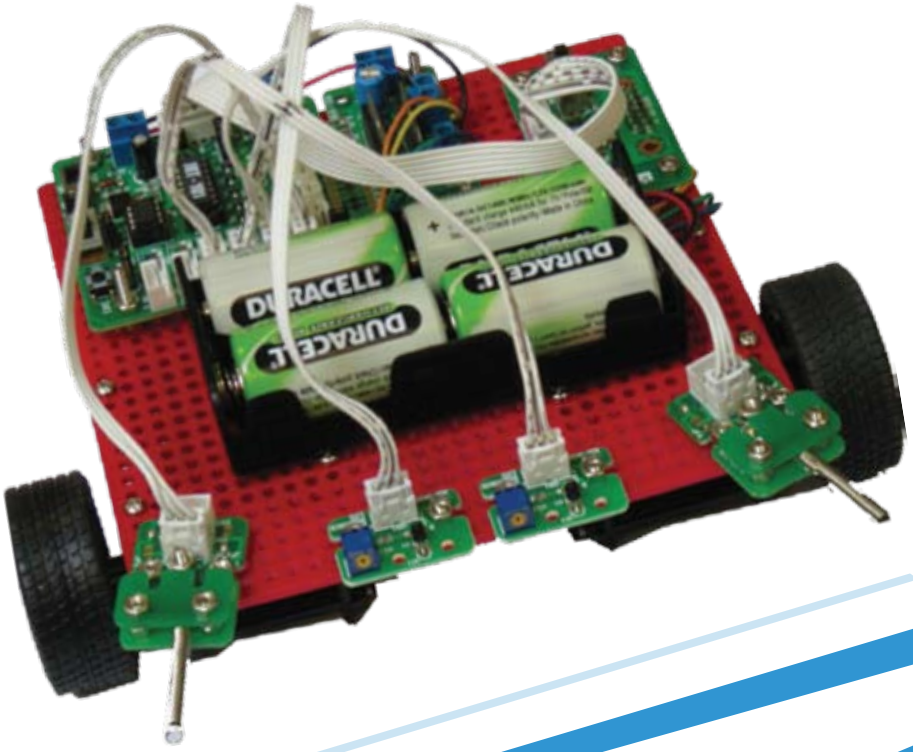
ROBO didactics



Education and Culture

Socrates
Minerva

Basic Course

Conceptual level

by

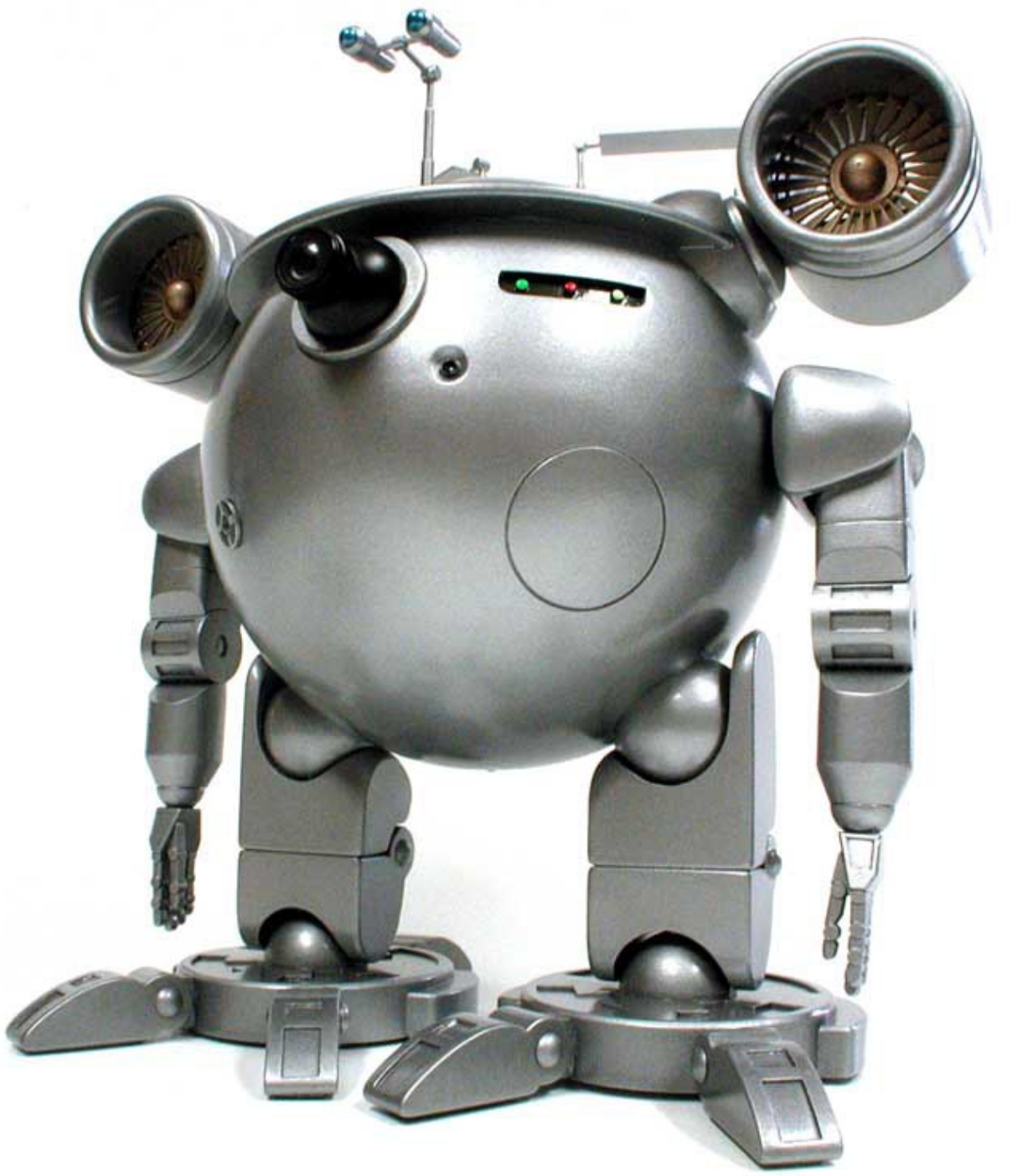
Peter van Lith

James Caska

Graphic design by Alessandro Romani,

Fondazione Mondo Digitale

Contents



Example of Robot - © by smallartworks.ca

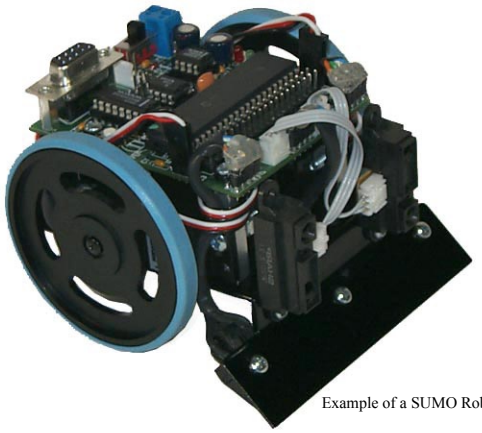
BASIC COURSE

Introduction

This introductory course is for YOU, students of 9 years old and older. It will guide you into the wonderful world of small robots, that is, little machines that you can give instructions to do things that you decide, like moving forward, moving backward, turning, etc. Robots can have many shapes as shown by the pictures below. In fact, you may have watched movies about robots and you may also have watched robot competitions on television.

Bumblebee - from "Transformers the movie"
© by Dreamworks™ SKG

ASIMO humanoid robot -
© by Honda Motors LTD



Example of a SUMO Robot

This course will teach you how to build small robots and then program them, that is, give instructions to them to do interesting things all by themselves. So these robots are not controlled by a wire or remote control, they have to do everything automatically.

We will show you, in a very simple way, how you can make your robot do these interesting things and you will learn how it is possible for small electric things to work **automagically**.

Once you have successfully completed this Basic Course, you will know you how your robot moves forward, backward or makes turns, reacts to a line drawn on the floor. You will also know how to make your robot push plastic cups outside a circle, drawn on a piece of paper. Once finished you are ready to start with the other courses and you will be prepared to participate in the exciting RoboCup Junior games Dance, Rescue and Soccer.

Course outline

This Basic Course will allow you to get started by teaching you about the robot and about the way in which you can program it with instructions using an entertaining, game-like, computer tool called roboPAL programming environment. It also gives you information about the sensors that you need to allow your robot to detect where it is. The sensors are little devices the robot uses to interact with its surroundings, like you use your eyes, nose, etc.

The course consists of a number of lessons, each treating a separate topic:

1. Introduction to Robotics
2. Making your robot move forwards
3. Detecting a dark area
4. Finding out other colors
5. A small game
6. The first line follower

Building and Programming robots

1. Introduction to Robotics

What you will learn

Many young children think that technology is difficult, boring and only interesting for nerds. Yet, almost everybody is using technology, for instance, by using mobile telephones, riding in cars, using computers, playing video games or using cookers and washing machines. At the same time, less and less people seem to understand how these devices work, nor do they seem interested on what is going on inside these ingenious devices.

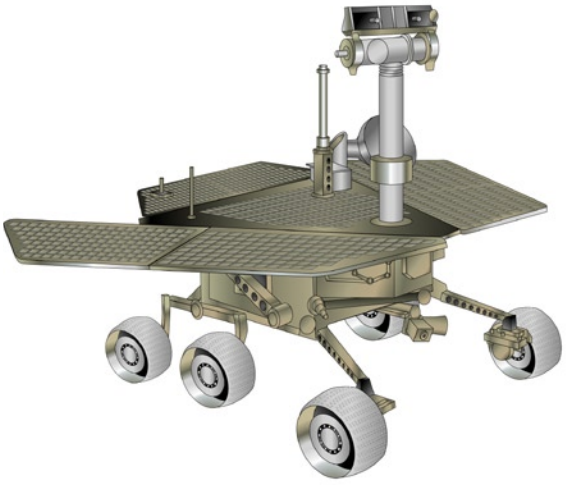
This course intends to show you how interesting and entertaining the world of technology can be. It will introduce you to the wonderful world of small robots, that is, little machines that YOU can program to do things that YOU decide, like moving forward, moving backward, turning, etc. You will learn some words such as robot hardware, software, icons and simulation and, by learning about robots; you will gain a better understanding of the modern technology around you. You will also realize that working with robots can be fun, entertaining and fruitful, since you will gain a better insight into what is required to make these devices work.

What is robotics

Robotics is the name given to all knowledge, activities and technologies involved in the research and construction of robots for all sort of purposes, for instance, toys, machines for industry, space explorers, etc. The actual physical robot is called **robot hardware**, while the program of instructions you give to the robot is called **robot software**.

Example of Robot Toy

Example of an Industrial Arm



The New Mars Rover
from National Science Foundation
www.NSF.org

In this course, you will work with small robots that are more like toys and you will have good fun while you learn something by playing with your robot. This is why the software that you will use in this course is called **roboPAL**, which stands for **Play And Learn**.

With roboPAL you will learn very quickly how easy it is to program a robot and make it obey your commands. For this, we will first give you a simple program that you will use to tell your robot what to do and then you will see on your computer screen, like in a game, a small picture of a robot that reproduces or simulates what your real robot will do once you load your program onto it. The performance of your program of instructions by the small 'game-like' robot on the computer screen is called **simulation**, because the real hardware robot is not yet involved and the 'game-like' robot acts as if it were the real thing.

Simulation with roboPAL is very useful because you can check very quickly if the program that you have created is doing what you expected before loading it into your real robot. In fact, you do not even need to have a robot available to work with simulations, so you can work at home or any place where you have a computer available.

Once the little robot in your computer does what you want it to do, you can then load your program into the memory of your real robot and you can watch it do the same things as the simulated robot on your computer screen.

Programming with roboPAL

Programming is telling your robot (or the little computer inside it) what to do. There are several ways in which you can do that and with roboPAL we have made it very simple for you. You will start with a number of building blocks that allow you to make simple programs very quickly. This helps you getting started really fast and once you have your robot doing what you want, we will gradually show you how to make your program better and more interesting. In this way you will learn to do more complex programs without much effort.

With roboPAL you will be able to get you first working robot program in just one lesson. And from there you will learn to program it do all kind of exciting things. We will also explain how things work, so you will be able to write ever more complicated programs later on. This Basic Course currently concentrates on roboPAL only.

You need a Robot

In this course we use the RoboDesigner RDS-X01 robot produced by RoboTech Japan or a compatible robots like the JelloBot or the JoBot Junior robots.

Before you can start the lessons you need to build a hardware robot, although you can do all examples with the roboPAL simulator. Please note that there are always small differences between a simulated robot and the real robot, so you should try out all examples wit a real robot too. How to build your robot is described in the documentation that you received with the robot you are using. Because the robot consists of many small parts, you are able to give your robot its own looks, thus making it different from robots from other students.

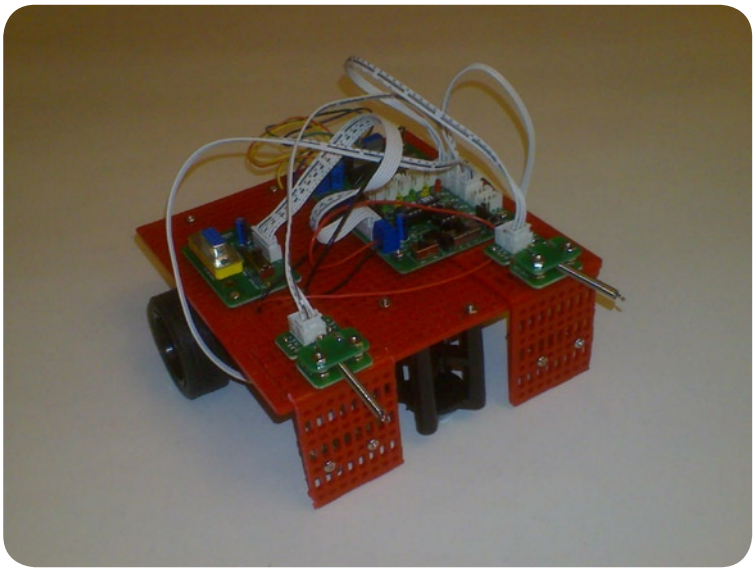
The RoboDesigner RDS-X01 robot

This robot is the standard robot used by the RoboDidactics courses. It comes as a building kit and to use it for roboPAL you first need to build your robot or use one that has already been assembled. Please follow the instructions that came with your robot and build one for

the activities you want it to perform. In this basic course we will be using a robot with two motors and at least one light sensor, facing down, so it can detect color changes on the floor.

You may program you RDS-X01 robot with the standard TiColla programming language. TiColla has no simulation capacity and its way of programming makes it necessary to think clearly about **what** you robot needs to do and **how** you are going to accomplish that. In this course, we will show you a simpler way which uses roboPAL and concentrates mostly on **what** you robot needs to do.

After you robot starts doing what you want you will be ready to learn more about **how** a robot actually works in more detail in later lessons. Without the need to know all these details, you will be using your robot very quickly.



The RDS-X01 robot has a small processor on board that stores the program that you develop. The standard language for the robot is TiColla but a special version of roboPAL is provided for the RDS-X01 as well. However, since the processor in this robot is limited, not all features of roboPAL will be able to run on this robot platform. A plug-in extension board is available that allows the processor to be upgraded and use the full potential of both roboPAL and the underlying Java programming language as well.

You also need some Software

Just having a robot is not enough. You also need a computer and a program that allows you to tell the robot what to do. The programs that you can select in this course are either in TiColla software or roboPAL software.

Your robot contains a small computer that can only understand some very primitive commands like starting a motor, turning on a small light, giving a beep, or add two numbers.

Making a program that does what you want with these commands is a lot of work and rather difficult.

So on your computer there are other programs that allow you to tell the computer what you want your robot to do and that program then translates these commands into a series of instructions that the small computer in the robot understands.

You will only have to deal with the programs that allow you to tell the robot what to do and not with the more difficult instructions for the robot's small computer chip.

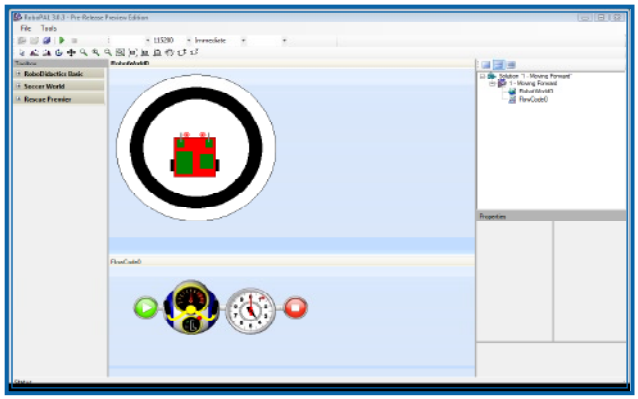
In fact, you will be working with roboPAL, which allows you to work with small pictures of what you want your robot to do, like the 'driver' that tells the robot to move forward, backward or sideways.

These small pictures are called **Icons** and they are like little paintings that use simple symbols to show you what the robot is supposed to do. In practice, icons are like a language that should be easy to learn and remember.

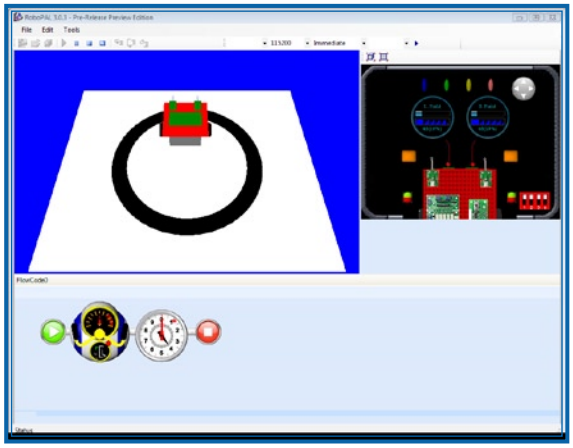
Another way to program you robot is to use TiColla, the software that comes with the RDS-X01 and that uses much more detailed and little more difficult to use instructions.

We will not be working with this language in the material of the Basic Course. Instruction manuals that come with the RDS-X01 robot are available for this purpose.

With roboPAL all you need to do to create a program for your robot is to put some icons together that tell the robot what to do and then roboPAL will show you a moving picture on your computer screen, simulating what your real robot will do when it runs the program that you have made. As we saw earlier on, the small moving picture is called a simulation, because it shows a reproduction of the behavior of your robot and acts as if it were the real thing.



In the picture, on top of the central column you can see your simulated robot, while the program is shown underneath using some dials that indicate what the sensors 'see'. Later on we will explain what these values mean and how you can use the robot's sensors.



When your program is executed (by pressing the run button) , a new picture appears in which you will see your robot moving according to your program On the right you see the values of the sensors and are some controls to activate the touch sensors. While the program is running you will see an arrow that points to where the program is currently active.

Programming your Robot

1. Making your robot move forward

R2 D2, from the movie "Star Wars" - © by LucasFilm

What you will learn

A robot needs to be told what to do. You are the one giving the instructions in the form of a number of steps the robot will follow. If you remember, this series of steps is called a program and making a program is called programming … and YOU are the program.

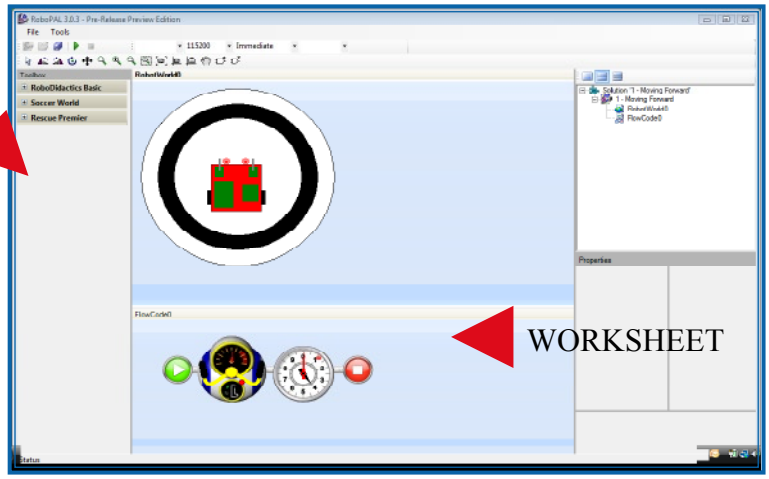
This course will show you how you can program your robot using roboPAL and we will start with a simple program that will make your robot move forward. But, first, to make the program work, you need to define a couple of things like on what field the robot will play and whether it needs any objects like a ball.

We will show you how to define a playing field and a robot in roboPAL. Then we will show you how to create a very small program and then run it inside the simulator.

Our first program will make your robot move forward for a fixed time of 4 seconds and then stop.

Programming with roboPAL

- Select from the File menu of roboPAL the 'Open' command and load 'Lessons\1 – Moving forward'
- You will get a filled-in worksheet and a Toolbox plus a number of other screens.



You use the File menu to load the demo program and after that we will make some small changes that will make the robot do what we want. Above you see a simple program and a screen on which you will see your robot move. You will use the Toolbox to make some small changes to the program.

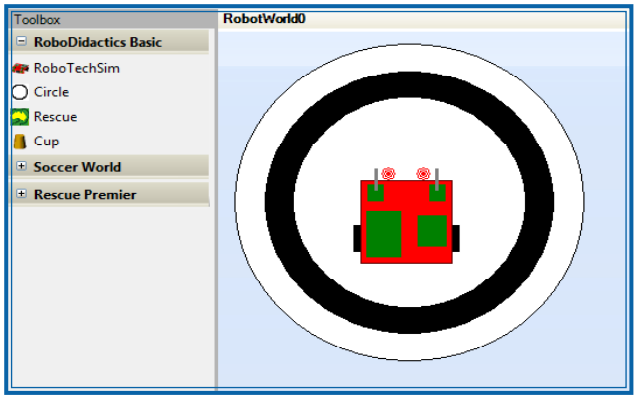
The components of a roboPAL program

A roboPAL program consists of a number of components that you will use to program your robot. For now the most important parts are the two worksheets called the World and the FlowCode and, then, the Toolbox that belongs to each of them.

The World

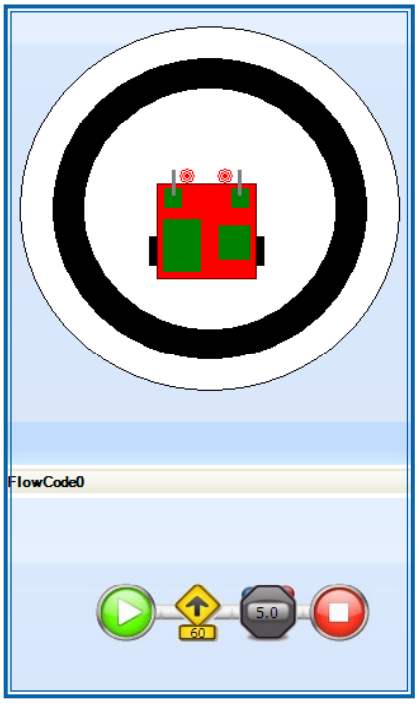
In the World worksheet you see a circular field on which the robot is placed.

On the left-hand side you see the Toolbox on which you will find the icons for several robots, playing fields and objects you can put on these fields. With these you can create your own world and in the example you are reading that has been done for you already.



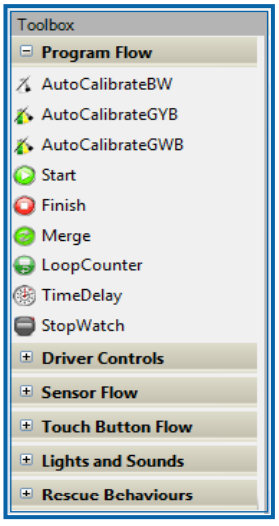
In the Toolbox of the World worksheet you may select the type of robot and the various fields and objects to be placed on the field.
In our first example we will use the white circular field, together with the RDS-X01 robot.
If you are using a different robot, you will have to replace the existing robot with the one you are using. As long as you are not using the real robot however the simulator will work fine with any type of robot.

The FlowCode



The second worksheet is the FlowCode worksheet seen here underneath the World. There is actually a third one, but we will not be using this one for now. The FlowCode worksheet is used to construct the program with icons that are selected from the toolbox. This toolbox contains different icons from those of the World worksheet.
You can select an icon from the Toolbox and place it on the worksheet and connect it with other controls already on the worksheet.
We will always start with the green "start" icon and ends with the red "stop" icon. Your program commands are placed in between and are tied together with the small 'tube-like' connectors.

The Toolbox



We use the Toolbox to select the icons to be placed on the worksheet.

Most icons have a short line (i.e., hyphen) next to them. If you press it you will see a list of icons representing the various **controls** that are available for the type of robot you have selected. They are called **controls** because they allow you to instruct and control the behavior of your robot. In the example, 'Program Flow' has been selected, showing nine controls.

In these first lessons, you will use a limited number of controls.

Selecting and moving controls

When you place a control on a worksheet, you will wish to move it to its logical position. To do this without modifying the control you should use the four-pointed arrow shown in the horizontal menu above. **Remember this four-pointed arrow allows you to move the control to another location and does NOT change anything.**

On the other hand, if you want to make a change to one of the controls, then, you should use the cursor arrow on the right-hand side of the horizontal menu.

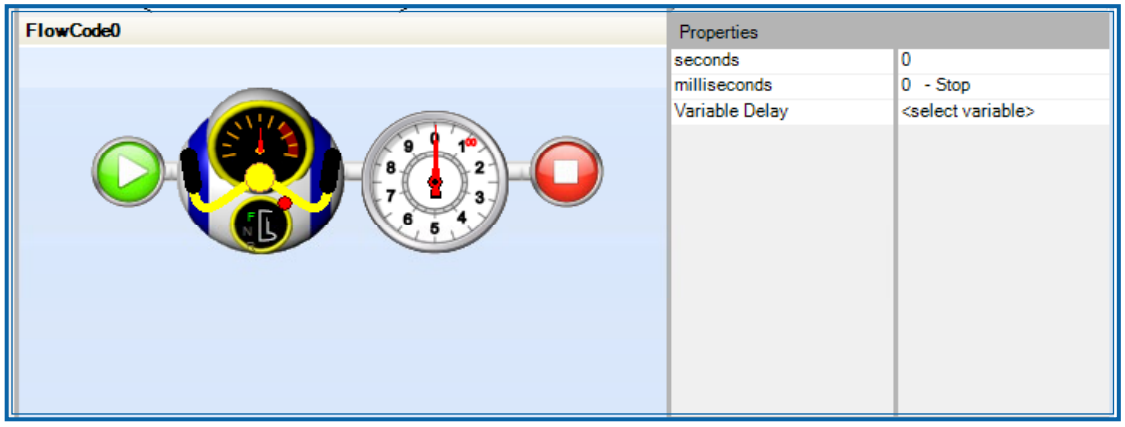
As examples, if you select the 'Driver' icon with the cursor arrow you will see that you can move the yellow steering wheel in various directions. The same is true for the speedometer or the dials of the stopwatch control next to it.

Thus remember, the cursor arrow makes changes to the properties of a control. If you do not wish to do this, selecting the cursor arrow is not good since it may accidentally make changes. For instance, if you click on the steering wheel it may change its position.

The special cursor with the circle is used when we want to rotate the control. It is generally used only to rotate the robot or other objects like cups.
We will now make a slight change to the small program that makes the robot moves forward for 4 seconds and then stops.

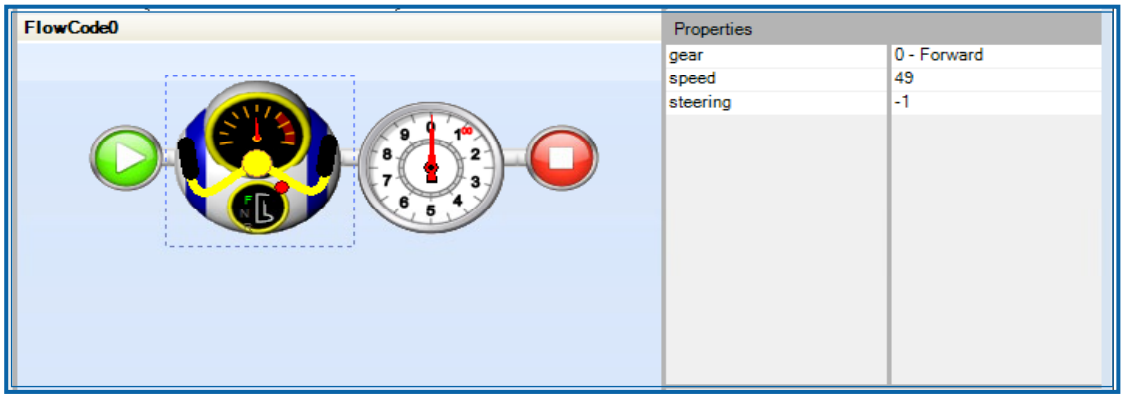
Modifying the program

If you want to make changes to the program, you first select a control and use either the cursor arrow or the values of the control shown in the **property box** that appears on the left-hand side of the screen as soon as you select it on a worksheet. The **property box** as it names implies contains the properties of the controls, since each control is defined by a number of properties. If you wish to make changes to the values of the properties, you can do it here.



For example, you can see that in the picture the stopwatch is set to zero and you need to change this. You wish the robot to move forward for 4 seconds. You have two options: you can change the black dial using the cursor arrow or you can change the value of **seconds** in the property box. To do this, you first select the stopwatch control and make sure the properties of this control are shown. If you are good with the mouse you may set the dial swing it, otherwise just change the value of seconds from 0 into 4.

Since you are working only in seconds, make sure that the milliseconds are set to zero. Milliseconds are only used for very brief amounts of time like half a second or shorter. Short periods are especially useful with dancing when you have to match the timing of the music.



If you wish to change the position of the yellow steering wheel under the black dial from its middle position, then, select the steering wheel with your mouse and move it around. You can set the direction and speed either with the dials or using the numbers in the property box. You simply select the 'driver' control with the steering wheel and the property box appears on the right-hand side of the figure. You see the items gear, speed and steering and you can set the gear to Forward, the speed to 50% and the steering to zero so it drives forward.

You will see that the position of the speed dial changes. This also works the other way. If you change the speed dial using the mouse, the values of the speed property on the left will change.

The same applies to the steering wheel. As you change its position, the direction of the robot changes. Setting the steering wheel in the middle position makes the robot drive straight ahead.

Wiring

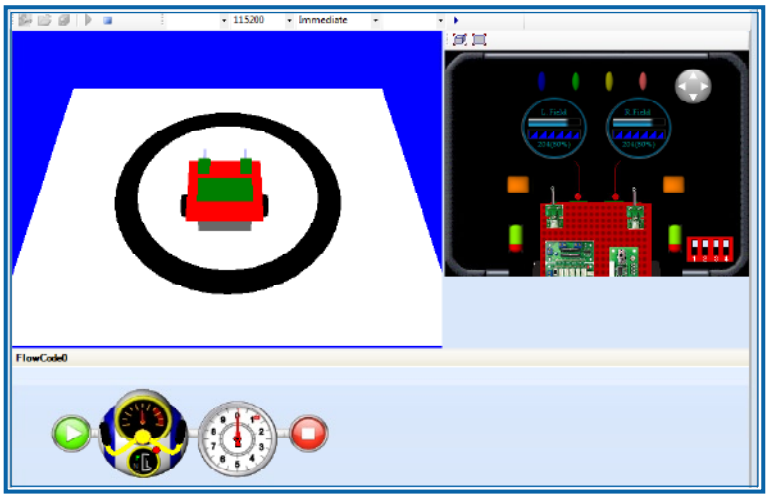
To make the program execute all steps in the proper sequence, you need to connect the controls to each other. Each control has two connectors that you can use to connect them. When controls are not connected, the program will stop since required steps are missing and cannot continue.

Wiring controls is generally the source of most problems, so we have made wiring very simple by directly connecting controls together. Later on when our programs get a bit bigger, we will need more facilities to change the wiring but for now we keep things simple. Your program is now ready to be uploaded to be tested.

Running your program

To test whether your program works, you do the following:

Select the little green triangle that indicates RUN and press it.



You will now see your program run inside the simulator.

If all goes well, the robot will start moving forward and will stop after 4 seconds.

The arrow shows where the program is at any given time and it will stop when it reaches the finish icon.

Further exercises

When this works well you may try the following additional exercises:

1. Try to run backwards
2. Try to make a turn
3. Make the robot run faster or slower
4. Try to make your robot run inside the circle.

2. Detecting a dark line

What you will learn

Just making a robot drive forward is not very interesting. We really would like the robot to react to its environment. People and animals use their eyes, ears and feelers to sense what is going on around them. Robots do this with sensors that detect light and darkness, sound, or a bump when they drive into something.

Of course, the robot will not be able to distinguish, for instance, between dark and light if we do not tell it exactly when something is supposed to be light or dark. Looking at something in bright sunlight for instance will give a different idea about a color than when you are inside a cinema, where everything is dark and a flashlight suddenly seems to be a very bright light.

In this section, we will look at differences in light levels and we will instruct the robot to react when it sees a change in light level. In this way, we need not define exactly what we mean with light and dark, we just say that when we see a large difference, something has obviously changed.

But does this solve the problem? When is a difference large and when is it small?

Using sensors

Let's start by working with a light sensor. A light sensor sends out a beam of light and receives the reflected light back in its foto-sensor. The strength of the reflected light is measured and depending on the color of the surface that reflects the light, more or less light will be transmitted back to the sensor. For example, a black surface reflects less light than a white surface. This fact allows the light sensor to differentiate several colors.

As an exercise, you will program your robot to move forward until it finds a darker area, where it will stop.

The program you use is similar to the one before, only now you use a light sensor to determine if the robot must stop. Depending on the value of the sensor, the program will continue until a darker surface is detected.

To determine which value to use, you use the 'Property Box' in which the level of the light sensor is defined. When using a white surface and a black line, the difference in light levels will be large and most values will work.

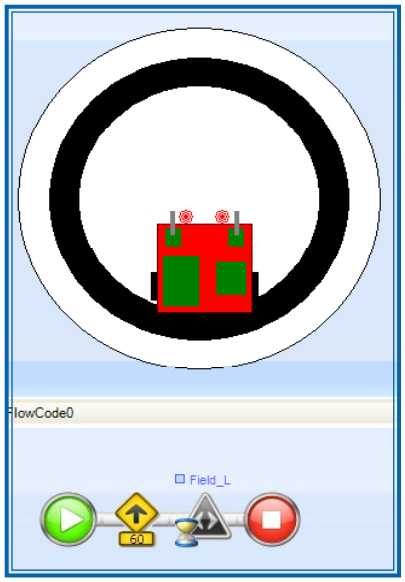
To test the program you again use the built-in simulator of roboPAL. In the circle field you see a black line. The robot starts in the middle and drives forward until the sensor detects the black line.

You see that the **Driver** control goes again forward at half speed. It then reads the sensor continually while driving forward until it sees a darker area and then executes the next step, which is the "stop" command.

Because a white surface reflects more light than a black surface, the value of the sensor gets lower over a dark surface.

As soon as a darker surface is detected the program stops and so the robot also stops. In roboPAL stopping the program also stops all motors.

Sensors

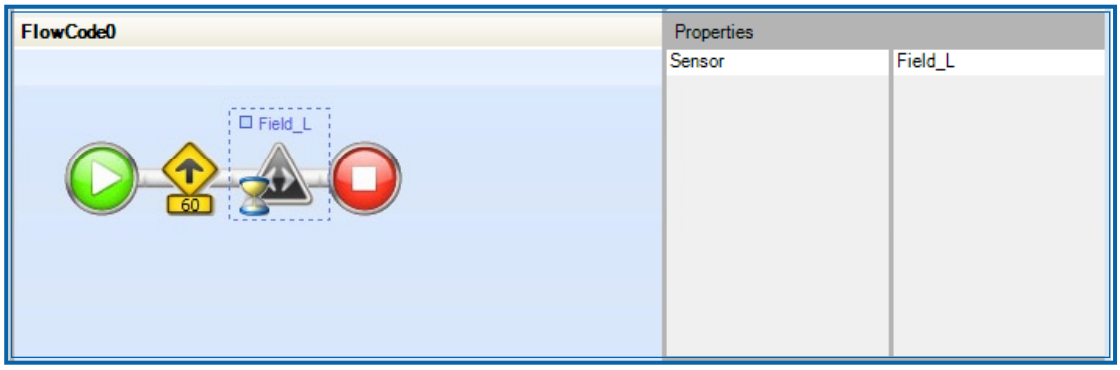


Sensors can be used in two ways generally. In the first way, the amount of light is expressed as a number ranging between 0 and 255. The higher the number the greater the light reflected.

The second way (the used here) is to take the value and check if it is lower or higher than the value before. So in our case the **Wait Until Darker** will continually read the sensor until it sees a darker area and then continues with the next step. This is like the stopwatch which waits for a certain time, this one waits until it sees less light.

Property Box

To use the property box you first select the icon in the program. This will show the control's parameters (the various properties are also called parameters). Above the control you will see "Field_L". This is the name of the sensor used in this example.



A property box contains a number of fields and the properties or parameters that can each have a value.

The **Sensor** field indicates the sensor being tested. You do not need to define a value here, because you only are looking for a change. The first field also shows the name of the sensor used, in this case the **Field_Left** sensor. Each sensor has its own name. For now we only use this **Field_Left** sensor.

Further tests

Once your program works well, you may want to try the following:

Change the speed of the robot and the width of the line. For instance, if the robot moves quickly and the line is very thin, you will see that the program may not have enough time to detect the line and will skip over it.

3. Finding out other colors

What you will learn

Detecting a difference in light levels is a good and simple way to react to changes in the environment. Animals do it all the time. When an insect sees a shadow coming up above it, the difference in light level signals danger and then it flees.

But suppose we now have different colors and we would like to react to a specific color. For instance, in the Rescue game, we have a playing field with a black path, a green field and a yellow path and a yellow swamp. How can we make our robot detect the differences in color?

The simplest way is to let the robot detect these colors automatically. We will show you how the difference in light level can be used to teach the robot the values for the various colors. We simply let the robot see three colors and make it remember the three colors, so it knows exactly what light level to look for.

Why is this important? Well every time you move your robot to a different place, the light level in the room may be different. As a result, the values that the robot detects for the colors may vary every time something changes in the environment. It may be a dark day or a very sunny day. You may be working in a dark area or in a room full of bright lights. In all these circumstances the colors that the robot will read are different.

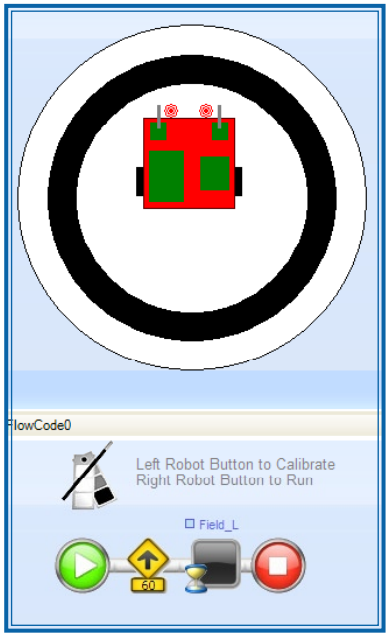
To make things worse, all sensors that we use are slightly different. No two sensors are the same. So if you would tell the robot how the color yellow looks like, this would not be useful for another robot, because that robot's sensors will be different and a different value would be needed.

We will program our robot in such a way that every time we start, we first show it the three colors, yellow, green and black and let the robot 'memorize' these values. This process of going from light to darker colors is called **calibration** and is a very important topic in robotics. Our robot will do it 'automagically' and in later lessons you will learn what actually happens when calibrating a robot.

Showing colors to the robot

What we are going to do is to make a program that makes the robot drive over the calibration field. This is a field that contains all colors.
We want the robot to detect a change in color and then remember the value of the light sensor, so it 'knows' what this color looks like. We must make sure that the colors we show are always in the same sequence. Here we assume that we are going from light to dark.
The robot drives over the field, looking for a change in light level. When it finds one, it registers the change in a variable that indicates the color. Once this is done, we can use these recorded values in the remainder of our program.

Calibrating the robot



So, before you can run a program that needs to know about the colors of the field, the robot needs to find out about the colors. For this you use the AutoCalibrate icon that will drive the robot over the field and records all colors. Once it has read the last color it will stop and wait for about 10 seconds.

If the robot cannot find the colors that it expects, it will stop and either blinks its lights or beep to signal that an error has been found. Then, it will not start the program.

If the calibration went OK, then the actual program will start. In this program you will see that the robot is first calibrated and then it waits until it sees the color green. That happens when you place the robot in the middle of the field. It then drives until it finds the color black on the green field.

Moving the robot around

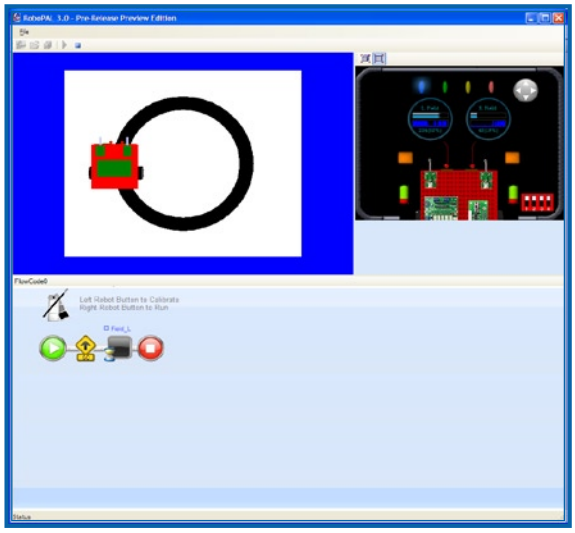
After your robot has calibrated itself it will stop and wait until it detects green.

The autocalibration field has a white part as the last field. Here it stops.

You now need to pick up the robot and place it in the middle of the field. You can do this in the simulator just by dragging the robot to the middle of the field, using the mouse.

Looking at the sensor values

While you are dragging the robot over the field, notice how the sensor values in the control panel on the right change value as they pass over the different colors. These values are the values that the program stores when it performs the calibration.



Further exercises

Once you got this working, you may try some other exercises like calibrating for another color line like gray or green.

Repeating program parts and Loops

4. A small game

Lego Mindstorm Robosapiens

What you will learn

Now that you have learned how to make the robot move forward and backward and how it can detect colors, it is about time to put your knowledge to use and give you a real challenge. What we are going to do is put a number of cups on a tabletop within a circle with a black line around it.

Your robot will have to drive back and forth criss-crossing over this field and push all cups out of the circle. How are you going to do this?

The technique you are going to use is that of **program loops**, where you use the same small program over and over until the task is done.

A program that uses jumps and loops can make exceptions to the normal sequence of the program that you have followed so far and allows you to program more complicated tasks. In this challenge you will need a table with a piece of white paper on it and a black circle of about 3 cm width. You need to place a number of plastic cups on

top of this table. These cups will be moved outside the black line by your robot. The simulator already contains such a field and you can program it using an example program.

We will provide you with a criss-cross driving program for your robot. You need only to tell it how wide the field is, so your robot does not go over the lines. We do this for additional safety, because the sensors are programmed to detect if the robot is going over the black line.

The Challenge

This program will instruct the robot to move forward and turn away a little bit when it reaches a darker area and continue to do this for about 10 times.

The program is used to do a fun table clearing challenge. It can also be used later in the Rescue challenge to scan for, and eventually rescue, the victim.

Tips

- If the robot arrives at the dark area on the field, it will start driving backward for a short period of time.
- Getting back in your program is always done using some kind of arrow, pointing back to an earlier step in your program.
- You can only connect with a single arrow.
- The program will repeat the cycle for 10 times or until you stop it manually, or, by switching off the robot.

Outline

You are now going to make a program that uses everything you have learned so far and will clear the cups from the table. You use the calibrated value you found earlier to determine the value for black.

The robot drives forward until it detects the black line and then drives backward for five seconds. It then makes a slight turn for 2 seconds and repeats this process 10 times. We assume that by then the robot have covered a large part of the field.

This exercise is very useful to use in the Rescue competition to scan the swamp for the victim.To perform criss-cross driving, we provide you with a special version of the Driver icon that does the back-and-forth movement by itself. You will need to tell this driver, using the properties, what speed it uses, how long it will drive backwards and what angle it needs to make when it reaches the black line. Every time it starts, it will drive backward for some time, make a slight turn and then move forward until it finds the black line. This is repeated 10 times.

In this program, **wiring** is used for the first time, other than the short connectors between the controls that we have been using so far. Wiring is selected from the icon bar by first selecting the Wiring tool cursor that you see underneath.

With the wiring icon next to the four-pointed arrow you can draw lines. Try to make the wiring for this program yourself now.