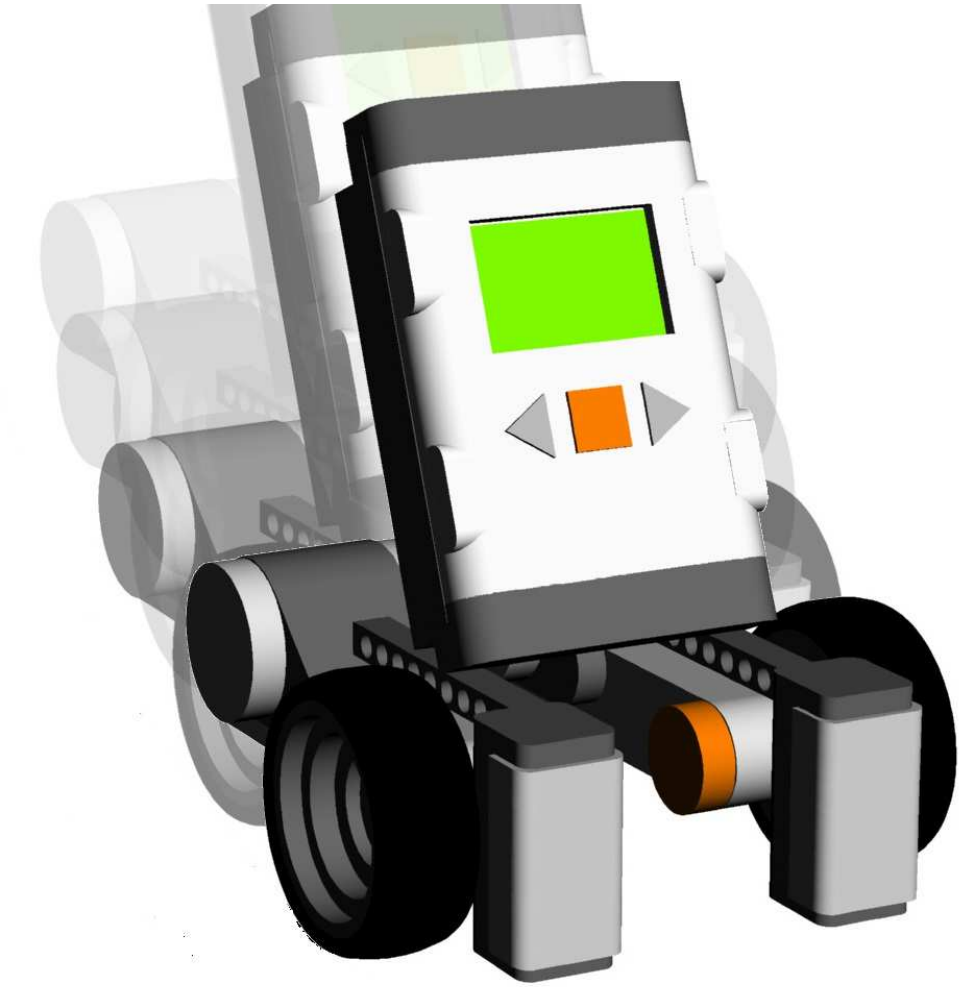
Robotics

Teachers Guide



RLT Robotics Learning Track

Module for Level 2
Version 4.3 - Sept 2012

Colophon

This Robotics module is part of the RoboDidactics Robotics Learning Track (RLT). The material presented in this module is based on the Dutch Robotics material developed by the author for the SLO Certified Robotics Module.

All the original and associated material for the RoboDidactics Learning Track may be downloaded from the Phyrtual site or from the RoboPal for NXT site at www.virtualbreadboard.com. It can also be found in the English download section of www.robocupjunior.nl. Teachers are permitted to modify this material for use in their own lessons, provided these changes are reported in the colophon of the modified material. The Phyrtual site can be reached through www.phyrtual.org

This module was developed and translated by the author (Peter van Lith) as part of a cooperation agreement with the Fondazione Mondo Digitale in Rome, Italy in 2010. The RoboPAL software used in this version has been developed with VirtualBreadBoard by James Caska.

This version is developed for use with the Lego MindStorms NXT and RoboPAL software. A more extensive version is available (currently only in Dutch). It is based on robots that can be programmed in Java, using the Java Simulator and Eclipse.

This NXT version is easy to use because it uses a graphical programming language.

The module consists of three parts. The first is the basic version needed for all further lessons. The second part deals with the RoboCup Junior Rescue challenge. The more demanding third part is optional and deals with a simple simulated organism, based on reactive behaviour.

Should, however, any person or organization deem to have rights to parts of the text or illustrations, they are advised to contact RoboCup Junior Netherlands (info@robocupjunior.nl).

This module has been compiled with care and has been tested extensively by the authors and several test schools. The authors accept no responsibility for incorrect or incomplete parts of this module, nor do they accept any claims for damages as a result of using this module or its associated software.

Contents

Introduction: RLT Module for NXT

This version of the RLT Robotics Module is not a substitute, but a self-contained integration to the existing Java version. It differs from the Java version as follows:

- It is based on the RoboPAL graphical programming language, which makes the Module much simpler to use and better suited to students in lower grades.
- It uses the Lego NXT robot that allows schools to pursue other projects, besides the RLT Modules.
- The number of lessons has been compressed to ten and subdivided into three parts that may be followed independently.

The NXT version of the RLT Robotics Module is based on the RoboPAL graphical programming language. Its content is identical to the Java version and all the examples and assignments are the same, but the programming language is easier to use. While reviewing the original material, the twelve Java lessons proved to be too much to treat in a single year. Therefore, the lessons have been reduced to ten and organized into three separate parts.

An important part of the Module is the use of a simulator. Students can use a simulator to test their programs on a PC without the need for a real robot. This allows students to work independently and also use the system to work on their projects at home. Without a simulator, each student would need a robot, making the robotics course far more expensive. Thanks to the simulator, the number of robots can be reduced to 4-5 per classroom.

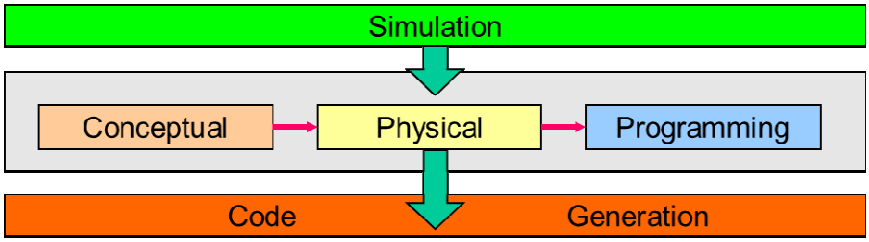
The training material in this Module is extensive and students often complain about the amount of text they have to read. Several users have further reduced the material. There also are many optional exercises that experienced students may safely skip. In response to some teacher remarks, we have also developed a much simpler version of parts of this material for students. It is important, however, to understand that some of the more complex subjects are not covered by the shorter version. If this creates subsequent problems with understanding the material, the best way is to refer the students to the extended version. Also, please note that the official certification of the material does not apply to the simplified version.

Commento [PvL1]: Certification only applies to the Dutch version, so maybe leave this out in this document.

Leaving out information to meet the decreased willingness of students to read books or follow lessons will result in more questions on how to solve certain problems. Moreover, many students who would like to pursue a scientific education will have to read and write scientific articles. Using simpler material and avoiding technical literature is not a very adequate way of preparing for such an education.

Learning Track

The RLT Robotics Module is part of a Learning Track. The students are introduced to programming simple robots on three different levels.



The use of a simulator is a fundamental to this approach.

The Learning Track is based on three levels:

The **conceptual** level focuses primarily on the basic educational level. At this level, students concentrate mostly on problems. They will be presented with partial solutions (from the programming environment) that need to be assembled correctly. Students will also have to learn to set up the necessary variables in the right manner. This will make them feel comfortable with programming a robot in a very short amount of time and they will learn to understand what is important in the practice of robotics. The conceptual level is not treated in the RLT Robotics Module. Separate educational material is available for this level and ,ay be downloaded from the www.phyrtual.org site.

At the **Physical** level, students learn how to use sensors and motors and gain a deeper understanding of how to program robots. The RLT Robotics Module for the NXT focuses on this second level. Students have to discover much more on their own and execute assignments, which will allow them to gain insight into how programs are developed. Here, they will have to create the solutions that were given to them at the conceptual level. The other parts of the Learning Track (at this level) are also available in RoboPAL, but are not part of the RLT Robotics Module. Separate educational material is available for further topics such as Dance and Soccer.

The **Programming Level** is the most advanced and difficult level of the Learning Track. Here, students will have to solve the same problems, but have to do so with the Java programming language. Java is widely used

both in industry and on the Internet. The existing Java version of the robotics Module was developed for this level, so it is more challenging.

The RLT Robotics Module is an integral part of this Learning Track and completes the physical (NXT) and the programming (Java) levels. The complete Learning Track is described in the schematic overview provided below. All parts of the Learning Track are available for RoboPAL, but not all of them are part of the RLT Module. Educational material for the other parts is available separately.

The current Java version of the RLT Robotics Module is meant for Level 3 (*programming level*), while the NXT version is meant for Level 2 (*physical level*).

Level 1 (*conceptual level*) is not a part of the Robotics Module, but can be used independently. Moreover, both the Java and the NXT versions can be used independently. We now will describe the physical level of the NXT version for RLT in great detail.

RoboPAL

RoboPAL (Play And Learn) is a complete development environment with a simple and easy graphical programming language. In RoboPAL, a program consists of icons that can be controlled by setting parameters to change their behaviour. The programs that students create gradually become more complex and reach the highest level of difficulty at the third level. The NXT version is based on Level 2. The structure of a RoboPAL program is identical to that of a Java program, which makes the NXT Module a good introduction for programming in Java.

Simulator

The use of a simulator is one of the most important aspects of the Robotics Module. Not only is it a frequently used technology in almost all robotics projects, but it also helps students to quickly assimilate the educational material.

In the daily practice of robotics, simulators are everywhere these days. First of all, they make testing new and important functions much quicker. A robot is subject to wear and tear, but this becomes less of an issue when using a simulator. Secondly, using a simulator is much quicker because you do not have to reprogram the robot for every test. In the classroom, this is a big advantage as all the lessons can be developed and tested on a PC. In

fact, a robot is only occasionally necessary. Moreover, students can also develop assignments at home on their own PCs.
The advantage of RoboPAL is its simple programming environment with an integrated simulator; indeed, the entire concept of a learning track is integrated into the system.

The program is developed with RoboPAL and then tested on the built-in simulator. If the program works correctly, it can be transferred to the memory of the NXT via a wireless connection and the special RoboPAL dongle. If necessary, a standard USB cable can be used as an emergency option. There also is a special Server version for classroom use in which a central computer takes care of connections with all the robots.

A program is developed with the RoboPAL development environment.

Programs can be run on the simulator and followed step-by-step.

The special RoboPAL dongle uploads programs to the NXT via Bluetooth.

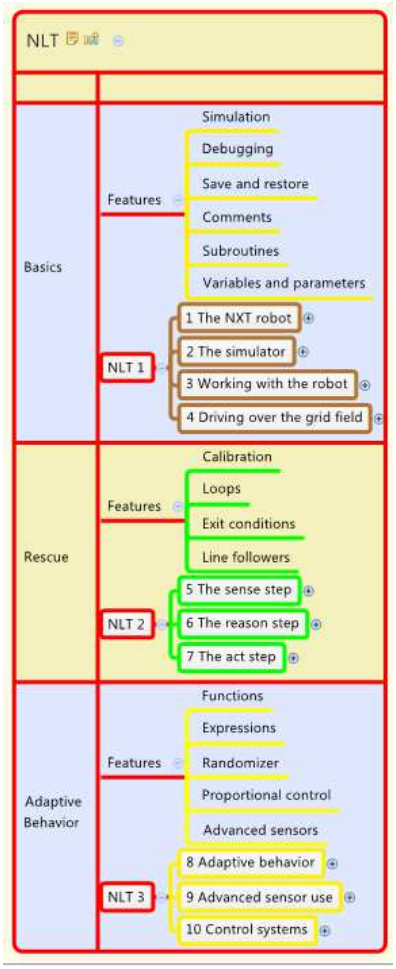
Lego NXT

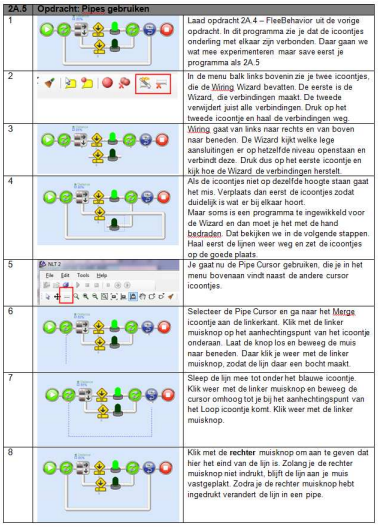
The Java version of the RLT Module uses the JoBot Nano robot. This small robot is based on an industry-standard microcontroller made by MicroChip. It is very technical and suited for modification and extensions. Students in the higher classes can construct these robots themselves. In fact, this is one of the reasons that many schools select the RLT Robotics Module.

However, programming in Java is not a simple task. Students who are interested in learning about robotics at a less advanced level can now do so, thanks to the new NXT version. This is especially suited not only to younger students, but also to older students with a less theoretically oriented education.

Using a simpler programming language and a standard Lego NXT robot not only makes programming simpler, but also allows the robot to be used for other projects. The Lego NXT is a flexible robotics construction set that can be programmed with a large range of programming languages. Moreover, a large variety of sensors and other equipment is available. This makes it possible for students, after following the RLT robotics course, to work on larger projects independently.

The RLT Module for Lego NXT





The RLT Module is subdivided into three parts that can also be followed independently:

- The *basics* are laid out in the first part: four lessons on the robot, the simulator, programming and the development of the student's first program. This is also a good introduction for teachers who are interested in participating in the RoboCupJunior competitions.
- The *Rescue* mission includes three chapters that explain how to use the Sense-Reason-Act Loop to program a Rescue mission.
- *Adaptive Behaviour*. The final three chapters concentrate on programming simple, insect-like behaviour employing techniques from the field of Artificial Intelligence.

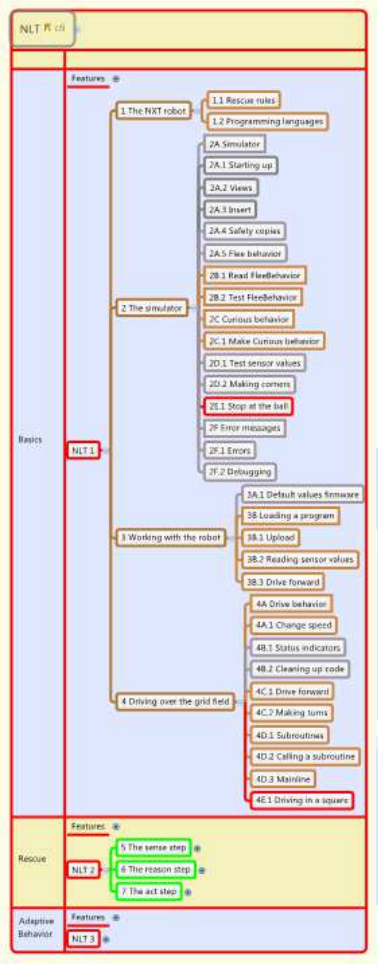
Each lesson treats a number of important concepts (see the *Features* in the diagram on the left). The Module may be followed as desired. The Basics part is required in all cases. Subsequently, students may continue with the Rescue Mission and then the more advanced part on Adaptive Behaviour. Another option is to move directly from the Basics to the part on Adaptive Behaviour.

The Rescue mission is about finding and removing a dangerous container from a swamp, while Adaptive Behaviour concentrates on allowing the robot to adapt its behaviour to changing circumstances.

In addition to this material, follow-up material is also available to introduce students to the programming of Service Robots. This part, called Robots@Home, is available separately for students who have completed the RLT Module lessons.

Basic Module

The Basics Module provides students with all the information necessary to develop programs, test them and run them on a NXT robot.

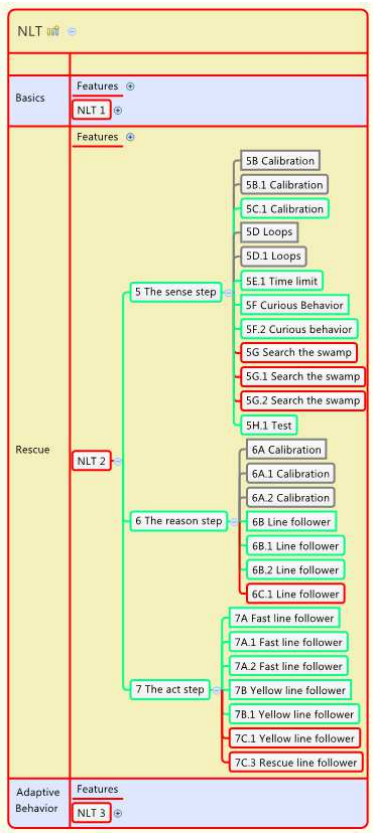


Each chapter includes an explanation and a series of steps that show students how to develop a program. A number of optional detail assignments are also included to make sure that students have a good understanding of the material. These may be skipped or followed, if one of the main assignments proves too difficult.
The optional assignments are shown on the left in the grey boxes. Each chapter ends with a do-it-yourself assignment in which students apply their newly learned skills. The mandatory assignments are boxed in red.
Moreover, there is a strong focus on saving programs frequently and using variables, subroutines and the debugger.

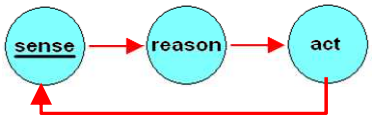
At this level, the assignments are very simple and treat elementary programming concepts. We begin with simple programming and move on to the use of loops and subroutines, variables and parameters.

The programs at this level are relatively easy to follow, and they teach students to develop simple programs. Often, it seems easier to be presented with a subject than having to come up with one. This is what happens in the remaining two parts.
In Part II, the Rescue Mission, students learn how a robot can locate a dangerous container. This is a fun assignment that can also be used by students who want to participate in national championships or RoboCup Junior. These competitions are organized annually in over 26 countries.

Making Your Own Programs (Rescue)

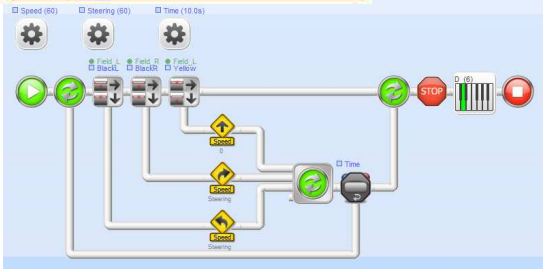


In Part II, the Sense-Reason-Act Loop is introduced.



This construction is used in almost all robot programs. It allows robots to use sensors to follow a line or find an object. Each of the three chapters treats one of these three steps.

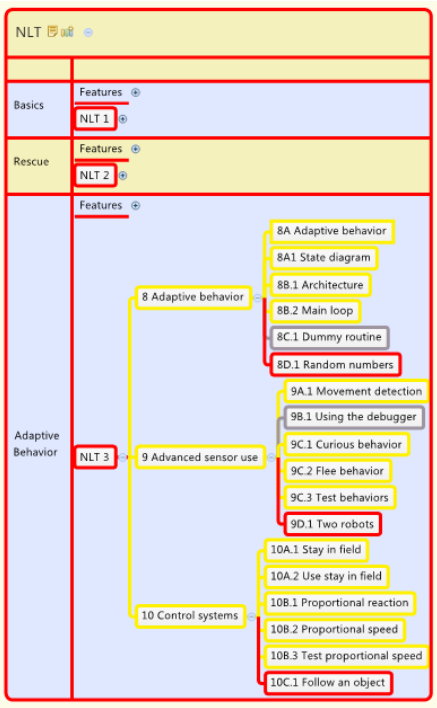
The second part focuses on the Rescue mission. The robot must follow a black line until it reaches a yellow swamp. There it must search for a can and push it out of the swamp. The can represents a container with a dangerous substance that may explode if it comes into contact with water. So, the container must be pushed out of the swamp as quickly as possible.



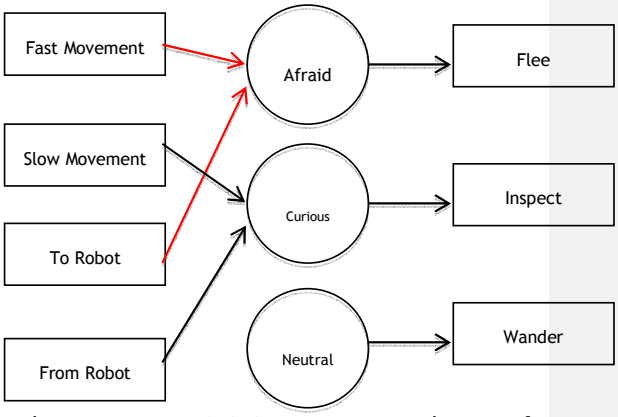
Students learn how to program a line-follower and calibrate sensors. The challenge is to let the robot follow the track as quickly as possible without losing it.
The programs in this part are more difficult than in the first part. They employ loops, subroutines and parameters to influence the behaviour of a program. Once again, it is important to identify the correct settings and correctly calibrate the sensors. The teacher may decide whether to follow the third part or skip it.

Students that like more challenging problems may skip the second part and move directly to the more advanced third part.

Adaptive Behaviour

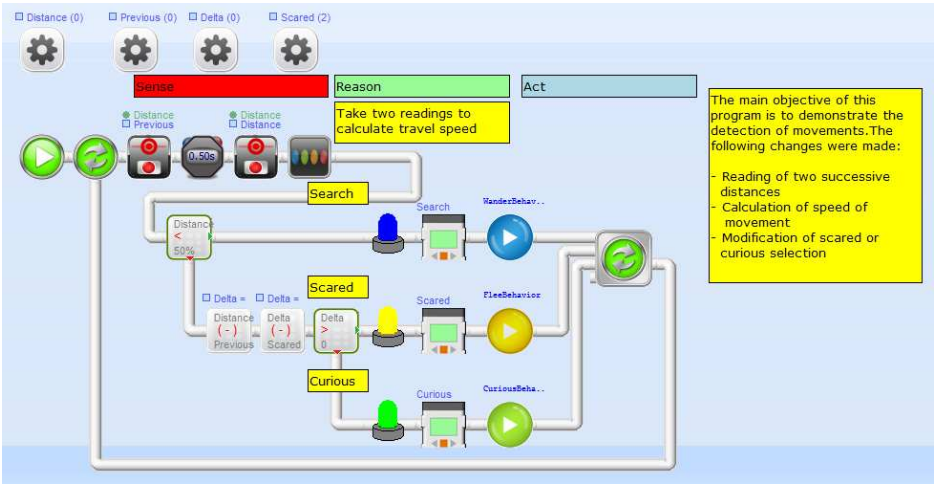


In Part III, students learn about how a robot can adapt to changing circumstances.



In these programs, it is important to make use of a State Diagram, which defines the states that a robot can adopt along with their associated actions. This is a further refinement of the Sense-Reason-Act loop.

A state-diagram shows which external stimuli lead the robot to a certain state and which actions it will then undertake (behaviour).



The programs in this part are both more complex and more theoretically oriented than in the previous parts.

Software and Documentation Download

All the software and documentation for the RLT Robotics Module is available on CD and from the download section of the RoboDidactics website (sections mentioned below).

- The RLT educational material can be found in the RoboDidactics section of the www.phyrtual.org site. You can also find the RoboPAL software at www.virtualbreadboard.com
- Learning Track: includes all other educational material for Levels I and II.

All information on the RLT Robotics Module for the NXT is also available on CD. For additional information on equipment, prices, etc., please check www.virtualbreadboard.com

The Teacher's Guide

This Teacher's Guide contains an overview of all the assignments given in the RLT Robotics Course and provides further background that may be useful.

The Module has been set up so that students can work independently and test their programs on a simulator without the constant need for a physical robot. So, while using a robot is not necessary, it does give the course an important extra dimension. Reality is always different from any simulation. The experience of making a real robot move is completely different and is far more rewarding.

Students can also install and use the software for the RLT Module at home. If you provide them with the CD for this purpose, please make sure you only give them a copy that does **NOT** contain the teacher's material, otherwise they will also have all the answers to the assignments and tests.

Learning by Discovering

The material always begins with a short explanation of the main goal of each lesson. Each lesson addresses a number of concepts that the students learn by discovering the difficult or interesting points.

So, there is relatively little explanation provided. The structure or purpose of programming languages is not explained. The students are immediately immersed in the subject matter. They learn how to program by studying and using existing code and expanding and modifying it. Gradually, they will gain more insight into how a robot works and what a programming language is.
This makes it more difficult for teachers to guide the students. There also is the possibility that a student does not 'discover' the subject of a lesson or that he/she may find out something else. It is important for teachers to realize this.

The Teacher's Task

The teacher acts as a guide and shows students around on this discovery tour of the world of robotics. The purpose is to learn by doing. This is especially hard at the beginning, because many students (and teachers) may not have any experience with programming.
The idea of gradually introducing more and more advanced code is that students will learn how a program is created while working with code. Continuous testing and gaining insight into what a program does is fundamental at this stage.

It is very important that students understand that programming is a discipline that requires a very accurate and systematic approach. A computer does exactly what - and only what - it is told. As we are used to dealing with the intentions of other human beings, we usually look at a program with the idea we have as a programmer. In doing so, however, we frequently miss the true content of a program and only see what we interpret to be the purpose of the program. Thus, we often overlook small mistakes. Nearly 95% of the time, we waste time to identify our own mistakes. This is something that students need to experience and adapt their behaviour to, by working more accurately and planning ahead.

How to Prepare as a Teacher

If you have no programming experience, it is a good idea to first follow the lessons yourself and make sure you are always at least four lessons ahead of your class. This will provide you with a first-hand experience of the problems that your students may encounter. If you do not master the lessons yourself, you will not be able to assist your students.

This can be hard, especially in the beginning, because programming requires a strict discipline and even minor mistakes can prevent a program from working as intended. Finding the cause of these problems will consume most of the time devoted to developing and testing a program. It is important that your students understand this.

If you have virtually no experience with programming or information technology, then first follow an RLT course for teachers. Several universities and educational institutes provide these courses. Often, there are one or more bright students in a school that have experience with computers and programming. Make sure that this kind of knowledge is used in your lessons, as both teachers and students can learn a lot in this way.

The Student’s Task

The student’s task is to work out the assignments step by step. The programs are not explained in great detail on purpose. If the students carefully follow all the steps in a lesson, small mistakes will occur to make things go wrong. They should not just follow the instructions. The idea is that the examples will help them understand what is going on and help them to gain insight into how a program works.

At the end of each chapter, there is a final assignment in which the students will have to apply all the knowledge they have gained in previous assignments.

Using FlowCode

Students use program code right from the beginning. This means that they have to read a FlowCode (and try to understand it). The code has to be changed and adapted frequently. The goal is for students to learn how to create an entire program. This is more likely to occur in the later lessons, after all the important concepts have been treated.

For the benefit of the teacher, all code that is relevant is supplied as tested modules on the CD. Students only receive the initial code in addition to that described in the lessons. Be careful not to accidently provide the complete code to your students.

If a student program does not work, compare it with the code that is supplied on the CD in the Teacher's version.

In the RLT Module for NXT, we use RoboPAL, a graphical programming language. This language is not only easier to use than Java, but it also provides less opportunity for making mistakes.

Using the Development Environment

As emphasised before, it is very important that students learn to work accurately. It is fundamental to always work in small steps and check that everything works correctly after each step. If too much code is created at once, it will become much more difficult to identify the cause of a problem.

Setup of Chapters

Each chapter in this Module has the same setup:

1. What you will learn
2. What you will need
3. What you will experiment with
4. What you will be able to do after completing the chapter
5. Explanations and Assignments
6. In Practice
7. Test questions

Explanation of Pictograms

The following pictograms are used in the Robotics Module. Here is what they mean:

Assignment Type	**Assignment**
	Detail assignment (may be skipped)
	Find something out
	Assignment on the Grid field
	Assignment on the Rescue field
	Assignment with RoboPAL (programming)
	Assignment with the simulator (testing)
	Assignment with the robot

What You Will Need

To follow this Robotics Module you will need the following equipment:

- At least one computer for every 2 students, but a computer for each student is preferable;
- License for the RoboPAL software in the form of a dongle;
- One NXT robot for every 4-5 students;
- One Rescue field.

Your didactical model will determine the number of robots and playing fields that you need. Do you let individual students create and demonstrate each program fragment? Do students create and test their programs independently with the robot? In the first case, you will just need a few robots; in the second case, you will need one robot for every 5 students. In the first case, you will need a single rescue field; in the second case, a field for each student. Experience shows that students are much more enthusiastic when testing their own programs, but the decision is up to you.

Contents of the Module

The table below indicates the focus of each chapter, describes the subjects and topics and lists the assignments that the students will have to complete. It also shows which subjects are treated in which lessons and how the knowledge is tested.

Halfway through the course, teachers may administer the included tests to check on student progress. At the end of the Module, a final test requires students to independently develop and demonstrate an entire program a program that conforms to a number of given criteria.

This manual contains examples and written tests that can be modified or extended as desired.

H#	Theme	Learning Assignment	Working Assignment	Examples
1	Getting to know the robot	Understanding Sensors and Processors	Finding out what a robot is	Where are robots used?
2	Getting to know the simulator	Using a simulator	Changing a program	Flight Simulator
3	How does your robot work?	What can a robot do?	Making the robot move	Bridge
4	Making the robot drive over the field	Following patterns	Following a fixed track	Welding robots
5	Sensors for the 'Sense' step	Sense-Reason-Act loop Robot behaviour vs. animal behaviour	Observing before acting	Automatic elevator doors
6	Processing for the 'Reason' step	Processing Information	Line-follower	Automatic Guided Vehicles
7	Actuators for the 'Act' step	Driving and Steering	Adaptive behaviour	Self-parking cars
8	Adaptive behaviour	Adapting to the environment	Not reacting in the same way every time	'Pick and place' robots
9	Advanced sensors	Sensory information	Calibration and object avoidance	X-rays, infrared, ultrasound
10	Control systems	Feedback in control systems	Reacting to disturbances	TV transmitters automatic programming

Most schools use the first 7 chapters; the remaining chapters can be used for Information Technology classes, for which this Robotics course is a good introduction.

The Assignments

The table below provides you with an idea of the assignments that are part of this course. It indicates where code is created in each chapter.

Type	#	Assignment	Description
	1A	TestBehavior	Designing a solution
	2A.1	CuriousBehavior	Basic settings in RoboPAL
	3B	**CuriousBehavior**	**Loading a program**
	3B.1	FleeBehavior	Testing on the simulator
	3B.2	Reading Sensors	Testing with the robot

The answers to every assignment are included. The code for all assignments is provided on the teacher's CD.

Scope of the Material

The course material consists of ten chapters in which various concepts, ranging from robotics to information technology, are addressed step by step.

Most schools use a selection of the provided material. The speed at which students absorb the material varies greatly, which makes it necessary to help students on an individual basis. This makes classical education very

difficult and puts a demand on guidance. Most students will not get any further than chapter 7. Indeed, the material in the last chapters is considered difficult by most students. It is fundamental to study the chapters on the Sense-Reason-Act Loop. The third part can be skipped, postponed to a later year or reserved for information technology courses.

Part III combines all the knowledge gained in Parts I and II and provides an overview of the way in which robots can adapt their behaviour to changing circumstances. Gaining insight into good programming techniques, frequent testing and working accurately are fundamental to developing working programs and are emphasised in each chapter.

Overview of the Lessons

Here is an overview of all the assignments that are provided in the lessons. The following chapters include all the assignments, all the answers to questions and tests and supplementary background information.

In parts I and II, the assignments are split into several categories:

a) *Main assignments* - must always be completed and demonstrated.
b) *Detail assignments* - are optional and only followed by students who find the main assignments too hard. After working out a detail assignment, a student may return to the main assignment that posed a problem. In some cases, extra instructions are provided in the detail assignments, so these assignments are not fully optional. Detail assignments are shown in **Grey**.
c) *Personal assignment* - is given at the end of every chapter and allows students to use their new knowledge independently. These assignments are shown in **Red**.
d) *Tests* - are given at the end of every chapter.
e) *Written Tests (WT)* - are included after every 2-3 chapters, but can be administered as necessary.

In part III, there are no detail assignments and the individual assignments become more important. The final assignment is a project that must be fully developed, independently, by each student.

There also are much simpler lessons available in the form of instruction cards that treat the same subjects, but in considerably less depth. They are intended for the higher classes of the elementary schools or the first classes of middle grade schools. The cards provide more instructions and there is less reading to done by the students. These simplified versions are not certified.

Part I

Type	#	Assignment	Description
	2A	Simulator	Working with the simulator
	2A.1	Simulator	Starting the Simulator
	2A.2	View	Studying the Simulator features
	2A.3	Insert	Adding parts
	2A.4	Making a Copy	Making a safety copy
	2A.5	Drawing Lines	Connecting icons with pipes
	2B	**FleeBehavior**	**Testing flee behaviour**
	2B.1	FleeBehavior	Making the robot flee
	2B.2	FleeBehavior	Studying fleebehavior with RoboPAL
	2C	**CuriousBehavior**	**Making curiousbehavior**
	2C.1	CuriousBehavior	Making the robot curious
	2D	CuriousBehavior	Reading Sensor values
	2D.1	Experiment	Experimenting with sensor values
	2D.2	Experiment	Experimenting with turns
	2E	CuriousBehavior	Making the robot stop in front of a ball
	2E.1	Personal Assignment	Stopping the robot stop in front of the ball
	2F	CuriousBehavior	Error messages
	2F.1	CuriousBehavior	Errors in your program
	2F.2	CuriousBehavior	Debugging a program

Type	#	Assignment	Description
	3A	RoboPAL	Loading the firmware
	3A.1	Basic Settings	Basic settings in RoboPAL
	3B	**CuriousBehavior**	**Loading the program**
	3B.1	Uploading	Uploading a program
	3B.2	Reading Sensors	Reading sensors
	3B.3	Testing	Making a robot move around
	3C	Test	Written test on chapters 1-3

Type	#	Assignment	Description
	4A	**DriveBehavior**	**Changing the robot's speed**
	4A.1	DriveBehavior	Adapting 'Drive' speed
	4B	DriveBehavior	Starting the program
	4B.1	Code Change	Showing the status
	4B.2	Code Change	Cleaning up the program
	4C	**DriveBehavior**	**Driving forward and stopping**
	4C.1	DriveBehavior	Making the robot drive forward
	4C.2	DriveBehavior	Making a turn
	4D	**SubRoutine**	**Using a subroutine**
	4D.1	SubRoutine	Making a subroutine
	4D.2	SubRoutine	Using the subroutine
	4D.3	SubRoutine	Making the Main program
	4E	SubRoutine	Driving in a square
	4E.1	SubRoutine	Making the robot move in a square

Part II

Type	#	Assignment	Description
	5A	**Discussion**	**How does a robot work?**
	5A.1	Group Discussion	Where are the sensors located?
	5B	CalibrateBehavior	Reading sensor values
	5B.1	Calibration	Reading sensor values using the simulator
	5B.2	Calibration	Reading sensor values with the robot
	5C	**DriveBehavior**	**Driving up to the black line**
	5C.1	DriveBehavior	Stopping on the black line
	5D	Loops	Making a loop
	5D.1	Loops	Using a loop
	5E	**DriveBehavior**	**Using a time limit**
	5E.1	DriveBehavior	Adding a time limit
	5F	CuriousBehavior	Making a graph
	5F.1	Measuring	Drawing a graph
	5F.2	CuriousBehavior	Modifying CuriousBehavior
	5G	**DriveBehavior**	**Cutting the reed in the swamp**
	5G.1	Search the Swamp	In practice: mowing the lawn
	5G.2	Modification	Using a subroutine
	5H	**Test**	**Test on Chapters 4 & 5**

Type	#	Assignment	Description
	6A	Calibrate	Automatic calibration
	6A.1	Calibrate	Automatic calibration
	6A.2	Calibrate	Using the calibration data
	6B	**LineFollower**	**The first line-follower**
	6B.1	LineFollower	A line-follower with a single sensor
	6B.2	Experimenting	Experimenting with the settings
	6B.3	Testing	Testing line-follower behaviour
	6C	**LineFollower**	**On the other side of the line**
	6C.1	Testing	What needs to change?

Type	#	Assignment	Description
	7A	**Line-Follower**	**A faster line-follower**
	7A.1	Line-Follower	A line-follower using two sensors
	7A.2	Experiment	Testing with the simulator and the NXT
	7B	**Line-Follower**	**Following the black line**
	7B.1	Line-Follower	Following the road using a subroutine
	7B.2	Testing	Following the black road
	7C	**Line-Follower**	**Following the yellow road**
	7C.1	Line-Follower	Following the yellow road
	7C.2	Testing	Testing YellowLineFollower
	7C.3	Testing	Following the rest of the track
	7D	**Test**	**Test on Chapters 6 & 7**

Part III

Type	#	Assignment	Description
	8A	**AdaptiveBehavior**	**State Diagram**
	8A.1	Study	Studying the state diagram
	8B	**AdaptiveBehavior**	**Architecture**
	8B.1	Code Modification	Setting up an architecture for adaptive behaviour
	8B.2	Code Modification	Making a first function act
	8C	WanderBehavior	Using a dummy routine
	8C.1	WanderBehavior	Testing the WanderBehavior dummy routine
	8D	**WanderBehavior**	**Random behaviour**
	8D.1	Code Modification	Driving around; random numbers

Type	#	Assignment	Description
	9A	**Movement Detection**	**Reacting to movements**
	9A.1	Movement Detection	Detecting movements
	9B	Movement Detection	Using the debugger
	9B.1	Movement Detections	Testing with the debugger
	9C	**Flee- and Curious**	**Becoming scared or curious**
	9C.1	CuriousBehavior	Becoming curious
	9C.2	FleeBehavior	Becoming scared
	9C.3	Testing	Testing with the robot
	9D	**Movement Detection**	**Scaring the robot**
	9D.1	Testing	Testing two robots on the simulator
	**9E**	**Test**	**Written test on Chapters 8 & 9**

Type	#	Assignment	Description
	10A	**Adaptive Behaviour**	**Avoiding obstacles**
	10A.1	StayInField	Staying on the field
	10A.2	StayInField	Creating and using StayInField
	10B	**Adaptive Behaviour**	**Proportional controller**
	10B.1	CuriousBehavior	Making Curious proportional
	10B.2	Obstacle	Making the robot's speed variable
	10B.3	Testing	Testing with an obstacle
	10C	**Object Tracking**	**Following and avoiding**
	10C.1	Curious Behaviour	Following an object

PART I

This first part is the basic module. It is necessary to learn to work with RoboPAL and a robot. Students who have already worked with RoboPAL may skip it.

It is also possible to follow the shorter version of the basic course that consists of instruction cards. It presents considerably less material and goes less in depth. It allows students to get started quicker, but it may also generate more questions during later assignments, as a number of basic principles will not have been discussed.

Introduction

1. Find out

Use a search engine to find the meaning of the word “robot”. Use what you have found to describe (½ A4) the similarities and differences between the old and modern meaning of “robot”.

Robot means ‘worker’ in Czech (as is explained in a footnote in Part 1). The original meaning of “robot” is a machine that performs jobs in a factory. Nowadays, we interpret “robot” as a machine that can perform tasks independently. They can be used in factories, organizations and in the house. In addition, robots are more and more in use both in the entertainment industry and as toys.

2. Find out

Use a search engine to find enough information to answer the following questions. (½ A4)

- What is a robot?
- Are there different types of robots?
- What is the difference between a robot and a human?
- If you had to divide all robots into two main groups, what would they be?
- What is the main difference between these two groups?

A robot is an industrial, domestic or entertainment machine that can perform tasks autonomously. Moreover, there also are service robots that are mainly used in health and personal care. However, the two largest groups are industrial and service robots. In industry, robots are mainly used to perform repetitive and dull tasks such as welding and assemblage operations. Service robots, on the other hand, need to respond intelligently to their environment in order to help people who are not able to perform certain tasks themselves.

1. Getting to Know Your Robot

The most important parts of a robot are explained in this chapter. You may cover this chapter together in the classroom, but it can also be read individually. In the first three chapters, all the most important hardware and software parts of a robot are explained, allowing students to get started with the equipment. The first chapter is mainly a reading task in which the most important concepts are treated.

The most important thing that must be discovered in this first lesson is the width and complexity of the development environment that will be used. Students will be introduced to hardware and software elements and find out more and more about these parts as they go along.

In this Module, we use the Lego MindStorms NXT robot. The lessons are based on the use of a standard robot that first needs to be assembled. The building process is not included in the course material, but separate instructions are available as a PowerPoint document that can be found on the CD. There are two versions and the exact form is not so important. What is important is that the motors and sensors are connected correctly; otherwise, RoboPAL will not work. The connections are clearly described in the lessons, but here they are again:

- Motors Left and Right are connected to ports A and C
- Light sensors Left and Right are connected to ports 1 and 4
- Distance sensor (Ultrasonic) is connected to port 3
- Extra sensor is connected to port 2 (not used in these lessons)
-

1.1 You will learn

- what a robot is
- why we are using a simulation program
- about various programming languages

1.2 You will need

- a computer
- a Lego MindStorms NXT Robot
- the RoboPAL development environment
- the simulation program (part of RoboPAL)

Fig 16: Lego NXT

1.3 You will experiment with

- the components of the robot and learn to name them
- the how and what of a simulator
- the language used to program a robot

1.4 After following this chapter, you will be able to

- explain the meaning of the terms “Robotics” and “AI” (Artificial Intelligence)

  AI or Artificial Intelligence is a scientific discipline that aims to gain insight into the way intelligence works. It is a multidisciplinary field that is related to biology, philosophy, neurology, mathematics and information technology. Robotics is the practical application of AI via the construction of autonomous robots. In addition to these fields, AI is also related to electronics, engineering and industrial design.
- point out the most important parts of a robot, name them and describe their functions

  In addition to general elements such as sensors and actuators, the processor plays an important role. Students must be able to name and point out these parts of a robot. It is also important to be able to point out the most important parts of a robot in general terms.

1.6 Test

1. What sensors do you know about and what is their function? Explain to what human senses they correspond.
   - ***Pressure sensor*** *- detects an obstacle and corresponds to touch or feeling in our fingers.*
   - ***Reflection sensor*** *- measures the grey value on a surface. It roughly corresponds to our eyes.*
   - ***Infrared sensor*** *(or Light sensor) - measures the strength of a light source. It also corresponds to our eyes, although we cannot detect infrared light.*
   - ***Ultrasonic*** *sensor - measures a distance by sending out sound waves. It corresponds to the senses that a bat uses to detect its environment. Our ears are more like a microphone.*
   - ***Compass sensor*** *- detects the magnetic north of the earth’s magnetic field. We have no sense for magnetism, but some birds do: pigeons, for example.*
   - ***Gas sensor*** *- detects certain gases. It corresponds to some extent to our nose.*
   - ***Distance sensor*** *- determines the distance to an object. We use our eyes to do this.*

2. Explain what the following robot components are used for: push buttons (three of them), batteries, processor, motors, connectors, distance sensor, reflection sensors and USB connection.
   - ***Push buttons*** *(three) - The orange button is used to switch on the NXT. The two triangular buttons are used to select system functions, while the rectangular grey button resets the NXT when pressed together with the orange button. Always switch off the NXT unit when it is not in use.*
   - ***Batteries*** *- are used to power the motors. Although different power supplies are often used for motors and electronics (to prevent disturbances), this is not done on the NXT.*
   - ***Processor*** *- the 'brain' of the robot. It stores the program and all sensors and motors are connected to it.*
   - ***Motors*** *- allow the robot to move around. The motors of the NXT can be used both as servomotors and as regular electromotors. In general, this is very unusual for a motor.*
   - ***Connectors*** *- connect the sensors and motors to the processor. In addition, we also have connectors for the power supply and the USB cable.*
   - ***Distance sensor*** *- sends out a high frequency sound to measure distance by calculating the time the echo takes to return to the sensor.*
   - ***Reflection sensors*** *- measure the grey values on the surface.*
   - ***Reset*** *- press the orange button and the grey one beneath it at the same time. This switches off the unit and puts it back in its original state.*
   - ***USB connection*** *- connects the robot to a PC. We only do this in case of problems or to load the firmware. Programs are uploaded to the NXT via a Bluetooth connection between the PC and the robot.*
3. What is the purpose of the LCD screen on the NXT brick?
   The LCD screen displays messages from the program. It allows you to check on what the program is doing. The sensor values are shown in this way, too. It also displays four simulated led lights that can be turned on and off from the program.
4. Why are we using a simulation program?
   A simulator allows you to quickly see what the program does without using a robot.
5. Which programming language is used to control our robot?
   In this Module, use the RoboPAL graphical programming language.

1.1 Rescue Assignment Rules

- Open ***Downloads*** via ►www.robocupjunior.nl
- Open Rules
- Open Rules Rescue 200X
- Study these rules

Students should study the rules for the RoboCup Junior Rescue competition. This will give them an idea of what is required from students who want to participate in the competition. This background is important to understand what the requirements are both in terms of robots and programs.

1.2 Find Other Programming Languages (search engine)

- Find out whether there are other programming languages and try to answer this question: why there are so many of them?
- Mention at least three and what they are used for
- What are the most used programming languages and why are they so popular?

Industrial:	Ada · Assembler · C · C++ · C# · COBOL · D · F# · Fortran · J# · Java JavaScript Lisp · Modula-2 · Object Pascal · Pascal · Perl · PHP · Python · Rexx · Ruby · Basic · Visual Basic
Academic:	Eiffel · Haskell · Logo · ML · Oberon · Occam · Prolog · Scheme · Smalltalk
Historic:	ALGOL · APL · BASIC · Clipper · MUMPS · PL/I · PowerBuilder · Simula

In industry, the most frequently used programming languages include COBOL, C (various dialects), Fortran, Java and Visual Basic. The reason for this is mainly historic. Often, they were developed specifically for business applications or specific application areas and as a lot of software has already been developed and developers have been trained in these languages, there is a strong resistance to change. In fact, it would cause large maintenance problems. Older programs often have a lifespan of over 30 years and must be maintained continuously. They are called ‘legacy’ programs.
Sometimes, a new generation of programming languages will handle a certain type of problem better, faster and/or cheaper and this will stimulate the switchover to a new language. This happened with Java, which took a giant leap as a result of its use in Internet applications and then spread to many other areas. A further reason is often the availability of new development tools.
In this Module, we will use the RoboPAL graphical programming language that has been developed specifically for educational purposes. Working with icons makes it easier for beginning programmers to understand what a program is doing. Programming languages such as Java are text-based and more difficult to use, but they are considerably more powerful. The Java-version RLT Module works with Java and Eclipse as a development environment.

2. Getting to Know the Simulator

In this chapter, students grow familiar with the simulator and the RoboPAL development environment. There are several detail assignments in this chapter that students with experience in information technology can quickly skim. Students that do not have this advantage should follow parts of the detail assignments if they seem useful. In it, they are given a code fragment that they must try to understand and then modify. This should eliminate the initial fear of program code and programming.

2.1 You will learn

- to interpret a RoboPAL program
- to understand *FleeBehavior* and *CuriousBehavior*, both as behaviour and as a program
- how to work and test programs on the simulator
- how to recognize errors in a RoboPAL program

2.2 You will need

- a computer with RoboPAL
- the Grid field (a white surface)
- two programs: *FleeBehavior* and *CuriousBehavior* (You actually get *FleeBehavior* twice and need to change one of them into *CuriousBehavior*.)

2.3 You will experiment with

- simulator features
- changing an existing program and testing it
- the behaviour of the robot on the simulator

2.4 After following this chapter, you will be able to

- explain how the simulator works and why it is useful
- work with the development environment and the simulator
- use loops and draw lines between program icons
- make small changes to an existing program

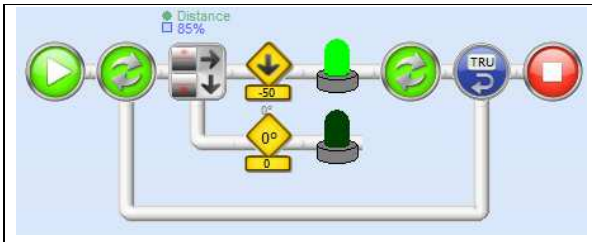
2.5 Test

For some of the questions below, it may be necessary to look at the detail assignments.

1. Explain what the following parts of RoboPAL are used for: FlowCodeSheet, WorldViewSheet, ToolBox, Simulation and Control Panel

- **FlowCodeSheet** - is the panel in which the program icons are inserted.
- **WorldViewSheet** - is the panel with the playing field and the robot.
- **ToolBox** - is the panel on the left in which all available icons are shown. The FlowCodeSheet and the WorldViewSheet panels each have their own set of icons.
- **Simulation** - is the panel that appears after starting the simulator. It displays the simulated field, the FlowCode being executed and the Control Panel.
- **Control Panel -** is the simulated control panel of the NXT and a representation of all sensors and lamps.

2. What is the meaning of this icon  in RoboPAL?
   This icon activates the simulated green LED on the NXT LCD screen.
3. What is wrong with the program in the FlowCode underneath:



   In this program, turning off the green LED is not connected with the Merge icon above it.
4. What will happen if you run this program?
   After turning off the green LED, the program does not know where to continue and will stop.
5. How are errors indicated in RoboPAL ?
   RoboPAL generates its own error messages; however, if it is a serious error, the program may crash and Windows will generate an error message.
6. What is the function of a debugger and what is a breakpoint?
   - *A debugger is used to be able to follow what a program is doing during execution, step-by-step, and inspect or change the contents of the variables.*
   - *A breakpoint is an icon that signals that the debugger must stop and wait for a command to continue.*

Type	#	Assignment	Description
	2A	Simulator	Working with the Simulator
	2A.1	Simulator	Starting the Simulator
	2A.2	View	Studying the Features of the Simulator
	2A.3	Insert	Adding parts
	2A.4	Making a copy	Making a Safety Copy
	2A.5	Drawing lines	Connecting Icons with Pipes

Type	#	Assignment	Description
	2B	**FleeBehavior**	**Testing flee behaviour**
	2B.1	FleeBehavior	Making the Robot Flee
	2B.2	FleeBehavior	Studying fleebehavior with Robopal
	2C	**CuriousBehavior**	**Making curiousbehavior**
	2C.1	CuriousBehavior	Making the Robot Curious
	2D	CuriousBehavior	Reading sensor values
	2D.1	Experiment	Experimenting with Sensor Values
	2D.2	Experiment	Experimenting with Turns
	2E	CuriousBehavior	Making the Robot Stop in front of the Ball
	2E.1	Personal assignment	Making the Robot Stop in front of the Ball
	2F	CuriousBehavior	Error messages
	2F.1	CuriousBehavior	Errors in Your Program
	2F.2	CuriousBehavior	Debugging a Program

The expedition through the land of robotics starts with the RoboPAL development environment. The simulator is integrated into RoboPAL. Students who want to learn more can install RoboPAL on their own home computer.

For students who think this is too easy, there are many other features of the environment to be discovered. In the lessons, we will discuss the most important components in greater detail, including the facilities for ‘debugging’, (tracking down errors). So there is a lot to learn and to discover.

The **Documentation** directory on the CD provides further background information and additional information may be found on the RoboPAL site.

3. How Does Your Robot Work?

After having explained robots, in general, RoboPAL and the simulator, we will now address the NXT robot and how to upload a program to its memory.
This will provide all the necessary information to get started in the following chapters.

3.1 You will learn

- how to make a connection with the Lego NXT robot
- how to upload a program and have the robot execute it

3.2 You will need

- a computer with RoboPAL
- a RoboPAL dongle
- charged batteries / adapter
- a table to make your robot move on the Grid field (a white surface)
- the *CuriousBehavior* program

3.3 You will experiment with

- making contact with your robot
- finding the NXT robot in RoboPAL
- uploading a program to the robot from the computer
- trying out the program on the robot.

3.4 After following this chapter, you will be able to

- explain how to write a program, how to test it, how to upload it to the robot and run it on the robot

  You use RoboPAL to create and test programs. When you start the Simulator, it will prepare the program for upload to the NXT. Then, you can upload the program to the NXT. Once it has been stored on the NXT, use the RUN button on the NXT to start the program and check if it performs just as in the simulator.
- use the RoboPAL dongle

  In the simulator, use the upload button to transmit the program to NXT. Select the name of your NXT. If it is not known, select <refresh list> or <refresh network list> to find the NXT. If the name of the NXT has been changed, use <clear list> to remove the old name from your PC. Then, use the upload button to send the program to the selected NXT. The NXT needs to be in the main menu.

3.5 Test

1. What needs to be done to the NXT before you can use it with RoboPAL?
   Using the Perspective Tab of the RoboPAL start-up screen, install the LeJos Operating System, followed by the RoboPAL Firmware. The teacher or the system administrator usually does this. If you are using your own robot at home you will need to perform these steps yourself.
2. Where can you find the 'number' of your robot?
   This number is specified when uploading the RoboPAL Firmware and is displayed on the start-up screen of the NXT.
3. What mode should RoboPAL be in to upload a program onto the NXT?
   RoboPAL must be running the Simulator, the dongle must be in the PC (or in the Server computer) and the NXT must be switched on the start-up screen.
4. If you are not able to upload a program to your robot, what could be wrong?
   The robot could be switched off or a program may be running. It may help to switch the NXT off and on again.
5. How can you read the values of the sensors?
   The CHK option in the start-up screen shows the sensor values on the LCD screen. On the simulator, this is done in Control Panel.

Type	#	Assignment	Description
	3A	RoboPAL	Loading the Firmware
	3A.1	Basic settings	Basic Settings in RoboPAL
	3B	**CuriousBehavior**	**Loading the Program**
	3B.1	Uploading	Uploading the Program
	3B.2	Reading sensors	Reading the Sensors
	3B.3	Testing	Making the Robot Drive Around
	3C	Test	Written test on the first three chapters

After becoming familiar with the development environment and the simulator, students move on to the hardware. Actions such as turning the robot on and off and uploading a program to the NXT are repeated very often, so students need to become familiar with them.
Please note how the NXT is connected to the computer. If a robot has not been discovered on a computer, this needs to be done first.
You can upload programs to the NXT in two different ways. The special RoboPAL dongle allows the programs to be transmitted. This dongle, which also serves as a license key, is very easy to use and requires no further drivers or additional software to be installed.
Before using the NXT, the Lego Java Operating System (LeJos) needs to be installed. Instructions for this are included in the Installation chapter.

The second method is to use the RoboPAL server. A different version of the dongle is connected to a central PC, which works with the RoboPAL server program. All PCs communicate through RoboPAL with the Server, which takes care of sending the correct program to the NXT.

4. Driving over the Rescue Field

Students are now ready to create their first program. We have already seen how the robot drives forward and makes turns. Now, we will see how you can determine at what precise moment something needs to be done and how long this action should take. In robotic applications, doing things at the exact time is essential. We use a method that involves counting how long or how often a robot must do something. This allows a precise control over the timeliness of an action. This is fundamentally different from administrative programs in which timing is hardly ever a critical factor.

4.1 You will learn

- to set the speed of the robot
- to make the robot stop, after a certain time, by counting
- to let the robot make turns
- to show a message on the display

4.2 You will need

- a computer with RoboPAL and the simulator
- a robot
- the rescue field
- the program *DriveBehavior*

4.3 You will experiment with

- making the robot move forward for 2 seconds and then stop
- letting the robot make turns
- programming it to follow part of a road

4.4 After following this chapter, you will be able to

- describe the properties of a robot environment on the simulator and on the actual rescue field
  The important thing here is that students discover the differences between the two. The simulator provides a good idea of what the program does and helps to identify the most important mistakes. However, a real robot suffers from disturbances and is not accurate. The sensor values for a robot always differ and the program will need to be adapted frequently.
- explain why following a fixed pattern is difficult
  Following a pattern by giving precise instructions does not work very well, because the track cannot be described in sufficient detail. If the exact starting position is not used, the entire pattern shifts. If there

is a small unevenness on the field, the wheels may slip, in which case the pattern will not work. If the batteries run down, the robot will start driving slower and the pattern will go wrong.

- explain what a subroutine is and be able to use one
  A subroutine is a piece of code that is reusable. It is always called by another routine, which is known as the Caller. Variables used in the subroutine may be influenced by the Caller that can supply the value as a parameter.
- make a simple program in which a robot drives straight ahead and makes turns
  This is the main topic of the exercises in this chapter.

4.5 Test

1. What does flowcode 9 do, describe it in pseudo code.

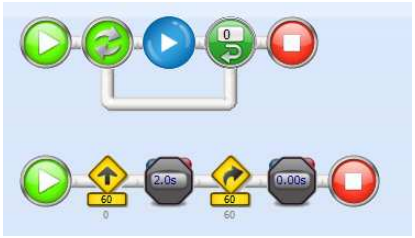
FlowCode 9

```
Repeat 1 time:
  The robot drives forward with a speed of 60%
  during 2 seconds
  Then it makes a turn to the right
  during 0 seconds (which does nothing)
Stop
```

Although the LoopCounter is set to 0, the program will first go through the code until it reaches the LoopCounter and will find out it is ready. So, it is not possible to skip the code in the loop in this way. Normally, the counter is decreased and when it reaches zero, the loop condition terminates. However, as the counter is already at zero, it will not be decreased and the code will have been executed. This construction is called a Do-Until Loop and always tests its condition at the end of the loop. A Do-While Loop performs its test at the beginning of the loop, but RoboPAL does not offer this facility. Making a turn for 0 seconds will effectively do nothing, since the motor is stopped right after is has been started.

2 How often is the code inside the loop of FlowCode 9 executed?
As the counter is set to zero and the test is done at the end, the code will be executed at least once.

3. Insert FlowCode 9 into a subroutine.



FlowCode 10

To do this, we need to make a new routine and call it "Main". This main routine then calls the subroutine that is inside the loop. In FlowCode 10 you therefore see two subroutines, the first is Main which calls the second one.

4. How can you make a robot do the same thing for a couple of seconds?
   By placing a stopwatch after an instruction and specifying how long it should wait before continuing. The instruction may, for example, be a Driver icon that starts driving forward. The program then continues and when the next instruction is to wait, the Driver icons remains active during that waiting time. If the next instruction is another Driver icon, then that one will be executed, but if the instruction is one that does not give a new Driver command, the original Driver instruction will remain active.
5. What kind of indicators can you use in RoboPAL?
   In RoboPAL you can use the following indicators:
   - *Turning a simulated LED on the display on or off*
   - *Sounding a beep*
   - *Showing a message on the screen: an LCDMessage*
   - *Showing a variable on the screen: an LCDDebugMessage*

Type	#	Assignment	Description
	4A	**DriveBehavior**	**Changing the Robot's Speed**
	4A.1	DriveBehavior	Adapting 'Drive' Speed
	4B	**DriveBehavior**	**Starting the Program**
	4B.1	Code change	Showing the Status
	4B.2	Code change	Cleaning up the Program
	4C	**DriveBehavior**	**Driving forward and Stopping**
	4C.1	DriveBehavior	Making the Robot Drive forward
	4C.2	DriveBehavior	Making a Turn
	4D	**SubRoutine**	**Using a Subroutine**
	4D.1	SubRoutine	Making a Subroutine
	4D.2	SubRoutine	Using the subroutine
	4D.3	SubRoutine	Making the Main program
	4E	**SubRoutine**	**Driving in a Square**
	4E.1	SubRoutine	Making the Robot Move along a Square

PART II

This second part mainly covers the Rescue mission. A number of important principles are discussed here. For this section, there also are more simple instruction cards that treat the subject in less detail.

Students that find this part not challenging enough may skip it and go directly to the third part, which is more advanced.

5. Sensors: the 'Sense' Phase

In part II, we introduce the so-called Sense-Reason-Act Loop. Subsequently, we will work out each of these subjects in greater detail as the programs gradually become more complex and more interesting. We start with the robot's sensors. These are to robots what our senses are to us.

5.1 You will learn

- how to use sensors
- how to handle sensor data
- what properties sensors have
- what sensors can do

5.2 You will need

- a robot and a computer with RoboPAL
- a rescue field
- the *Calibrate DriveBehavior* and *CuriousBehavior* programs

5.3 You will experiment with

- calibrating the left reflection sensor for yellow, green and black
- making a table with measurement values
- programming the robot to drive up to the black line
- setting a time limit
- adjusting the sensitivity of the sensors
- searching the swamp in the rescue field

5.4 After following this chapter, you will be able

- to calibrate sensors
  This mainly means adapting the robot to circumstances such as the lighting situation; matching the sensors is a bit more difficult. This may involve scaling the sensor values to make them equal.
- to make the robot react to colours and objects
  After having calibrated the sensors, we can read the sensor values and determine what colour is on a surface or what the distance to an object is with the distance sensor.
- to use loop constructions
  We use a loop or repeat construction to repeat parts of a program. We can make loops using a counter, a timer or a condition with LoopCounter, LoopTimer and LoopConditional.

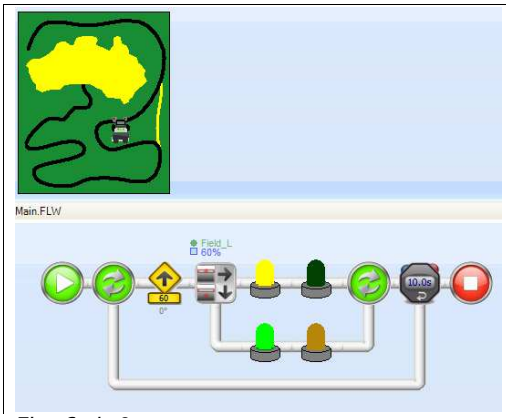
- to draw a graph of the measured values of a sensor and determine the linearity
  By making a graph you can check if the sensor characteristics are linear. In most cases, this will not be true for the entire range of values: with the distance sensor, for example, you may see that at less than 10 cm the data becomes unreliable.
- to explain the working of the Sense-Reason-Act loop.
  Animals and people collect information on their environment through their senses. We determine what is relevant and what reaction is required on the basis of this information. Then, we perform an action. This may result in a change in the situation, after which another observation is required. This cycle takes place instantly and in most cases almost simultaneously. In a robot, these actions take place sequentially in the sense-reason-act loop.

5.5 Test

1. The senses of living organisms look a lot like the sensors of robots, but there also are big differences. Provide some examples.
   Examples include our sense of balance, but also smell and taste are difficult to replace by sensors. On the other hand, we do not have very good night vision, nor can we see temperature in different colours like some animals.
2. When do you have to calibrate sensors?
   Every time that the circumstances change, like the lighting situation or moving to another location
3. What kind of information is collected by robot sensors? Explain, for every sensor, what physical property is recorded and what values will be presented.

Distance sensor	*distance*	*number*
Reflection sensor	*reflected light*	*number*
Touch sensor	*0 or 100*	*number*

4. Look at the program below and explain what happens when you run this program and the robot is in the indicated position.



FlowCode 3

The robot drives forward at half speed and uses the left floor sensor. If it detects a value that is higher than 60%, the yellow lamp is turned on and the green lamp off. If it detects a darker colour, then the green lamp is turned on and the yellow lamp off. This is repeated for 10 seconds.
The robot will spot the black line first, so the green lamp will be turned on. Then, it will detect green, so the green lamp will remain on. Immediately after that it will detect the yellow field and the yellow light will come on and the green lamp go off. It continues to drive and then stops on the yellow field.

5. What happens to the program if you lower the value of the sensor?
   Depending on the value, it will detect the colour green and turn on the yellow lamp.
6. And if you increase the value?
   Depending on the value, the robot will not recognize the colour yellow and the green lamp will stay on.

Type	#	Assignment	Description
	5A	**Discussion**	**How Does a Robot Work**
	5A.1	Group discussion	Where are the Sensors Located?
	5B	CalibrateBehavior	Reading Sensor Values
	5B.1	Calibration	Reading Sensor Values using the Simulator
	5B.2	Calibration	Reading Sensor Values with the Robot
	5C	**DriveBehavior**	**Driving up to the Black Line**
	5C.1	DriveBehavior	Stopping on the Black Line
	5D	Loops	Making a Loop
	5D.1	Loops	Using a Loop
	5E	**DriveBehavior**	**Using a Time Limit**
	5E.1	DriveBehavior	Adding a Time Limit
	5F	CuriousBehavior	Making a Graph
	5F.1	Measuring	Drawing a Graph
	5F.2	CuriousBehavior	Modifying CuriousBehavior
	5G	**DriveBehavior**	**Cutting the Reed in the Swamp**
	5G.1	Search the Swamp	In Practice: Mowing the Lawn.
	5G.2	Modification	Using a Subroutine
	**5H**	**Test**	**Test on chapters 4 & 5**

One of the most important things that must be done when working with robots is calibrating the sensors. Students will discover the relative unreliability of sensors. It would be nice to offer some background information from physics about factors such as noise and material properties and how they affect sensors. In order to take these differences into account in a program, it is important to investigate the measured values under different conditions.
Then, students will discover how a robot can (seemingly) read out sensor values and keep track of the time limit at the same time. It is important to

understand that an optimal control over when things happen in the program is essential to programming a robot.
These are aspects that are not so relevant in administrative applications. An administrative program will often 'wait' until a certain amount of time has passed, whilst remaining idle. If you do that with a robot it will not be able to do anything else while waiting and will therefore not notice any changes in sensor values at the right moment.

5A.1 Assignment: Group Discussion

- Which sensors do we know in practice?
- In what equipment can we find them?

1.	Heat sensor	e.g. washing machine
2.	Light sensor	e.g. garden lighting
3.	Humidity sensor	e.g. car windshield wiper
4.	Infrared sensor	e.g. security system in building
5.	Pressure sensor	e.g. under doormat for shop door beeper

We are observed by sensors almost continually. The shop door opens automatically. The garden light turns on when it gets dark. When the coffee machine gets too hot it switches off. There exists equipment that will open a door when you come near it via a chip (the size of a grain of sand) placed under your skin that detects who you are and if you have permission to enter.

Here you see a special reading light, developed by the DQL research group. The user can change the light by brushing over the back of the lamp. Using sensors, the lamp 'searches' the book that is being read and follows the movement of the reader as he/she changes position.

sync.nl/wp-content/uploads/articleimages/586.jpg

SO 5H: Written exam on chapters 4 & 5

These chapters end with a written exam.

6. Processing: the 'Reason' Phase

We have looked at the information that sensors deliver to a robot in the Sense phase of a Sense-Reason-Act loop. In the Reason (or processing phase), a robot decides what to do with this information. We are going to make a first, simple Line-Follower that the robot can use to follow a track on a field.

6.1 You will learn

- to read the values of robot sensors
- how to make a robot remember its calibration data
- how to control its motors
- how a line-follower works

6.2 You will need

- a NXT robot
- fully charged batteries
- a computer with RoboPAL
- a rescue field
- a *LineFollower* program

6.3 You will experiment with

- Using the calibration values of the robot in a program
- making a program that follows a line using a single sensor

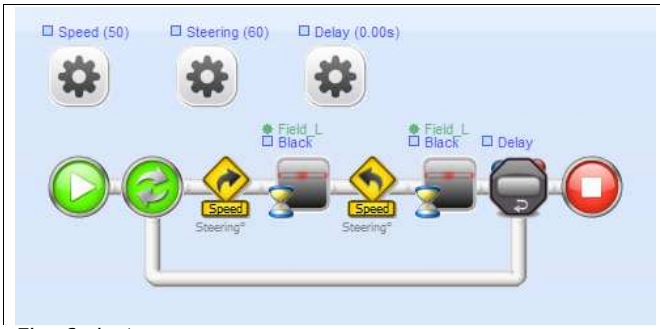
6.4 After following this chapter, you will be able

- to explain the operation of a line-follower
  It is important that students understand what the difference is between a line-follower with a single sensor and one with two. They also need to gain insight into the factors that may influence the behaviour of a line-follower: especially the angle at which a robot follows curves on a line that must be followed. Students should understand that an optimal setup can only be reached by using information on the nature of the curvatures. To do this they need to know approximately what the robot's position is on the trajectory; without this information a solution is possible, but it will be a very slow one.
- to use a simple line-follower
  A line-follower has a number of parameters that can be modified. The lessons have been designed to learn how to deal with these settings and at the same time understand how the line-follower works. In this lesson, we start with a simple line-follower with a single sensor.

- to read the sensor values of the robot
  The sensor values can be inspected on the simulator as well as on the NXT. This information is important to set up the line-follower and to gain an understanding of how the sensor values are collected and used.
- to calibrate the field sensors of the robot
  By gaining insight on the values that sensors return, students will discover that sensors differ from each other. The calibration process is meant to eliminate these differences.

6.5 Test

1. Suppose you have used your robot and have read the following sensor values: **`Black 200, Green 280, Yellow 420`**. Explain what values you will use in the line-follower FlowCode for the colour black.
   As black reflects the least amount of light, it will result in the lowest value. To detect black we need to check for a value that is lower than the value for green, but higher than the lowest value for black. The best choice is between these values, so around 240 in the given examples.
2. What is wrong in FlowCode 4?



FlowCode 4
The colour black is used twice; it should be black and green.

3. What will happen if you do not change this?
   The robot will go to the right and as soon as it detects the black line go back to the left. It will now continue going to the left until it detects black again. This will probably occur when the robot has made a full circle, but it will certainly not follow the line.
4. Under what conditions can something go wrong with a line-follower?
   Things will go wrong if the colours have not been calibrated correctly, if the robot travels too fast or if the software reacts too slowly. Moreover, the contrast between light and dark must be sufficiently large. If the ambient light is intense, like sunlight, the sensor may be blinded.
5. What can you do to resolve that?
   Perform the calibration more accurately and check that the contrast is at least 5%. The robot can drive more slowly or the sensor frequency

can be increased. If there is 'false' light, you should construct some kind of sunscreen around the sensor.

Type	#	Assignment	Description
	6A	Calibrate	Automatic calibration
	6A.1	Calibrate	Automatic Calibration
	6A.2	Calibrate	Using the Calibration Data
	6B	**LineFollower**	**The first line-follower**
	6B.1	LineFollower	A line-follower using a single sensor
	6B.2	Experimenting	Experimenting with the settings.
	6B.3	Testing	Testing LineFollower behaviour.
	6C	**LineFollower**	**At the other side of the line**
	6C.1	Testing	What needs to change?

In chapter 4, students made the robot follow a trajectory in a 'naïve' way. We have seen that this does not work very well and promised to show them a better way. In this chapter, we fulfil that promise and show how the information from a reflection sensor can be used to follow a line.
We start with a very simple line-follower and will move onto a better line-follower in the next chapter. At the same time, we will show how the information about the robot's internal state can be used to make a program gain insight on what the robot registers. This is necessary to test the program properly.
Testing a program is the most difficult part for most students. This is because a program is created with a certain idea in mind and you are inclined to look at it with that idea. Students must experience that 'you see what you want to see' and that their mistakes are missed very easily, because they do not look at what is written in the program, but see what is meant to be there. The computer and the robot, of course, do not know what that was supposed to be and interpret everything literally. It is important to realize and experience this and at the same time learn the procedures that can help one break away from this way of looking at a program.
We have tools such as the LEDs that show what is happening and the LCDMessage icons that display what is happening to help us to do this.
The hard part in this independent assignment is to decide which method to use to solve the problem. There are several methods of doing this and they are all fine.

First of all, the robot will go to the left every time it does not see the black line. If you put the robot at the other side of the line, it will move away from the line. This may be solved by not going to the left, but to the right. Another solution is not to use the left sensor, but the right one and work in a mirror image. This is the preferred method, because we will be working with two sensors in the next chapter and we will need the mirror image there too.

7. Actuators: the 'Act' Phase

After the robot has decided what needs to be done with the sensor information, it will perform an action as the last part of the Sense-Reason-Act Loop. In most cases, executing an action means performing a movement and for this we need so-called actuators (mainly motors).

7.1 You will learn

- what type of actuators exist
- how actuators are controlled by information from the reason phase

7.2 You will need

- a NXT robot
- a computer with RoboPAL
- a rescue field
- a *Line-Follower* program

7.3 You will experiment with

- controlling motors based on data from the reflection sensors
- using a line-follower with two reflection sensors

7.4 After Completing This Chapter, you will be able

- to explain how a line-follower with two sensors works
  By using two sensors, a robot can keep the line in between its two sensors and react more quickly as it will need to implement less corrections than if it were using a single sensor.
- to use the information from the sensors to control the motors
  Not only does the line-follower react to the black line, but the robot also checks for yellow or black to identify the end of the line.
- to explain why a line-follower with two sensors can make the robot go faster
  Because with two sensors the robot can continue to drive straight as long as it does not touch the line and thus is not delayed. The zigzag movement with a single sensor transforms the real distance travelled into a much longer distance.

7.5 Test

1. What is the difference between a servomotor and a standard electrical motor?
   An electric motor runs forward or backward at a certain speed. To travel the correct distance, it is necessary to count how long the

motor has been running. A servomotor has a built-in position sensor that can determine what the position of the axis of the motor is. That way the axis of the motor can be put in the desired position without counting. It is also much more accurate because the position is measured precisely.

2. Why is a line-follower with two sensors faster?
   Because the robot no longer has to make a zigzag movement and can drive at top speed more often.
3. Which three variables determine the behaviour of a line-follower?
   The speed of the robot, the angle at which the robot makes turns (to correct the direction) and its sensitivity for the colour of the line
4. What happens when both sensors detect the black line at the same time?
   Normally this should not happen, but as we block the motor on the side where the line is detected, detecting it on two sides will stop the robot, or it will wiggle back and forth.
5. Could this really happen and if it does, what can be done about it?
   You cannot prevent this from happening, but you can develop a test for this situation. You can make sure that the motor does not stop completely, but that it takes its corners less sharply. You can also let the robot make a turn to free both sensors from the black line.
6. Why do we have a stop condition in the line-follower?
   The line-follower must continue to follow the line until a certain condition is met. Most of the time that is a certain time limit or a colour that is detected.

Type	#	Assignment	Description
	7A	**LineFollower**	**A faster line-follower**
	7A.1	LineFollower	A line-follower using two sensors.
	7A.2	Experiment	Testing with the simulator and the NXT
	7B	**LineFollower**	**Following the black line**
	7B.1	LineFollower	Following the road using a subroutine.
	7B.2	Testing	Following the black road
	7C	**LineFollower**	**Following the yellow road**
	7C.1	LineFollower	Following the yellow road
	7C.2	Testing	Testing YellowLineFollower
	7C.3	Testing	Following the rest of the track
	7D	**Exam**	**Exam on chapters 6 & 7**

By crossing over between light to dark to follow the line, the forward speed of the robot is decreased. If you make fewer turns you will go faster. Thus, the line-follower can go much faster by using two sensors and having the robot react only if one of the sensors touches the line.
In this lesson, students will discover what factors influence the speed of the line-follower: the threshold value of the line colour, the power

assigned to the motors and the angle at which the robot makes a correction.
The sharper the corner, the larger the distance travelled, but a smaller correction means that a sharp turn cannot be followed reliably. The real challenge is to find the optimal situation.
The robot can go even faster if you use the properties of the field. You can follow the yellow road and cut off a part of the road, but this means the line-follower does not follow the black line, but the yellow line. With a yellow line, the logic is reversed. Also, the rescue field contains some difficulties at this point that are not directly obvious to everyone.

What is important is that the line-follower has a so-called stop condition. With the black line that condition is the yellow line, because when you detect that you have to switch to following a line of a different colour. Which sensor you use for this is important, because the yellow line can only be detected on one side of the line. In addition, the stop condition for following the yellow is the beginning of the black line at the other end of the yellow line. HOWEVER, the yellow line begins exactly on the place where the other sensor will still detect the black line. Without any changes to the program, the robot will interpret this (wrongly) as the end of the yellow line and it will look like the program is not working.
To discover that such small, but very rapidly occurring situations may have a large influence on how a program works is an important part of this lesson. Many students find this very frustrating, because they do not understand what is happening and maintain that their program is correct and that the robot is not doing what it is supposed to do. It is important that they find out that the cause of the problem lies entirely in their own actions, but that is hard to communicate.
This is a second lesson in working accurately, thinking deeply about what is going on and reasoning about what is happening. It helps a lot to use the LEDs, the display and if possible sound to see exactly what is happening.

Exam 7D: Exam on chapters 6 & 7
These three chapters end with a written exam.

PART III

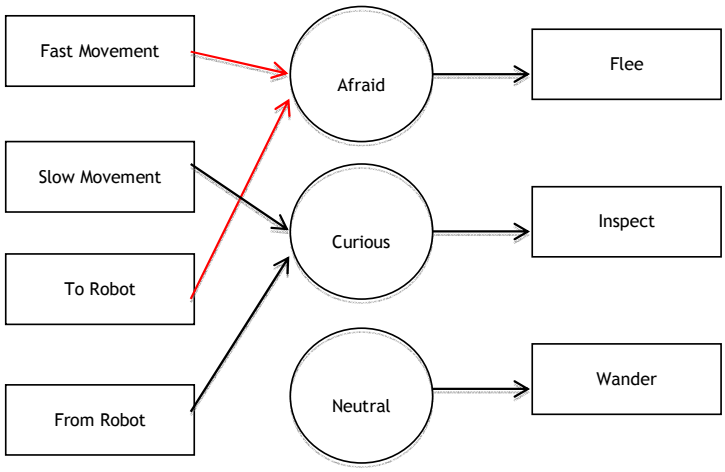
The third part is the most interesting, but also the most difficult part of the Module. It is best suited to students with a talent for technology or who are following a more theoretical education. Students with less interest in technology are better off concentrating on Parts I and II.

For students who find part III not challenging enough, there is another Module (not part of RLT) that contains information about the @Home competition, in which robots have to preform simple domestic tasks in a home. It provides many examples of so-called Service Robots that we will find in our houses in the future and that also could take care of the elderly- or other people who need help.

This third part is completed by a final assignment in which the students put their new knowledge into practice and develop a scenario of their own design.

8. Adaptive Behaviour

Now that we have seen the Sense-Reason-Act loop in action, we are going to adapt the robot's behaviour so that it reacts better to events that occur in its environment. We call this adaptive behaviour. This is schematically represented in a state-diagram, which is the main topic of the next three chapters.



8.1 You will learn how to

- program adaptive behaviour
- use random numbers
- program a robot to become scared, curious or indifferent

8.2 You will need

- a NXT robot
- a computer with RoboPAL
- a Grid field
- the *CuriousBehavior* and *FleeBehavior* programs

8.3 You will experiment with

- copying small parts of the behaviour of an ant
- modifying *FleeBehavior* and *CuriousBehavior*
- distinguishing between behaviours and states

8.4 After completing this chapter, you will be able

- to read and interpret a simple state diagram
  From here on, the state machine and the accompanying state diagram will play an important role. Everything will become a lot clearer for students if they first draft their design in the form of a state diagram.
- to use dummy routines
  Dummy routines are used when one needs to test completed parts of an incomplete program. A dummy routine temporarily takes over the task of the routine that still has to be developed, but without implementing its functionality. Subsequently, the dummy routine will be replaced by the designated function.
- to explain when random numbers are useful
  In any case in which it is important to 'just do something,' a randomizer is very useful. Without a randomizer, you would have to tell a robot in what direction and how fast it needs to move. In nature, a lot of animals display random behaviour. Microbes, for instance, look for food in their environment randomly. Lawn-mowing robots and robot vacuum cleaners imitate this behaviour, so that they can search an area without using a map.

8.5 Test

1. What is a state-diagram?
   A state diagram depicts the situations in which a robot can find itself and how the transitions from one to another state take place. It is a design for the logical structure of a program.
2. How clever is a robot? What kind of things will it not notice?
   A robot just reacts to its sensors, so it will not react to the size of objects or to sounds. Just as with animals it becomes scared by things that move fast, but it cannot distinguish between things that move fast or that suddenly appear in its field of sight.
3. What is the importance of an architecture?
   An architecture makes sure that you have a good overview of the design of a system and a clear layout of the components that need to be created, together with their interconnections. In our example, we first create the architecture and then its various parts.
4. What is a randomizer?
   A randomizer is a function that generates an arbitrary number. The numbers need to have a 'normal' distribution and there should be no repetitive sequences.

5. If a randomizer generates the following sequence of numbers: "1, 3, 5, 7, 1, 1, 1" is that an example of a good randomizer? Explain why.
   No, a good randomizer generates numbers that have a 'normal' distribution and therefore are not predictable. In this example, there is regularity and the second part contains a repetition.

Type	#	Assignment	Description
	8A	**AdaptiveBehavior**	**State Diagram**
	8A.1	Study	Studying the state diagram
	8B	**AdaptiveBehavior**	**Architecture**
	8B.1	Code modification	Setting up Adaptive behaviour architecture
	8B.2	Code modification	Making a first Act function
	8C	WanderBehavior	Using a dummy routine
	8C.1	WanderBehavior	Testing the WanderBehavior dummy routine
	8D	**WanderBehavior**	**Random behaviour**
	8D.1	Code modification	Driving around; random numbers.

In order to properly program this behaviour, we use a type of model called an architecture. This chapter is one of the most difficult ones in the entire Module. A lot of code is presented. The most important thing for students is to understand how the idea of a Sense-Reason-Act loop is used to adapt behaviour. This is, once again, an example of working in an organized manner and preventing that too many errors creep into a program that we will have to detect later on. The later you discover a mistake, the harder it is to solve it. Therefore, it is very important to check if the program works after every step and make sure that no errors have been introduced.
From this chapter onwards, there are no final assignments. The entire assignment should be developed independently.
Up to this point, programs have been based on a single behaviour. The actions of people and animals are more complex and composed of many steps. With the introduction of 'finite state machines', described by a state diagram, the students have been introduced to more complex forms of behaviour. Combining FleeBehavior with CuriousBehavior shows that the sensors may be used to collect more information about objects near the robot.
This lesson is the most extensive and is therefore split into a number of parts. It is important that the students make new versions of their code for each step, so that they can easily turn back to an earlier version if things go wrong. It is often very hard to find out where an error originates. Students must recognize that taking one step at time and frequent testing can prevent lengthy searches for the cause of an error.
A second important concept than needs to be learned is how to design a sound structure or architecture in which the logical connection between the parts is clear.
Finally the use of 'dummy' routines is important to test a program with unfinished parts. This approach is also called 'stepwise refinement' and

helps develop a working version of a program in a short amount of time. Making many small changes works considerably better than a ‘big bang’ approach. The latter method increases the chances of making mistakes and, in the end, it will cost considerably more time to locate the cause of an error.

9. Advanced Sensors

Now that we have looked at the basics of a robot program, we will further investigate sensors. We will begin with a sensor that looks at the internal status of a robot, namely its battery status. Unfortunately, this sensor is not accessible from inside the program, so we can only discuss what the robot can do with it. It is an example of a robot reacting to its own internal state instead of reacting to objects in the outside world. Moreover, we will also be looking at how we can use the distance sensor to collect additional information on the properties of objects. The principle that we use here is analogous to the way in which a radar works.

1.1 You will learn

- to use sensors to determine the direction and speed of an object
- to use the distance sensor in a different way
- something about sensor input ports

1.2 You will need

- a NXT robot
- a PC with *RoboPAL*
- a Grid field

1.3 You will experiment with

- making the robot react to movements
- detecting and avoiding an obstacle

1.4 After completing this chapter, you will be able

- to explain the difference between intrinsic and extrinsic sensors and their functions
  A digital sensor only returns a 0 or a 1, a true or a false, an object or no object. This means that the sensor itself has some kind of a threshold value that indicates if some situation is present or not. An analogue sensor returns a proportional signal that indicates a measure of how strong a given situation is.
- to use various sensor properties
  We have seen this issue before. Here, however, we want to clarify the difference between the various types of sensors. The light sensor (which we have not used) returns the amount of ambient light that it detects. The reflection sensor does the same, but only with reflected light. Silently, we assume that radiated light is reflected, but if the ambient light is stronger than the light source of the sensor, we will not get enough light back and the sensor will mostly function as a light sensor. The distance sensor indicates at what distance an object

is detected. As a result of the properties of a sensor, interpretation errors can occur when a robot gets too close to a given object.

- to make a robot modify its own behaviour
  By using sensor information, such as the distance to an object or the colour of lines on a surface, a robot can adapt to a given situation by avoiding an obstacle or reacting to surface colours.

9.5 Test

1. How reliable are the sensors on your robot?
   Not very. The reflection sensor can sometimes detect less than 10% difference between green and black. The distance sensor is more accurate (about 1 cm), but not over the entire range.
2. What factors may disturb sensors?
   The amount of ambient light influences the reflection sensor; however, the properties of the ink used for the rescue field are also important. The distance sensor cannot measure a distance of less than 3 cm. The battery level may also influence the behaviour of sensors.
3. How can you eliminate these disturbing factors?
   Calibration allows sensors to adapted to the existing circumstances. If we use a scale of measured values for every sensor between 0 and 100, it makes comparisons easier.
4. What does a robot need to measure the speed and direction of a moving object?
   To do this you need to take at least two sensor readings in succession and then calculate the difference between them. If the second measurement is higher than the first, the object is moving towards the robot, otherwise it is moving away from it. The speed is determined by the absolute difference between the two readings.
5. Provide an example of a useful intrinsic (internal) sensor for a robot
   The value of the battery voltage, but this information is not available on the NXT as a sensor value. The pushbuttons with which we give commands to the robot also are a type of sensor.

Type	#	Assignment	Description
	9A	**Movement Detection**	**Reacting to movements**
	9A.1	Movement Detection	Detecting movements
	9B	Movement Detection	Using the debugger
	9B.1	Movement Detections	Testing with the debugger
	9C	**Flee- and Curious**	**Becoming scared or curious**
	9C.1	CuriousBehavior	Becoming curious
	9C.2	FleeBehavior	Becoming scared
	9C.3	Testing	Testing with the robot
	9D	**Movement Detection**	**Scaring the robot**
	9D.1	Testing	Testing with two robots in the simulator
	**9E**	**Test**	**Written test on chapters 8 & 9**

We can use sensors to inspect the environment, to control other elements and more. Besides measuring grey values and distance, for example, if we combine a distance measurement with movement, we can also detect the contours of an object. This allows us to determine the size and form of an object and can be used to manoeuvre around an obstacle and avoid it. This requires a fixed pattern. We have already done something similar. Now, a pattern will be used to detect an obstacle and avoid it, according to a predetermined plan. To prevent deviations from the track, we use the sensor to continually monitor the position of the robot.

Exam 9E: Written exam on chapters 8 & 9
These chapters end with a written examination.

10. Control Systems

The way we control a robot, using a state machine, is a flexible way to make sure that it can react to its environment. Many machines have embedded systems to control them. Control Systems are used in many devices such as thermostats, for instance. You will find more complex controllers in cars and in servo systems. Control loops are often developed as PID controllers that are very efficient, but not as adaptive as our robots.

10.1 You will learn

- several aspects of control algorithms and real-time control
- the basic principles of feedback loops in control systems

10.2 You will need

- a NXT robot and a ball
- a computer with RoboPAL
- a Grid field

10.3 You will experiment with

- making the robot avoid obstacles
- developing a program to follow a ball

10.4 After completing this chapter, you will be able

- to explain the principle of a feedback loop with your own example
  There are many examples. The simplest is a thermostat, although it usually will not be a proportional controller. A proportional controller is often found in the temperature control of an oven. To get a more accurate control, feedback should be proportional. Other examples include the controller for the speed of a drill and the power supply stabilization for your computer.
- to indicate what happens when a robot controller only works with 'on' and 'off' and the motors run at full speed
  This requires some reasoning. If students use the speed control of a simulator, they will immediately see what happens. Reactions are exaggerated and if the robot is following a wall the behaviour is more like jumping than carefully staying close to the wall.

10.5 Test

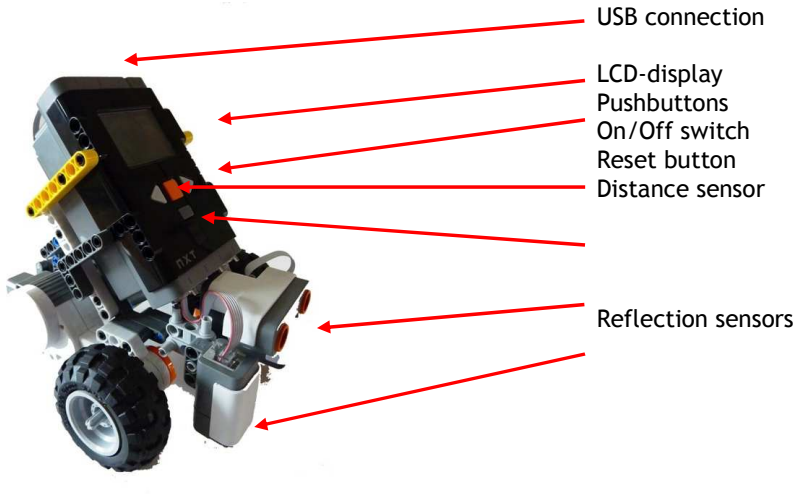
1. Why do we use feedback systems?
   The feedback principle is used to eliminate errors or small deviations. In a device that has to keep something stable, such as the power supply of a computer, a controller uses feedback to eliminate any deviations from the correct voltage supply.
2. What is the difference between a discrete and a proportional feedback system?
   A discrete controller returns a yes/no signal and triggers a relative action; while in a proportional controller, the strength of the signal deviation determines how much the controlled signal has to be modified.
3. Provide an example of the use of a feedback loop in your home and explain what kind of feedback it is.
   As already mentioned above, a thermostat; other examples include oven temperature, refrigerator temperature, computer power supply, speed of a power drill and water level of a toilet flush.
4. The final part is based on your program: the better it allows the robot to chase the ball, the higher your grade will be.

Type	#	Assignment	Description
	10A	**Adaptive behaviour**	**Avoiding Obstacles**
	10A.1	StayInField	Staying within the field
	10A.2	StayInField	Creating and using StayInField
	10B	**Adaptive behaviour**	**Proportional controller**
	10B.1	CuriousBehavior	Making Curious proportional
	10B.2	Obstacle	Making the robot's speed variable
	10B.3	Testing	Testing with an obstacle
	10C	**Object tracking**	**Following and avoiding**
	10C.1	Curious Behaviour	Following an object

Tests

Part I

Robotics Test Ch 1 – Getting to Know your Robot

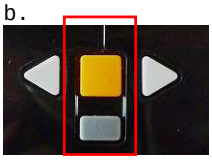


1. Which sensors are used the most in the RLT lessons?
   - ***Reflection sensor*** *- measures the grey values on the field*
   - ***Ultrasonic*** *- measures the distance to an object*
2. Which parts of the robot are used to manually operate it?
   Pushbuttons (four) *are used to switch the robot on and off and to select a choice from the menu.*
3. What is the purpose of the LCD screen on the NXT brick?
   The LCD screen displays messages from the program and allows you to check what it is doing. The sensor values can be displayed, too. The screen also displays four simulated lights that can be switched on an off via program control.
4. Why would you use a simulation program?
   To be able to see what a program does without having to use a robot.
5. What programming language do we use to control the robot?
   In this Module, we use RoboPAL.

6. Name the NXT parts shown in the pictures.

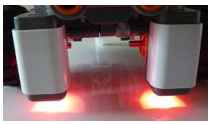
a.

b.



c.

d.



a) LCD display
b) Pushbuttons to turn the robot on- and off
c) Ultrasonic (distance) sensor
d) Light sensors or reflection sensors.

7. *Explain the function of the following parts of a robot*
   - *USB port - connects the unit to a PC*
   - *Processor - computer, controls all electronics and executes the program*
   - *Pushbuttons - select the robot's behaviour or functions*
8. NXT Sensors: how do they work and what can you do with them
   - ***Distance sensor** - sends out a sound wave (which you cannot hear). Objects reflect this sound, the receiver detects the echo and the distance is calculated.*
   - ***Reflection sensor** - sends a light beam, senses reflected light and determines a grey value.*
9. Four simulated LEDs are displayed on the LCD. The first one (red) is the Heartbeat that indicates that the processor is working. What is the function of the other three colours: yellow, green and blue?
   They show the robot's behaviour or status and the processes that are running.
10. If the robot does not react, how can you make it start again?
    By pushing the orange and the grey button underneath it at the same time. This switches the robot off.

Robotics Test Ch 2 – Getting to Know the Simulator

1. What is function is of FlowCodeSheet? of the WorldViewSheet?
   - ***FlowCodeSheet -** is the panel in which the icons of the program are placed.*
   - ***WorldViewSheet** - is the panel in which the playing field and the robots are placed.*
2. How are errors reported in RoboPAL?
   RoboPAL generates error messages; however, in the case of severe errors, the program may crash and Windows will generate an error message.

3. What is a debugger, what is it used for and what is a breakpoint?
   - *A debugger is used to follow the operation of a program step-by-step while it is running and to inspect and/or change the values of program variables.*
   - *A breakpoint is an icon that indicates that the debugger must stop execution and wait until a command is given to continue.*
4. What are the advantages and disadvantages of using a simulator compared to using a real robot?
   Advantages: *Easier and faster development & testing of programs; No real robot needed; Everything is always stable and works the same way.*
   Disadvantages: *It is not the real world; Cannot simulate all circumstances; Robot reacts differently than on the simulator.*
5. How can you see if there is a mistake in a RoboPAL program?
   - *a RoboPAL error message*
   - *the program does not work or crashes*
6. Here is a program that contains an error that RoboPAL can detect. What caused this error?

   A subroutine is called that does not exist.
7. Here is a program that contains an error that RoboPAL *cannot* detect. What is wrong? Look carefully. What needs to be changed?

   The first icon with a stopwatch is a LoopTimer that needs a wire running back to the beginning of the program, but the line is missing.
8. What will happen if you run this program?

   There is an icon that is not followed by a stopwatch so it will have a default waiting time of 1 sec. The robot will drive forward for 1 sec and then make a turn for 0.5 seconds.

9. What is wrong with this program?

The output of the sensor test is connected to the output of the stopwatch, but it should be connected to an action later in the program, which is missing. The construction shown here has no sense, so the program will ignore the sensor test.

10. Name three programming languages that are used often
Java, C or C#, Pascal, COBOL, Assembler, Python, Fortran, PHP, Basic

Robotics Test Ch 3 – How Your Robot Works

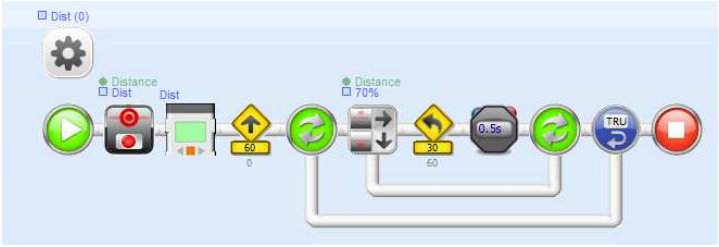
1. What needs to be done, in RoboPAL, before you can use a NXT robot?
First, you have to locate the robot using the dongle or the server and then select that robot on the simulator.
2. Where can you find the name of your robot?
The name is displayed on the start-up screen of the NXT.
3. Can you send a program to the NXT while you are in the development environment?
RoboPAL must be running the Simulator, the dongle must be inserted in the PC and the NXT must be switched on with the RoboPAL start-up screen.
4. How can you read the sensor values on the simulator?
The simulator has a series of dials in the Control Panel that show all the sensor values.
5. Where can you select the name of the robot that should receive your program?
Start the simulator, go to the robot selection box and select your robot from the list.
6. What should you do if your robot does not appear in the list?
Select <refresh list> or <refresh server list>
7. What do you need to upload a program to the robot?
A robot, RoboPAL operating the simulator and a dongle
8. How do you upload a program to a robot at school?
Use a white dongle (or a black server dongle) depending on the situation
9. Where do you store your programs at school?
Depending on the situation:
   - *Laptop*
   - *USB stick*
   - *Network drive*
10. If you are working with a server, where do you send your program?
Depending on the situation, the pathname of the server

Robotics Test Ch 4 – Driving over the Rescue Field

1. How can you make your robot perform the same action for a few seconds?
   By giving a command and placing a stopwatch after it that specifies the duration. A command such as a driver icon begins an action and then immediately moves on to the next icon. If this icon is a stopwatch, the command will continue to be performed for the duration indicated by the stopwatch.
2. How can you make a loop that counts the number of iterations?
   Use the LoopCounter icon
3. What is the function of the Merge icon?
   It provides a connection to the place where the program should continue after a branch.
4. Show the code in which you use a loop that lets the robot drive in a square.

5. You often see a program with a Driver icon. How can you make the robot: go slower; stop, go backward, make a right turn?
   Go slower - change the Speed
   Stop - Set the Speed to 0, or use a Stop icon
   Go backward - Speed negative or DriveReverse
   Make a right turn - Increase Steering or use DriveSwurveRight
6. Describe what this program does.

   It makes the robot drive forward 0.3 seconds, make a left turn for 0.3 seconds and then turn on the green lamp. Subsequently, it makes a left turn for 1 second (default), displays "Ready" on the LCD screen, turns off the green lamp and stops.
7. Develop code to do the following:
   - *Stand still and indicate the value of the distance sensor on the LCD.*
   - *Drive forward and as soon as the robot detects an object, drive backward slowly counter clockwise during 0.5 seconds*
   - *Drive straight ahead.*



8. How can we find out about the robot's status with RoboPAL ?
   - *LCD display with LCDMessage or LCDDebugMessage*
   - *Simulated LEDs on the screen*
   - *MusicalNote to sound a note*
9. What is the difference between a Subroutine and a normal Routine? Can every program be used as a Subroutine?
   A Subroutine is always called from another part of the program. In principle, any program can be used as a subroutine.
10. The program with which the robot starts is called the Mainline. Where do you specify which program this is?
    In the WorldView, use the robot Application properties to specify which program it should start with.

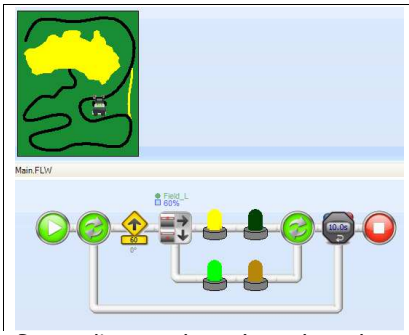
Part II

Robotics Test Ch 5 – Sensors: the Sense Phase

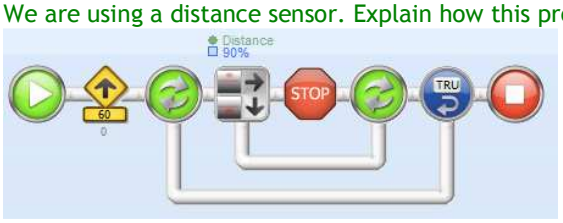
1. Robot sensors are similar to the senses of living organisms, but there are also large differences. Give some examples.
   For example, the sense of equilibrium, smell and taste are difficult to reproduce using sensors. On the other hand, humans cannot see well in the dark, nor can we see temperatures as colours.
2. When do you have to calibrate the sensors?
   Every time that the external circumstances change (i.e., ambient light or a different location), the sensors must be calibrated.
3. What kind of information do robot sensors collect? What do the distance sensor and the reflection sensor read and in what units is it being measured?

Distance sensor	*distance*	*number 0-100%*
Reflection sensor	*reflected light*	*number 0-100%*

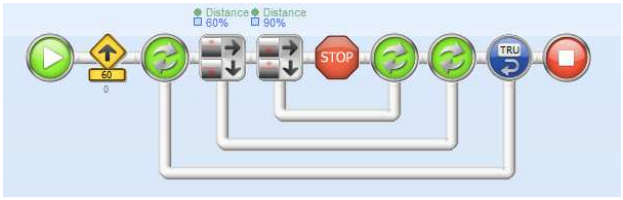
4. Explain what happens if you run this program and increase the sensor value in the test.



   Depending on the value, the robot may not recognize the colour yellow and the green light will stay on.
5. What will happen if you lower the sensor value?
   Depending on the value, it will detect the colour green and turn on the yellow lamp.
6. We are using a distance sensor. Explain how this program will behave.



   The robot drives forward slowly. If the distance sensor detects a value higher than 90%, the robot will stop.
7. What do the two sensor icons in this code do and under which circumstances do you use this?



This code checks whether a value is higher than 60% and lower than 90%. It is used to see if a value is between two limits.

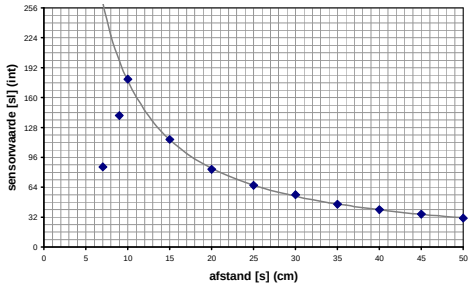
8. What happens in this program if an object is closer than 50%? And what happens if the object is further away?

If the value is less, the program will continue and try again. The program stops after 10 seconds. However, if the value is greater than or equal to 50%, it will wait until something changes, even if the 10 seconds have already passed.

9. How can you read the sensor values on a robot?
   Use the CHK menu. You will see a value for every sensor as you place the robot on different points of the field.
10. How can you display the sensor values on a robot during a running program?
    While a program is running, you can use the LCDDebugMessage to show the contents of 3 variables. You save the sensor value as a variable and display it on the screen.

Robotics Tests Ch 6 – Processing: the Reason Phase

1. Suppose, you have used the robot to read the following sensor values: **`Black 200, Green 280, Yellow 420`**. What value do you choose for black in the line-follower?
   *The value needs to be higher than 200, but lower than 280, so we select 240. However, as we prefer to work with percentages we need to calculate the value: 240 / 1023 * 100 = 23%*
2. What can you prevent the robot from losing the track in a tight curve?
   The robot can drive slower or you can increase the correcting angle.
3. Given the calibration data below for the distance sensor, which value must be used to stop near an object at a distance of 15 cm. Give the value as a percentage.



*According to the graph, the value at 15 cm is about 120. Although the graph shows a maximum value of 250, the standard maximum is 1023. So: 120 / 1023 * 100 = 12%*

4. Develop code to do the following:
   Drive around in a wide circle clockwise; turn until something appears in front of the robot. Then stop and turn on the blue LED.

5. Develop code to do the following:
   - *Drive forward; if an object is detected, drive towards it slowly and stop in front of it.*
   - *Stand still and show the value of the forward pointing sensor on the LCD.*

6. Develop code to do the following:
   - *Drive forward and as soon as the robot comes close to an object, slowly turn backward counter clockwise for 0.5 seconds and then continue to drive forward.*
   - *While making the turn, switch on the yellow LED.*

7. What colours do you need to calibrate for the line-follower with a single sensor?
   This depends on the set-up of the line-follower. If you only check for the presence of black, then black is all that you need. If you are testing for black and green, then you will need these two colours.

8. How do you perform an automatic calibration?
   Assuming you are using two colours (black and green) put the robot on the green part of the field and let it drive forward for a short while. It will then read the value for the colour green. Then, it will wait for a change in colour (at least 5%) and record the value of the second colour.
9. A line-follower with a single sensor checks for the line. If we only use black, how can you determine if you see the line or not?
   You can do this by comparing the calibrated value for black with a higher or lower value. Use the Margin property of the icon to make the comparison.
10. If you want the robot to drive on the other side of the line with a single-sensor line-follower, what do you need to change?
    You can make the turns the other way around. You can also reverse the program's logic.

Robotics Tests Ch 7 – Actuators: the Act Phase

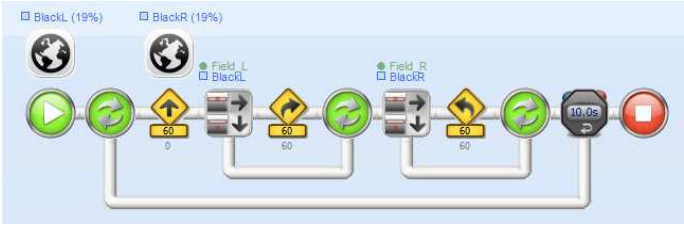
1. What is the difference between the numbers used in RoboPAL to control a normal electric motor and a servomotor?
   In an electric motor, the numbers defines the speed and turning direction; with a servomotor, the number represents a position.
2. How do you choose to use the third motor as servomotor on the simulator?
   In WorldView, select the Servomotor from the Head Motor property in the property box of the robot.
3. How do you select a servomotor in your program?
   Select a servomotor by using the appropriate motor icon.
4. What happens if the distance between the two light sensors is decreased?
   When following the line, the robot will detect the black line more often and therefore make more, smaller corrections. With sharp turns, it may detect the black line sooner with two sensors.
5. What happens when you increase the distance between the light sensors?
   The robot keeps on driving straight ahead much longer and makes corrections to its path through sharper turns. This means it can follow the line faster, but has to make bigger corrections in sharp turns.
6. What is the purpose of a stop condition in a line-follower?
   The line-follower must follow the line until a certain condition is met. Usually, this is the expiration of a certain time limit or the detection of a given colour.
7. Why do you have to make a separate line-follower to follow the yellow line?
   The line-follower must compare its sensor values with the colour of a line. These values are always higher than the surrounding field

with a yellow line, while with a black line these values are always lower. Therefore, the logic is reversed and cannot be processed by the same code.

8. The yellow line-follower has as a stop condition: the detection of the black line at the end of the yellow line. What can go wrong there?

   At the start of the yellow line, the right sensor still detects a part of the black line. This may make the line-follower believe that it has reached the end of the line. Therefore, the stop condition needs to be activated only after that first part of the black line has been passed.

9. This is a very simple line-follower, but it can get into trouble. What could go wrong?



   As each iteration of the loop starts with the robot driving forward, a correction is made as soon as the black line is detected. However, the delay that is caused by the two tests, while the robot is already driving, may cause it to lose the line in sharp turns.

10. How is this problem solved in the lessons?

    In the line-followers that are used in the lessons, the motors are only started after a test for both sensors is made first.

Part III

There are no specific tests for Part III. In this section, the main challenge for students is to develop a program of their own. The questions are the same as in the lessons.

Robotics Tests Ch 8 – Adaptive Behaviour

1. What is a state-diagram?
   A state diagram depicts the situations in which a robot can find itself and how the transitions from one to another state take place. It is a design for the logical structure of a program.
2. How clever is a robot? What kind of things will it not notice?
   A robot just reacts to its sensors, so it will not react to the size of objects or to sounds. Also it cannot distinguish between an object moving fast towards the robot of one that suddenly appears from the side.
3. What is the importance of an architecture?
   An architecture makes sure that you have a good overview of the design of a system and a clear layout of the components that need to be created, together with their interconnections. In our example, we first create the architecture and then its various parts.
4. What is a randomizer?
   A randomizer is a function that generates an arbitrary number. The numbers need to have a 'normal' distribution and there should be no repetitive sequences.
5. If a randomizer generates the following sequence of numbers: "1, 3, 5, 7, 1, 1, 1" is that an example of a good randomizer? Explain why.
   No, a good randomizer generates numbers that have a 'normal' distribution and therefore are not predictable. In this example, there is regularity and the second part contains a repetition.

Robotics Tests Ch 9 – Advanced Sensors

1. How reliable are the sensors on your robot?
   Not very. The reflection sensor can sometimes detect less than 10% difference between green and black. The distance sensor is more accurate (about 1 cm), but not over the entire range.
2. What factors may disturb sensors?
   The amount of ambient light influences the reflection sensor; however, the properties of the ink used for the rescue field are also important. The distance sensor cannot measure a distance of less than 3 cm. The battery level may also influence the behaviour of sensors.
3. How can you eliminate these disturbing factors?
   Calibration allows sensors to adapted to the existing circumstances. If we use a scale of measured values for every sensor between 0 and 100, it makes comparisons easier.

4. What does a robot need to measure the speed and direction of a moving object?
   To do this you need to take at least two sensor readings in succession and then calculate the difference between them. If the second measurement is higher than the first, the object is moving towards the robot, otherwise it is moving away from it. The speed is determined by the difference between the two readings.
5. Provide an example of a useful intrinsic (internal) sensor for a robot
   The value of the battery voltage, but this information is not available on the NXT as a sensor value. The pushbuttons with which we give commands to the robot also are a type of sensor.