ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

AUTONOMOUS MEDICATION DELIVERY SYSTEM IMPLEMENTED WITH SUBSUMPTION ARCHITECTURE AND FAIL-SAFE CONTROL BASED ON LEGO NXT 1.0 PROTOTYPE

του

ΕΜΜΑΝΟΥΗΛ ΣΑΜΑΝΗ ΤΟΥ ΚΩΝΣΤΑΝΤΙΝΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΜΑΝΟΣ ΡΟΥΜΕΛΙΩΤΗΣ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του μεταπτυχιακού διπλώματος ειδίκευσης στα Πληροφοριακά Συστήματα

Φεβρουάριος/Σεπτέμβριος 2014

(Οκτώβριος 2014)

Acknowledgments

Finally I would like to thank my Professor Manos Roumeliotis for his substantial help and support which guided me to accomplish this diploma thesis

Preface

ABSTRACT

The drug delivery system presented in this thesis envisions autonomous systems which will deliver inside hospitals medication to the patients without any fault or mistake. The drug delivery prototype robot can autonomously deliver from the pharmacy dispenser a medication to a current bed. The development of such an autonomous system has many benefits. It can reduce the scheduled delivery of medication to the patients from the nurses in order to increase the productivity of their timetable. In this case the nurses can give more care to some patients. But the development of such system must be strict, with respect to safety issues because we use this in association with human beings and there is no tolerance to hazards, because the result will be catastrophic. So, a system like that must be fail-safe. For this reason, a fault tree analysis is used in order to eliminate any possible fault or lead to a fail-safe state, may happen in such autonomous system, ensuring a safe operation. Fault tree analysis is a widespread approach for analyzing reliability and safety. Furthermore in this thesis, subsumption architecture is used in order to implement the JAVA code with behavior programming that runs into the drug delivery prototype robot. There are many advantages in this type of programming. The prototype design we use for the Lego NXT.1 system is based on the original Forklift Design. To sum up, in this thesis we present an autonomous robotic system, which can deliver medicines from one point to the patient bed and give it to the patient safely in addition with the actions taken to ensure safe operation without faults.

Table of Contents

Table of Figures

Table of Tables

Overview

Chapter one presents the general motivation of this thesis and analyzing the problem of medicine delivery system and what are the thoughts that made in order to contribute in the solution. Chapter two introduces various theoretical and practical methods in robotics, programming and risk management. Most of these methods will assist in the implementation of Autonomous medication delivery system. The NXT is introduced extensively; every technical part of the robot is presented. Likewise the software of the NXT is presented also with a briefly reference to other programming languages that can be used in the robot. Lastly this chapter gives an insight on Fault tree analysis. In chapter three the actual implementation of the delivery system takes place combining the methods from the previous chapter in the creation of the code. Also in this chapter we see the prototype and how the delivery is done in this thesis. Chapter four describes how the risk management ensures fail safe operation and how a patient is protected from receiving a false medicine. Chapter five talks about the form of the autonomous system with future work. Chapter six gives finally a conclusion and the future prospects of the autonomous medication delivery system in health sector. Chapter seven talk about other similar or relative projects comparing them with this thesis. Lastly, chapter eight gives the bibliography that has been used.

Chapter 1

Introduction

In this chapter, we present what inspired and motivated us to embark on this project and we analyze the challenges and problems we dealt with, while implementing the medicine delivery robot.

1.1 Motivation

The NXT 1.0 LEGO motivated us to; at least theoretically, construct a robotic system that is unique and innovative but also simple enough to be reproducible and extensible by a future analyst.

Aware of the lack of funding and staffing in most European public health services, we came up with the idea of a robot which will have the ability to deliver medication from starting point at nurses' station to the patients of a hospital ward with an absolutely fail proof procedure.

The nature of the diagrams and the structure of the code we created to deliver our plan allow our tools and methods and risk management procedures to be effective at any stage without the need to deconstruct the system in order to expand it. It is an autonomous robotic system simple enough for someone to understand its purpose and functionality but with the potential to easily become more sophisticated adding further functions.

Our goal in this thesis is to solve the lack of nursing staff and lack of uninterrupted completion of tasks that is responsible for most incidents in the hospital environment and thus improve quality and safety of care.

1.2 Analyzing the problem of medicine delivery system

With most European countries going through recession for the past eight years cost cutting and reduced funding has hit public health with disastrous results even in the once reputable National Health Service of United Kingdom. (BBC,2014)

In 2007 Julie Bailey lost his mother in Stafford Hospital in Stafford. Because of that she started a campaign called Cure the NHS, that was the trigger. The Stafford Hospital scandal became widely known because of an investigation by the Healthcare Commission in the operation of Stafford Hospital in Stafford, UK. The commission started the investigation after many warnings about high mortality numbers in the hospital. The responsible organization for running the hospital, which is the Mid Staffordshire NHS Foundation Trust, couldn't provide to the commission a sufficient explanation and a full-scale investigation was conducted between March and October 2008. The outcome of this investigation revealed the poor conditions inside the hospital. Furthermore a severe criticism of the management was made by highlighting hospital's deficiencies. Many press reports said that because of the poor care the hospital provided, between 400 and 1200 more patients died in the years from 2005 and 2008 that it would be expected, for that type of hospital. Although, the inquiry continues, an average compensation of £11,000 was paid to some of the involved families. (Wikipedia,2014)

Mid Staffordshire would not have happened if there were enough and well rested nurses. Staff that is overworked and underpaid and has to perform several tasks simultaneously will eventually fail. The wrong medication will get to the patient or the patient will die whilst the nurse is handing out medication at the other side of the ward or will hand out incorrect drugs as she/he had to stop and deal with something else. (BBC,2014)

Our system saves them time and effort and allows them to concentrate on delivering the high quality of care they want to the patients instead of handing out pills.

It is an automated system that due to the safety nets and fault tree analysis we used to create it, will deliver medication to patients who are conscious and reasonably mobile to move their hands, always on time and correctly.

Chapter 2

Relative Work

Relative projects that have been implemented are few in robotics area; fewer are those with the greatest resemblance. As such we compared this thesis project with other quite similar technological robotic projects.

First of all, we will talk about one very interesting robotic prototype from Victor Emeli,Alan R. Wagner,Charles C. Kemp. This robot can autonomously can deliver pills and water to a person inside his home. This robotic system includes a tray, a stationary dispensing robot and a smartphone for the user. This system is for elder people with chronic illnesses that require tactical their medication. With this robot they overcome problems like under dose or over dose, distinguishing between pills, difficulty to open their pills container, or dehydration. This autonomous delivery robot scopes to deliver pills to a patient which is very similar to this thesis, the difference is that is that this system is intended to be used in a home environment unlike our project is intended to be used as an aid for certain tasks for nurses in a hospital. The project we are examining is divided in two parts. Firstly the robot must dock with the dispensing robot to acquire the pills and a cup of water in its tray. For the navigation this robot uses a labeled map to move through the house. In the contrary with our thesis, we avoided to use this way of navigation because we need absolute accuracy in the navigation process, because of the importance of the medication delivery to patients, as it was explained thoughtfully in a previous chapter. In addition, the second part of the project is the developments of a smartphone interface that and a friendly user interface. While the elder people carry their smartphones, the robot alerts them when it is the time to deliver the pills and the water and ask the user where he is located. After that, the user gives the appropriate information via speech. When the robot enters the room is trying to identify the user via a gesture using 3D location process. Similarly to our project, we made a referring in future work chapter explaining how our robot embracing the technology that supports smartphone interconnection via android and what this feature can offer in a future implementation. The whole design was based in the Willow Garage Turtlebot robot. This robot has low cost and open source hardware and software. It comes with the Robot Operating System that is open source that has libraries and tools in order to create custom robotic applications. The similarity in both cases is great, as in this thesis we made the same thing, we used the Lego Mindstorms NXT, which comes also with open source Lejos software with libraries and tools for every construction. Furthermore in this paper, actual implementation has been done according with experiments in actual home environment in contrast with our thesis which only has remained on a theoretical level. At last, as we saw in the outcomes of the paper that we examine, there is percentage of failure deliveries that is

unacceptable in our project due to the importance of delivering treatment to patients, that their life depend on. That is why we implemented additionally the Fail Tree Analysis. (Victor,Wagner,Kemp,2012)

But there are also some real robotic models with actual resemblance with our project. These models are already for sale by tech companies. First of all, we examine the Automated Delivery and Tracking Solution, from AETHON Company called TUG. This automated model can navigate through the whole hospital via elevators. This feature has not examined in our scenario case. Basically TUG delivers 24 hours a day scheduled or not any kind of deliveries that are needed for the hospital staff, but no actual medication to patients. TUG is perfect tool for the staff that partially shares our motivation, to facilitate the heavy working schedule of the nurses at least on routine jobs, allowing them to focus on other patient's needs. We have not much technical information for TUG except that is controlled via a touch monitor or via hospital PC's if the hospital is networked. TUG also supports control via smartphones. To conclude, the basic idea of this product that our thesis also shares is that a robot can make the deliveries inside a hospital when the medical stuff can really focus in the patients with no disturbance. (Aethon Inc,2014)

To end with, one more robot will be discussed called Hospi constructed by Panasonic Company. Hospi is an autonomous cargo deliverer/container. It can deliver small things only in his tray like drugs, in contrast with TUG that delivers everything. Both robots are very different with our project that is focused only in delivering medication to patients. But we recognise that a hospital has many automated delivery needs that can be done using a robotic model freeing nurses of non-critical activities, enabling the nurses to quality attend the patients. Like TUG, Hospi also can move easily through the hospital and has many sensors in order to avoid people or obstacles like our project in this thesis. Unfortunately, we don't have access to technical information for Hospi, but is very significant to say that five Hospi robots have been purchased at Matsushita Memorial Hospital where they reduced delivery time over 30% of the usual. The actual cost of each robot was $100,000 with an addition cost for altering hospital infrastructure, to be robot friendly. As a conclusion we see that the robotic delivery process is a field under constant development, but hopefully we have a very advanced technology ready to enrich and improve the robotic inventions, that can be used as a tool to serve and preserve human life also inside a hospital where it needed the most.(IEEE Spectrum,2014)

Chapter 3

Methodology – Design

Mindstorms can be used both as toy but also as an engineering tool. There are many examples of engineers and scientists that used Mindstorms as prototype in inventions or projects

3.1 Meet NXT

The Mindstorms project began in 1987 in MIT Media laboratory called Programmable Brick with contribute of Lego Company. The basic designer was Fred G.Martin who made many versions of the Programmable Brick between 1987 and 1998. Below is the final version with four outputs for the motors and six inputs for sensors. (Bangall,2013)

Fig 1: MIT Programmable Brick (Bangall,2013)

Soon the next version developed as Robotics Invention System. The version inside called RCX brick and it was very primitive. It used an 8-bit processor at 16 MHz and 32 kilobytes of memory. (Bangall,2013)

Fig 2: RCX Brick (Bangall,2013)

The Lego Mindstorms has been used since then in many schools as an education tool, but also in research and graduate projects because of its many advantages and benefits. Later a very important improvement was made by the hacker community when they unlocked the RCX brick and more programming languages were available since then as C and Java. On August 2, 2006 Lego developed the NXT 1.0 kit and upgraded in NXT 2.0 kit in July 2009. These kits used standards as Bluetooth and I^2C. The hardware between the NXT 1.0 and NXT 2.0 remained the same but the parts, sensors and software changed. The dimensions of NXT brick are 7.2 cm by 11.2 cm. It uses six AA batteries and weighs 160 grams. Also, the NXT contains an ATMEL 32-bit ARM processor at 48 MHz with 256 KB of flash memory. The flash memory is used to store the programs and data. Moreover the NXT brick contains also an ATMEL 8-bit AVR processor at 8 MHz. This processor is used to run the servo motors and rotation sensors inside the motors. Also the processor uses a 4 KB flash memory and 512 bytes of RAM. This extra processor is used in order to monitor the optical tachometer constantly so as to remain accurate. This was made in order the main processor to be unaffected while it deals with other things, as sensor data processing or running programs avoiding any fault in keeping track of the rotations. (Bangall,2013)

3.2 NXT parts

In this chapter every part of the NXT robot will be discussed. All these parts help the procedure to be completed.

Batteries

The NXT 1.0 uses six AA batteries which provide to the robot 9 volts. Also there is another one solution for energy; NXT can use also rechargeable lithium ion battery. This option provides at least 7.4 volts and 8.2 volts when it is just recharged. The downside of the rechargeable battery for the NXT is that the motors will not operate the same as the batteries that provide 9 volts. When the robot uses rechargeable batteries lacks of power or speed while it operates. (Bangall,2013)

Speakers

Inside the NXT brick there is an amplifier chip which can play a sampled sound. The quality of the sound is low, but the NXT has a variety of sounds inside a library. Also someone can record a sound and incorporate it inside the library of sounds. (Bangall,2013)

Ports

There are four ports for the sensors in the NXT brick and three ports for the motors. The ports use the standard protocol called Inter-Integrated Circuit. I^2C is used for transmission of the data for and to the sensors. The I^2C protocol can control more sensors than four as long as the sensors use the Auto Detecting Parallel Architecture through an expander. (Bangall,2013)

Cables

The NXT kit contains seven wires with three different lengths, 20 cm, 35 cm 50 cm. The cables are very similar to RJ12 but not identical. (Bangall,2013)

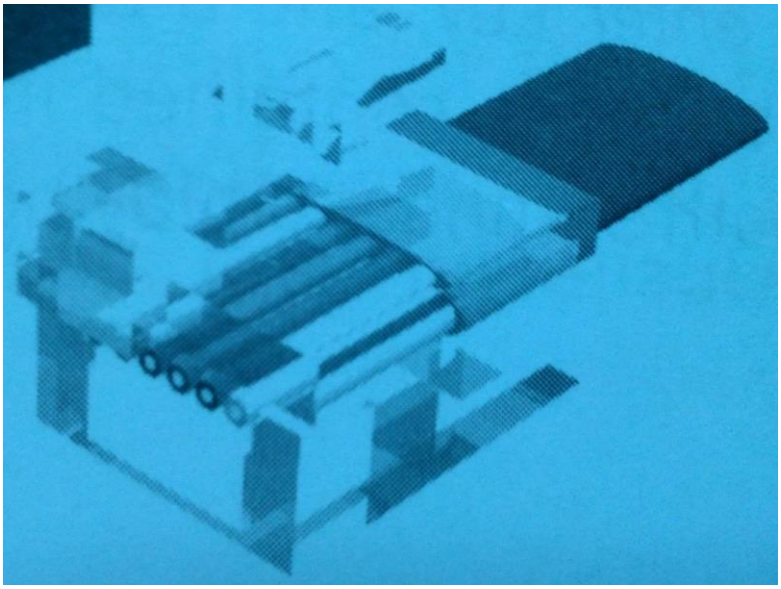


Fig 3: RJ12 connector (Bangall,2013)

LCD Display

The LCD display of NXT is in black and white pixels it can support bit mapped images and has resolution of 100x64 pixels over an area of 26.0x40.6 mm. The LCD screen of the robot requires 17 ms to draw a new screen. The display refreshes 60 times every second (60 Hz) and supports animations also. (Bangall,2013)

USB

The NXT brick supports a standard USB cable, which can be used to upload code, data or firmware to the robot. The USB can transmit data at 12 Mbits per second. (Bangall,2013)

Bluetooth

The NXT contains a Bluetooth chip but not an adapter. Bluetooth is slower than other protocols. It has low consumption as 0.01W and is very cheap technology. The Bluetooth can cover a very small area of few meters wide. It can transmit data up to 1Mbps in addition with the transition of sound. It operates in the frequency band of 2.4 GHz with the communication between two devices as point to or the communication between multiple devices as point or point to multipoint. It supports eight devices per network and this technology allows more than one network to coexist in an area. Such kind of network called piconet. The minimum distance between the transmitter and the receiver is 10 cm and the maximum distance is 10 meters. These two devices communicate after a validation is done between them with the use of a PIN code. This feature is used in the implementation of this thesis as a safety net in medication delivery. (Διακονικολάου,Αγιακάτσικα,& Μπούρας,2007,p.270)

Bluetooth chip inside the robot gives the ability to interact with PC applications and devices of every kind if they support Bluetooth protocol. In addition Bluetooth adapters use frequency hopping to avoid conflicts. This is very useful in this project because in a room where there are too many beds with Bluetooth adapters there will not be a problem while the robot delivers the medicine. This will be explained later thoroughly. (Bangall,2013)

Lego Building Parts

The NXT 1.0 kit has 577 plastic parts which are used in order to build every design. The quality of NXT kit parts is very high. The accuracy of LEGO's injection molding process is flawless with no defects. Every part is almost indestructible and cannot even scratch, dent, bend or break. The kit contains Beams, Liftarms, Pins, Axles, Tires and Wheels, Gears, Cables and other parts or accessories, all of them are very useful for the assembly of any kind of robots. (Bangall,2013)

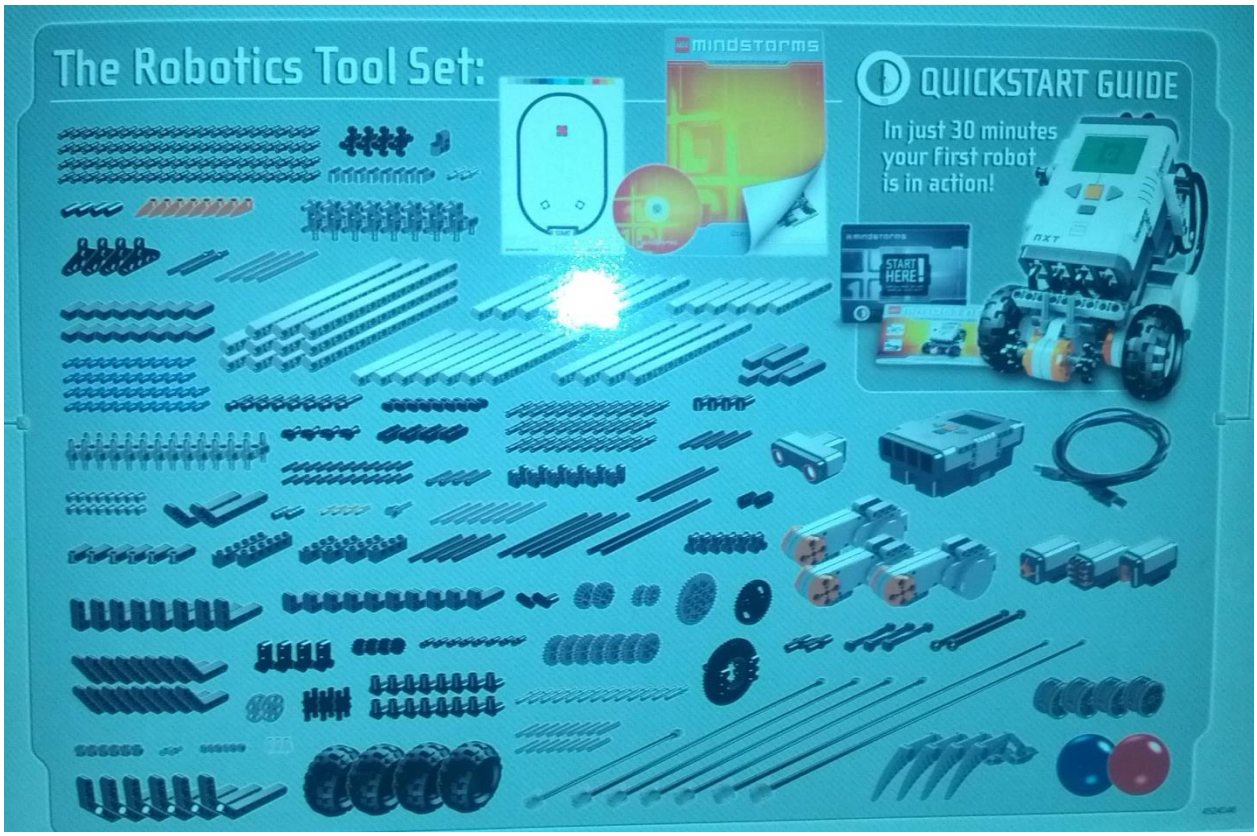


Fig 4: The parts of NXT 1.0 LEGO (Bangall,2013)

Motors

The NXT motor has inside encoder in order to keep track of axle rotation. Also this encoder helps the robot to come back in the exact starting position even if the motor turns for thousands of rotations. This feature helps the robot with navigation or in the preciseness of movements. The NXT kit includes three motors, each motor weight 81 grams. In addition the every motor contains gears that can reduce the speed while increase the power ability. The unique is that the motor speed is controlled and altered through code. (Bangall,2013)

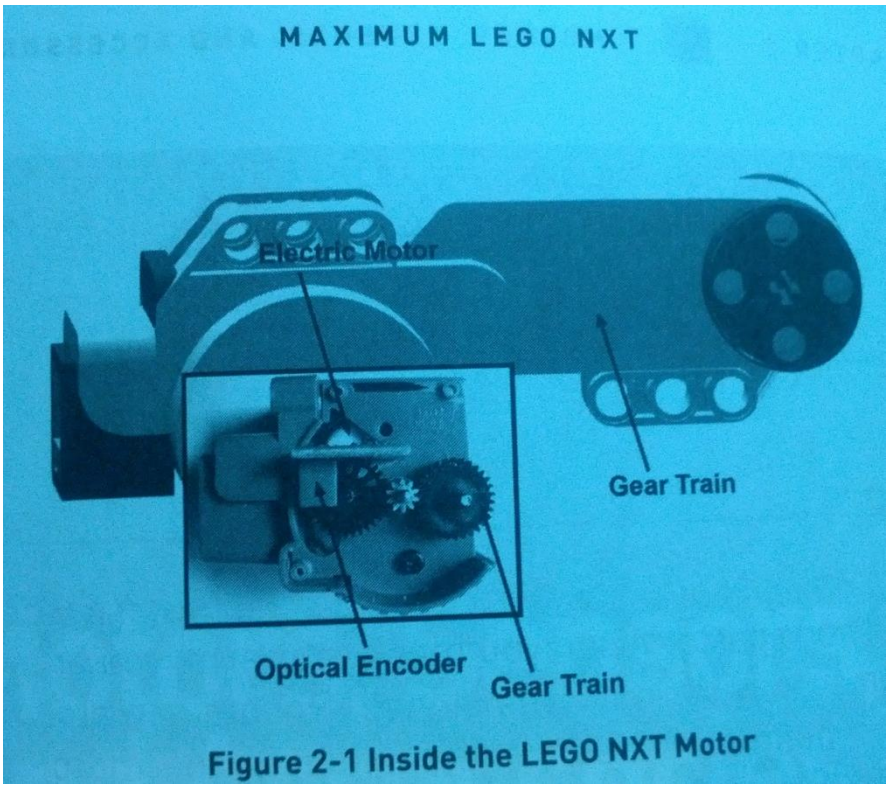


Fig 5: The NXT Motor (Bangall,2013)

Sensors

Most of NXT sensors have the same size and shape and the same means for attaching the robot. The NXT kits have one touch sensor, a light sensor, a sound sensor, an ultrasonic sensor. (Bangall,2013)

Touch Sensor

The touch sensor is very basic inside the kit. It includes a switch that can be activated if the orange button is pressed on the front. (Bangall,2013)

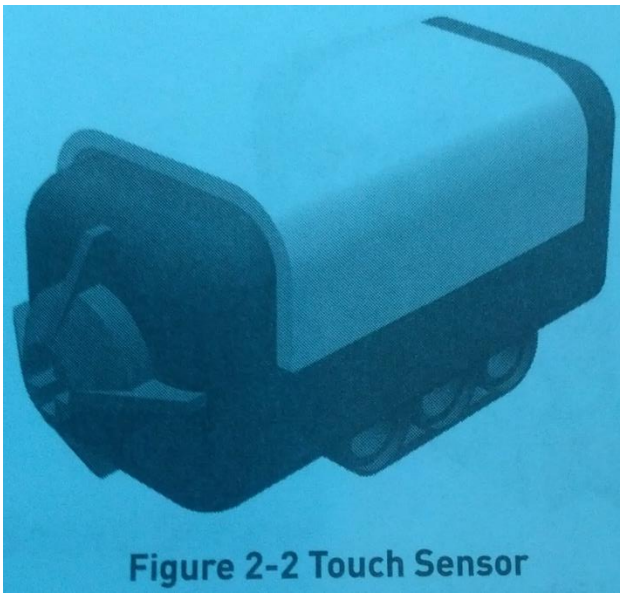


Figure 6: The touch sensor (Bangall,2013)

Light Sensor

The light sensor measures the intensity of the light entering a tiny lens on the front of the sensor. The sensor also has a red light-emitting LED diode which illuminates everything in front of the sensor. The sensor also supports IR light detection feature. In this thesis the light sensor is used in order to follow a black line for the maximum precision in navigation because of the importance of the medication delivery, no navigation fault is allowed. The light sensor has two modes:

- ✓ Active mode when the LED sensor is illuminated. More accurate the sensor measures the reflection of the light that emitted and so it can follow a black line over the thresholds that have been set in the code.
- ✓ Passive mode when the LED is off. The sensor is used to detect ambient light and IR light.

(Bangall,2013)

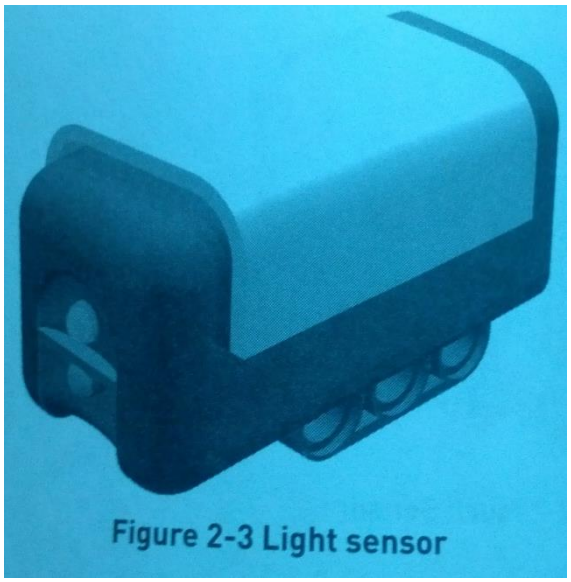


Fig 7: The light sensor (Bangall,2013)

Sound Sensor

The NXT sound Sensor looks like a microphone, it can measure the loudness of sound in decibels. Also it can be used as a sound recorder and the files are stored in the brick. In addition, the sensor can play sound files that are stored in the brick. Lastly, it can react in any ambient sound. (Bangall,2013)

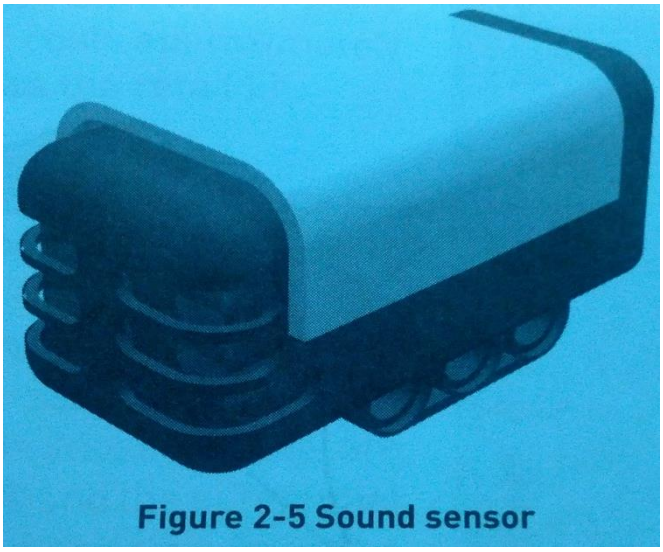


Fig 8: The sound sensor (Bangall,2013)

Ultrasonic sensor

The ultrasonic sensor looks like a pair of eyes. I can send a sound signal in low frequency, inaudible to humans. Then it measures the time the signal needs to return back. The sensor has stored the speed of sound and so it can make the calculation. The ultrasonic sensor uses the I^2C bus protocol. It can measures the distances in centimeters or inches. The measurement ability is up to 255 centimeters but in these kind of distances the ping signal is weak and the results are not trustable. The high accuracy of the robot relies between 6 cm to 10 cm. The measurements beyond that threshold are not reliable. The accuracy is plus or minus three centimeters but better when the objects are close. The sensor produces signals like a cone shape, as you can see in the picture below. The cone has 30 degrees angle. The cone shape of sound signal emission is ideal because it can scan large area for possible collisions. (Bangall,2013)

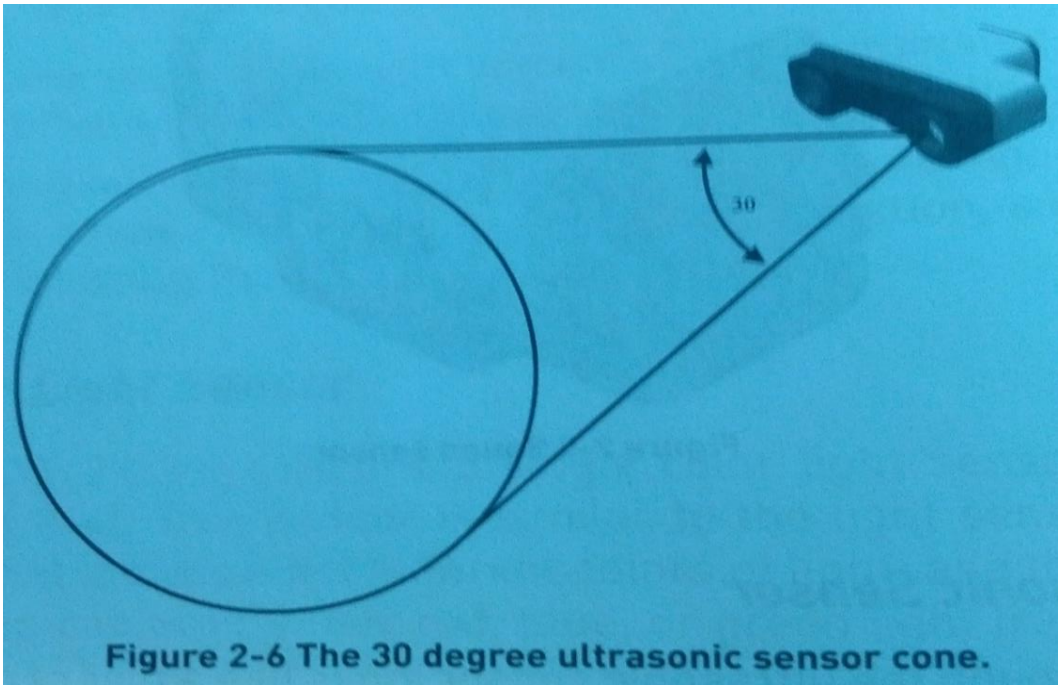


Fig 9: The ultrasonic sensor and the emission cone (Bangall,2013)

3.3 Lejos NXJ

The software package that is used in this thesis is known as LeJOS NXJ. This software has many capabilities and contains also a premade library of routines for many tasks. LeJOS NXJ is a Java based environment with threads, arrays, floating point numbers, garbage collection, control of motors, control of sensors and many others used to control the robot. For this thesis and for the development of the code, the lejos NXJ set up on PC and the firmware uploaded in NXT brick via the software. The lejos platform can be installed in many platforms like Windows, Linux and Macintosh. (Bangall,2011)

Installing the NXJ

For this thesis the installation was made in Windows platform, here are the steps for the installation.

1. Installation of a USB driver from www.mindstorms.lego.com/en-us/support/files/Driver.aspx, and connection with the robot and PC with USB cable
2. Installation of Java Developer Kit, the latest JDK from www.oracle.com/technetwork/java
3. Installation of the Setup.exe from www.lejos.org , when the installation is completed then firmware will be installed in the robot.

(Bangall,2011)

In the next chapter we will deal with how the compilation and the upload of the code will be done via eclipse.

3.4 JVM- IDE-API - (eclipse)

The LeJOS JVM is based on C code. This is made in order to be ported to other hardware also. Java VM runs the Java code, let's explain briefly the VM. (Bangall,2013)

Memory

The leJOS VM requires only 55 KB of memory and leaving at least 200KB for the programs and data to be stored. (Bangall,2013)

Speed

In robotics there is no need for extremely computing speed. For example with the Java VM needs to check the sensors and update every action every 8 ms and the code for this uses only 1 ms, so in this thesis where the robot uses the behavior code to check every time what sensor is triggered in order a behavior to take over the computing time of the Java VM for this is more than sufficient. (Bangall,2013)

Floating Point Numbers

The VM supports floating point numbers both to the left and right of the decimal. This allows trigonometry functions such as tan, cos and sin. The trigonometry functions are very important for the robot to navigate through an environment. (Bangall,2013)

Threads

The Java VM supports also Multi-threading, this feature allows many parts of program to be executed simultaneously sharing the CPU. The leJOS thread scheme allows synchronization and interrupting. Because the NXT contains only one processor every thread must take turn while using the processor controlled by a thread scheduler. The thread scheduler switches from one thread to another using a native C function called switch_tread() and every thread can execute up to 128 instructions before the next thread takes over. Lejos allows the creation of maximum 255 threads, more than enough for robotic projects. (Bangall,2013)

Garbage Collection

JVM supports garbage collection in order to discard from the memory the objects that are not required anymore. When objects in object oriented languages are no longer in use then must be removed from memory this feature is called garbage collection. (Bangall,2013)

An IDE or Integrated Development Environment is used in order to compile an upload the code in the robot. The IDE that has been chosen is eclipse by IBM which is free and open source.

Installation and use of Eclipse

1. First the download of eclipse from www.eclipse.org and then the installation follows.
2. Then follows the installation of leJOS Plugin. In eclipse we select Help then Install New Software, click add and enter the name leJOS NXJ and in the location field enter the following link http://lejos.sourceforge.net/tools/eclipse/plugin/nxj and then ok.
3. Place a checkmark in the box to add the new item, we click next and the plugin will be installed.
   (Bangall,2011)

When the installation is completed then we can proceed in the creation of our project with its own directory to store the class and data files. The creation of the project is done via File > New > Project.

In addition, after the project is created, we can add classes via File > New > Class. Then if we want to upload the program in the robot we can right click in the name of project and the click on run as LeJOS NXT program and then it uploaded on the robot via USB cable. The eclipse IDE can offer many other tools, plugins and features from debug, errors warnings to anything a programmer needs. Also eclipse supports functions to generate try catch blocks and getter/setter methods. (Bangall,2013)

The LeJOS NXJ API is consisted of many classes each one has multitude of methods. Some of them are standard Java classes while the other classes are designed by the LeJOS developer team. This is the link to LeJOS NXJ API: http://www.lejos.org/nxt/nxj/api/index.html as showed below. The classes and methods in LeJOS API that have been used in this thesis, have direct access over the functions of the NXT brick. (Bangall,2013)

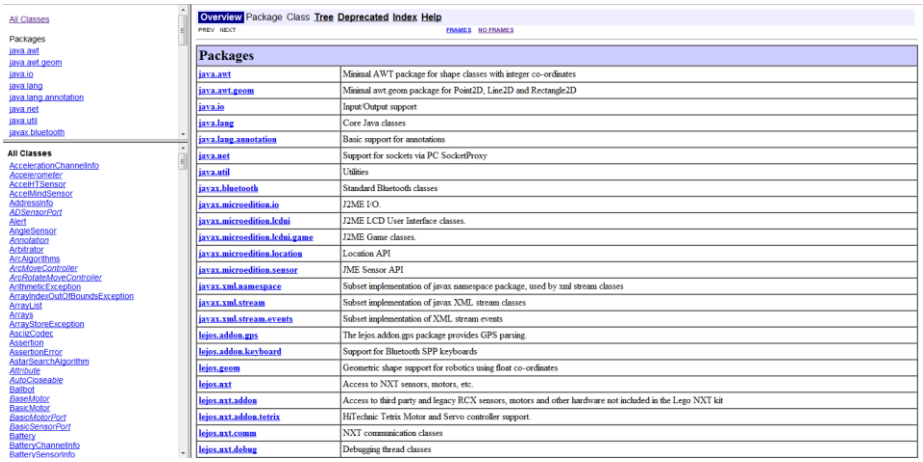


Fig 10: LeJOS NXJ API
(Java for LEGO Mindstorms,2014)

3.5 About navigation-localization

Localization is the ability of everyone to determine the location or the place they stand using names. Robots cannot determine the location as humans but they can understand numbers, as a result with the use of Cartesian coordinate system we can make robots to find a certain location. Two main strategies are used to ensure the ability of localization, the blind proprioception and landmark navigation. Localization methods of a robot will be analyzed more in order to explain why have been chosen the line following method for the project of medicine delivery robot. (Bangall,2013)

Firstly let's discuss about navigation by using landmarks. This kind of navigation is made with the use of landmarks for example with the use of a grid patterns with poles, we can set every pole a letter and then the navigation can be made easily. For example the way to point A in the grid is three poles south and two poles east. This method is more reliable than proprioception, with the use of blind proprioception there a great possibility of way loss. (Bangall,2013)

A way for this method to implement is localization via beacons with the use of a mathematical formula called the Generalized Geometric Triangulation Algorithm. This method works in a way of placing three beacons in three certain spots in a room, then the robot use a laser sensor that emits a beam to reflect the light off every beacon inside the room.

As a result the robot calculates the angle of the beacons and with the use of triangulation algorithm produces the position as you can see below. (Bangall,2013)

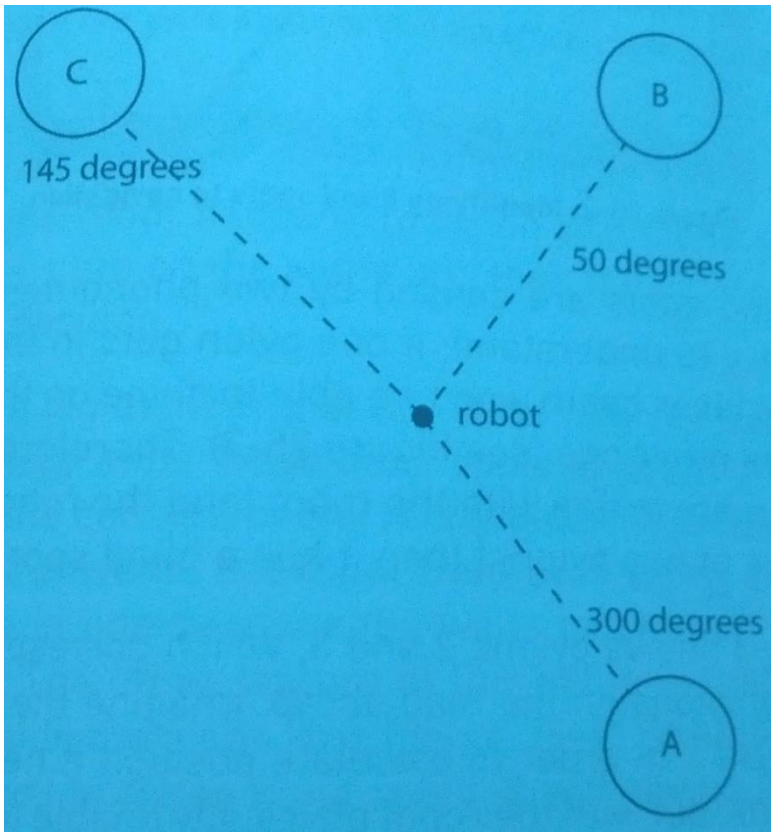


Fig 11: Triangulation algorithm scheme (Bangall,2013)

Because this algorithm has many blind spots, as you can see below the results are not very reliable we cannot trust triangulation method because we need 100% sure navigation method for this project. Also one downside more is the cost of an extra laser sensor plus the three beacon materials inside every room the robot passes through which is a problem for a corridor in a hospital where is more like a large rectangular room.

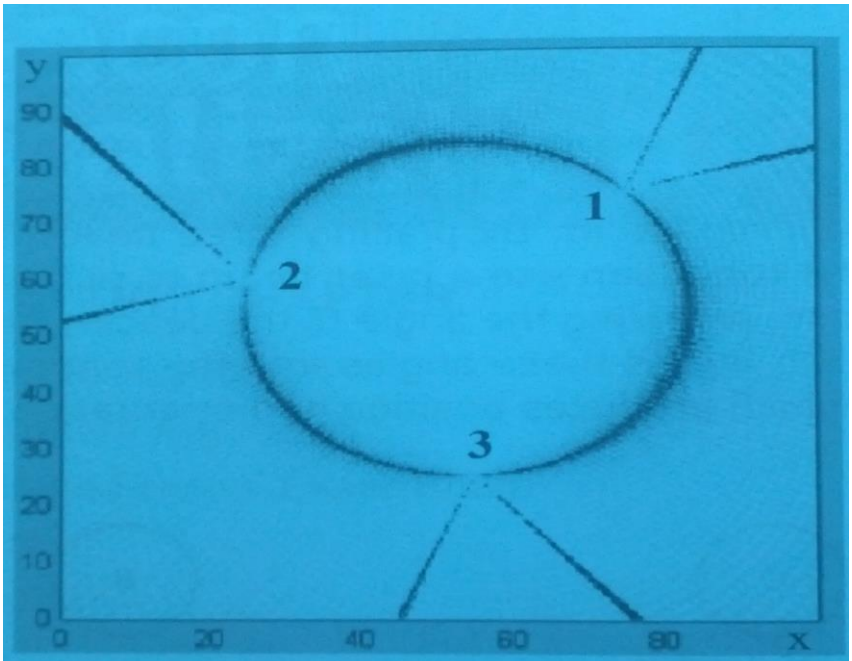


Fig 12: Blind spots scheme (Bangall,2013)

In leJOS API there is feature where we can make a map of a room or a place as an SVG map using an editor and then we can control the robot via a class called NXTNavigationModel which connects the PC with the robot. Also there is PC program called Map Command. This application is used for the display of the map data inside the PC. Map Command can make the robot rotate, move and travel through different coordinates by clicking on a point in the map to make the robot travel in this direction. (Bangall,2013)

The blind proprioception is a navigational method helps the robot keep track of every movement it makes from the starting point. This navigational method called dead reckoning or orienteering. The only information this method needs to be implemented is direction and distance. Direction is provided by compass and distance is obtained by keeping track of the rotation of the robot wheels using the tachometer that is inside the motors. For consistency to be applied while robot moves, it must always update the coordinates that where stored in the navigation process. For the coordinates to be updated the robot must implement trigonometry calculations. The leJOS software supports these features via the Navigator API. (Bangall,2013)

Dead reckoning navigation method is depending on good round wheels and good tachometer, also because of floor type the robot will always have the risk of losing the way from one point to another because it is not able to correct the path on its own. (Bangall,2013)

To conclude there is a downside to these two methods that makes them not very appealing choice for the navigation of the medicine delivery robot. They could not ensure us accuracy in the navigation process.

We also need the robot to perform other jobs as well while it travels, for example object detection and obstacles avoidance if we use another navigation method it is likely that the robot will lose its way. As a result we choose the line following navigation method because it can offer many advantages. Firstly, the robot never loses its way because while it moves towards the black line it auto calibrate using the sensor to determine darkest and brightest light values of the black line. In case of derailing it has a mechanism to find again the black line by rotation and this ensures the medicine delivery. The steering procedure also depends on the thresholds we set. In addition because of the facility we want the project to take part, the line following is ideal because we don't want the robot to distract the other actions that take place in the hospital. So when the robot follows a visual black line painted or embedded on the floor it can always find the way to the patient from the starting point and back without any concern that it might lose its way of avoid any possible collision or obstacle due to the behavior we added, this behavior it could not applied in other navigation methods without the possibility of losing its way to the patient. The disadvantage of the line following method is that we use a light sensor and this creates extra cost in the project.

3.6 Bluetooth in Lego

Bluetooth protocol is very useful in order to create a communication between two devices without any cable. This protocol can create a Wireless Personal Area Network which is a portable wireless network which can be created everywhere between the NXT Bluetooth chip and with a portable adapter. An adapter to be compatible with NXT it must be Bluetooth 2.0 or higher also for Windows users it must be Widcomm Bluetooth stack version 1.4.2.10 or higher. (Bangall,2011)

The following list gives some the uses of Bluetooth technology for the NXT brick.

- Upload data and programs to NXT via PC
- Control the NXT with PC
- NXT data feedback to PC
- Communication or pairing between two different NXT bricks
- Control of NXT with a mobile phone with Bluetooth protocol
- NXT data feedback to mobile phone
- Connection and control of the NXT with Serial Bluetooth keyboard
- Pairing the NXT with GPS
  (Bangall,2013)

LeJOS NXJ API contains classes and methods to control the connection between the NXT and a Bluetooth adapter anytime is needed. This feature has been used in this thesis in order to create a safety net to ensure a fail-safe delivery of the medicine to the patient.

3.7 Object Detection

There are many classes used to detect objects and obstacles. The package in the Lejos API for object detection called lejos.robotics.objectdetection. There are multiple sensors and methods to detect obstacles. (Bangall,2011)In this thesis we are going to use the ultrasonic sensor for the object avoidance.

The ultrasonic sensor is used here as Object Detector, this feature has the class RangeFeatureDetector from the API. We can set the parameters in the code that controls the ultrasonic sensor as the maximum range the sensors scans and the time required between the scans. (Bangall,2011)

3.8 RFID

Radio frequency identification or else known as RFID, is a method to identify an item or a person or anything you want with the use of radio frequencies. There are numerous uses for RFID technology. (Bangall,2011)A brief overview of how RFID systems work will be placed in this section.

First let's talk about components of an RFID system. Such a system is made of two components.

- the transponder which is located to item we want to be identified, there are many kinds of transponders active or passive.
- the interrogator or reader, this device varies upon the need, the design or the implementation every time needed.

(Finkenzeller,2010)

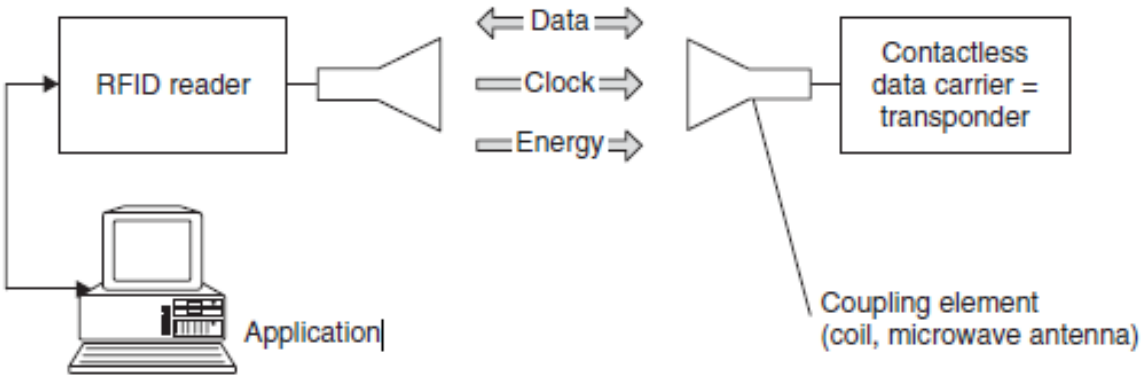


Fig 13: The main components of every RFID system (Finkenzeller,2010)

The reader is consisted of a radio frequency module the transmitter and receiver, a device for control and a coupling element on the transponder in order the communication to implement. In addition the reader might have an extra interface in order to forward the received data to another system for processing or analyzing like a PC, Robotic systems etc. (Finkenzeller,2010)

In this thesis we made something similar we used a reader which is the RFID sensor attached to the robotic brick via cable and the transponder is the RFID tag in the patient and bed used for the validation as a safety net.

The transponder is the data-carrying device in an implementing RFID system consisted also by a coupling element and an electronic microchip. The transponder is usually running under its own

power supply. It's active only when the reader approaching it, for example an RFID sensor, elsewhere it's passive. The transponder activated through the coupling unit when it powered it up by the reader, with the timing pulse and data. (Finkenzeller,2010)

Three different transmission bands exists in RFID systems, the LF (low frequency, 30–300 kHz) the HF (high frequency)/RF radio frequency (3–30MHz) and UHF (ultra-high frequency, 300MHz–3 GHz)/microwave (>3 GHz). (Finkenzeller,2010)

Also the range coupling thresholds are the following, the close-coupling (0–1 cm), the remote-coupling (0–1 m) and long-range coupling (>1m). (Finkenzeller,2010)

Two kinds of transponders exist active and passive. The active transponders contain battery to power up on the contrary passive transponders are depend on external electromagnetic field and usually are powered by the RFID reader/sensor. The advantage of passive RFID transponders is that are quite cheaper than the active and most commonly used. This logic we use for this thesis for the theoretic implementation of the medicine delivery robot. We relied on the RFID kit from Lego which contains an I²C RFID sensor and many passive transponders with unique ids. The transponders are activated from the electromagnetic field from the sensors when it approaches them. Because the battery of the brick has limited abilities the velocity between the sensor and the transponders is up to three centimeters which is ideal for this project, as we use the remote-coupling method. (Bangall,2011)

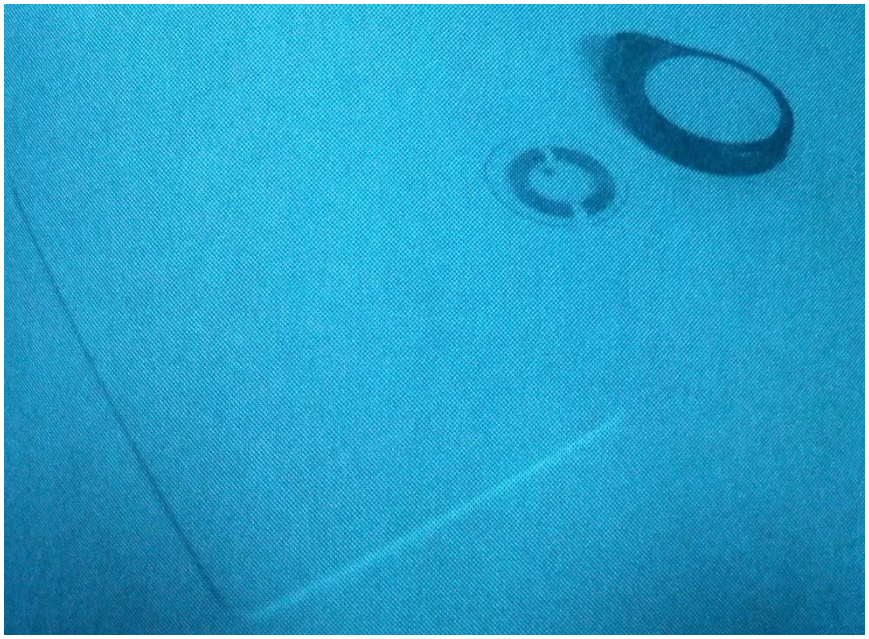


Fig 14: The Lego NXT RFID transponders (Bangall,2011)

The RFID systems are very good solution for object recognition in robotic systems. As for the code the NXJ leJOS API supports RFID systems by a package called lejos.addon which contains the class RFIDSensor. This class has one method that allows the data reading from the sensor in order the brick to analyze the values from RFID transponder. (Bangall,2011)

Fig 15: The Lego NXT RFID sensor (Bangall,2011)

3.9 Sound in NXT

The NXT brick supports the play of tones. The leJOS NXJ contains in the package lejos.nxt many methods for the control of sounds. Sound.playTone() method can play a tone with two values frequency and duration and by adding a delay after the tone plays. The NXT can also store playable sound files which can be uploaded to the brick. The LEGO software preferably uses files with .RSO extension. This format contains less sound information than others as MP3 and WAV but are ideal for this use because they occupies less memory in the brick. (Bangall,2013)

3.10 Subsumption architecture

"The subsumption architecture is a behavior-based decomposition, using a flat organization with suppression on the outputs". (Champandard,2010,p.65)

In 1986 Rodney Brooks implemented an approach in Robotics by using a design called behavior based robotics. The basic idea is the design of a robot is not consist of basic functions as perception, learning, planning but of behaviors like obstacle avoidance, line following, navigation as we made in this thesis. Each behavior has immediate access to sensors inputs independently and constant interaction with sensor data when are needed. Behaviors are set in a hierarchy diagram with priorities where the higher level behaviors can interrupt and override the lower level behaviors. The most expensive goal of robotic architecture is to eliminate the reliance of real time situation of the world state. (Russel,& Norvig,1995,p.791)

The corollary of Rodney Brooks Behavior Based Robotics is the subsumption architecture. This implementation of this architecture requires the most complex behaviors to be analyzed to simpler behaviors. (Bangall,2013)

The subsumption architecture is also called arbitration mechanism. With this technique selects between behaviors, every behavior wants to take the control every time. This architecture is based over control networks which are called "augmented finite-state machines (AFSMs)", also they include timers that allow the behavior rotation after some time. The AFSMs are triggered through sensors values and give the output to the actuator, for example in this thesis the NXT brick. The subsumption architecture is implemented through two techniques the "suppression overrides inputs" and the "inhibition restricts the output". Many of the AFSMs are ideal for Robotics. (Champandard,2010,p.65)

A behavior simply takes over also when the condition that has been set for every behavior comes true. In addition a behavior does not always depend on the sensors triggers in order to become true. The robot with the use of behaviors can monitor many external or internal situations, as the internal timer monitor time. (Bangall,2013)

More about how the management of priorities of the behaviors is done in the implementation chapter according with the development of diagrams which explain the procedure thoroughly.

3.11 The Subsumption API

The behavior API consists of only one interface and one class. The interface defines the methods. When the behaviors are defined then the Arbitrator regulates them. The interface of the behavior API it lies on lejos.subsumption package. It has three methods to control each behavior, when a behavior takes over, what to do and when to stop. (Bangall,2013)

- boolean takeControl() : Returns a Boolean value, when is true then the behavior becomes active.
- void action() : this method initiates an action when a behavior initiates.
- void suppress(): this method can stop immediately the method action(), also used to update any data to the brick before the behavior ends. (Bangall,2013)

Also there is one class in the lejos.subsumption.Arbitrator package in order to arrange the hierarchy of behaviors, when each one must be active. (Bangall,2013)

- Public Arbitrator(Behavior [] behaviors) : This class creates an object to control, the higher index number has a behavior the higher behavior priority is. (Bangall,2013)

When the Arbitrator object is instantiated then an array of behavior objects is declared in the code also. Then the start() method is called, which starts the arbitrating as described based on the hierarchy diagram. The Arbitrator object calls the takeControl() method of each object of the behaviors until it finds a behavior that wants to take the control. When this happens it calls the action() method of this behavior for one time. But if two or more behaviors want to take the control the same time then only the higher priority behavior will take control. (Bangall,2013)

3.12 Managing Risk

The risk management method that is been used in this thesis is divided in two techniques but the use of only one is done in the implementation chapter because it fits more for the project of medicine delivery robot. A briefly image provided below of how these techniques work. These two methods are suitable to identify the cause of faults in a procedure alongside with the ability in some cases to predict the outcome of some events. These methods which can work also complimentary are the fault tree analysis and the event tree analysis. (Danaher,n.d)

3.13 Fault tree analysis

The FTA method is part of risk management process. This method can analyze and understand and even prevent an unwanted event to occur.

The Fault tree analysis method, analyses a fault in a process by visualising the hazardous event in a logic model and estimating the frequency of occurrence. This fault analysis model examines many fault instances in basic systems, safety systems and human reliability. (Danaher,n.d)

The construction and the creation of such a model is done with the use of logic gates such as AND, OR. The result we want to analyse is called Top Event. We give a probability in this event as an outcome of the calculation of the probabilities of the basic event through the gates. The basic events are the total of the events that are interact in order the Top Event to be produced. (Danaher,n.d)

The basic hypothesis that is made in order to use the FTA method is that the system faults are binary in nature. It means that there are only two outcomes for a procedure either it will be done successfully or it won't. Furthermore it is a presumption that the system is will be successful and performs its task if all the sub events or sub systems are working perfectly. The FTA method can only deal with instantaneous faults not to examine the operation of a system over a time period. (Danaher,n.d)

The purposes that can be accomplished with the FTA method are following:

- Probability of the occurrence of a certain incident
- Specifying combinations of system faults, operation condition, human errors that can lead to an unwanted outcome
- Evaluation and provision of a variety of solutions

(Danaher,n.d)

The construction of a fault tree diagram is made through five steps:

1. **System analysis, including definition and quote of limits**

This is the first step in order the FTA method to be implemented. The causes that can lead in fault must be clear and totally understandable, this can be done through the analysis of the system the method will be applied. This is the step where the boundaries and the data needed of the system are set. The information that is required for the system analysis consist of the

description of the process, fault properties of the materials operating maintenance process etc. (Danaher,n.d)

2. **Identification of faults and determination of the top event**

A major variety of procedures are used to identify faults in a system and determine the top events. These procedures are site inspections, check lists, hazard operability studies and accident investigation. (Danaher,n.d)

3. **Actual implementation of the fault tree**

The fault trees are logic diagrams which can explain how an undesired event could take place and from which events consist of, also is constructed from the top down. The undesired outcome firstly is chosen to become the top event alongside with the causes and the logical relationship that lead to it.(Danaher,n.d)

4. **A qualitative analysis of the structure**

When the Fault tree is completed then it can be qualitatively analyzed as well for better understanding of the events and causes. Some fault trees are too complicated and can be applied Boolean defining expression for the top events as an outcome of the lower events. The top events are the logical Boolean sum of all lower events. (Danaher,n.d)

5. **Quantitative assessment of structure (statistical)**

Once the fault tree has been created and determined and a probability weight has been set to every basic event, it is easy then the probability of the top event with the use of Boolean algebra. All the lower event logic gates must be calculated before the calculation of the probabilities of the next higher level, addition for OR gates and multiplication for AND gates. (Danaher,n.d)

As a conclusion the FTA method is widely used as an analysis tool. There are many theoretical publications and papers according this technique which describing very well the methodology. In addition in the implementation chapter a furthermore analysis exists in this thesis. A significant advantage on this method is the additional data that are extracted after the implementation of the fault tree analysis. These data are the outcome of the qualitative and quantitative analysis as described above. There is also a downside while using this analysis method which lies in the complicity of developing the fault tree if the analyst make wrong in calculations over the probabilities or fault assumptions on basic events. (Danaher,n.d)

3.14 Event Tree Analysis

Event tree analysis creates a logic model diagram which examines the outcomes of a primary event. It can provide the time sequence of the event or the spreading events after the initiating event, with the use of protective actions or any interventions that can be made or the consequences of the event. But this methodology cannot guarantee the 100% avoidance of an unwanted fault or event and in contrast that is a valuable method complimentary to the FTA method is not used in this thesis because the robot model is an application that has interaction with humans and this requires zero tolerance in errors. In order to base this perception, the ETA method is applied in two versions. Firstly is applied in pre incident and secondly in post incident applications. For the pre incident the cause is to check the system while an incident precursor exists and then try to prevent it of becoming a real incident. In this thesis where we have a system that is not allowed to make no mistake this method is not very attractive. Finally the post incident application can give a variety of possible outcomes dependable on events. (Danaher,n.d)

Chapter 4

Implementation

This chapter illustrates the implementation of the medicine delivery robot. Thus, here all the methods that were used in order to design are been presented. The first subchapter analyzes the prototype I theoretically used so as to exploit all the capabilities that Lego NXT.1 has to offer. Then, the next subchapter cites the programming method used in order to ensure a 100% delivery of a medicine to a patient. Lastly the quote of the source code in Java is done.

4.1 NXT Forklift as prototype

The autonomous medication delivery robot was made having the NXT.1 form Lego Mindstorms as prototype. It was programed using the NXJ software which gives almost everything from standard Java.

- a Java Virtual Machine where the code runs
- variety of classes
- tools to compile and debug the code
  (Bangall,2011)

Moreover the code was structured with subsumption architecture. This architecture is composed from many simple behaviors in order to create more complex behaviors. There are two ways to build a behavior. First use on an input, secondly use of an output. This also constructs the way all living beings behave. In robotics the input means the data stream from the sensors either digital or analogic and the output means the reaction of the robot. This reaction is triggered by the sensors and it could be for example a movement using the motors. This condition that the sensor is triggered and a reaction follows is called behavior. (Bangall,2013)

Below the algorithm procedure of medication delivery robot where the code is based in, is explained thoroughly. In this algorithm FTA technique is applied in order to eliminate the risk and ensure fail safe operation.

The medication delivery robot is a theoretical model only the code and the fault analysis were implemented actually. I used the capabilities of NXT in order to make the code with behavior programming. This model was based on NXT 1.0 Lego design called Forklift, but altered a little

bit for the purpose of the project. I added one RFID and one light sensor more in contrast with the original design in order the design to deliver the medicines efficiently to the patients without any faults. (Nxtprograms.com,2014)

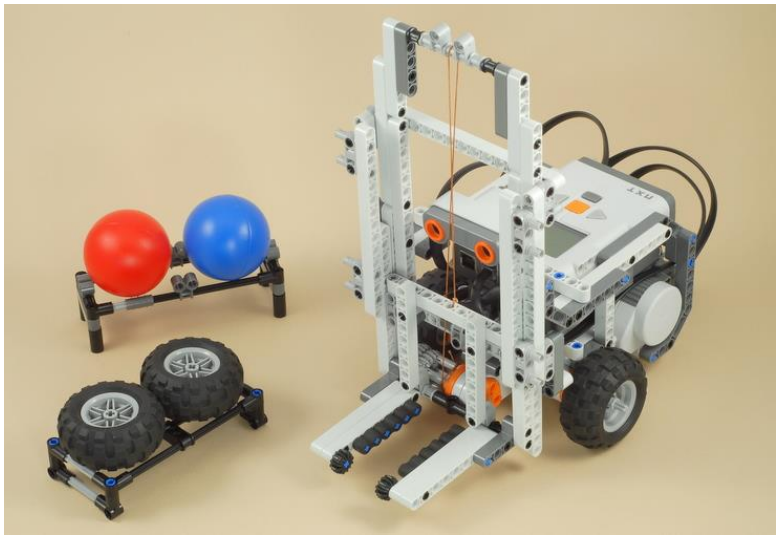


Fig 16: The original Forklift Design for NXT 1.0 (Nxtprograms.com,2014)

Briefly I refer the materials I used for the theoretic construction of the medication delivery robot:

- NXT Brick: The NXT Brick contains an Atmel 32-bit ARM processor 48 MHZ with 256KB of FLASH memory and 64KB of RAM and 8-bit Atmel AVR ATmega48 microcontroller
- The Bluetooth CSR Blue Core™ 4 chip inside the NXT brick
- Sound amplifier chip with the ability to play a sampled sound
- Three servo motors
- Touch sensor
- Light sensor
- Ultrasonic sensor
- RFID reader sensor
- Seven RJ12 Connectors

(Bangall,2013)

The code was made in order to implement the following process. To begin with we first suppose that we have made the match with the tags from the bed and patients hand with the RFID sensor, because different values applied in every tag for the validation process. Also we make sure that the Bluetooth chip has been paired before with the bed adapter. Every paired device is added in an array inside NXT memory. (Moral,2009,p.154-p.155)

This is the calibration we need before we start the process. Hence the process is automated. The medication delivery robot is placed in the nursery where the drug dispenser is based in. The nurse prepares the medication for a specific patient with his own RF-tag in his arm, who is placed in a specific bed with a Bluetooth adapter and RF-tag attached in the bed. The nurse place the medicine over the platform of the robot, then she power up the robot and the automation delivery begins. The robot is designed to follow a black line. A black line must be painted from the starting point across the bed of the patient and backwards in circular way in order the robot to find the way back to the starting point. The robot follows the black line until the Bluetooth chip is paired with the Bluetooth adapter of the bed, if pairing is not successful the robot continue to follow the line. If the pair is successful the robot moves toward to the bed. The range of the Bluetooth devices is 10 cm and the maximum distance is 10 meters in addition the range of the RFID sensor is varies from 0–1 cm, 0–1 m or up to >1m. (Διακονικολάου,Αγιακάτσικα,& Μπούρας,2007,p.270)

These ranges both from Bluetooth or RFID devices are ideal for the project because it can be used as we need, we first set the first safety net, the Bluetooth to check in 5 meters range and then we set the RFID reader to check from 0-1 m for validation as the second safety net. (Διακονικολάου,Αγιακάτσικα,& Μπούρας,2007,p.270)

If the validation with the RF-tag of the bed and the RF-tag of the patient is correct (the validation of both RF tags is required in order the robot to move forward) then the robot continues to move forward to the bed. If the validation is not correct then the robot turns back and tries to find again the black line in order to keep following it. This is another major check point that serves the cause of ensuring the safe automated operation of medicine delivery. Then the robot moves forward to the bed until the touch sensor is pressed. If the touch sensor is pressed then it raises the platform with the medicine near the patient. When the platform rises up it plays a prerecorded tone for 5 sec that warns the patient to receive the medicine. The robot waits for additional 10 sec for the patient to take the medicine then the robot turns back and rotates until it finds again the black line to follow. For safety reasons the robot is coded to avoid obstacles using a specific routine. The ultrasonic sensor is always open as explained in behavior programming tactic. If the sensor returns a certain value, a specific threshold was placed in the code then the avoid obstacles and collision behavior takes over. This behavior ensures the obstacle and collision avoidance guaranteeing the elimination of a hazardous event.

This is the automated delivery of a medicine to a patient without the interference of any human from nursing room to the bed.

4.2 Programming with behavior using Subsumption Architecture

The Subsumption Architecture is an excellent programming method very simple to implement providing predictable results. The programmer can organize the code in independent parts called behaviors which can be combined in order the desirable outcome to be produced. (Champandard,2010,p.65)

This is the basic advantage and this is why is been chosen for this thesis. We want to ensure that no fault will take place, and this way of programming ensures safe outcome. The Subsumption Architecture includes a mechanism called arbitrator. This mechanism selects every time which behavior will take over. (Champandard,2010,p.65)

All the behaviors are included in an array, the programmer sets the hierarchy, and the arbitrator selects the behaviors based on this. When certain stimulation occurs in the robot then the arbitrator gives the control in the behavior that will deal with the situation based in the hierarchy array and suppress the behavior that is running. (Bangall,2013)

The delivery process was explained thoroughly in the previous chapter; the basic point is that we need the robot to deliver the medicine from one point to another without a fault or be lost in its way and finding the right bed and patient.

The above table shows the behaviours that we based in so as the delivery to be implemented. We can also see there the priorities we gave for every behavior. We based in this table in order to create the array list inside the code which the arbitrator choses from every time.

Table 1: Drug delivery Robot Behavior priorities

Behavior	Condition	Action	Priority
Follow line	Seek the black line with light sensor	Follow the black line	low
Avoid Collision/Object	Ultrasonic sensor detects object	Stop moving for 5 sec if the object is still there makes an avoiding movement to find again the black line	high
Find the right bed	Bluetooth search for a connection	Try to bridge with bed's Bluetooth adapter for confirmation	high
Find the right patient	Rf id sensor searches for a validation with both bed and patient	Confirm the id of the patient's bracelet and the id of the bed	high
Deliver the medicine	Touch sensor is pressed	Give the patient's medicine and play a tone so as to warn the patient to take it	medium

When a condition becomes true then a behavior takes over the control, depending on the external sensors or Bluetooth chip. Furthermore the flowchart of the code is described above. When this happens the lower priority is been suppressed.

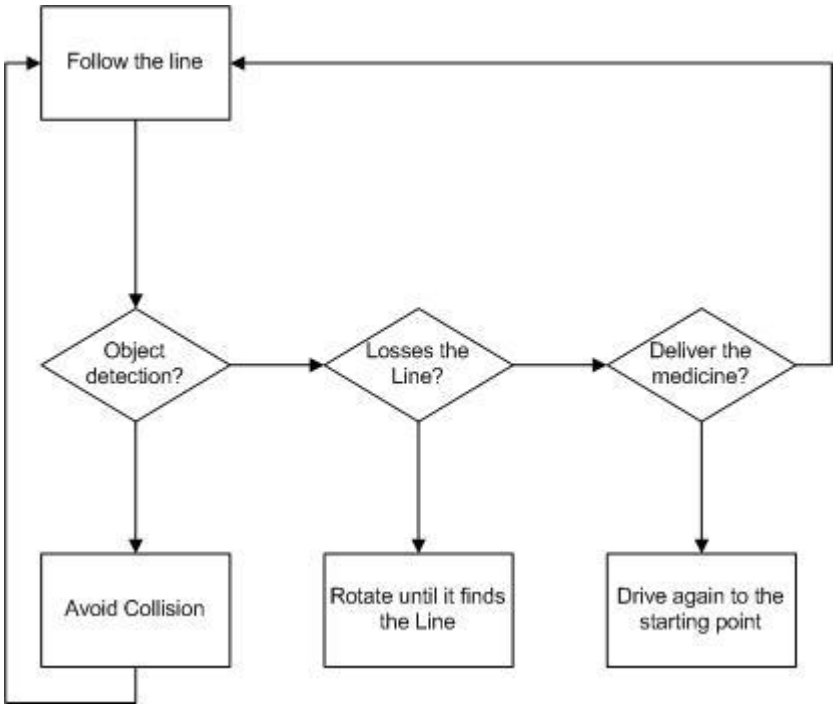
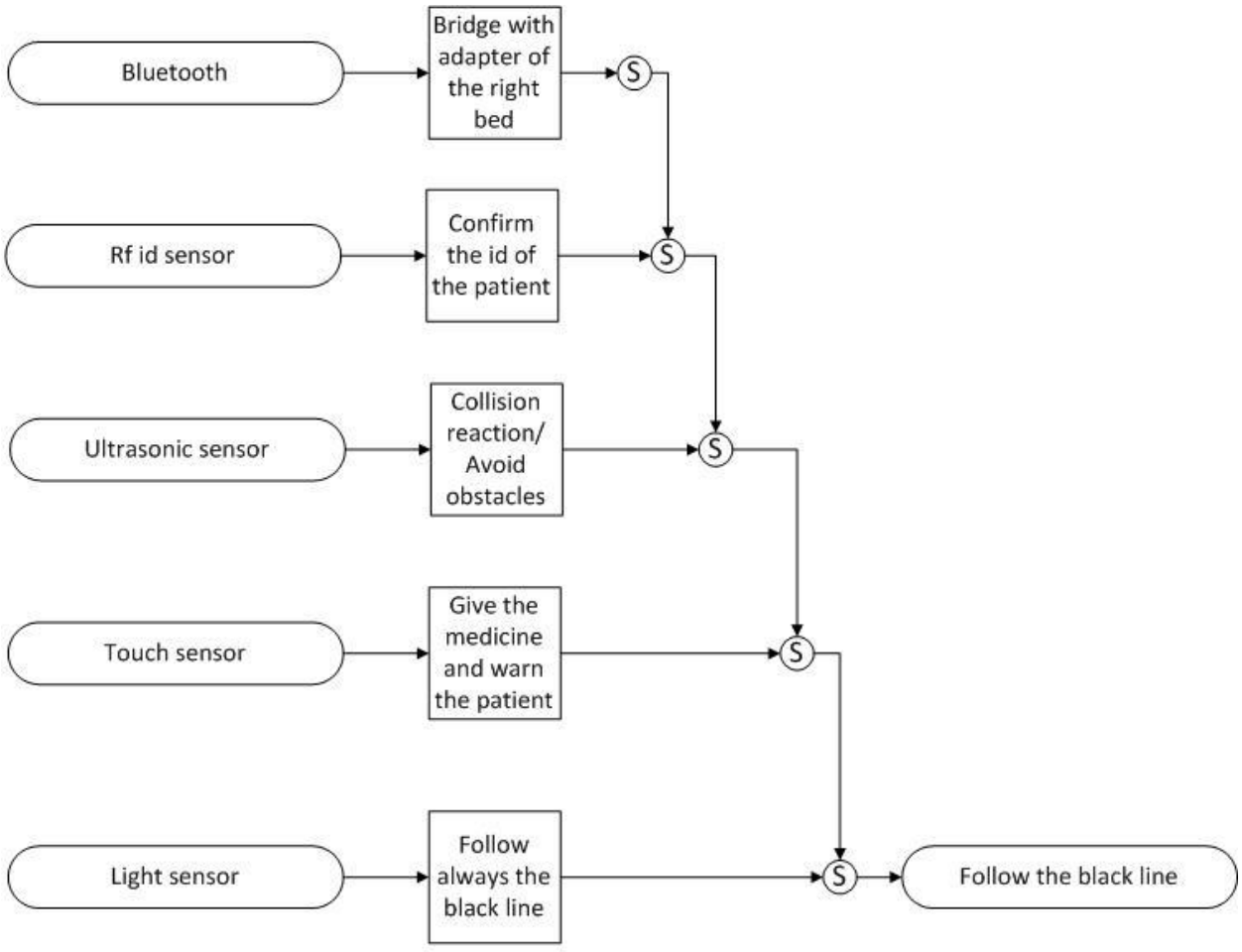


Fig 17: Structured programming visualized

Rodney Brook developed a diagram to represent the hierarchy of the behaviors. The standard diagram shows all the behaviors and the priorities that have been made. In the diagram when one behavior is suppressed by another we show that we an S. Hence the basic rule of the Subsumption Architecture is that the lower level behaviors are suppressed when a higher based behavior takes over. Only one behavior can have the control every time. (Bangall,2013)

The following diagram is based on the standard Diagram of Rodney Brook for representing hierarchies of behaviors for medicine delivery robot.



(S) = Point of Suppression

Figure : Drug delivery robot hierarchy

Fig 18: Drug delivery robot hierarchy

The higher priority behaviors starts form low in this diagram and suppressed when a sensor is triggered.

4.3 Source Code

The lejos software supports a subsumption framework. The subsumption architecture uses behavior based code based on Java. Each behavior is defined by a separate class which contains at least three methods.

- action – this method defines what to do
- suppress – this method defines how to stop doing it
- takeControl – this method defines when something will be done

Each behavior is added in an array by priority. The arbitrator is in charge of the behaviors. The arbitrator starts and then calls the behaviors to check which one is triggered every time. (Joseph,Schad,Fullon,2014)

For this project, the medicine delivery robot, we have five Behavior classes in addition one class with arbitrator and on class is the RFID which is used for the validation of the RFID sensor with the tags. Below the classes will be explained more along with the source code from eclipse.

Drug delivery robot Class

AvoidObstaclescollision.java | *DrugDeliveryrobot.java | FindBed.java | FollowLine.java | GiveMedicine.java | RfidMatchbed.java | RFID.java

```
package com.manos;


import lejos.nxt.SensorPort;
import lejos.robotics.subsumption.Arbitrator;
import lejos.robotics.subsumption.Behavior;


public class DrugDeliveryrobot {
    public static void main(String [] args) throws Exception  // An exception is an event, which occurs during the execution of a program,
                                                               //that disrupts the normal flow of the program's instructions and
                                                               // must be considered for examination of the error message
    {

        Behavior b1 = new FollowLine (SensorPort.S1);
        Behavior b2 = new GiveMedicine(SensorPort.S2);
        Behavior b3 = new AvoidObstaclescollision(SensorPort.S3);
        Behavior b4 = new RfidMatchbed();
        Behavior b5 = new FindBed();
        Behavior [] bArray = { b1, b2, b3, b4, b5}; // the lowest priority behavior takes the lowest array index
        Arbitrator arby = new Arbitrator(bArray);
        arby.start();
     }
   }// end of Arbitrator
```

Fig 19: Drug delivery robot Class (Barnes,Kolling,2011)

Here is the main method containing the arbitrator. This class instantiates the behaviors, next the behaviors are placed in an array with the lowest priority Behavior taking the lowest index in the array. Finally in the code the arbitrator is created, and the last line starts the arbitration process.

Avoid Obstacles class

*AvoidObstaclescollision.java | *DrugDeliveryrobot.java | *FindBed.java | FollowLine.java | GiveM

```
package com.manos;

import lejos.nxt.SensorPort;
import lejos.robotics.subsumption.Behavior;
import lejos.util.Delay;
import lejos.nxt.Motor;
import lejos.nxt.*;

public class AvoidObstaclescollision implements Behavior {
    public static boolean suppressed;
    private UltrasonicSensor sonar;

    public AvoidObstaclescollision(SensorPort S3) {
        sonar = new UltrasonicSensor(SensorPort.S3);
    }
    public boolean takeControl()
    {
        return  sonar.getRange() < 5;//check for a certain value to start
    }
    public void suppress() {
        suppressed = true;

    }
        public void action()
    {
         Motor.B.setSpeed(700);
         Motor.C.setSpeed(700);

         if (sonar.getRange() < 5) { //check for a certain value to wait and avoid obstacle
            Motor.B.stop();
            Motor.C.stop();
             } else {
             Delay.msDelay(2000);
             Motor.B.backward(); // go back and avoid any obstacle
             Motor.C.backward();
             Motor.B.rotate(180);
             Motor.C.forward();
             Motor.B.forward();
             Motor.C.rotate(90);
             Motor.B.forward();
             Motor.C.forward();
             Motor.B.rotate(-180);
             Motor.B.forward();
             Motor.C.forward();
             Motor.C.rotate(-90);
         }}}
```

Fig 20: Avoid Obstacles class

This behavior is used for the robot to avoid an object when following the black line or avoid a collision. When the sonar sensor realises that an object above a certain threshold, the take control method returns true no matter what is happening. The suppress method is used to stop this action when is called.

The behavior of avoiding the obstacles will take over in action method when the takeControl method returns true. Then the robot stops for a little to avoid a collision goes backward rotates 180 degrees and then returns in the line in order to avoid any obstacle. This is the behavior when the sonar is triggered.

Follow line class

*AvoidObstaclescollision.java | *DrugDeliveryrobot.java | FollowLine.java | *FindBed.java | GiveMedicine.java | RfidMatchbed.

```
package com.manos;  // inspired by the code of Juan Antonio Brena Moral, Book: Develop Lejos programs Step by Step p162

import lejos.nxt.SensorPort;
import lejos.robotics.subsumption.Behavior;
import lejos.nxt.*;

public class FollowLine implements Behavior {
    public static boolean suppressed;
    private LightSensor light;

    public FollowLine(SensorPort s1) {
         light = new LightSensor(SensorPort.S1);
         light.setFloodlight(true); // Use the of light sensor as a reflection sensor
    }


    public boolean takeControl() {
        return true;
    }


    public void suppress() {
          suppressed = true;

    }

    public void action() {            // follow the line

        while (light.readValue()<45){          // 45  used as Threshold
            MotorPort.C.controlMotor(0, 3);
            MotorPort.B.controlMotor(80,1);
        }

        while (light.readValue()>=45){
            MotorPort.C.controlMotor(0, 3);
            MotorPort.B.controlMotor(80,1);
        }
    }


}
```

Fig 21: Follow line class (Moral,2009,p.162)

This class is very important because through this class the robot moves in the black line. This behavior uses the light sensor as a reflection sensor. If the takeControl method returns true, this behavior takes control in the robot then pilots the robot accordingly to the black line in order to deliver the medicine. The suppress method stops the robot to move when another behavior is triggered. While the action method is in action the robot reads a value when the light returns to the sensor. A standard threshold has been set, if the value that returns to sensor is lower than the threshold then the motor B in controlMotor low level method moves forward with a value of 80 in power and the ControlMotor stops the motor C. In addition if the returned value in the sensor is equal or higher of the threshold then exactly the opposite as described above happens. As a result the robot always travels toward the black line without losing the line. You can also find more details about LeJos API in Appendix A.

Find bed class

```
package com.manos;
import javax.bluetooth.RemoteDevice;

import lejos.robotics.subsumption.Behavior;
import lejos.nxt.Motor;
import lejos.nxt.comm.Bluetooth;
import lejos.nxt.comm.NXTConnection;
import lejos.nxt.comm.BTConnection;

public class FindBed implements Behavior {
    public static boolean suppressed;
     String name = "Btdevice";
      public boolean takeControl() {
         final byte[] pin = {(byte) '0', (byte) '0', (byte) '0', (byte) '0'}; //the pin of the BTdevice in the bed
         RemoteDevice device = Bluetooth.getKnownDevice(name);  // the device in the bed that have been paired before
                                                                //the automation drug delivery begins
         if (device == null) {

             return false;             }
                   BTConnection connection = Bluetooth.connect(device.getBluetoothAddress(), NXTConnection.RAW, pin); // makes authentication with
                                                                                        //the pin or address of the Bluetooth device of the bed
            if(connection == null) {
                return false;      }
              return true; //Connection
              }
       public void action() {
       suppressed = false;
       final byte[] pin = {(byte) '0', (byte) '0', (byte) '0', (byte) '0'};
       RemoteDevice device = Bluetooth.getKnownDevice(name);
       BTConnection connection = Bluetooth.connect(device.getBluetoothAddress(), NXTConnection.RAW, pin);
       Motor.B.setSpeed(700);
       Motor.C.setSpeed(700);
           if (!suppressed ){  // the robot moves forward until it suppressed by the Rfid behaviour,
           //                then the Rfid validation takes over, if the validation is not correct other lower behaviour takes over
           Motor.B.forward();
           Motor.C.forward();  }
              else {                  // the bluetooth connection is closed and a lower connection takes over
          Motor.B.stop();
          Motor.C.stop();
          connection.close();
             } }
    public void suppress() {
        suppressed = true;
    }}
```

Fig 22: Find bed class

(Wooldridge,2012) (Bangall,2013) (Java for LEGO Mindstorms,2014)

This is the class where the Bluetooth chip bridge with the adapter of the patient bed. While the Bluetooth chip is always on, the pin of the Bluetooth device paired with the robot chip and the method takeControl takes over. Then the action method runs and the robot moves forward until it finds the RFID sensor for the second validation. If the pairing fails, the robot stops and closes the Bluetooth connection. Then another behavior takes over as line follower and the medicine will not be delivered to the patient.

Give medicine class

*AvoidObstaclescollision.java | *DrugDeliveryrobot.java | FollowLine.java | *FindBed.java | *GiveMedicine.java

```
package com.manos;

import lejos.nxt.SensorPort;
import lejos.nxt.TouchSensor;
import lejos.robotics.subsumption.Behavior;
import lejos.util.Delay;
import lejos.nxt.Motor;
import lejos.nxt.Sound;

public class GiveMedicine implements Behavior {
    public static boolean suppressed;
    TouchSensor touch;
    public GiveMedicine(SensorPort s2)
    {
         touch = new TouchSensor(SensorPort.S2);
    }
        public boolean takeControl() {
           return touch.isPressed();
        }
    public void suppress() {
        suppressed = true;
    }   /*  Motor.A.stop();
        Motor.B.stop();
        Motor.C.stop();
        }*/
        public void action() {
            suppressed = false;
            Motor.B.setSpeed(700);
            Motor.C.setSpeed(700);
            Motor.A.setSpeed(500);
            Motor.A.forward();  // rise the medicine platform near to the patient
            Motor.A.stop();
            Sound.playTone(220,5000); // play a tone for 5 sec to warn the patient about taking the medicin
            Delay.msDelay(10000);  // wait 10 sec for the patient to take the medicine
            Motor.B.backward(); // go back
            Motor.C.backward();
            Motor.A.backward(); // lower the platform with the medicine
            Motor.A.stop();
            Delay.msDelay(2000);
            Motor.C.rotate(-360, true); // rotate to a position where is able to find again the black line
                }
        }
```

Fig 23: Give medicine class

(Wooldridge,2012) (Bangall,2013) (Java for LEGO Mindstorms,2014)

The use of this behavior is to deliver the medicine to the patient after the robot passes the two safety checkpoints. When the robot moves forward after the validation from RFID sensor is true, the Give medicine class takes over when the Touch sensor is pressed. Then the takeControl method takes over and this behavior is activated. The action method is very simple.

The robot raises the medicine platform near to the patient. Then is plays a tone for five seconds to warn the patient about the medicine. Then the robot waits for ten seconds for the patient to take the medicine. After that the robot turns back, lowers the platform using the motor and rotates until it finds again the black line. Then the line follower behavior takes over and the robot returns to the starting point after it delivers the medicine.

FRID match bed class

*AvoidObstaclescollision.java | *DrugDeliveryrobot.java | FollowLine.java | *FindBed.java | *GiveMedicine.java | *RfidMatchbed.java | *RFID.java

```
package com.manos; //inspired by the book Intelligence Unleashed p 430

import lejos.nxt.Motor;
import lejos.nxt.TouchSensor;
import lejos.robotics.subsumption.Behavior;

import com.manos.basic.RFID;

public class RfidMatchbed implements Behavior {
    public static boolean suppressed;
    TouchSensor touch;



    public boolean takeControl() {


        if ((RFID.detectRfid() == RFID.patientRFID) && (RFID.detectRfid() == RFID.patientbedRFID)) { // This is double confirmation that
                                                                                                     //         is required in order the robot to give the drug to the patient
            return true;
        } else {
            return false;
        }


    }

        public void action() {   // We send the robot forward in order to touch the bed and the touch sensor to be triggered.
                                 //This trigger also the GiveMedicine Class
        suppressed = false;
        Motor.B.setSpeed(700);
        Motor.C.setSpeed(700);

        while (!touch.isPressed() ){
        Motor.B.forward();
        Motor.C.forward();  }

    }


    public void suppress() {
        suppressed = true; // here is the point where the validation with Rfid is not right so the robot will not proceed
                           //to give the medicine but a lower behaviour will take over from array

    }
}
```

Fig 24: FRID match bed class
(Bangall,2011,p.430)

The utility of this class is to ensure that the medicine will delivered to the right patient. Is been used as secondary safety net beyond the Bluetooth. This behavior is suppressed but when the robot passes the Bluetooth validation and moves forward, it takes over if the validation with the RFID of the bed and the RFID of the patient are correct and true. This is the role of the takeControl method, if this method becomes true the FRID match bed class runs the robot. Then the action method force the robot to move forward until it finds the bed and the touch sensor in front of it is pressed. If the touch sensor is pressed then accordingly to the hierarchy diagram the Give medicine class takes over in order to deliver the medicine to the right patient.

RFID class

AvoidObstaclescollision.java | DrugDeliveryrobot.java | FollowLine.java | FindBed.java | GiveMedicine.java | RfidMatchbed.java | RFID.java

```
package com.manos.basic;
// the way inspired from https://code.google.com/p/lejosgroup02/source/browse/lejos/CarrierBlue/nl/fontys/prj31/?r=26#prj31%2Faction



import lejos.nxt.SensorPort;
import lejos.nxt.addon.RFIDSensor;

public class RFID {

    public static final long patientRFID        = 123456L;        // Tag patient   We use 2 tags 1 in the bed and 1 in the patient's arm,
    //this because we want the robot to make double confirmation with the rfids before it gives the medicine to patient
    public static final long patientbedRFID        = 213456L;        // Tag bed


    public static long detectRfid(){
    RFIDSensor rfid = new RFIDSensor(SensorPort.S4);
    rfid.startContinuousRead();
      return rfid.readTransponderAsLong(true);
}
}
```

Fig 25: RFID class
(lejosgroup02 Lejos Project with NXT Programming,2009)

Next, the RFID class is added. This class is used to initialize the RFID tags of the patient and bed with a standard code. Also it has the detectRfid method which initiates the sensor to start seeking the tags. We call this method in RFID match bed class when the robot in this behavior is trying to validate the tag of the patient and the bed in order to continue the procedure of giving the medicine.

Chapter 5

Risk management ensuring fail safe operation

Risk Management is a very important part of every project in addition to if the project has any relation with humans. Fault Tree Analysis (FTA) and Event Tree Analysis (ETA) are used as tools in this project so as to eliminate the risk. These tools are complimentary and usually used by most project managers. Both these tools implemented and focus on opposite sides of a fault. (RMS Publishing,2011)

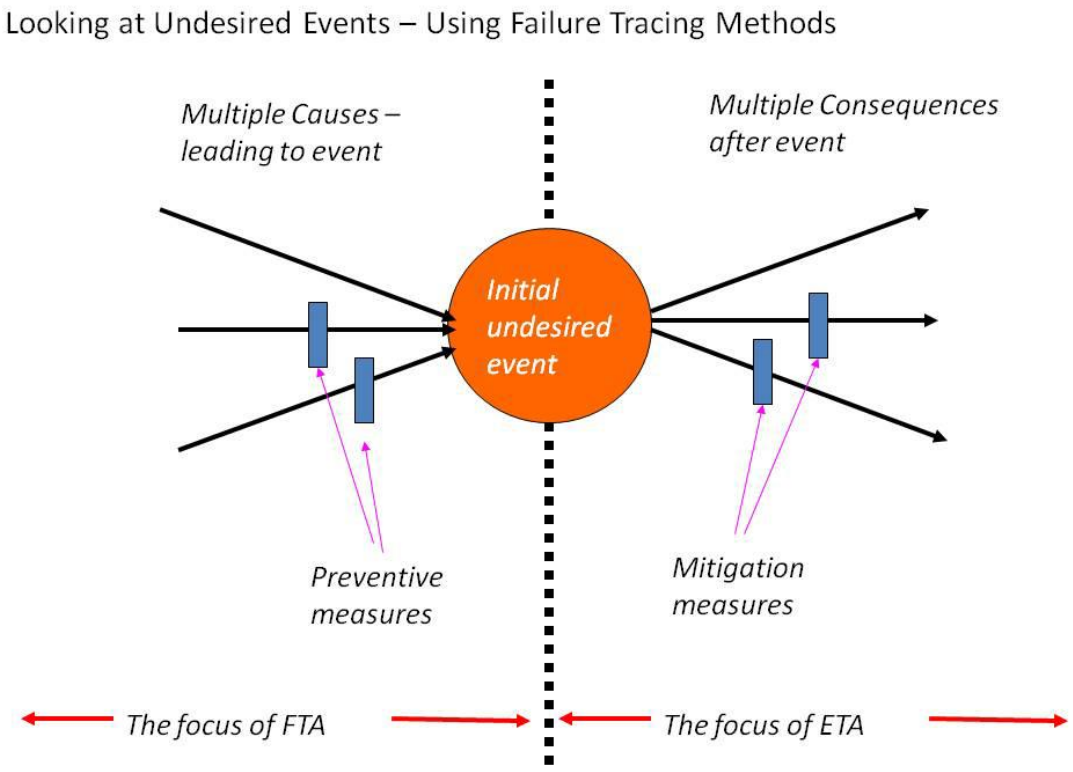


Fig 26: Failure Tracing Methods (RMS Publishing,2011)

The diagram above called bow-tie model. When both fault analysis techniques used together, this method called “bow tie technique”. The diagram in fact shows only single events at first. Every cause can of course lead to a different initial undesired event with multiple consequences after the event. Every event can be analyzed with the use of FTA and ETA. Generally FTA technique is used to analyze a fault which may lead to an undesired event. On the contrary ETA is used to stop the fault before it becomes catastrophic. The results are better in fail safe control when these methods are used together when is possible. (RMS Publishing,2011,p.1)

However i used only FTA analysis in this thesis, because it analyses the causes that lead to an event in contrary with the ETA that analyses the consequences after an event and we don't want an undesired event to take place. Also with FTA we can take prevent measures in order to avoid the undesired event instead with ETA analysis we can take measures to minimize repercussions of a fault or hazard. For this reason this method is not useful in this project.

The fail safe project for the medication delivery system ensures the safe operation in association with humans. No fault or error is acceptable in this autonomous delivery system. I have separated the whole procedure in two sections first the fault delivery of a medicine analyzed where we want 100% safe operation for the patient, the unwanted event is the patient to receive a wrong medicine. We ensure this by using behavior programming and several safeguards in the code. The second part is what hazards could happen in the delivery process which is also exceptional important when an autonomous system is used in a hospital. I implement the FTA analysis in both cases.

5.1 Fault Tree Analysis (FTA) in medication delivery system

There are many reasons always that can lead to a fault or accident during a process. FTA is an analysis method capable to investigate all the reasons, responsible for a failure or hazard and take preventive measures. The fault tree uses logic for analysis focusing in multi causality of events. Actually investigate all branches of the events that could lead to fail of a process. This method uses sort of symbols, labels and identifiers. The complete reference of the tools this method uses was made in previous chapter. (RMS Publishing,2011)

Very important in this risk analysis is to identify the hazardous events in the delivery process this is one part of the whole medication delivery system. In the other part the delivery of a medicine we cannot afford a hazard so we try another analysis approach. Before the implementation of the FTA method in the delivery process we perform a hazard analysis from which can produce a hazard list. But, firstly we can make a hypothesis that the autonomous system is a closed control loop. We divide into parts the intermediate events that can lead to a malfunction of the delivery. The parts are the Environmental impact, the Human error and the Hardware error.(Henke,2008)

These will be analyzed further while the FTA method will be conducted.

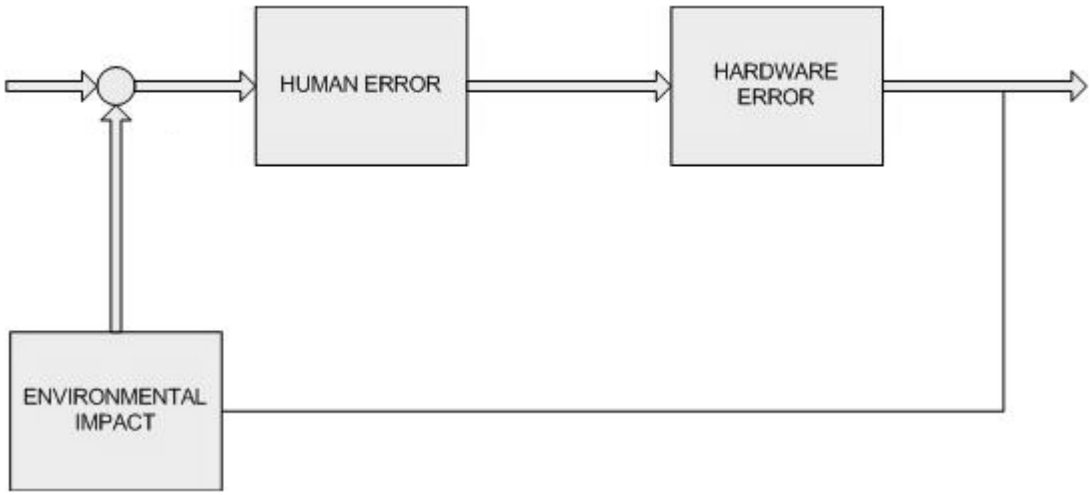


Figure 27: Medication delivery process Closed control loop analysis

From this hazard analysis we can extract of the primary events. These are the faults that may happen in accordance with the resulting accidents. With this list we have a better understanding about the faults that may occur and how we will prevent them.

Table 2: List of primary faults with resulting accidents

Process	Faults	Accidents
Human error	Nurse forgets to place the medicine Nurse mix up the RFID tags Patient cannot hear the Robot Patient is absent Technician made installation error	Patient does not receive the medicine Patient does not receive the medicine Patient does not receive the medicine Patient does not receive the medicine Patient does not receive the medicine
Hardware error	RFID not valid Bluetooth pairing problem Sensor/motor failure Battery/Power off	Authentication problem Authentication problem Delivery stops Delivery stops
Environmental impact	Faded line Obstacle collision	Robot not able to find the patient Collision robot off the course/fall of medicine

Fault Tree Analysis (FTA), delivery process failure

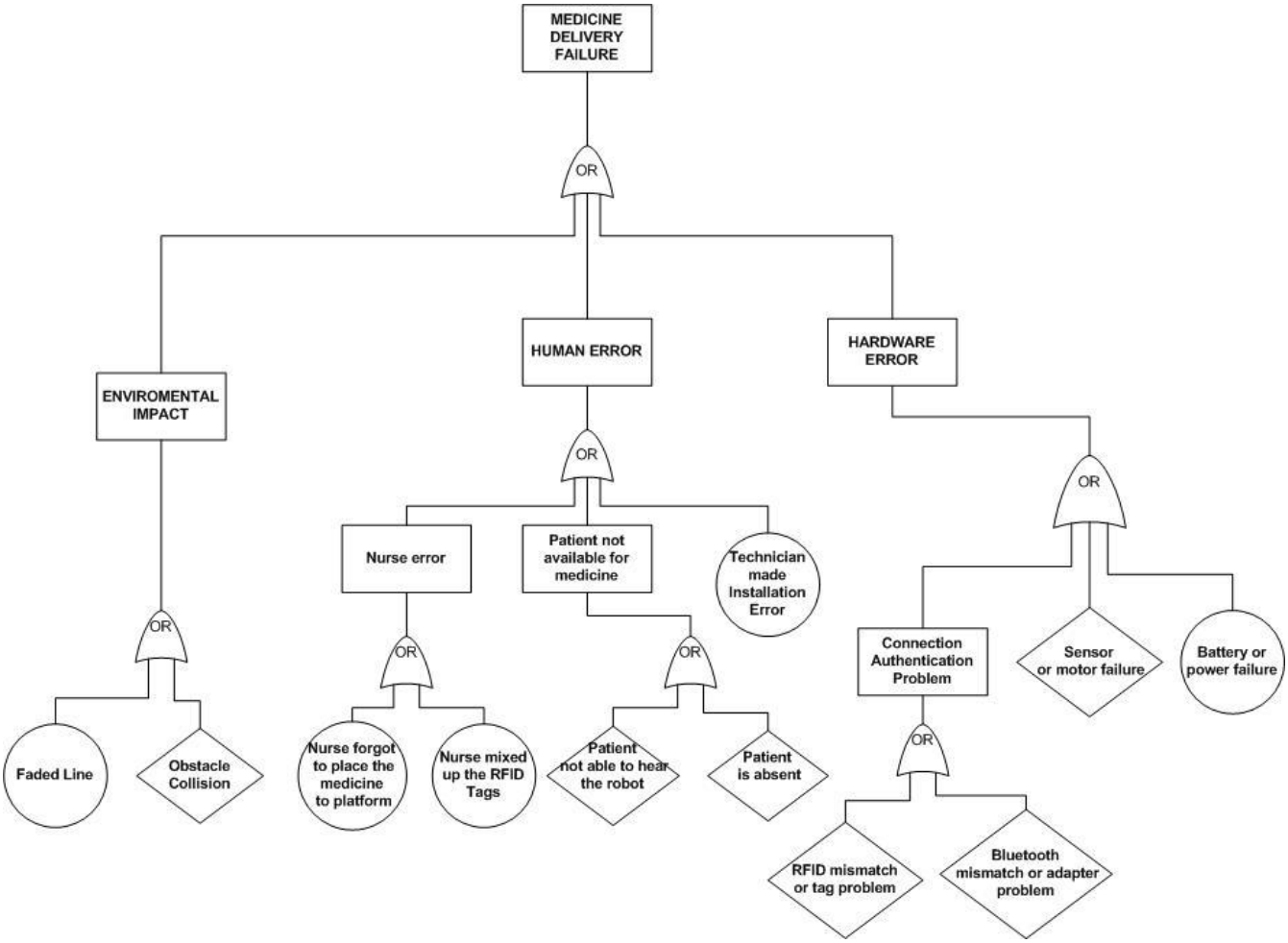


Figure 28: Fault tree analysis for the medication delivery system

The medication delivery system fault Tree Analysis starts from the top down. In the starting point is the undesired event called as top event, which is the failure of delivery. This failure embodies also the fault of medicine delivery to a wrong patient. There are several checking points in the whole process, which ensures that a medicine will not be given to a wrong patient. If the patient is wrong the robot will not deliver the medicine and it will return in the starting point and continue to run until it finds the right patient. I have drawn the rest of the diagram logically using logical ports with the affiliated conditions that lead to the top event. These conditions also caused by other faults and so on. These conditions are:

- Environmental impact
- Human error
- Hardware error

For our system to fail one of these failures is enough to be present, so I use an OR gate to connect them. To prevent the failure all these conditions must be prevented.

Going down to the diagram we can see for each condition the primary failures that lead to these conditions. Firstly, the Environmental impact failure. It is an intermediate event consists of faded line situation which means that the robot is not able to find its way and the collision with an obstacle. This is an undeveloped event which means that the robot is out of the way or medicine fallen out of the platform as a result of the collision. An OR situation also applies here because it would only need one situation here to be present in order the Environmental impact failure happens. Analyzing the Human error condition we see that it consists of a basic event that requires no further analysis, technicians fault during the installation and two events that are results due to the interaction of other events, Nurse Error and the unavailability of the patient to receive the medicine. I use the OR condition here because if one situation is true there will be a human error that will lead to medicine delivery failure. Analyzing further the nurse error consists of two basic events first the nurse forgot to place the medicine in the platform and the nurse mixed the RFID tags of the patients, any of these error if applied may lead to nurse error. Further intermediate event of patient's unavailability consists of two undeveloped events with or situation due to lack of suitable information. The patient is absent due to any reason or the patient is not able to hear the robot. Lastly the intermediate event Hardware error consists of a basic event which is battery or power failure, an undeveloped event which is sensor or motor failure due to any reason and intermediate event which is a connection authentication problem. All these faults consists the human error connected with an OR gate. The connection authentication problem is composed of two undeveloped errors connected with OR gate, the RFID mismatch or tag problem or the Bluetooth mismatch or adapter problem. To sum up these are all the faults explained that must be prevented in order the delivery to be successful. But the fault trees can also be quantified as well as in the RMS Publishing and Dr. Danaher article. (Danaher,n.d) (RMS Publishing,2011) The quantification can implemented from previous experience using statistical analysis in the matter or just estimation.

A probability in every primary failure will be given to Fault tree analysis for the medication delivery system based in the rules from Fault Tree Analysis (FTA) NASA HQ, Fault Tree Analysis (FTA):Concepts and Applications and RMS Publishing. These two sources agree in the procedure. (NASA HQ,2013,p.125) (RMS Publishing,2011)

We can get the final probability of the top event adding the probabilities which sit below an OR gate and multiply the probabilities below an AND gate. (NASA HQ,2013,p.125)

In our case Fault tree analysis for the medication delivery system has only OR gates. The range of the probabilities I set is from 0.01 to 1. This was made was arbitrarily. So the quantified diagram is the following.

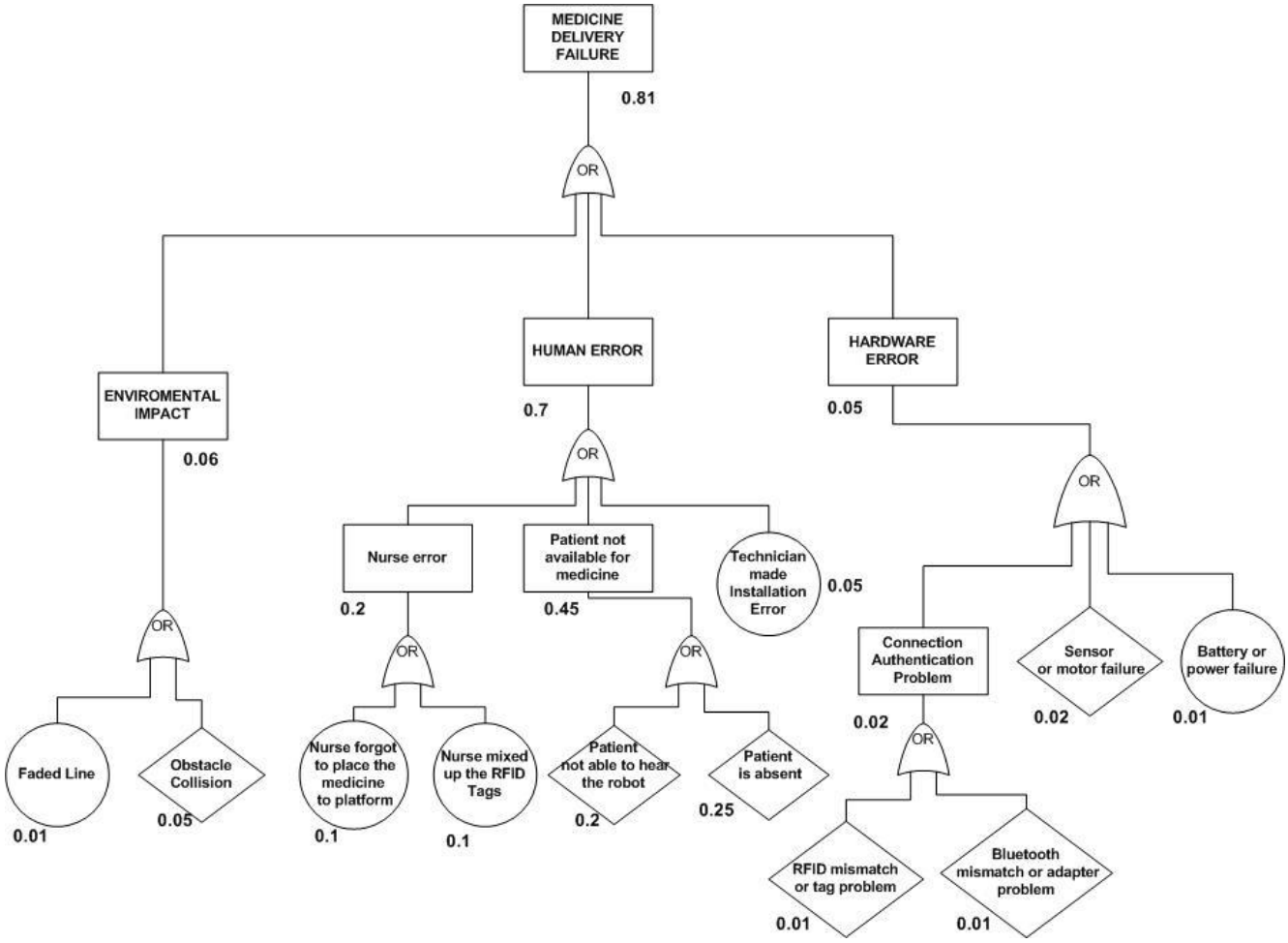


Figure 29: Fault tree analysis for the medication delivery system with probabilities

P(Environmental impact)= P(faded line)+P(obstacle collision)= 0.06

P(Human error)=P(nurse error)+P(Patients availability)+P(Technician Error)=0.7

P(nurse error)=P(Nurse forgot medicine)+P(Nurse mixed up the RFID tags)=0.2

P(Patients availability)=P(Patient is absent)+P(Patient cannot hear)=0.45

P(Hardware error)=P(Connection fail)+P(sensor/motor fail)+P(battery/power fail)= 0.05

P(Connection fail)=P(RFID mismatch/tag problem)+P(Bluetooth problem/adapter)=0.02

P(Medicine Delivery Failure)= P(Environmental impact)+ P(Human error)+ (Hardware error)=0.81

This is the quantification of Fault tree analysis for the medication delivery system, where we can extract our conclusions. I have given more weight in Human error intermediate probability event because is more difficult to control and predict.

Of course our primary objective is no false delivery ever happens, a false delivery of medicine is not allowed in our system. Because of that I set many checkpoints in order to prevent a false delivery, the system just do not deliver the medicine if everything is not right. We could not prevent a false delivery to a person with FTA. We have ensured that using behavior programming. FTA helps to take measures in order to prevent hazards in the process that can lead to a fail in the delivery operation.

Fault Tree Analysis (FTA), fault delivery of a medicine

With FTA we want to show here, how the medicine delivered to the patient succesfully and the safegurds that were used in order to ensure a 100% safe delivery without any mistake. Because of that FTA analysis for fault delivery of a medicine is not quantified. Below the events that can contribute to a false delivery presented.

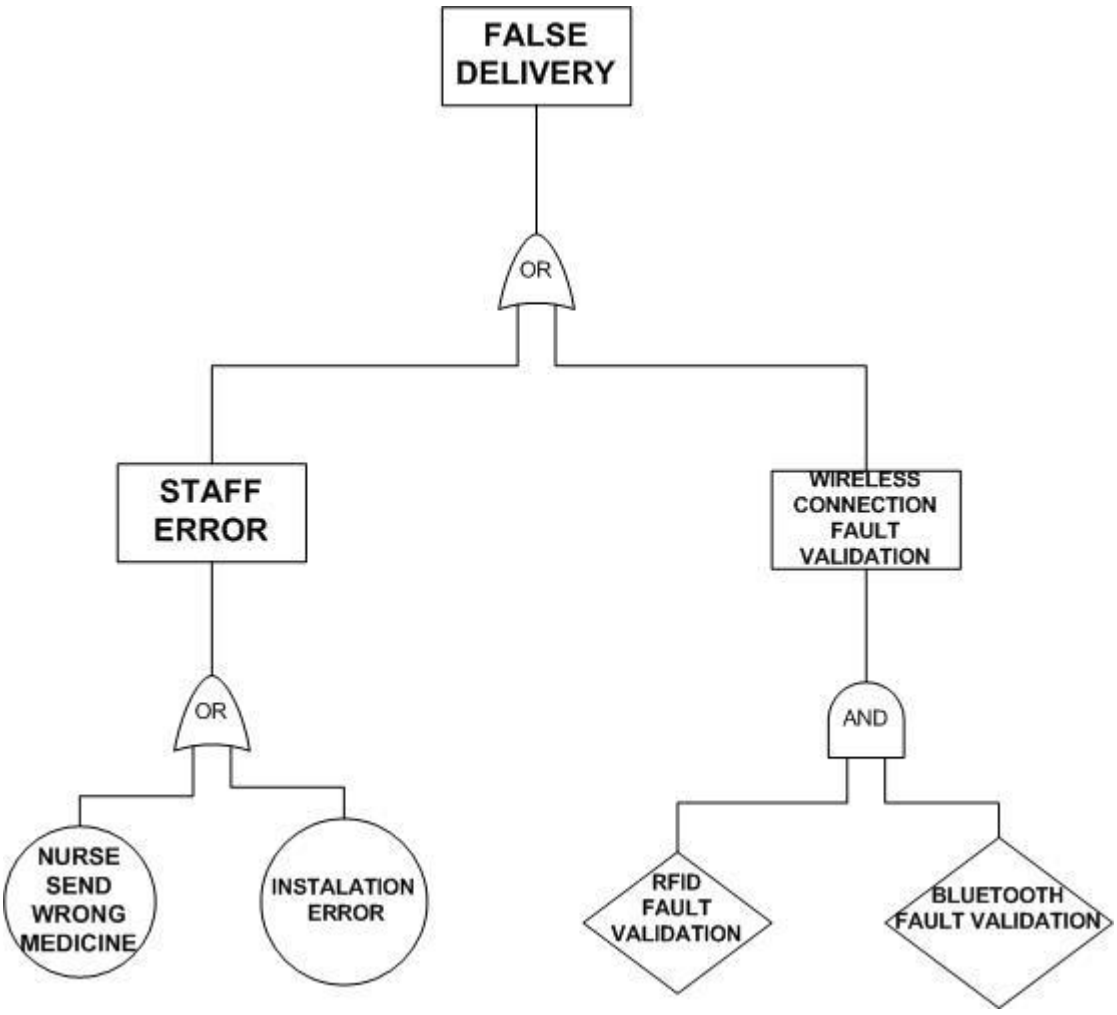


Figure 30: Fault tree analysis fault delivery of a medicine

If we examine one by one the primary events, we see that the second branch can be true and can lead to false delivery if for some unexplained causes exists. The calibrated RFID sensor in the robot must make a faulse validation with the tag in the bed and with the tag of the patient hand, in adition the blutooth chip of the robot must malfunction and connect unexpectetly with the bluetooth adapter of the bed.

This unlikely condition must exist in order a wrong medicine to be delivered to a patient. The RFID sensor and tags must malfunction but find a way to fault connect without validation also the bluetooth chip and adapter of the bed must malfunction but to find a way to make a connection without validation in order the robot to procced forward to the bed of the patient.

About the first branch if we examine the primary events that can lead to an intermediate event and then to false delivery. For a staff error event to happen one of these cases must become true. First a nurse must send a wrong medicine to the patient. If we suppose that a nurse is capable of this mistake then we also suppose that nurses make mistakes with patients medicines with or without the help of autonomous system. As for the other primary error we can avoid it by making many tests after the installation implemented.

Chapter 6

Future work - Conclusions

This chapter analyzes the future work before any conclusions are been placed. Here, this chapter also gives suggestions for future work based on the Mindstroms NXT 1.0 medicine delivery robot. The conclusions of the project are placed after that.

6.1 Future development

The medicine delivery robot has been implemented according the specifications of the Lego NXT 1.0 brick. However there are far more sensors and software tools that can evolve the project and can be applied for future work making the whole procedure more efficient and precise.

There is a wide list of third party sensors that can make the project more sophisticated and complex, with the additional added cost. Below are placed the list of sensors or adapters from other companies than LEGO. Some of them will be discussed as an extension of this thesis project in order future work to be implemented. (Bangall,2011)

HiTechnic

- Compass Sensor
- Color Sensor
- Tilt Sensor
- Barometric sensor
- Force sensor
- Magnetic sensor
- Passive infrared sensor
- Cables
- Cables
- IR Seeker
- Port Expander
- Motor Multiplexer
- NXT Prototype board

Website: www.hitechnic.com (Bangall,2011)

Mindsensors.com

- Camera subsystem
- Touch panel for NXT
- Numeric pad
- Real-time clock
- Tilt Sensor
- Compass
- Pneumatic Pressure Sensor
- Infrared Distance Sensor
- Temperature Sensor
- NXT motor multiplexer

Website: www.mindsensors.com (Bangall,2011)

Dexter Industries

- 9V solar panel to power NXT brick
- dSwitch to control household appliances
- dThermometer
- dPressure sensor
- dFlex bendable sensor
- dGPS
- NXTBee
- Thermal Infrared Sensor
- dIMU Inertial Motion Sensor
- WIFI adapter
- Compass Sensor
- Laser Sensor

Website: www.dexterindustries.com

(Bangall,2011)

Codatex

- RFID sensor

Website: www.codatex.com (Bangall,2011)

Vernier

- pH probe
- Low-g Accelerometer
- Conductivity Probe
- Dissolved oxygen Sensor
- UV Sensor
- Temperature Probe
- Magnetic Field Sensor

Website: www.vernier.com (Bangall,2011)

First of all in this thesis the robotic prototype based on NXT 1.0 Lego design called Forklift, but other designs can be used depending on the availability of sensors or motors and input ports. For example, if we use the port expander or motor multiplexer, more complex designs could be implemented. We can use more motors to deliver the medicine not just in front of the patient, but the robot could deliver it in the patient hands. In addition with the color sensor extra safety net could be implementing. We can link the medicine of a certain patient to a color bottle that contains it and the robot could select the medicine via color sensor from a dispenser and then deliver it to the patient. Moreover, the robot could load many of medicines in its tray and then deliver them in many patients in the same time selecting them with the use of color sensor with the condition that we have linked a certain color to every patient. Furthermore, we can use the force sensor in order the robot to understand whenever the nurse loads it with medicines. If it realizes that it is loaded then it will start the delivery process. This sensor also helps the robot to understand if the patient took the medicine while it checks the weight before and after the delivery. If the patient takes the medicine the robots tray should be lighter but if the weight is the same for example the robot could make one more turn so as the patient to have one more chance of getting the medicine. The force sensor could be used as a fail-safe mechanism as well. While the robot moves, we can make a behavior that checks the weight of the tray. If the weight is lower from the starting point before the robot delivers the medicine then something is wrong, and if we can adjust the priority of this behavior for example to return back to the nursery asking someone to check what is wrong. Thus, the nurse can load in a robot the whole daily treatment of a patient, while the robot moves always in the line and when it is time for a delivery to approach the patient and never get back in the nursery if it doesn't deliver the patient's daily treatment or when it has a problem. One more valuable addition in the navigation is the compass sensor. The robot can use this sensor to release when it has drifted away of his path. One more possible addition is the IR seeker in the robot and the passive infrared sensor in the patient's bed as an extra validation and confirmation that the bed where the delivery will be done is the right one. Admittedly, one more addition as the camera

subsystem could be used in order the nurses to have a remotely look in the patients; also, for monitoring purposes we can use the dThermometer sensor to check the patient's temperature and the oxygen sensor to check the oxygen concentration in the room. Additionally there is one more great future we can use with this project in a future implementation. The Lejos can also be used in Android applications as the Lejos API supports android. This is a serious benefit for the medicine delivery project because the nurses can use their mobile phones in order to check whether the medicine has delivered or to monitor the oxygen levels or the temperature of the patients without the need to go to every room as the robot can do this job and send data to android mobile phones via Wi-Fi or Bluetooth protocols.(Bangall,2011)

Another future thought based on this technology, is the robot to take orders from doctors like frequency or dosage parameters, via an android app and then simply choose the treatment from a dispenser and deliver it to the right patient based on an internal database, excluding almost the nurse of this procedure.

Furthermore because the NXT MINDSTORMS is very popular in the scientific sector many simulation software have been developed. Simulation is very useful in order to eliminate any problems or software errors before the actual implementation. There are many simulation software but they do not support all the Lejos projects. Let's see two that are compatible with Lejos. Firstly the LMS - Lego Mindstorms Simulator, this software is created in Java for the simulation of Lejos programs by using 3D models and is free. In this software all are configurable by the use of XML files for example the control of the robot, the scene, lights, the platforms etc. The software is consisted of the simulation engine and the broker. The simulation engine actually shows and coordinates the simulation and the broker coordinates the controllers. This software is very suitable for the simulation of Lejos programs like the project of this thesis. It was created by the research group Didactics of Informatics at the University of Paderborn. (Universität Paderborn,2007)

Another great choice for simulation is the Microsoft Robotics Developer Studio 4. This software is also free, it is NET based for the simulation of robotic applications generally but it supports LEGO NXT as well. It can implement many scenarios or it allows the users to drag and drop onto a canvas.(Microsoft,2014)

6.2 Conclusions

To conclude the primary objective of this thesis was the creation of a system that can deliver safely a medication in a clinic or hospital sector. This goal was achieved basically with the use of LEGO MINDSTORMS NXT 1.0, and the use of the FTA method which ensures the safe delivery. In the second chapter was given all the tools that helped this project from the analysis of the NXT parts to the software that can be used. Also in this chapter all the capabilities of the robot and the sensors were discussed. Furthermore, all the tools that were used as the eclipse software and the programming method called Subsumption architecture explained thoughtfully. Lastly a detailed analysis was made for the managing risk method FTA, which is very useful to minimize or eliminate the system's possible failures. As we can see the NXT as prototype was ideal for the project, because in this theoretically stage we don't need to deal with the primary construction of a robot, our goal was to implement a system that can deliver a medicine to a patient without any concern of fault because in the health departments no mistakes are allowed; for the outcome may be a human loss and this is unacceptable. If this project in future may be implemented then a different prototype more cheap and customized would be chosen, but this is a different scope.

The benefits of this project are many; also this kind of autonomous system of delivering medicines is very unique and is an innovation in the health sector. First of all, as we saw the robot can deliver in this thesis the medicine to a patient when the nurse loads it with the treatment and starts it. So the nurse does not lose any productive time of her day and can deal with other things beyond the delivery process. She can make only a final check on the patient if needed. This valuable time is precious in overcrowded, understaffed hospitals. In addition as we discussed in the future development the robot can send valuable data to nurses phones such as oxygen levels or patients temperature. Another great innovation was the use of the Fault tree diagrams in order to ensure that no mistaken would take place. For this reason in this project we set many safety nets in order for the robot not to mix up the bed or the patient and deliver a treatment to a wrong person. The use of Bluetooth and RFID protocols are very helpful in this procedure. Also a very important implementation was the use of two RFID tags one in the bed and one in patients arm. So, the robot in order to deliver the treatment must validate the codes of the two tags, so practically it is impossible for the robot to make any fault.

Moreover there are many designs for LEGO robots in order to create a prototype, and someone can use one to make tests before the creation of customized robot in the production. Another great advantage in this thesis was the use of the programing technique with behaviors using the Subsumption Architecture on Lejos Platform.

Firstly, the Java language has the benefit of compiling which gives faster performance and safe operation. Secondly, the best advantage of behavior programming is that the code is very easy to be understood. Additionally behavior based programming allows to add or remove behaviors without to deal with the rest of the code. So, for example someone can get this code of this thesis and enrich it easily, this makes this project very interesting because it always can become better. There are also some generalized behaviors that can be used in all projects and can be used without the need to be written again like navigation or pilot behaviors. The downside is that the more behaviors added the more likely is the behaviors interfere with one another, so it need much careful.(Bangall,2013)

Another downside is that the behavior programming is based on the data received from motors and sensors, so if we want more behaviors, we must have more sensors and motors. This was experienced also in this thesis as we were restricted of the use of extra sensors and motors as a result and behaviors due to theoretical extra cost or the limits of the sensor and motor inputs in the brick. To sum up the possibilities in robotics are endless, we can create anything to serve our causes, and a very unique tool in order to test or to derive useful ideas or inspiration, because is wide well used from researchers and others as a prototype, and a large community supports it, is the Lego NXT MINDSTORMS kit.

Chapter 7

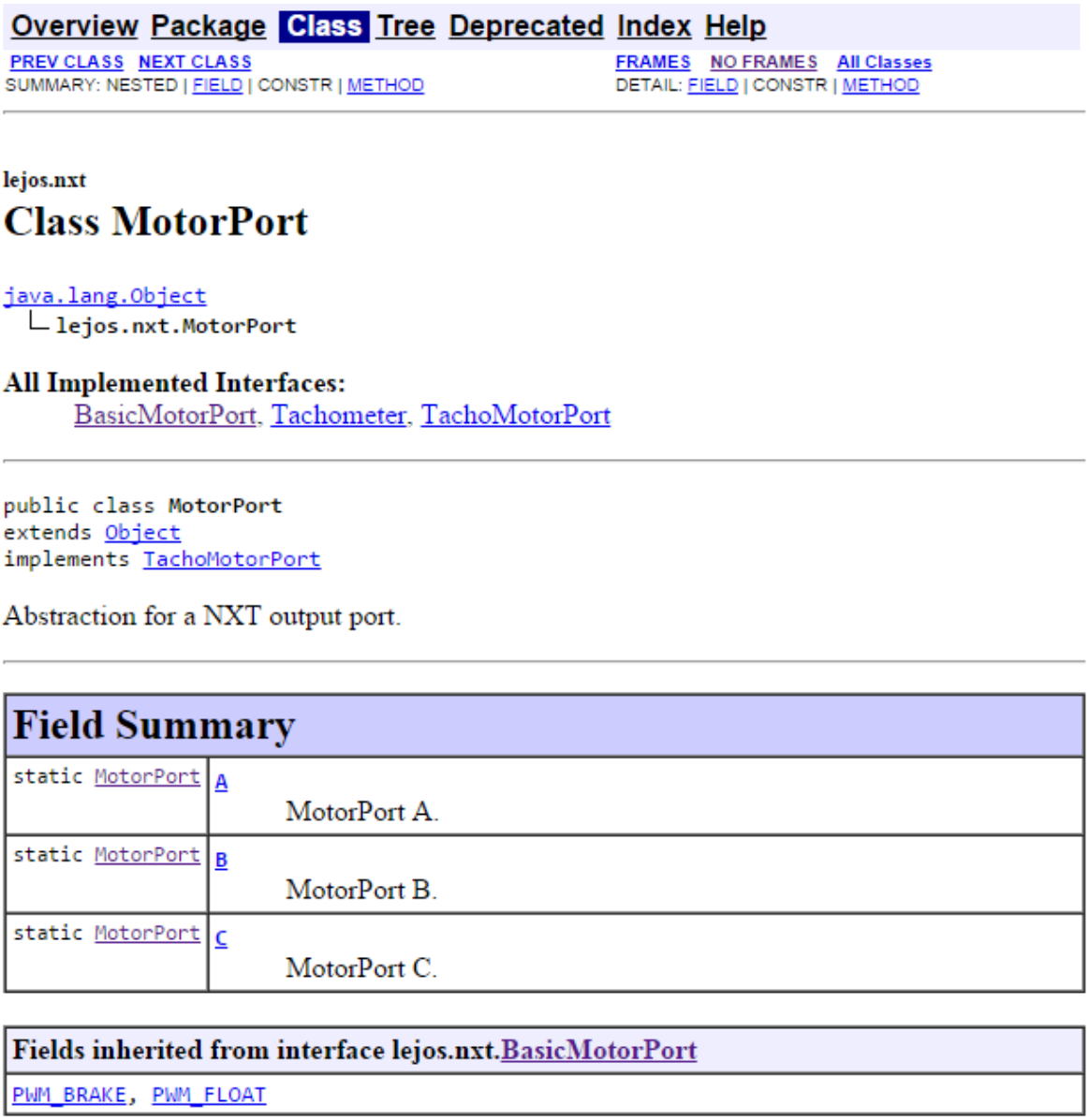
References

1) Aethon Inc.(2014).TUG ROBOT.Retrieved from http://www.aethon.com/tug/benefits/
2) Bangall,B.(2013).Maximum Lego NXT: Building robots with Java brains.Manitoba,Canada:Variant press.
3) Bangall,B.(2011).Intelligence unleashed: Creating LEGO NXT robots with Java.Manitoba,Canada:Variant press.
4) Barnes,D.,Kolling,M.(2011).Objects First with Java: A Practical Introduction Using BlueJ(5th ed.).Jefferson City.,USA:Pearson Education, Inc(Barnes,& Kolling,2011)
5) BBC.(2014).Stafford Hospital timeline in profile 2014. Retrieved from http://www.bbc.com/news/uk-england-stoke-staffordshire-20965469, 20-09-2014 Retrieved.
6) Champandard,A.(2003).AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors.USA:New Riders Publishing
7) Danaher,B.(n.d).Using Fault trees and Event trees to manage risk.Retrieved from http://www.qrc.org.au/conference/_dbase_upl/1995_spk019_Danaher.pdf
8) Finkenzeller,K.(2010).RFID Handbook:Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication.Wiltshire,UK:John Wiley & Sons
9) Henke,C.(2008).System Architecture and Risk Management for Autonomous Railway Convoys. doi:10.1109/SYSTEMS.2008.4518986
10) Howat,C.(2008).Lecture: Three Class Periods[PDF][Lecture notes] Kansas,USA:University of Kansas
11) IEEE Spectrum.(2014). Panasonic Revives Hospital Delivery Robot.Retrieved from http://spectrum.ieee.org/automaton/robotics/medical-robots/panasonic-hospital-delivery-robot
12) Joseph,C.,& Schad,C.,& Fullon,S.(2014).10-BehaviorBasedProgrammingJava in profile 2014.Retrieved from http://www.learningace.com/doc/2685237/90138d40b3c249f5f235f05a52e9ce25/10-behaviorbasedprogrammingjava
13) Lejos Java for LEGO Mindstorms.(2014).Java for LEGO Mindstorms in profile 2014.Retrieved from http://www.lejos.org/nxt/nxj/api/index.html
14) lejosgroup02 Lejos Project with NXT Programming.(2009).Prj31 in profile 2009.Retrieved from

https://code.google.com/p/lejosgroup02/source/browse/lejos/CarrierBlue/nl/fontys/prj31/?r=26#prj31%2Faction,16-08-2014 Retrieved.
15) Lejos Java for LEGO Mindstorms.(2014).Java for LEGO Mindstorms in profile 2014.Retrieved from https://www.lejos.org/nxt/nxj/tutorial/index.htm
16) Lejos Java for LEGO Mindstorms.(2014).Java for LEGO Mindstorms in profile 2014.Retrieved from http://www.lejos.org/nxt/pc/api/index.html
17) Lejos Java for LEGO Mindstorms.(2014).Java for LEGO Mindstorms in profile 2014.Retrieved from http://www.lejos.org/p_technologies/nxt/nxj/api/lejos/nxt/MotorPort.html
18) Microsoft.(2014).Microsoft Robotics Developer Studio 4 in profile 2014. Retrieved from http://www.microsoft.com/en-us/download/details.aspx?id=29081
19) Moral,M.(2009).Develop LeJos Programs Step by Step.Retrieved from http://www.juanantonio.info/lejos-ebook/
20) NASA HQ.(2013).Fault Tree Analysis (FTA):Concepts and Applications. Washington,USA:Bill Vesely.
21) Nxtprograms.com.(2014).Fun Projects for your LEGO MINDSTORMS NXT 2014. Retrieved from http://nxtprograms.com/forklift/index.html
22) Russel,S.,& Norvig,P.(1995).Artificial Intelligence:A Modern Approach.New Jersey,USA:Prentice-Hall.
23) RMS Publishing.(2011).A Study Book for the NEBOSH National Diploma: Unit A - Managing Health and Safety.Stourbridge,Great Britain:RMS Publishing Limited
24) Universität Paderborn.(2007).LMS - Lego Mindstorms Simulator in profile 2007. Retrieved from http://ddi.uni-paderborn.de/en/software/lego-mindstorms-simulator.html
25) Victor,E.,Wagner,A.,Kemp,C.(2012). A Robotic System for Autonomous Medication and Water Delivery.Retrieved from http://hdl.handle.net/1853/45009
26) Wikipedia.(2014). Stafford Hospital scandal. Retrieved from http://en.wikipedia.org/wiki/Stafford_Hospital_scandal
27) Wooldridge,M.(2012).Lecture 2:Threads & Behaviours[PDF][Lecture notes].Liverpool,UK:University of Liverpool.Retrieved from http://www.cs.ox.ac.uk/people/michael.wooldridge/teaching/robot
28) Διακονικολάου,Γ.,Αγιακάτσικα,Α.,& Μπούρας,Η.(2007).Επιχειρησιακή Δικτύωση.Αθήνα,Ελλάδα:Κλειδάριθμος.

Appendix A – Lejos API

This appendix gives some packages, classes and methods that were used for the implementation of medicine delivery robot code.



Overview Package Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

lejos.nxt

Class MotorPort

java.lang.Object
  └ lejos.nxt.MotorPort

All Implemented Interfaces:
BasicMotorPort, Tachometer, TachoMotorPort

public class MotorPort
extends Object
implements TachoMotorPort

Abstraction for a NXT output port.

Field Summary	
static MotorPort	A MotorPort A.
static MotorPort	B MotorPort B.
static MotorPort	C MotorPort C.

Fields inherited from interface lejos.nxt.BasicMotorPort
PWM_BRAKE, PWM_FLOAT

(Java for LEGO Mindstorms,2014)

Method Summary

void	controlMotor(int power, int mode) Low-level method to control a motor.
int	getTachoCount() returns tachometer count
static int	getTachoCountById(int aMotor)
void	resetTachoCount() resets the tachometer count to 0;
static void	resetTachoCountById(int aMotor)
void	setPWMMode(int mode)

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait

(Java for LEGO Mindstorms,2014)

Field Detail

A

```
public static final MotorPort A
```

MotorPort A.

B

```
public static final MotorPort B
```

MotorPort B.

C

```
public static final MotorPort C
```

MotorPort C.

(Java for LEGO Mindstorms,2014)

Method Detail

controlMotor

```
public void controlMotor(int power,
                         int mode)
```

Low-level method to control a motor.

Specified by:

controlMotor in interface BasicMotorPort

Parameters:

power - power from 0-100
mode - 1=forward, 2=backward, 3=stop, 4=float

---

getTachoCount

```
public int getTachoCount()
```

returns tachometer count

Specified by:

getTachoCount in interface Tachometer

---

getTachoCountById

```
public static int getTachoCountById(int aMotor)
```

---

resetTachoCount

```
public void resetTachoCount()
```

resets the tachometer count to 0;

Specified by:

resetTachoCount in interface Tachometer

(Java for LEGO Mindstorms,2014)

setPWMMode

```
public void setPWMMode(int mode)
```

Specified by:

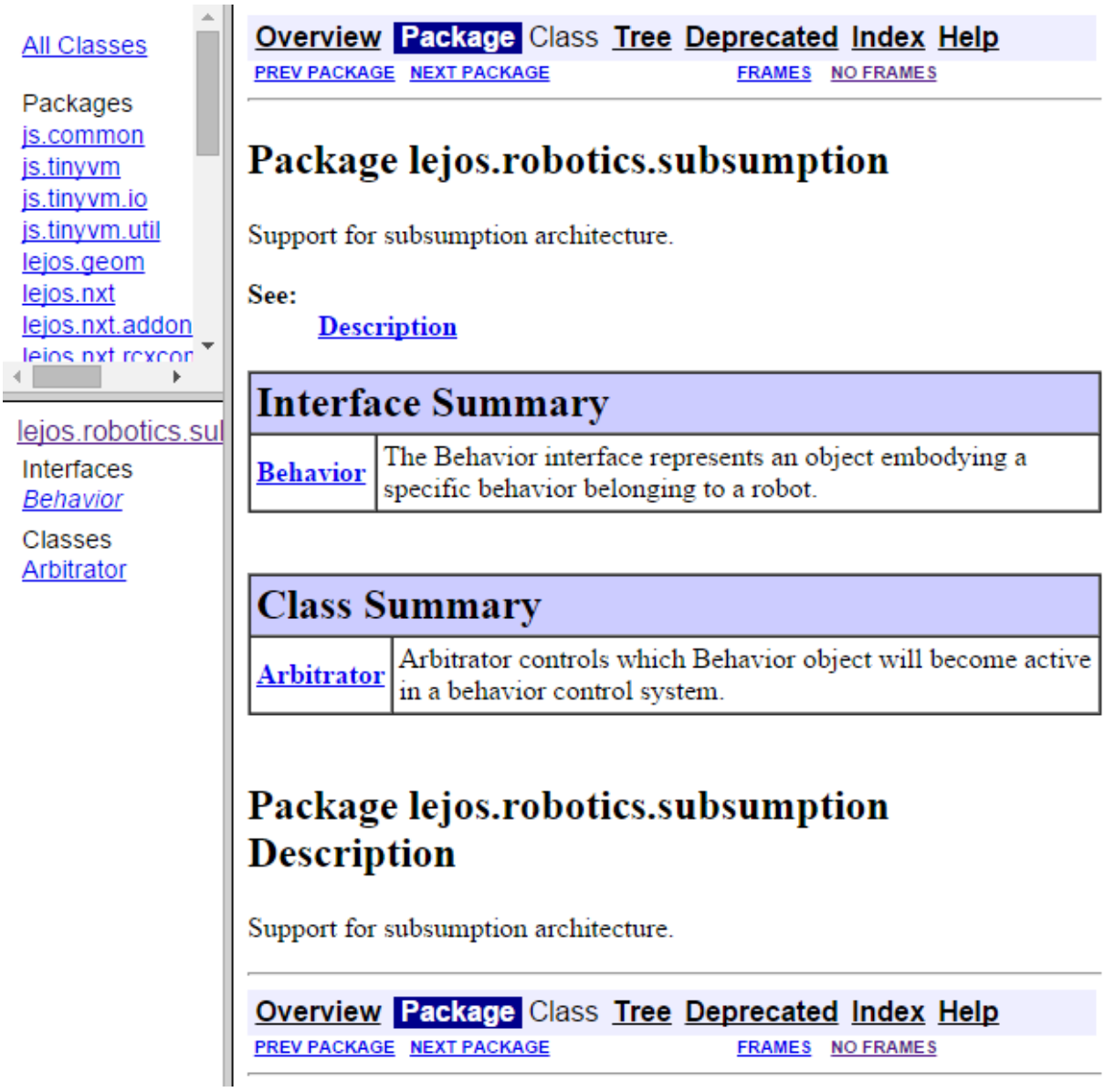
setPWMMode in interface BasicMotorPort

---

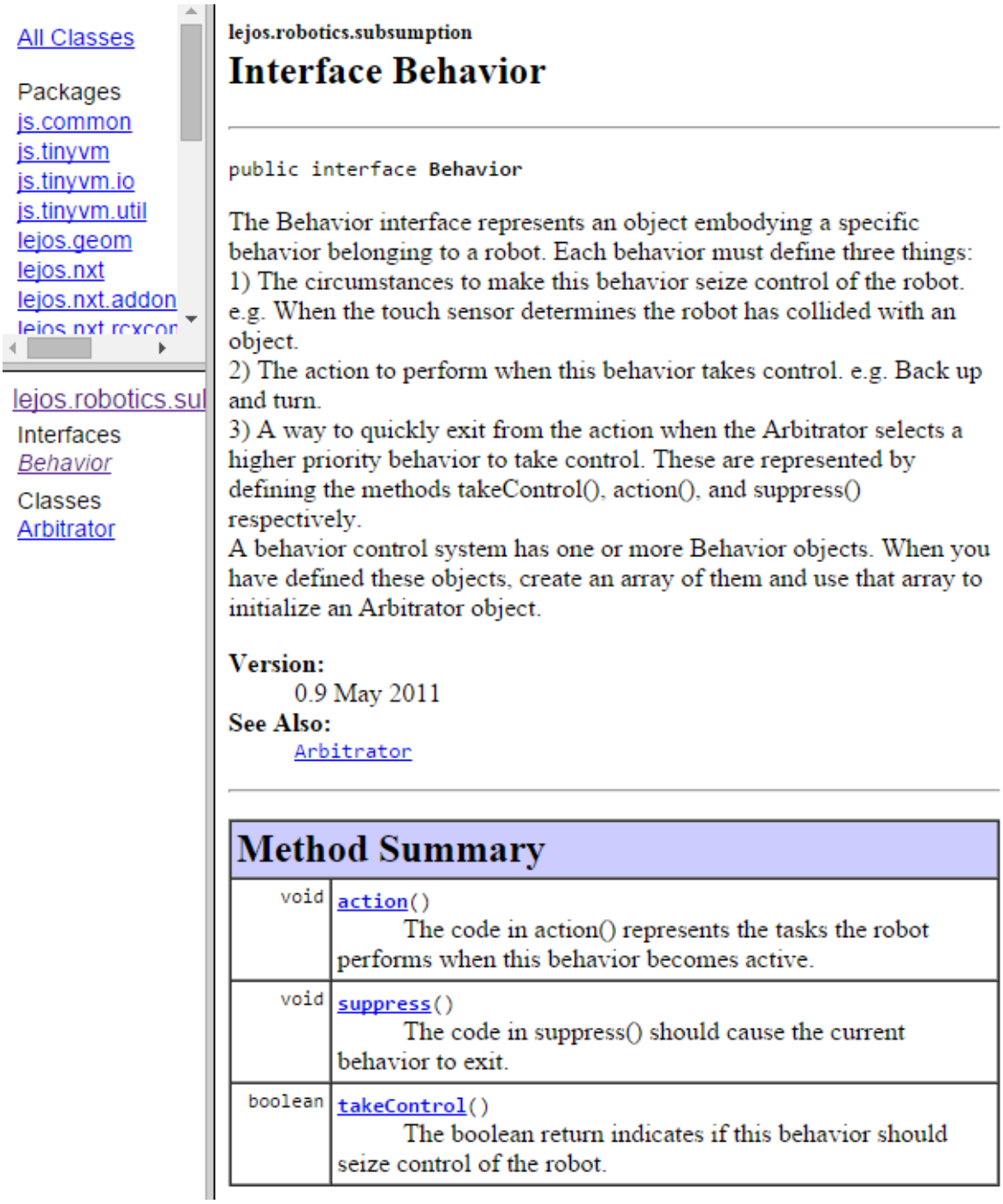
resetTachoCountById

```
public static void resetTachoCountById(int aMotor)
```

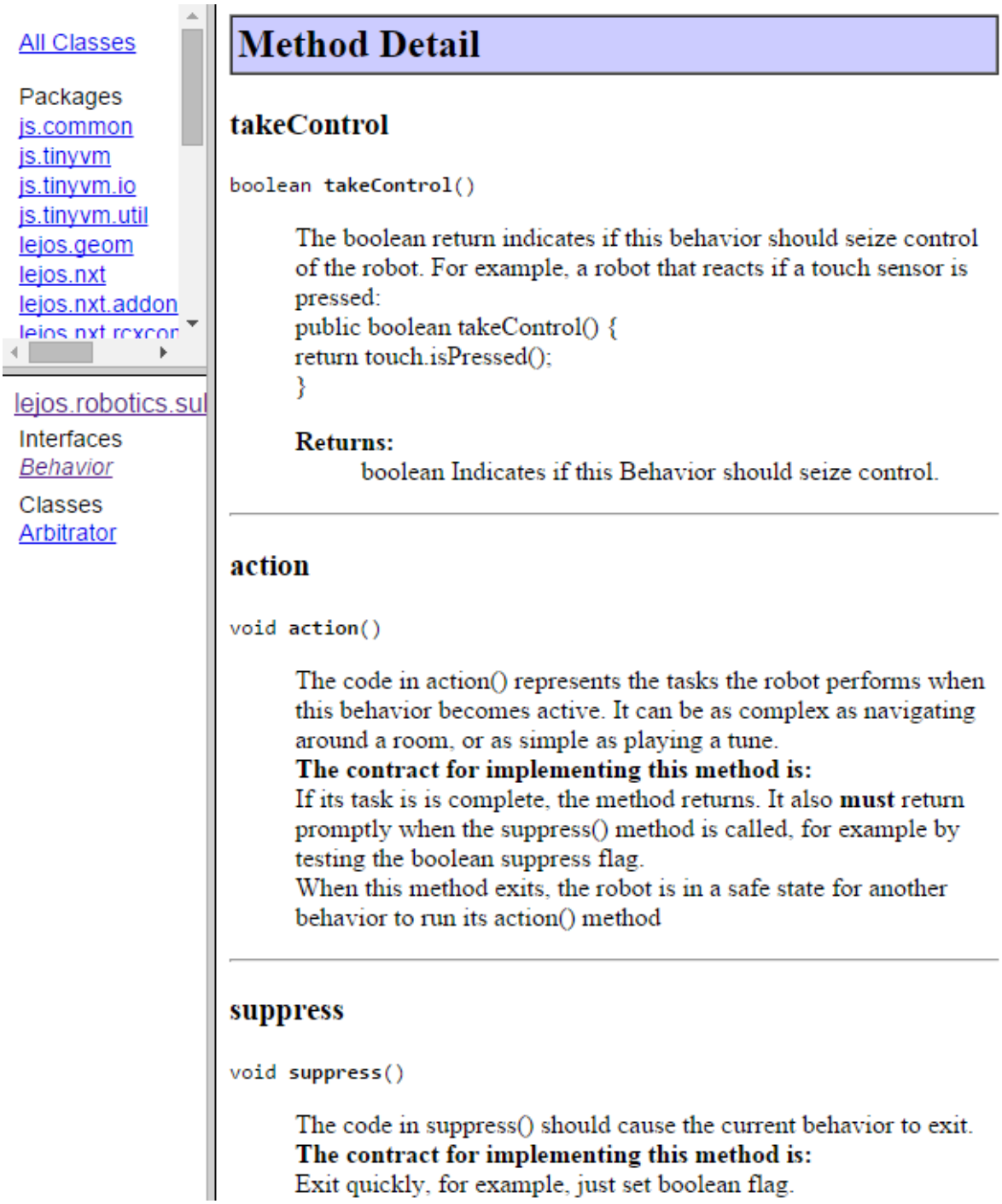
(Java for LEGO Mindstorms,2014)



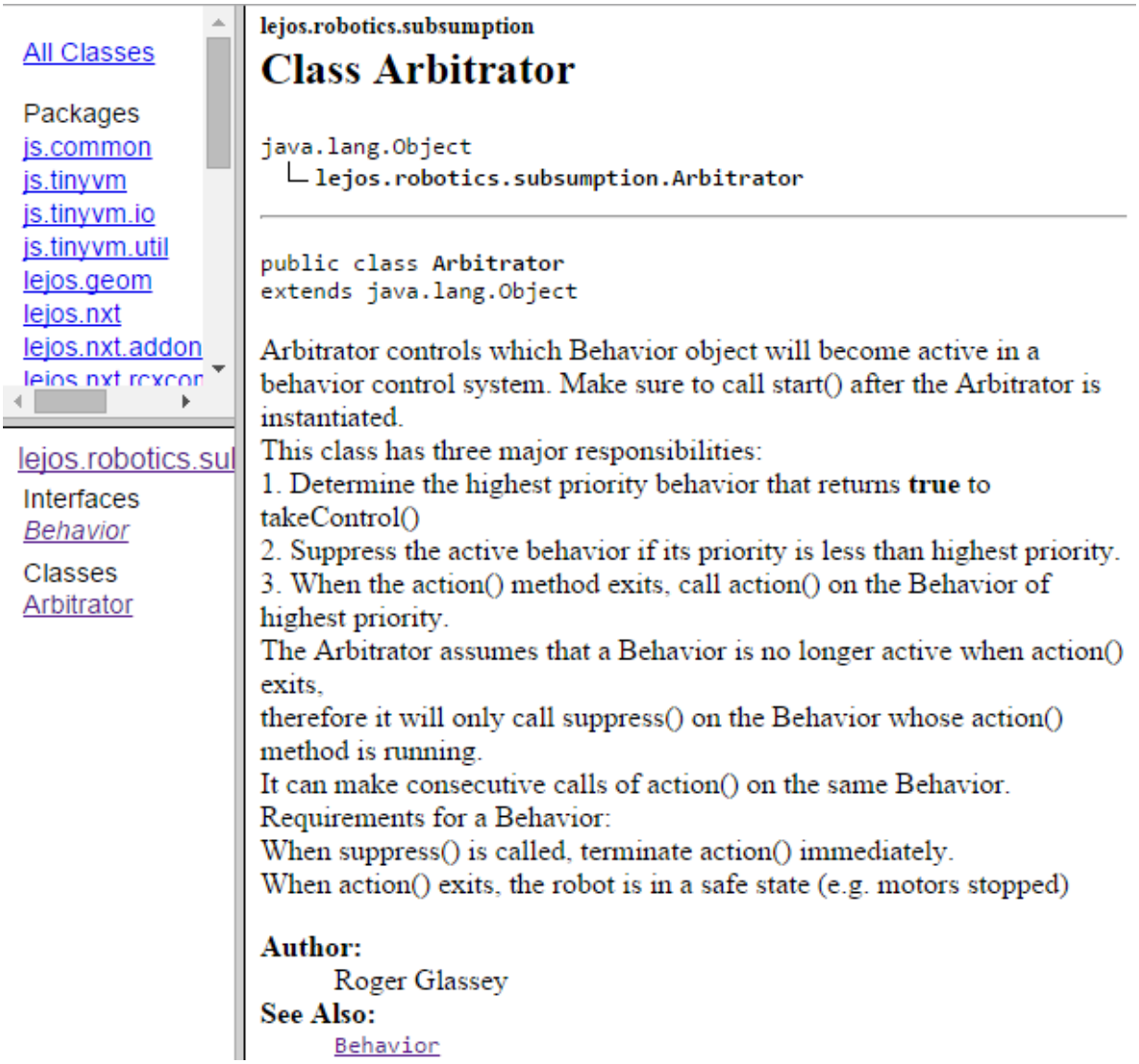
(Java for LEGO Mindstorms,2014)



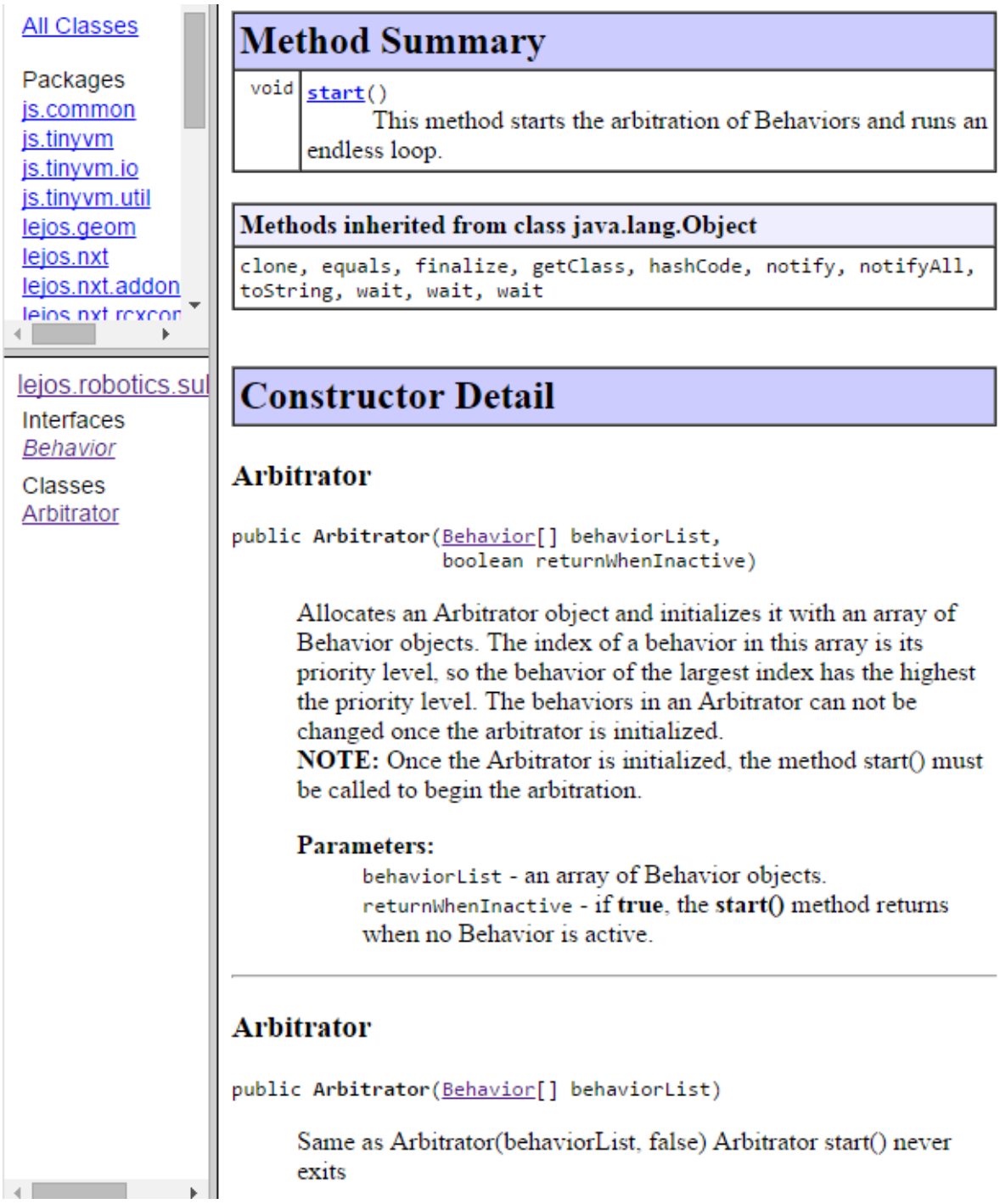
(Java for LEGO Mindstorms,2014)



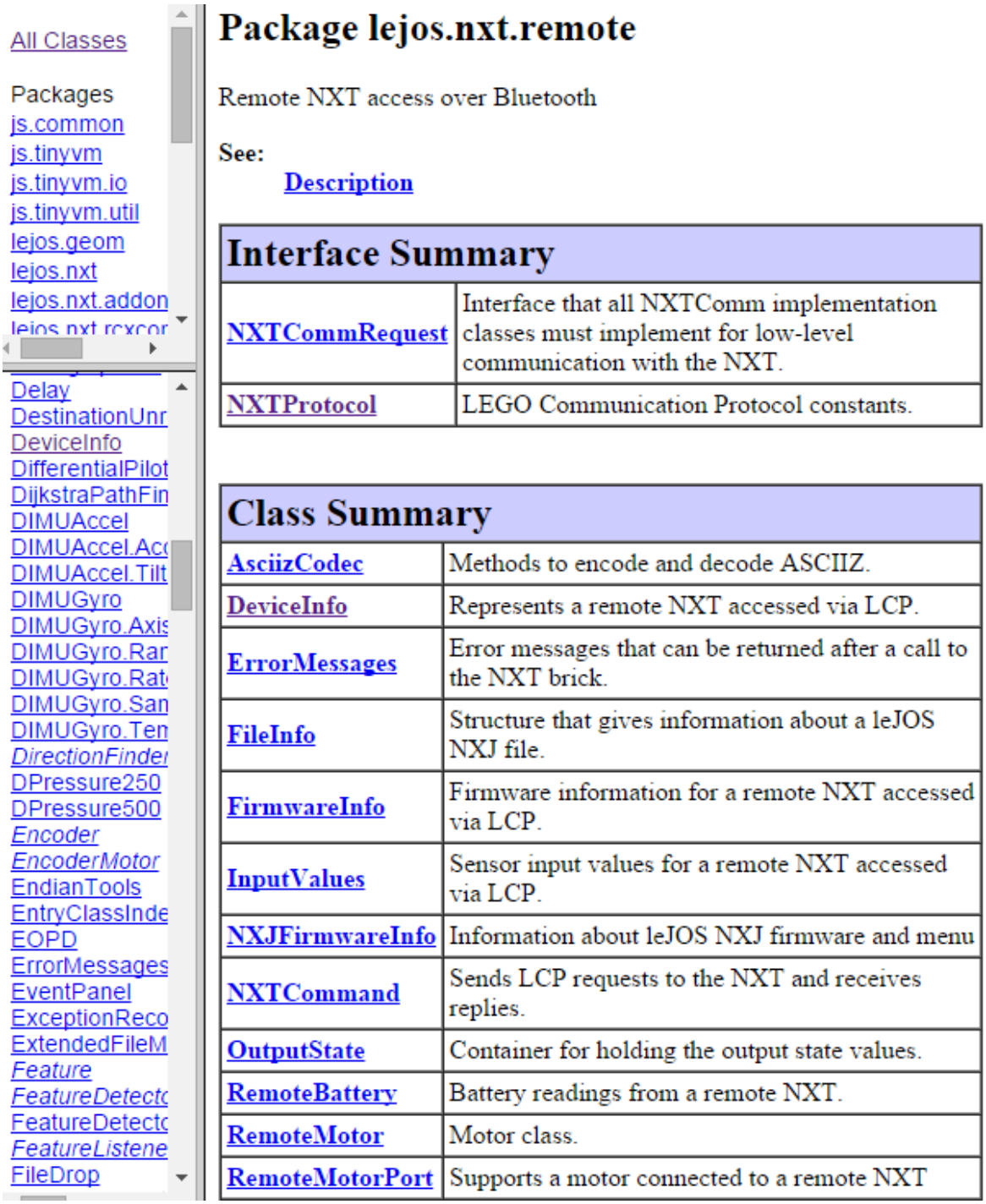
(Java for LEGO Mindstorms,2014)



(Java for LEGO Mindstorms,2014)



(Java for LEGO Mindstorms,2014)



(Java for LEGO Mindstorms,2014)

(Java for LEGO Mindstorms,2014)

(Java for LEGO Mindstorms,2014)