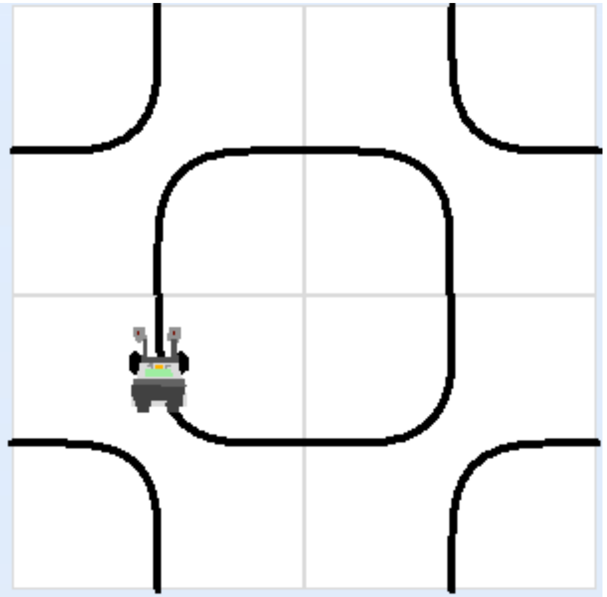
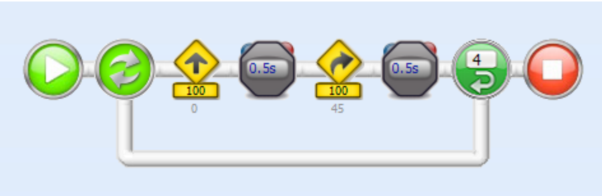
1. The First Line-Follower

How to make your robot follow a pattern



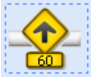
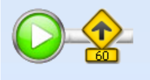
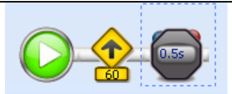
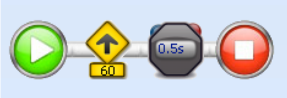
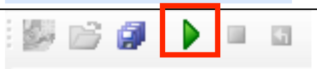
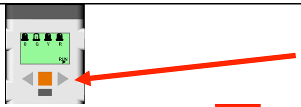
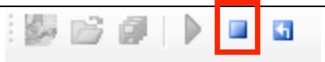
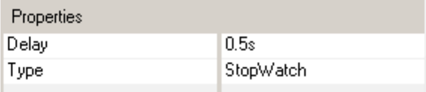
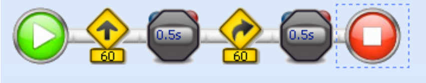
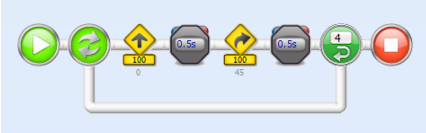


If you want your robot to follow a fixed pattern, you can try to describe precisely what the pattern looks like. With the pattern on the left, most people will start to make the robot move forward for a short while, then make a turn and then move forward again, until it is back where it started.

The program to do this is simple and does exactly what you would expect. We will develop this program and then test it with the robot to see how well it works. This way of working is very natural, but it has a number of disadvantages:

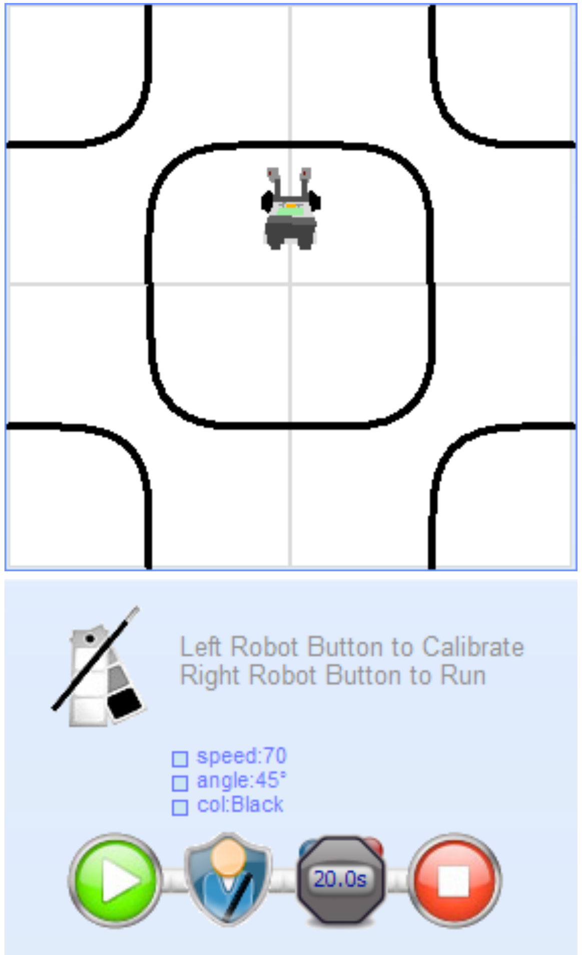
1. You always have to start at exactly the same place. If you accidently start a few centimetres too far from the beginning, the robot will make the turns at the wrong place.
2. If the batteries are running low, the robot will slow down and the whole pattern will be misaligned.
3. If there is something on the field and the wheels slip, the robot will lose track and not be able to follow the pattern.

We will develop this program first and then see if this is a better way of following a pattern.

1	 RoboPAL4NXT.lnk	If you have not started the program yet, double click on the RoboPAL icon on your desktop. Otherwise go to step 2.
2		Select 'New' and double click on 'QuickStart New Square World Project'. Then, click on the lower window with the green and red icons. This is the 'FlowCodeSheet' in which you will develop your program.
3		Select the 'Driver Controls' from the left column (Toolbox) and click on 'MoveStraightAhead' with the left mouse button.
4		Move the mouse pointer to the 'FlowCodeSheet' panel and drag the icon next to the start icon. Click on the left mouse button to leave it there.
5		Select the 'stopwatch' from 'Program Flow' and place it in the 'FlowCodeSheet' next to the previous icon.
6		Place the Stop icon at the end. Your first program is ready.
7		Now, press the green arrow button in the icon bar. This starts the simulator. If the green button is not enabled, first click somewhere in the World panel.
8		Press the grey triangular button under 'Run' and watch the robot move over the field.
9		Return to design mode by using the blue rectangle on the icon bar.
10	 Properties Delay 0.5s Type StopWatch	Click on the stopwatch and select 'Delay' (on the right side). Change the time in 'Delay' and experiment with it on the simulator to make the robot move for a longer or shorter period of time.
11		You can let your robot move more as long as you use the stopwatch to tell it how long every movement has to be.
12		If you want to follow the entire square, just add more movements. You can also add a LoopCounter to execute the movement four times. This will make the robot return to its starting point.

2. A Better Line-Follower

Using a sensor to find the line



Left Robot Button to Calibrate
Right Robot Button to Run

□ speed:70
□ angle:45°
□ col:Black

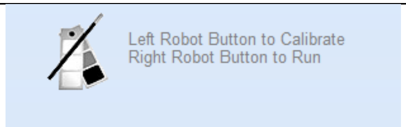
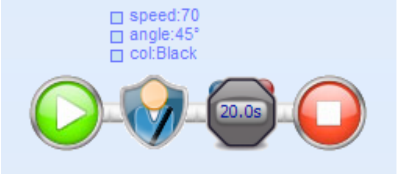
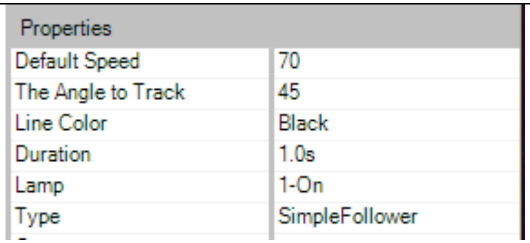
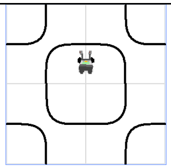
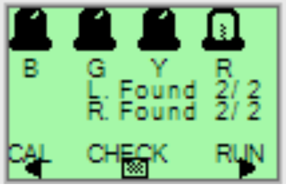
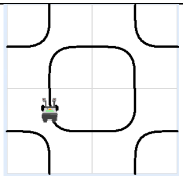
Following a pattern by giving the robot precise instructions looks like a simple solution, but the robot will very quickly lose track of the pattern.

So we have to come up with a better way to follow a pattern. We can do this by using the robot's sensors to find the line and follow it.

We will start with a simple line-follower that keeps the line between its two sensors. When either of the two sensors detects the black line, the robot will move forward in the other direction.

However, the robot first needs to know how to recognize the colour of the line. To do this, we use a calibration program that makes the robot move forward and remember the colour of the line that it detects.

The robot will then be ready to follow the line and you can tell it how fast to move and how sharp to turn to keep track of the line.

1		You are going to change the program from the previous lesson.
2	 Left Robot Button to Calibrate Right Robot Button to Run	First, the robot must find the black line and remember what a black line looks like. It must find out how dark the line is compared to the white field and remember this. This is called calibration. Choose the AutoCalibrateBW (Black/White) icon from Sensor Flow to make the robot detect black and white. The icon for the auto-calibration programme is placed on the top line.
3	 □ speed:70 □ angle:45° □ col:Black 20.0s	Remove everything under it between the start and finish icons. Choose the SimpleFollower icon from Program Flow. Do NOT use any of the other line-follower icons, as they are more difficult to use. We will use a stopwatch to determine for how long the line-follower must follow the line.
4	 Properties Default Speed: 70 The Angle to Track: 45 Line Color: Black Duration: 1.0s Lamp: 1-On Type: SimpleFollower	You can change the Speed in the properties of the SimpleFollower. A second field, the Angle, tells the robot how sharp to make its turns. If you make the turns too wide (low Angle), the robot may not be able to follow the line, while if you make the corners too sharp (high Angle) then it may start to zigzag (just like it happens with your bicycle). The value of the Angle can vary between 0 and 100.
5		First, test your program on the simulator. The first step is the calibration process. Start the simulator and make sure that the robot is at the spot shown in the picture. Now, press the triangular start button of the NXT on the simulator. You will see a menu appear on the screen.
6	 B G Y R L. Found 2/ 2 R. Found 2/ 2 CAL CHECK RUN	Choose the left option: CAL. The robot will start to move forward and use its sensors to detect the line. It will remember this and if everything works correctly, you will see the line on the screen, together with the number of colours that the left and right sensors have found. For both sensors, this should be the colours black and white. If you see 2/2, it means both colours were detected so everything went OK. Otherwise try again.
7		The robot has now been calibrated and you can start following the line. Place the robot on the spot with your mouse and select RUN from the menu. You will see the robot follow the line. Experiment a bit with the Speed and Angle values, to see what the robot does if you take the corners more or less sharply.

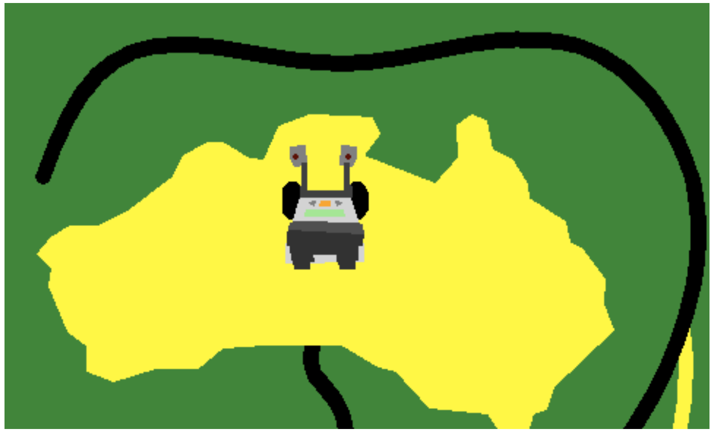
3. The Rescue Auto-Calibration

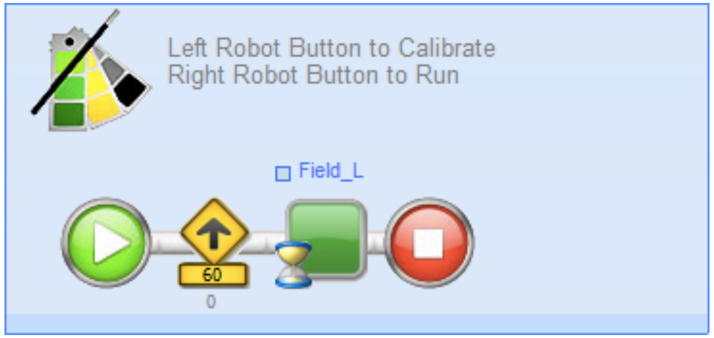
Discovering colours

You have seen how the robot can follow a line. The robot needed to explore the field to see what the line looked like. This process is called calibrating the sensors. On the green Rescue field, calibration is not only more difficult, because we have more colours, but in order to work properly, it is also important to perform the calibration at the right spot on the field.

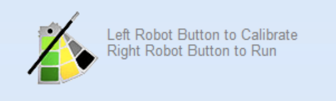
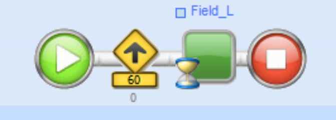
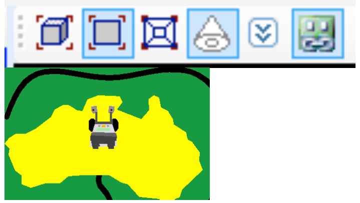
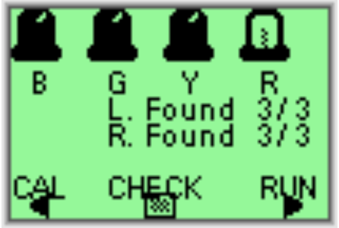
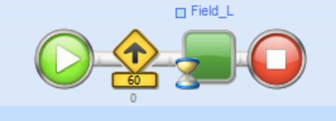
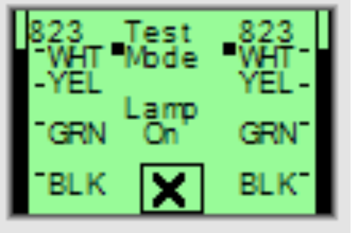
In the rescue mission, the robot needs to follow the black line until it reaches the yellow area. The yellow field is a swamp in which there is a container that will explode (not really) if it comes into contact with water. Your robot must get to the swamp as quickly as possible, search for the container and then push it out of the swamp.

There are three different colours on the Rescue field, which makes following the line more difficult. The robot has to know what the yellow area looks like, but it must also be able to recognize green and black to follow the black line. Place the robot on the yellow area and make it move forward, so that it detects yellow first, then green and then black. The calibration program will remember these colours and use them to find the road and the yellow swamp.



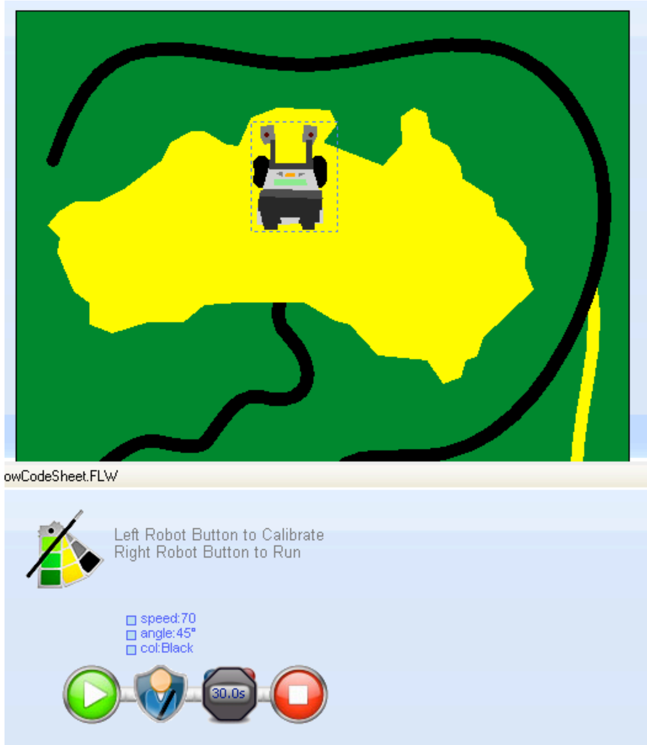


Left Robot Button to Calibrate
Right Robot Button to Run

1		To make the new programme, first select the ‘Close Solution’ option from the ‘File’ menu. Then select ‘File’ - ‘New’ - ‘QuickStart New Rescue World Project’.
2	 Left Robot Button to Calibrate Right Robot Button to Run	The ‘AutoCalibrateGYB (Green-Yellow-Black)’ icon from ‘Sensor Flow’ that calibrates the three colours is already part of the program. We just need to make a simple program so that it will recognize the colours.
3	 Field_L 60 0	Select ‘MoveStraightAhead’ from ‘Driver Controls’. Then, choose ’WaitForCalibratedGreen’ from ‘Sensor Flow’ and place it after the driver control. Complete the program by placing the Finish icon next to the green icon.
4		Start the simulator and place the robot on the yellow field that is supposed to be a swamp. The robot will move forward and start to detect the colours. If you put the robot too far in, it may miss a colour. Calibration is very important, so put your robot at the right place, so that it will always find all three colours while moving forward. To clearly see where you place the robot, it’s best to look at the field in 2D mode (using the second icon directly above the control panel of the NXT).
5	 B G Y R L. Found 3/3 R. Found 3/3 CAL CHECK RUN	Calibrate the robot by pressing the CAL option. If everything works correctly, the robot will have detected three colours. It is very important to put the robot at the right point. If your robot is too far back, it will miss the black road, only detecting two colours. If it is too far to the front, it may miss the Yellow. Also note that the light in the room can influence the results. If there is a lot of light coming from outside or a strong lamp, the sensors will not work as expected. If the lighting conditions in the room are not uniform, it may be necessary to calibrate the robot at a different point on the field. Always make sure that each sensor on the robot has detected all three colours.
6	 Field_L 60 0	Put your robot on the yellow area and press RUN. The robot will start moving forward and should stop as soon as it detects green.
7	 823 Test 823 -WHT ■Mode ■WHT- -YEL YEL- Lamp -GRN On GRN- -BLK ☒ BLK-	When things go wrong during these lessons, it’s often because the calibration was not performed correctly. And even if the robot did detect all three colours, if the lighting conditions change when it begins to move over the field, it may still lose its way. You can double check which colours the robot has detected by using the middle CHECK button. This will show you how the robot sees the colours on the field after calibration. Put the robot at any point on the field and see if recognizes the colour. If not, the calibration process must be repeated. The sensor values for the left and right sensor (in this case 823) are shown on the screen together with the black bars indicating the signal strength. Moreover, if the robot can detect a colour it will place a black dot next to its code: WHT, YEL, GRN and BLK. In this example, the robot recognizes the colour WHT.

4. The Rescue Line-Follower

Following the black line



owCodeSheet.FLW

Left Robot Button to Calibrate
Right Robot Button to Run

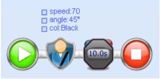
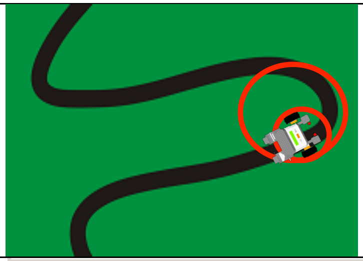
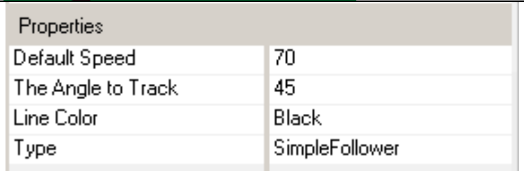
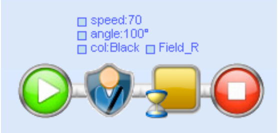
□ speed:70
□ angle:45°
□ col:Black

We have seen how a robot can find a line and remember the colours on the field. Now, the robot has to follow the black line. This means we must make some changes to the program.

We want the robot to follow the line. In this program, this lasts 30 seconds, but you can also make it move for a longer or shorter period.

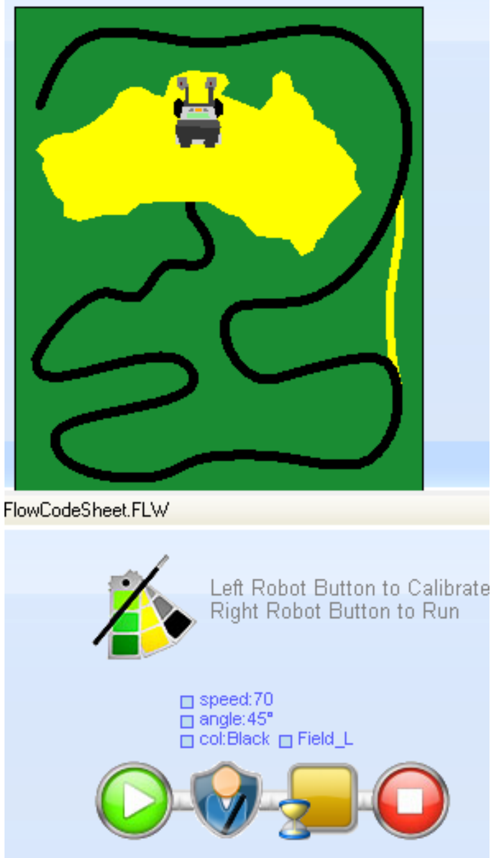
Try to make the robot follow the black line until it reaches the yellow area. You will notice that at the sharp turns towards the end this is not so easy. If you manage to make the robot move all the way to the swamp, then try to make it stop at the swamp. You can do this by replacing the stopwatch icon with a sensor command to detect yellow. If the robot cannot follow the sharp turns, don’t worry. We will show you a better way to do this later on.

Find the best speed and angle for the robot to follow the bends. Besides the speed it is moving at, the angle is important because if tells the robot how sharply to turn. If you use an angle that is too wide (a small Angle) then it will lose the track. If you make the turn too sharp (a large Angle) then it may shoot over the line. So you must find a good balance between the Angle and the Speed at which your robot moves.

1	 	Change the program by replacing the two icons with a SimpleFollower followed by a StopWatch icon. The latter tells the robot how long it needs to keep moving. You did this before.
2		In the previous lesson, you calibrated the robot. If the calibration was successful, you do NOT need to do it again, the robot will remember it. But if any of the circumstances have changed - the lights are different or you have moved to another field - you will need to repeat the calibration process. If you are using the simulator, you only need to recalibrate if you start a new program. Put the robot at the starting point with a sensor on each side of the line, press RUN and watch the robot follow the line.
3		At the first sharp turn after the yellow road, something will probably go wrong. The robot may be driving too fast, but the Angle that it uses to follow the line is also important. If the angle is too low, it will not return to the line fast enough, if it is too large, it will shoot over the line. The Angle has the same effect that a steering wheel has on a car (or your bike handles have on your bike). If you steer too little, you cannot follow the road and you'll hit the side of the road. If you steer too much, you end up zigzagging over the road and may end up on the other side of the road. The pictures shows the importance of robot's turning circle to correctly make the turns.
4	 Properties Default Speed: 70 The Angle to Track: 45 Line Color: Black Type: SimpleFollower	Try changing the Angle until it works correctly. Your robot will now make it a little farther and then probably go wrong again at the next turn. Try to set the angle so that the robot can follow the line and reach the swamp. If it gets stuck, the Angle is probably too large. Try to find the best combination of Speed and Angle to follow the road to the swamp. If this does not work, try to understand why. Sometimes the lighting on a certain part of the field may differ from that where you calibrated the robot. Check to see if the robot recognizes the colours at that point. If this does not work, continue with the next lesson, where we will make things a bit easier to take these sharp turns.
5	 	If you succeeded in reaching the swamp, have the robot to stop there and use the **<u>right</u>** sensor to detect to colour yellow. Be sure to select the correct sensor, otherwise it will stop at the yellow road. To do this, replace the StopWatch with a 'WaitForCalibratedYellow' icon from 'Sensor Flow'. Make sure to set the 'Properties' in the proper values.
6		If it all works correctly, you can also try the robot out on the real Rescue field. You will find that calibration is even more important now, as when the robot looks in a different direction any changes in lighting may block it.

5. A Faster Line-Follower

Detecting and following the yellow line



FlowCodeSheet.FLW

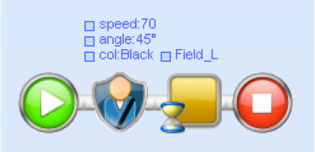
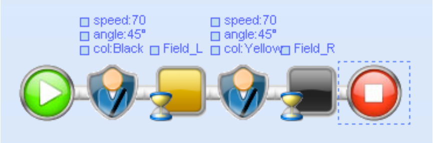
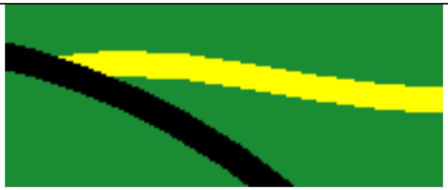
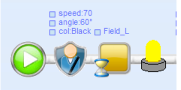
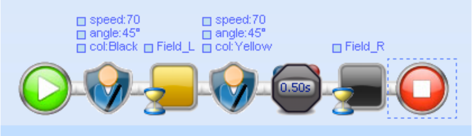
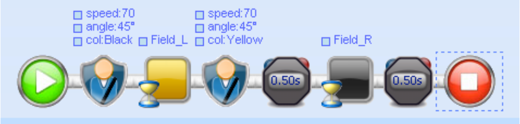
Left Robot Button to Calibrate
Right Robot Button to Run

□ speed:70
□ angle:45°
□ col:Black □ Field_L

Your robot has followed the black line and might even have reached the yellow swamp. However, you probably had to slow it down because of the sharp turns. As we want the robot to reach the swamp as quickly as possible, it is best if it follows the yellow line. You can save a lot of time by skipping the first bend in the road. Let’s develop a programme that allows the robot to find the beginning of the yellow line.

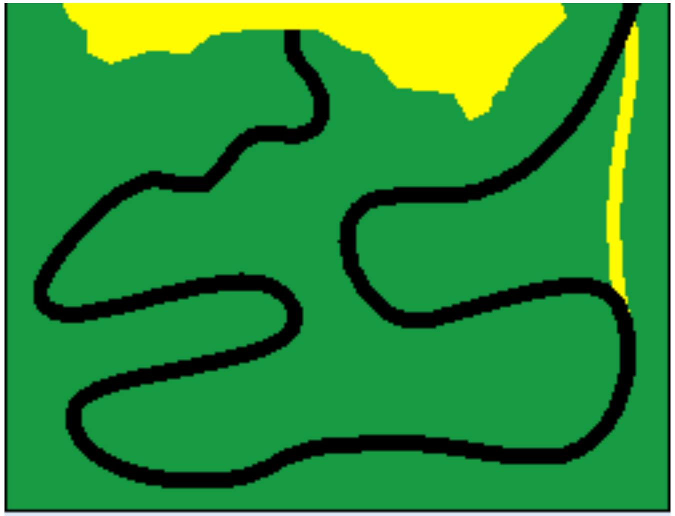
To do this, we first have to make the robot stop as soon as it detects the yellow line. Then, it has to follow it. We do this by using a second line-follower that will stop after a few seconds. You can include a stopwatch icon so the robot knows when it has reached the end of the yellow line, but it is better to use the sensor to detect the end of the yellow line.

You do this by having the sensor detect the colour yellow, as you did in the previous program. In this case, however, you have to do it on the other side of the line, because on the right side the yellow line is not visible. Once you start following the yellow line, you also need to know where it ends. You can count the time, but it is better to use the right sensor to detect the black line at the end of the yellow road. Beware, though, there is a sneaky problem hiding here!

1	 □ speed:70 □ angle:45° □ col:Black □ Field_L	Let's develop a programme to find the yellow line and have the robot stop when the left sensor detects it. Your previous program was looking for yellow with the **right** sensor. As the robot sees the yellow line on the left, the left sensor will have to detect it, as when it is using the right sensor, it will not see the yellow line.
2	 □ speed:70 □ angle:45° □ col:Black □ Field_L □ speed:70 □ angle:45° □ col:Yellow □ Field_R	As soon as the left sensor finds the yellow line, the robot must start following it. We do this by adding a 'SimpleFollower' and set the Properties of 'Line Colour' to 'Yellow'. Then, we must make sure that the robot will follow the black line again as soon as the yellow line ends. Therefore, we tell the robot that when it sees the black line again, on the right side, it must stop following the yellow line.
3		If you try this, you will find that things go wrong. It looks like the robot does not want to follow the yellow road. If you remove the command to stop when it sees black on the right side and replace it with a stopwatch indicating, for example, a half a second pause, it works OK. What is happening? The right sensor moves over the last part of the black line at the beginning of the yellow line. This makes the robot think that it has already reached the end of the yellow line, before it even starts following it. The robot thinks that it is ready to continue following the black line.
4	 □ speed:70 □ angle:60° □ col:Black □ Field_L	As this all happens quite quickly, it's hard to follow what is going on. Therefore, it is handy to include a lamp or a beep in the program. Do this by selecting a lamp or beep from Lights and Sounds.
5	 □ speed:70 □ angle:45° □ col:Black □ Field_L □ speed:70 □ angle:45° □ col:Yellow □ Field_R 0.50s	We need to make the robot skip the first part of the black line at the beginning of the yellow road. It must follow the line for half a second and only then check to see if the black line is visible. In this way, the robot will ignore the black line at the beginning of the yellow road.
6	 □ speed:70 □ angle:45° □ col:Black □ Field_L □ speed:70 □ angle:45° □ col:Yellow □ Field_R 0.50s 0.50s	And at the end of the yellow road, you will face the same problem. There still is a small part of the yellow line left when the robot detects the black line. If you start the black line-follower immediately, it may miss the beginning of the black line. So continue to move forward for a short while and then start following the black line again. You will, of course, have to add this into your program.
7		You can finish this assignment by following the rest of the black line. Keep in mind that when a number of sharp turns arrive, you will have to limit the robot's speed. The advantage of following the yellow road is that the robot knows exactly where it is. The first part of the track can be taken much faster than the rest of it. So try to follow the rest of the road in the next lesson.

6. The Rescue Mission

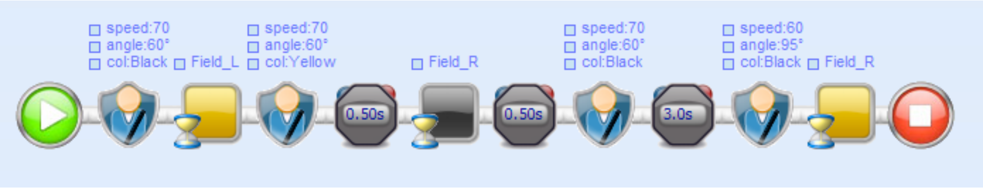
Following the rest of the track



To follow the rest of the track, you have to take the sharp turns into account. During your first attempt to reach the swamp, you may have noticed that you had to slow down to follow the entire track.

To follow the remainder of the road and the sharp turns to the yellow swamp, you had to adapt the line-follower to these turns.

To do this, you have to know where you are on the field. At the end of the yellow road, you have no further clues and have to count the time to see if you have already reached the winding road. Then, you had to change the line-follower to fit the Speed and Angle of the road.



□ speed:70
□ angle:60°
□ col:Black

□ Field_L

□ speed:70
□ angle:60°
□ col:Yellow

0.50s

□ Field_R

0.50s

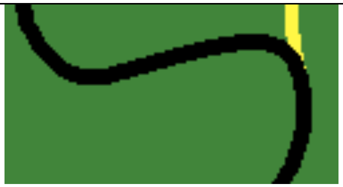
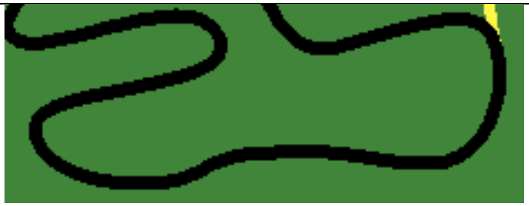
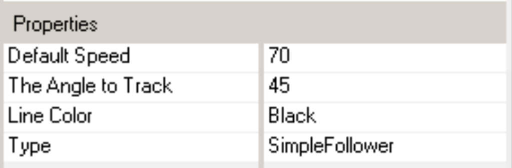
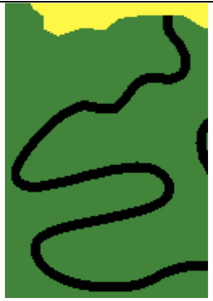
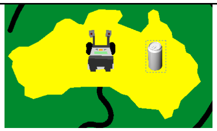
□ speed:70
□ angle:60°
□ col:Black

3.0s

□ speed:60
□ angle:95°
□ col:Black

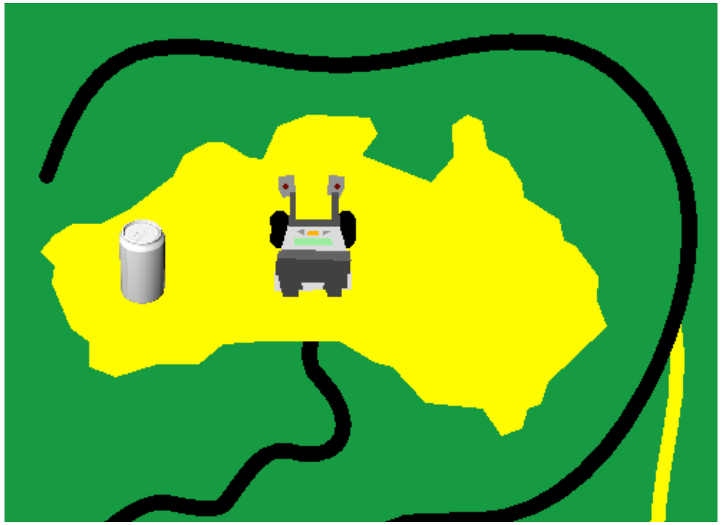
□ Field_R

The entire program is shown above, but you will learn more by trying things out yourself. Each robot is a little different from every other; you will have to find the right values for speed and angle for every part of the track. And if it works correctly on the simulator, this does not mean that your robot will operate correctly. You will have to adapt the program to your individual robot and the lighting conditions.

1		After completing the yellow road, you can follow the black line at a high speed, as there are very few bends left in the road. When the turns reappear, though, you have to change the properties of the line-follower.
2		To do this, you need to start counting how long the robot takes to reach the winding part of the road. When the robot gets there, start a new line-follower with different settings for speed and angle. In an earlier lesson, you tried to follow the entire track with the same settings and found out that this is very hard to do. So you must adapt your speed and steering (just as in a car) to the road ahead.
3	 Properties Default Speed 70 The Angle to Track 45 Line Color Black Type SimpleFollower	However, you can only change the speed and angle if you know where you are on the field. This will help you keep the highest possible speed without driving off the road. In the first part, we used the yellow road to find out where the robot was. In the last part, you will have to keep track of the time by using the stopwatch.
4		If everything works correctly, you will search the container in the swamp. You will do this in the last lesson. Make sure that when you follow the line, you understand why you are following the yellow road and what can go wrong at the beginning and at the end of it. When you reach the sharp turns, it is harder to follow the road at a high speed, especially when you have to move against the light. If the robot detects its own shadow, the sensors will react differently.
5		The goal of the Rescue Mission is to remove a dangerous container from the swamp. You will have to search for the container and then push it out of the swamp. The next lesson will show you how to do this.

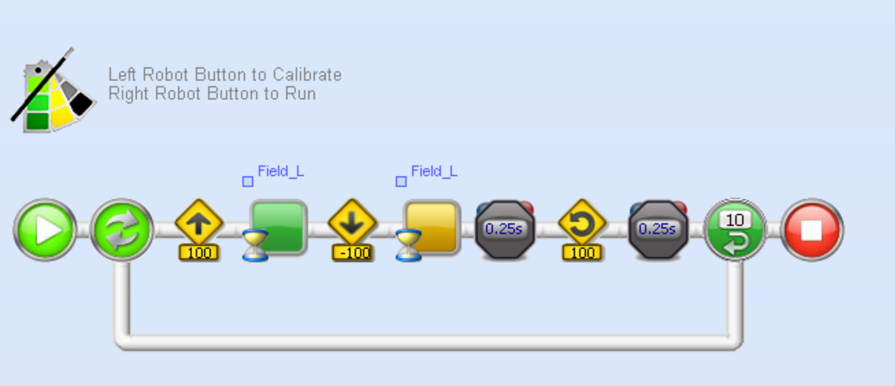
7. Clearing the Rescue Field

Searching the swamp



If you managed to reach the swamp at the end of the black road, you must now look for the container. You can also place a number of cups on the field. Just make sure that the robot searches the entire field and then pushes the container or all the cups outside of the field.

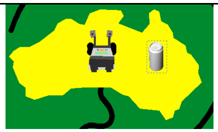
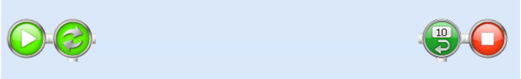
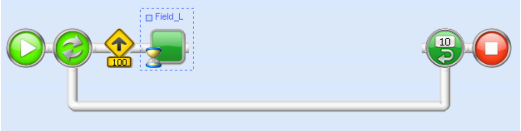
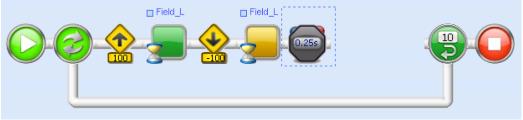
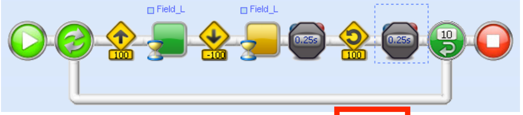
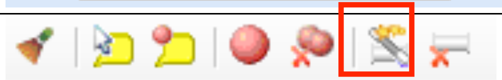
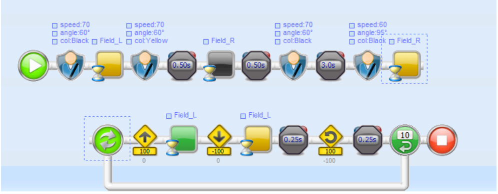
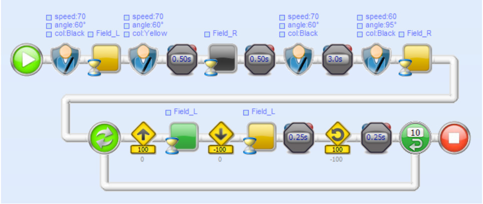
Move forward until you reach the green border. When you detect it, move backwards until you are back on the yellow field and make a slight turn. If you repeat this pattern, you will search the entire field. To do this, you need a repeating program or a Loop. You will test this program only on the yellow area.



Left Robot Button to Calibrate
Right Robot Button to Run

If everything works correctly, you can place the program at the end of the line-follower from the previous lesson.

You have now completed the rescue mission and are ready to participate in the Rescue Challenge. Try to make your program as fast as possible and change your robot to move as fast as possible.

1		Make a new program with the Rescue field. Select the container from 'Basics and Rescue' and place it somewhere on the field. You can also place a number of cups on the field. The robot has to push them out of the yellow area.
2		First, add the 'Merge' icon from 'Program Flow' and a 'LoopCounter' icon at the end of the program. We will develop the program to search the swamp between these two icons. We want to repeat this part 10 times and this is just what these two icons do together. We call this construction a 'Loop'. Remember to include the 'AutoCalibrate' icon so you can calibrate the robot (you will find it in 'Sensor Flow').
3		Make the robot move forward until it reaches the green field. As you see in the picture, there is a wire connecting the Merge icon and the LoopCounter. We will add this in the last step with the Wiring Wizard.
4		When the robot reaches the green area, let it move backwards until it reaches the yellow field again and then move backward a little more.
5		Now, let the robot make a slight turn so it faces a different direction. The 'Loop' makes the robot repeat this action 10 times, allowing it to crisscross over the swamp. And while it is moving around, it will find the container and push it 'ashore'.
6		You must now connect the ends of the icons at the bottom to close the loop. You do this with the Wiring Wizard at the right end of the toolbar. Test this part of your program both on the simulator and on the robot.
7		You are now ready to combine the two programs. First, select the first part of your program, which searches the swamp and then select Edit | Copy. Then, close the program and reopen the program that you created in lesson 6. Select the Edit | Paste option from the menu. All copied icons will then be placed in your program, but in the wrong place. Select the cursor with the four arrows from the toolbar, press the right mouse button and drag all the icons together to a free place in the rest of your program.
8		Now, select the last icon of the first line and the first icon of the second row. If you want to select multiple icons, use the Shift key together with the left mouse button. Then, select the Wiring Wizard. It will neatly connect the two selected icons. Save your program and now watch the robot perform the entire Rescue mission.

Introduction for teachers

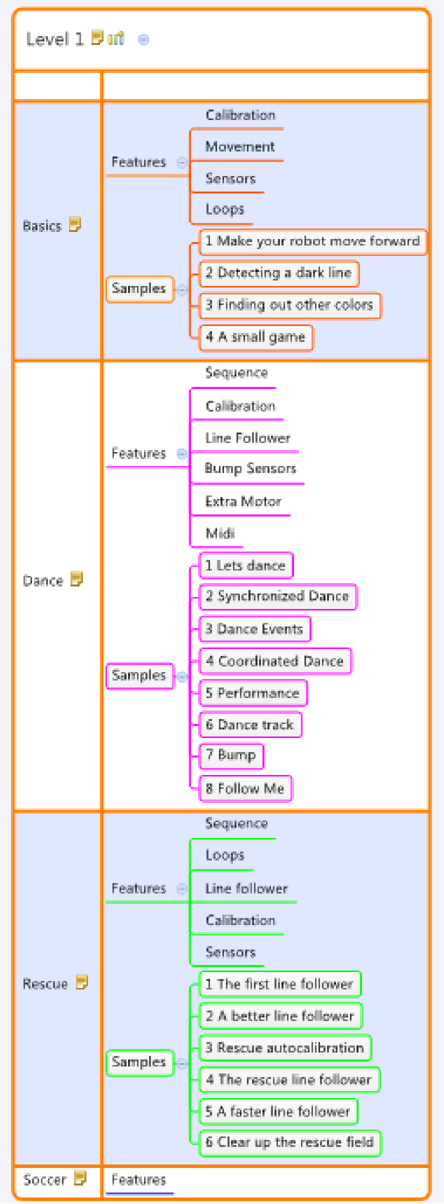
Rescue

The RoboDidactics workshops are based on individual lesson sheets. This allows each student to work at his or her own pace.

This is the Basics Level 1 workshop and is part of a series including:

1. Basics – basic concepts of RoboPAL programming and use of the Lego NXT robot
2. Dance – how to program a Dance robot
3. Rescue – how to program a Rescue robot

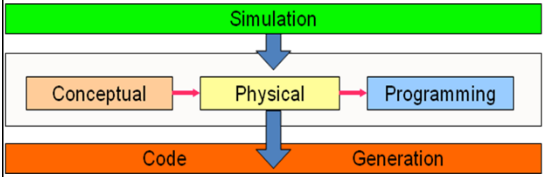
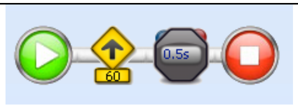
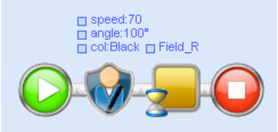
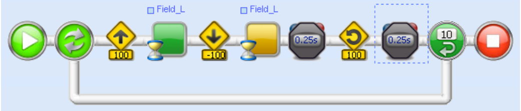
The picture on the right lists all the workshops included in the Conceptual Level (Level 1) together with their features and the individual lessons treated by each workshop.



Contents of the course

1	The first line-follower	How to make your robot follow a pattern
2	A better line-follower	Using a sensor to find a line
3	The rescue auto calibration	Discovering colours
4	The rescue line-follower	Following the black line
5	A faster line-follower	Detecting and following the yellow line
6	The rescue mission	Following the rest of the track
7	Clear the rescue field	Searching the swamp
8	The complete rescue mission	Completing the rescue mission

Setup of the course material

1		Each card includes an explanation about what will be done during the lesson on the front side. A brief description minimizes the amount of reading the students need to do. The RoboPAL software is available for the Lego NXT robot.
2	 Simulation Conceptual Physical Programming Code Generation	An important part of the teaching material is the use of the integrated simulator. This allows students to see the result of their program on a PC without having to use a robot. It makes testing easier and also reduces the number of robots needed in a classroom. In addition, students can develop and test their programs at home. The Conceptual Level concentrates on WHAT needs to be done. The Physical Level concentrates more on HOW things are done, while the Programming level concentrates on how to CODE a program in Java.
3		Programs are created using easy to understand icons in Levels 1 and 2, while in Level 3 programs are coded in Java. Each Level adds new capabilities, while Dance, Rescue and Soccer have their own lessons and facilities as well.
4		Programs are simulated on the PC and can also be run on a NXT robot. Once the program is ready and working, it is uploaded to a NXT, using a special dongle (memory stick) that takes care of all complicated software installation and driver issues.
5		For Rescue, a simple line-follower is provided to show students what a line-follower is and how they can influence its behaviour by changing the steering and speed at which the robot follows the line. Once students have mastered the rescue mission, they learn how to make their own line-follower with the Physical Level.
6	 □ Field_L □ Field_L 100 100 0.25s 100 0.25s 10	The students gradually learn about loops, subroutines and passing parameters. They also learn about calibration, testing and issues that have to do with timing and noise in sensors.
7		After each lesson, students are stimulated to reflect on what they have learned and what the program does. They do not merely follow instructions, but gain an understanding of what they have actually done. Once the students have completed a set of lessons, they are ready to participate in the Dance, Rescue or Soccer competitions of RoboCup Junior.