to detect errors after transmission or storage. CRC is computed and appended before transmission or storage, and verified afterwards by the receiver to confirm that no changes occurred on transit. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. For substantiality, we used the 16-bit CRC-16-CCITT polynomial $x^{16} + x^{12} + x^5 + 1$ (0x1021h).

6. COMMUNICATION PROCESSES

Before we move to the steps of the project let's see something about communication process.

6.1. Simulation Side (Robot Side) Communication Process

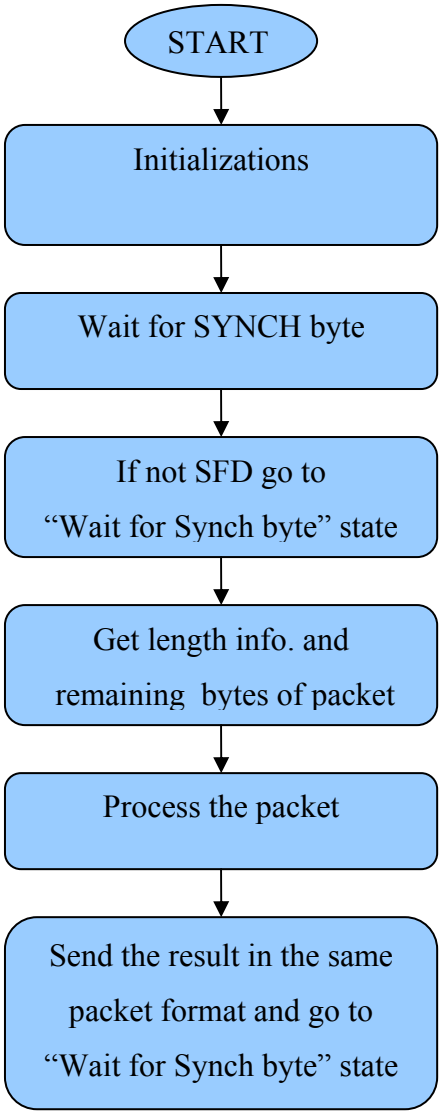

Figure 6.1. Flow Diagram of Main MicroController(18F452)

When robot is turned on, initial values of variables must be assigned. Also timers, ADC and interrupt settings must be done. After this configuration, robot waits until one of the

packets is correctly received. First it waits for SYNCH byte. When it comes, it checks next coming byte. If it is SFD, it considers the next coming byte is length of remaining bytes, else it goes back and waits for an other SYNCH byte. When packet is completed, it starts to process the packet. All of these waitings functions are implemented by using USART interrupts when new byte comes it goes to interrupt service routine and takes the new byte. It increases the system performance.

When robot starts to process the packet, first it verifies the data integrity by using crc checksum. If packet is corrupted it informs computer about the corrupted packet by using ERROR control byte. If there is no crc error, it controls address information, if it is correct, it processes the payload or else it does nothing because the packet is not sent to this robot. Addressing is an important point if there is more than one robot.

If everything is ok, it checks control byte, if it is HELLO byte it resend the same packet to the sender. If it is ACK or NACK, it processes the payload. Check the address of device if there is readable device; it reads the value of that device and writes to the output packet with device address. If there is writable device, it sets the value of that device to the new value.

After packet processing finishes, it sends the output packet to computer with ACK control byte. When sending packet to computer it uses interrupt mechanism also.

6.2. Computer Side Communication Process

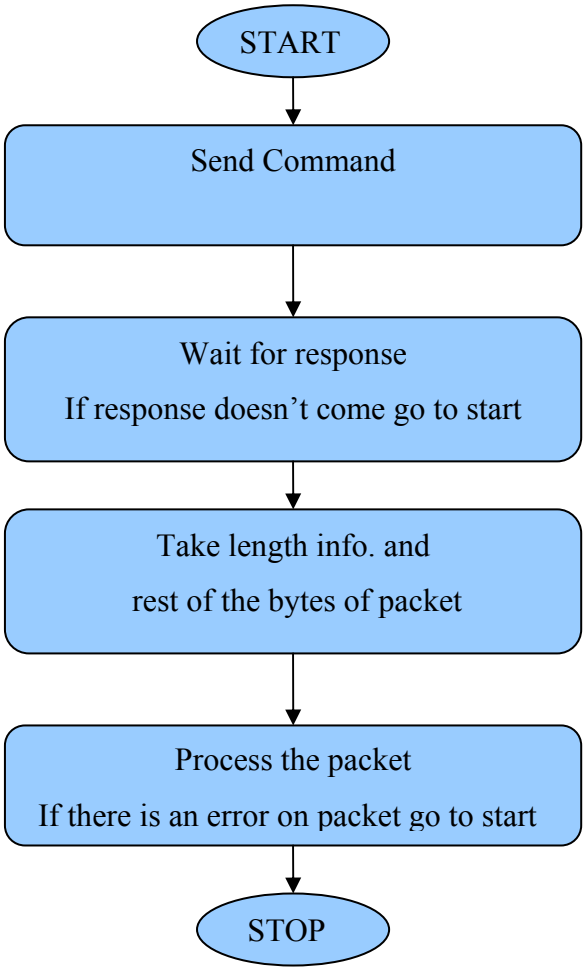

Figure 6.2. Flow Diagram of Computer Side Comm. Process

User interface and pc side protocol was written in Java. When we want to send command to robot, we have to create a packet in correct packet format. After we send the packet, computer waits for a predefined time if response does not come it resends the same packet. If response packet comes, computer verifies the data integrity by using crc checksum. If packet is corrupted it resends the packet else it verifies the address. If address is incorrect it resends the same packet, else it processes the packet and gets the results from the payload.

7. PROCESSES OF SYSTEM

We need to analyze the flow of the program from two perspectives:

1. Command interface
2. Simulation

First we need to analyze the command interface. As it can be seen in the UML diagram the command interface consists of 15 classes. The main class of the interface "Nterface1" class has been created using the swt technology through windows API. In the 5 tabs that comes to screen when it is run has remote control functions in the first 3 tabs and functions in the other two tabs used to change the command interface and program dynamically.

The following are done in the first tab, second tab and fifth tab;

1. When the button on the bottom left "portları bul" is pressed (find ports) "getPorts()" is called in from the "GetAndOpenPort" class.
2. The port selected from the combo box calls "openPort()" method from the "GetAndOpenPort" class and the port is opened. At this stage "helloPacket()" from the CreatePacket class is called and a hello pack is sent and the link to simulation is established.
3. Following the connection we press "Başla" (start) button and communication with simulation is initiated and the interface is refreshed through the incoming and outgoing data packages. Also the map is drawn on the screen. The following are the steps taken when the button "start" is pressed.
    3.1 First the thread "AnaThread" is created and the thread starts to run.
    3.2 A package is created within the thread to be sent to the simulation. The package is sent. Response for the package arrives, the response is evaluated and the interface is updated, the map is updated through the incoming response.
4. Communication can be stopped by pressing the "stop" button.

5. “start”, “stop” and “pause” buttons can be used to control the music player embedded in to the interface.
6. We can calibrate the speed and angle through the “calibration” button
7. we can differently use the joystick in the fifth tab. The data from the joystick controls the speed and angle and controls the simulation. The updated data is sent to the robot so that it can follow the joystick’s speed and angle.

In the third tab all labels and scripts can be updated. That is through this update the interface can be translated into any language.

In the fourth tab we can dynamically add devices to the program. First we should add a groupbox control for the devices to be added and the devices should be added to this groupbox. In this stage the addresses of the devices can be changed. Through this the robot will be able to fit in the address changes easily.

For above mentioned update process 5 tables have been created in Access database. You can find the ER diagram of these tables in the “ER DIAGRAM” title.
Secondly we need to analyze the robot simulation. Robot simulation program runs on another computer and responds to the control commands like a “slam robot”. The processes of the simulation are as follows:

1. When program is run simulation is ready to receive commands. “getPorts()” and “openPort()” functions are started within the main thread and a port is opened and readied to receive commands from the command interface.
2. When a request from the command interface is received the “receiveAndSend()” method of the Main class is used to respond.
3. In addition the speed and the direction of the robot can be changed based on the commands from the command interface. This change is also seen on the map screen. This map simulates the environment that the robot is moving within. The

robot’s sensors and switches updates the data to be sent to the command interface through “cevreBilgisiAl()” method.

4. Simulation is designed to respond the requests from the command interface. When a request is received (which values are being set which values are being get is determined) a suitable respond is created and sent to the command interface.

8. DIAGRAMS AND SCREENSHOTS

8.1. Flow Diagram

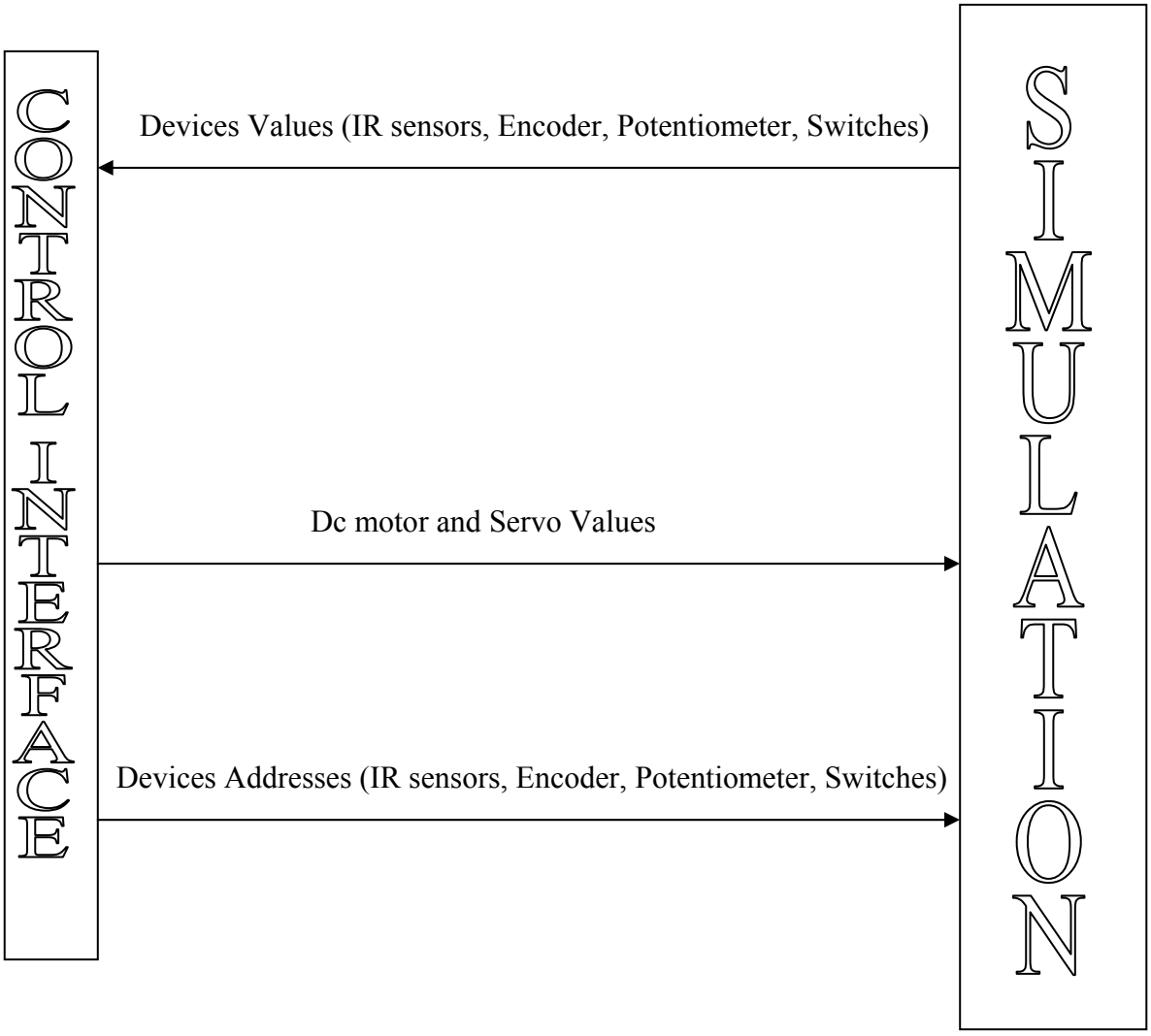

Figure 8.1. The flow diagrams of the designed system

8.2. ER Diagram

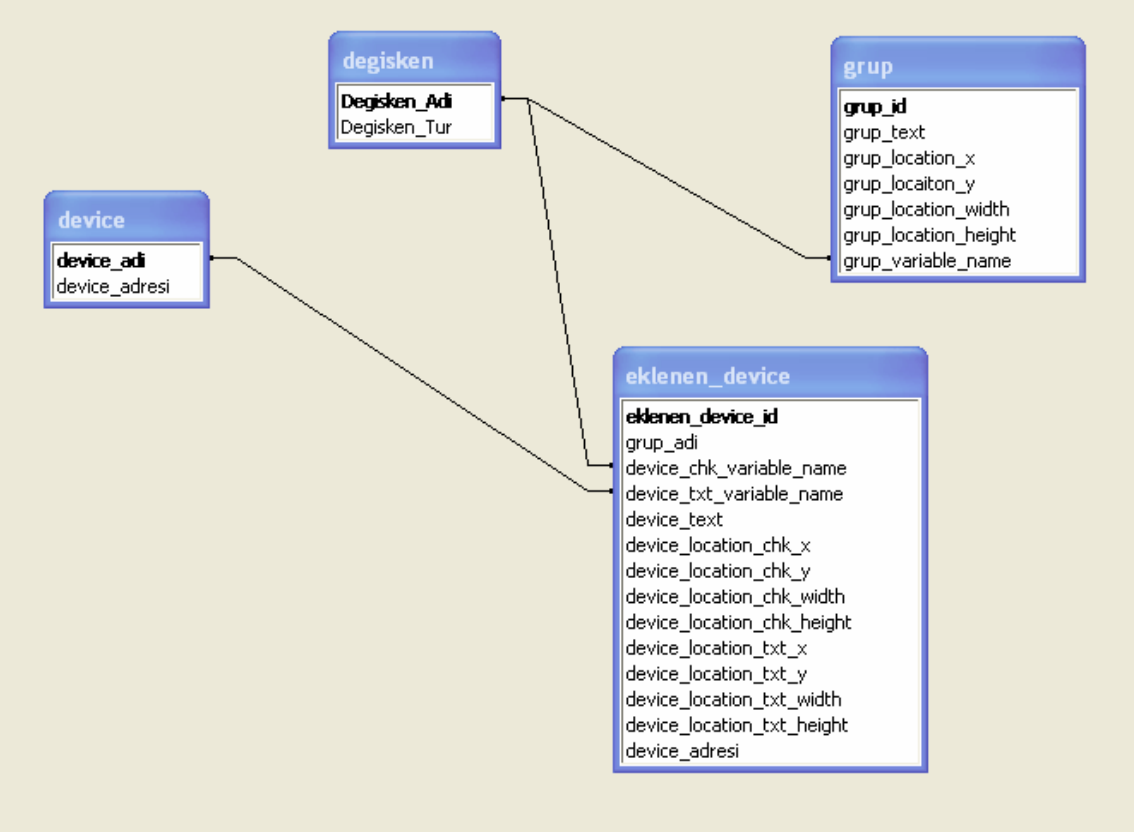

Figure 8.2. The ER diagrams of the designed database

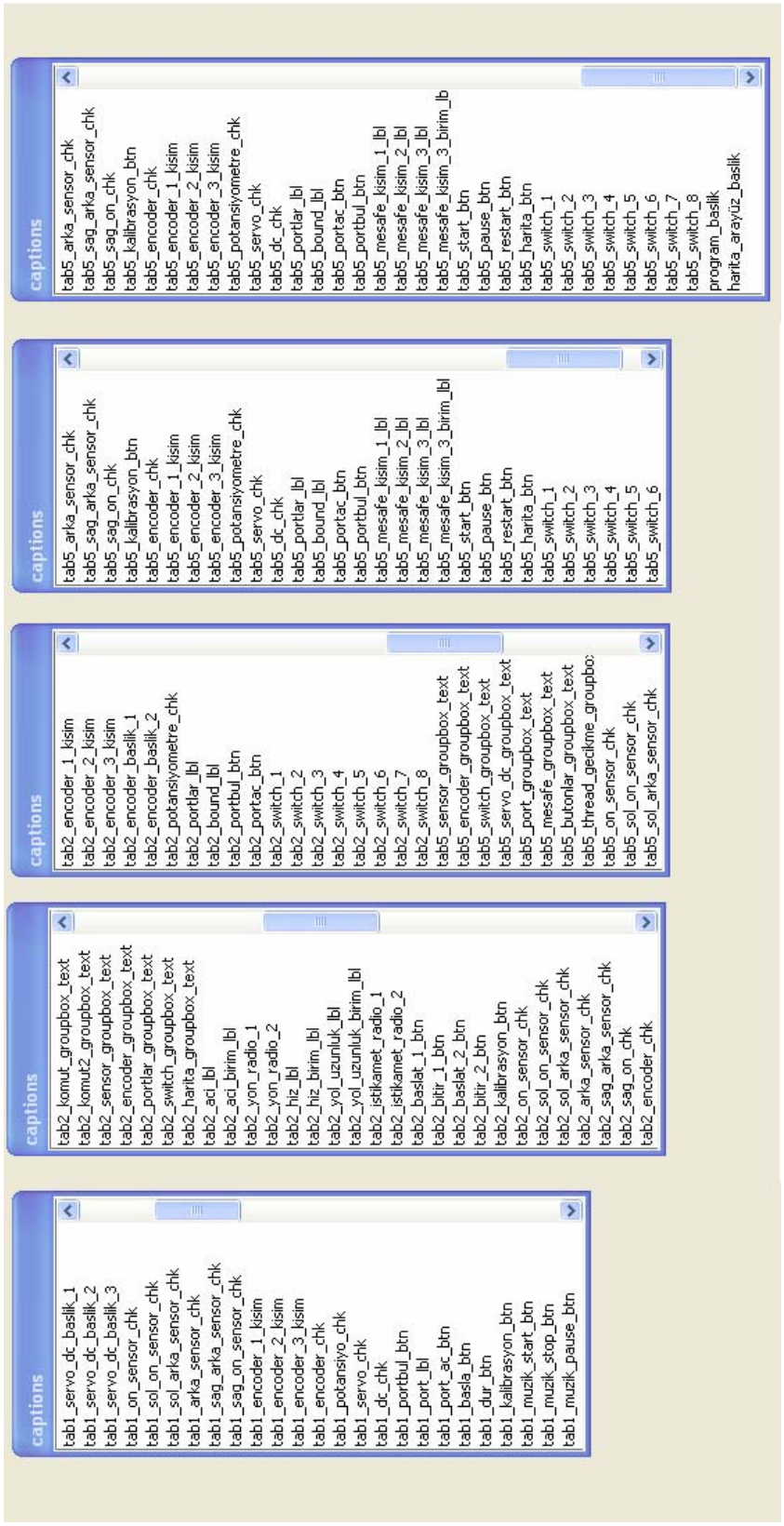

Figure 8.3. The ER diagrams of the designed database

8.3. UML Diagrams

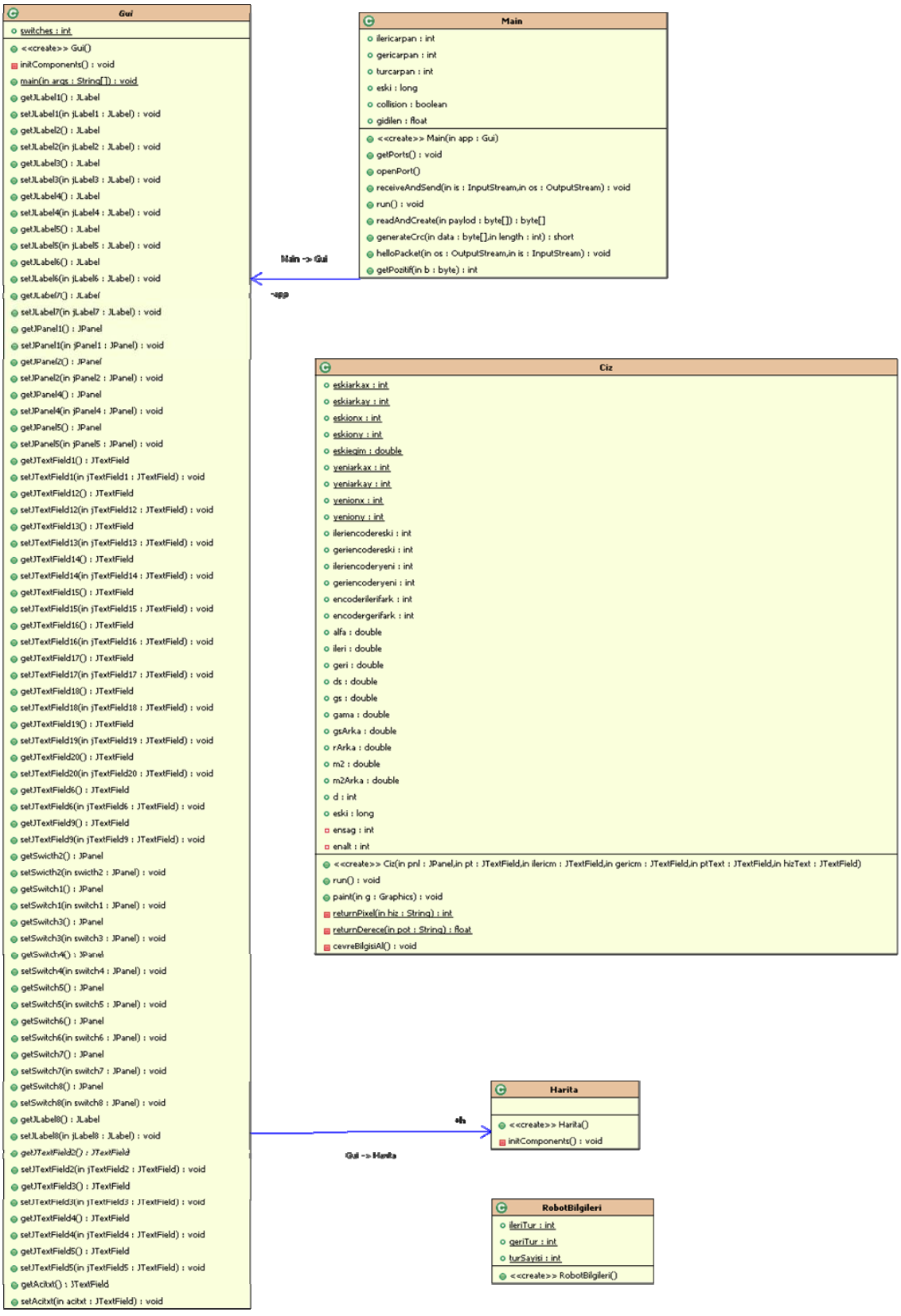

Figure 8.4. The UML diagram of the designed classes for Simulation Side

Figure 8.5. The UML diagram of the designed classes for Control Interface Side

8.4. Screenshots

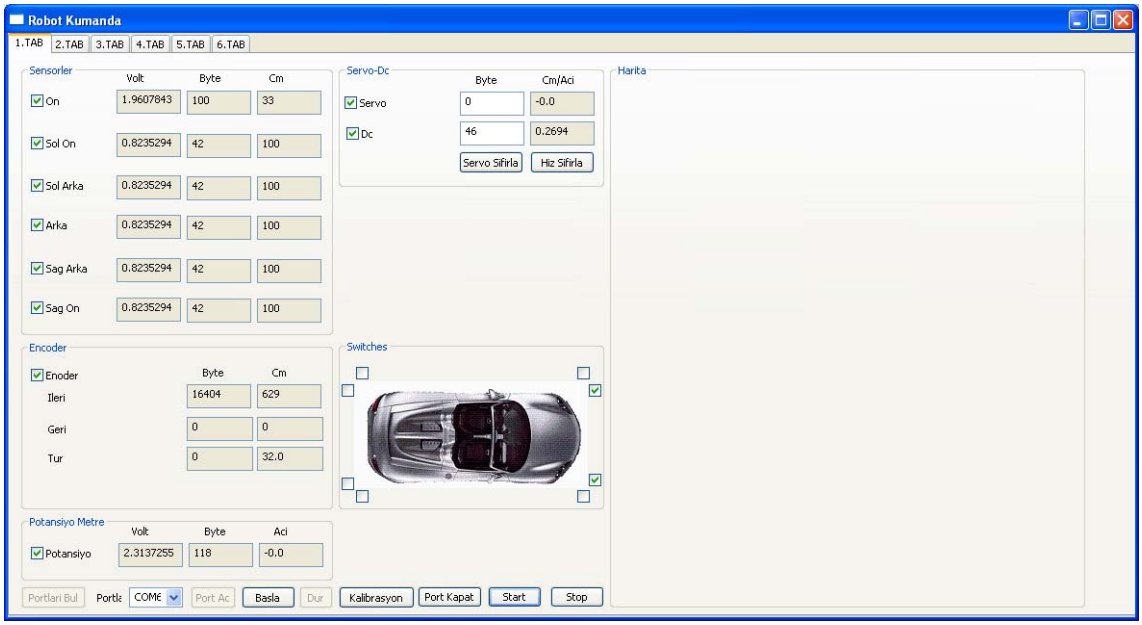

Figure 8.6. The First Tab Of Control Interface Side

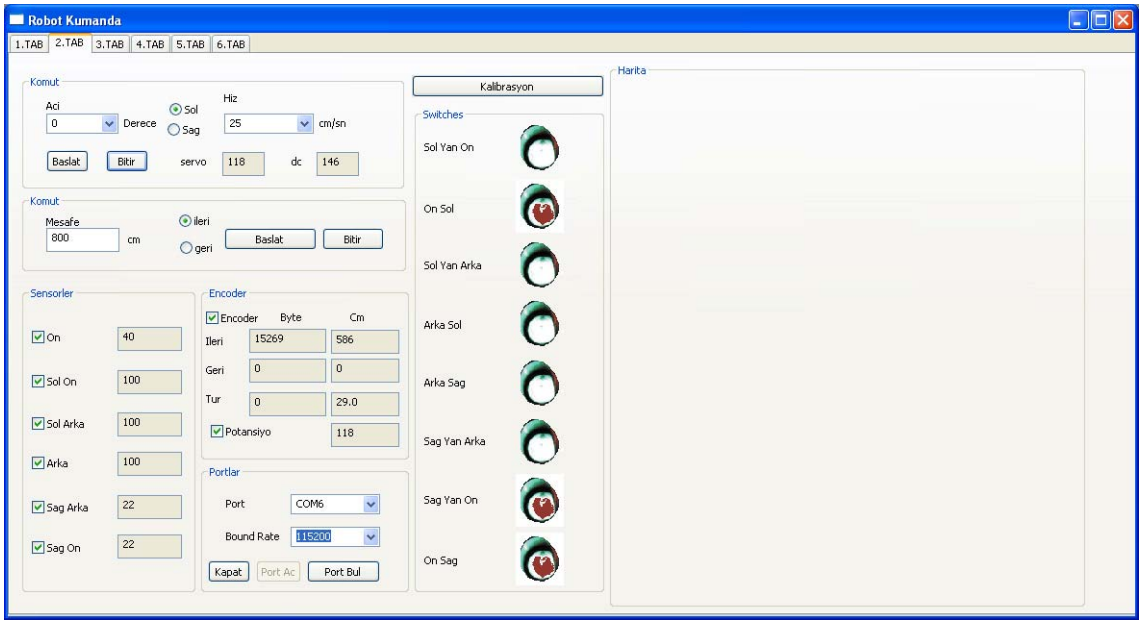

Figure 8.7. The Second Tab Of Control Interface Side

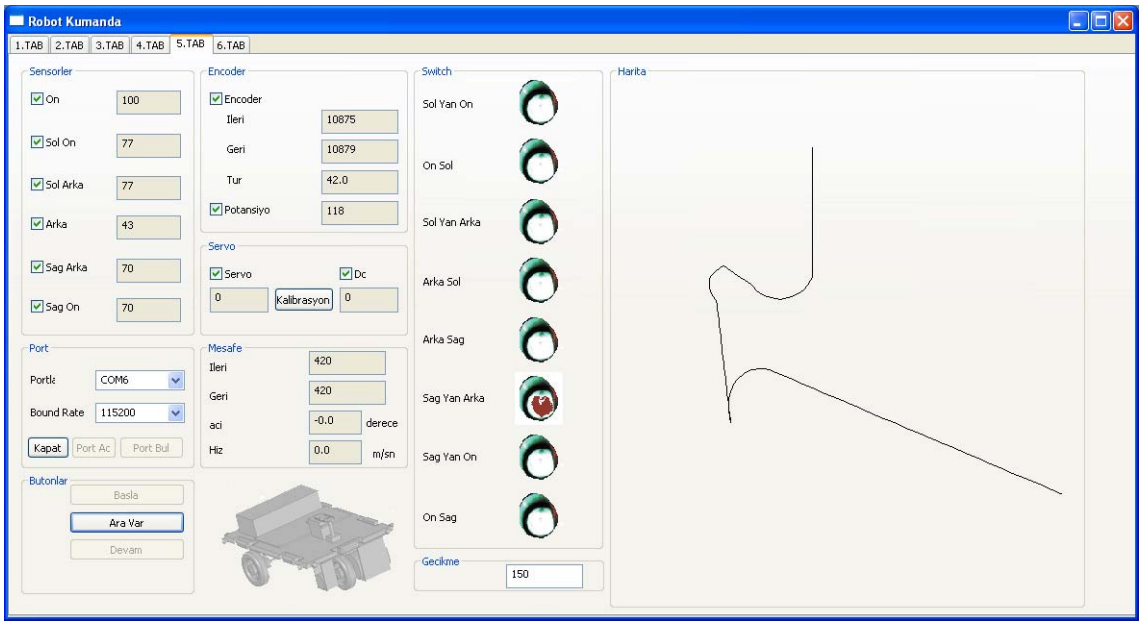

Figure 8.8. The Fifth Tab Of Control Interface Side

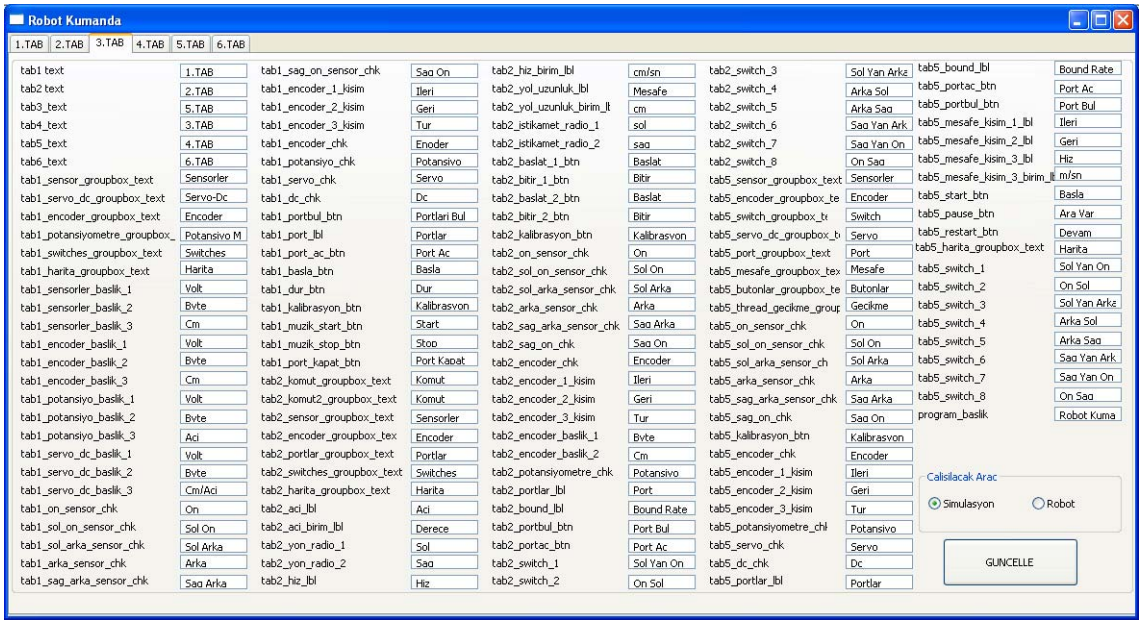

Figure 8.9. The Third Tab Of Control Interface Side

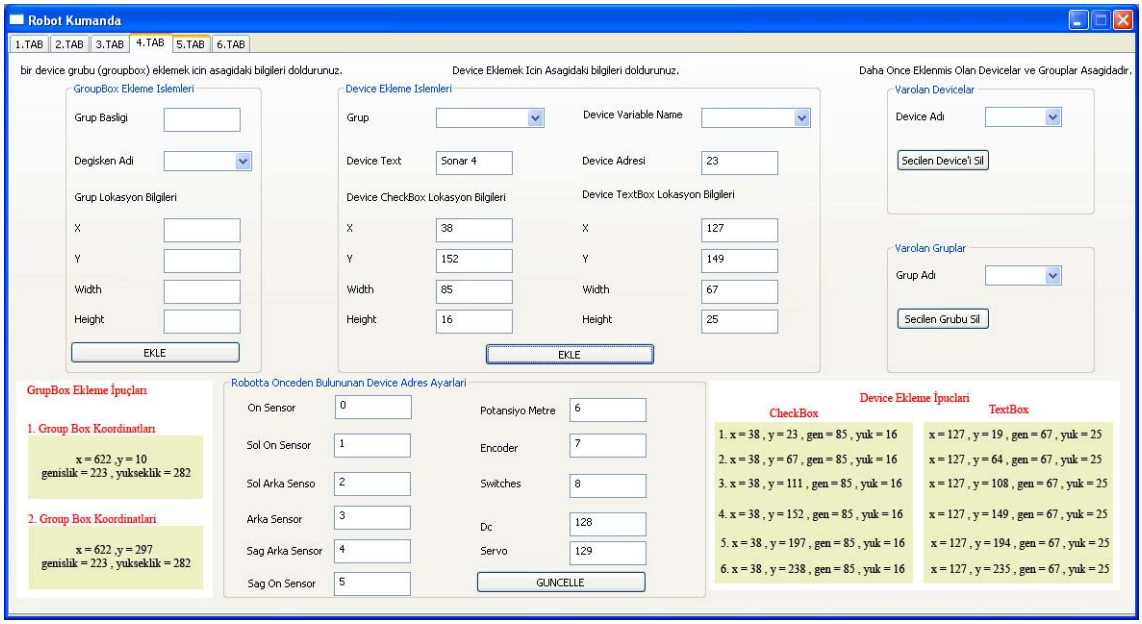

Figure 8.10. The Forth Tab Of Control Interface Side

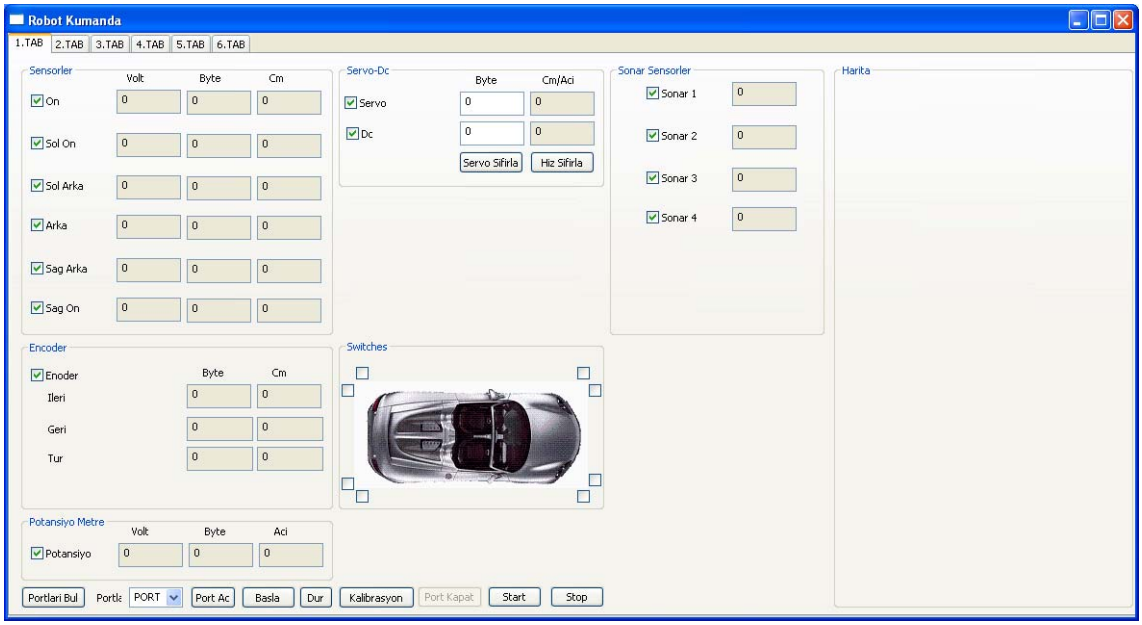

Figure 8.11. First tab after device insertion

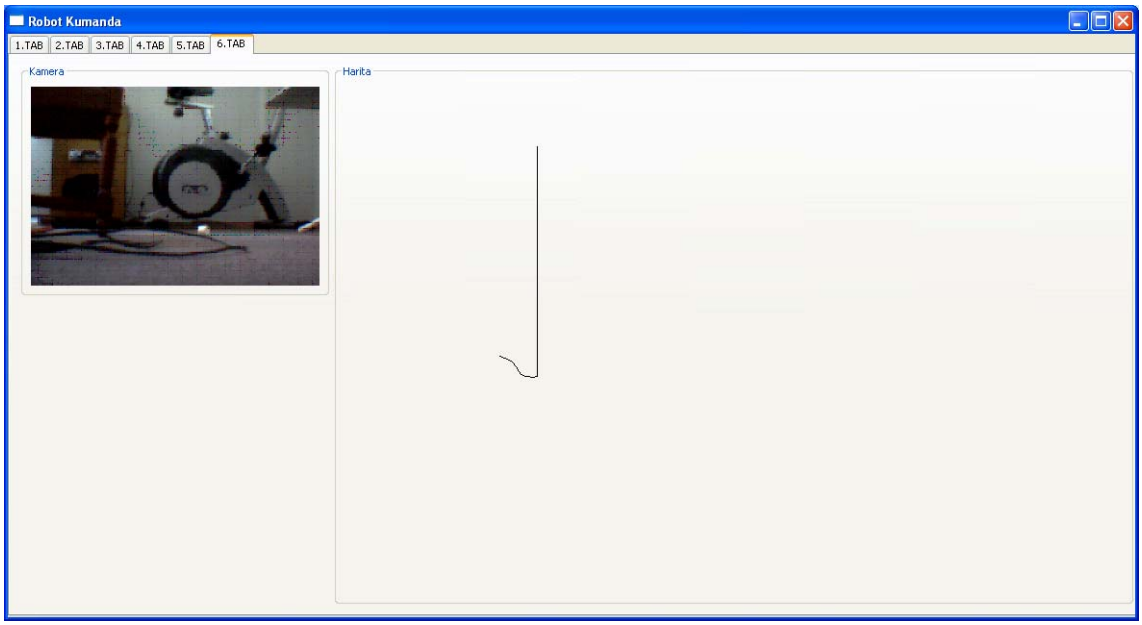

Figure 8.12. The Sixth Tab Of Control Interface Side

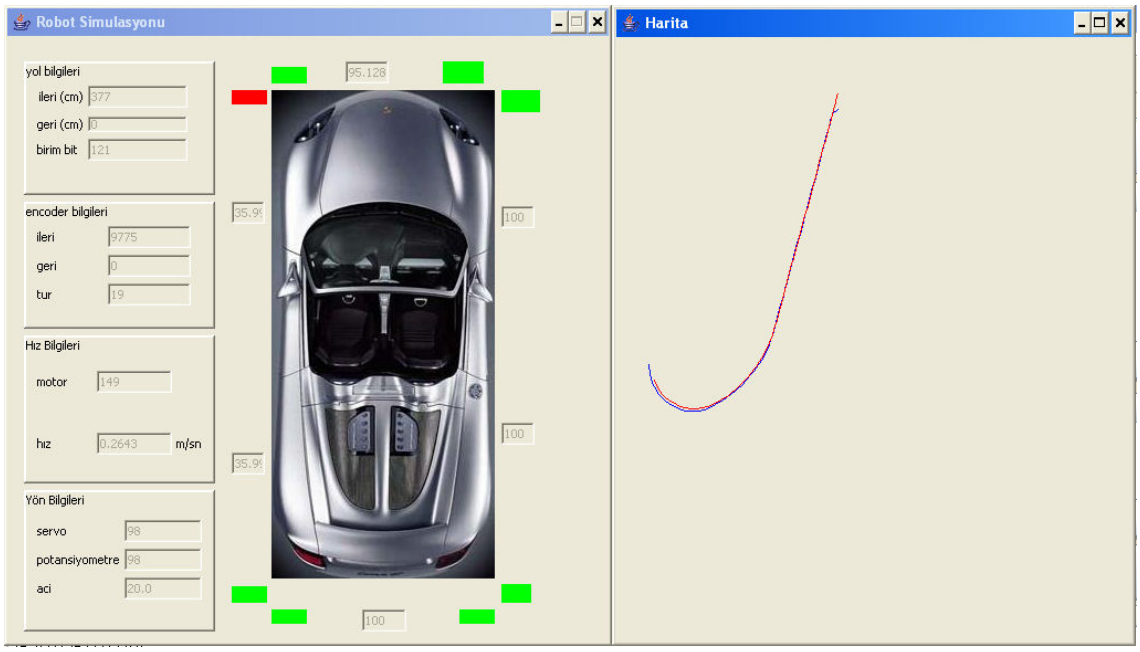

Figure 8.13. Interface of Simulation Side

9. CONCLUSION

Firstly, I am glad to send my best wishes to my instructors by the help of whom I have learned a lot.

Developing the Senior Project is a mile-stone to cope with the difficulty of getting started to direct an overall study of B.Sc. towards a solution for a real problem. I am glad for finishing successfully off the project and having the chance to set it up into the real life.

The Slam robot of our department has been simulated, a command interface for the robot and the simulation has been designed for this project.

Thanks to the project, a deep knowledge of the serial port communication has been obtained, many facilities of the Java program has been used, a firm experience has been obtained on the SWT subject of the java technology, a beginner level experience for game programming has been obtained, a sense of experience has been obtained on the real time systems and a project to be remembered proudly has been completed.

I personally want to thank all who contributed.

10. REFERENCES

[1] MANNING - SWT JFace in Action - GUI Design with Eclipse 3.0

[2] Wrox.Professional.Java.Native.Interfaces.with.SWT.JFace.Feb.2005.

[3] O'Reilly - SWT - A Developer's Notebook

[4] DEVELOPMENT OF A MAP BUILDING ROBOT' S MICROCONTROLLER SYSTEM(Senior Project Kemal Hilmi Yıldız /2006-1)

[5] http://java.sun.com/j2se/1.5.0/

[6] Jim Zyren, IEEE 802.11g Explained. Director of Strategic Marketing Intersil Corporation, Wireless Networking. December 6, 2001.

CIRRICULUM VITAE

Name : İbrahim
Surname : Ök
Birth date : 06.09.1983
Birth place : İstanbul
High School : İzmir Konak Hürriyet Lisesi

Work Experience :

July 2005 - September 2005 : CMS A.Ş. (IT Department)
March 2007 – May 2007 : Ferda Bilgisayar LTD. ŞTİ (IT Department)