



Teacher's Guide

mowayworld

*Educational Robotics
Exercises*



mOway[®]
education

Teacher's Guide

Educational Robotics Exercises

Educational Robotics Theory and Practice Manual

Created by MiniRobots, S.L.

© 2011 MiniRobots, S.L.

Av. de la Ribera de Axpe

ED. 11-B mod.310-312

48950 - Erandio

Bizkaia, España

D.L.: BI-2366/2011

INDEX

1. Introduction	4
2. What is mOway?.....	5
2.1 Processor	6
2.2 Drive system.....	6
2.3 Set of sensors and indicators	7
2.4 Power supply system	11
3. Technology in Secondary Education	12
3.1 How to work with mOway.....	13
3.2 Flow diagrams. MowayWorld	15
4. Exercises	20
4.1 Exercise I. Movement of a robot	20
What we have learnt	25
4.2 Exercise II. Conditions. The enclosure.....	26
What we have learnt	31
4.3 Exercise III. The light sensor	32
What we have learnt	35
4.4 Exercise IV. Line Tracker.....	36
What we have learnt	42
4.5 Exercise V. Variables	43
What we have learnt	46
4.6 Exercise VI. Accelerometers. Touch Parking.....	47
What we have learnt	53
4.7 Exercise VII. The Copy	54
What we have learnt	62
4.8 Exercise VIII. Speaker	63
What we have learnt	66
4.9 Exercise IX. Defender / Fighter	67
What we have learnt	71
4.10 Exercise X. Maze.....	72
What we have learnt	76
4.11 Exercise XI. Remote data station.....	77
What we have learnt	82

1. Introduction

A new era is beginning, the era of the mini robot. There are an increasing number of mobile robot applications in our daily lives. Today, there are robots to help us with simple tasks such as cleaning the floors of our homes, mowing the lawn or keeping the swimming pool clean. As the technology of these small contraptions becomes more and more sophisticated, with their mixture of mechanics, electronics and software, they are capable of undertaking more complex tasks. Little by little, they are relieving us of many of our less gratifying tasks and are becoming more and more useful in our lives.

It would not be foolish to think that robots will have the same profound effect on our daily lives over the next decade as computers and telecommunications. Today, we have sufficient technology to manufacture these devices and the public is more and more used to seeing them on the market.

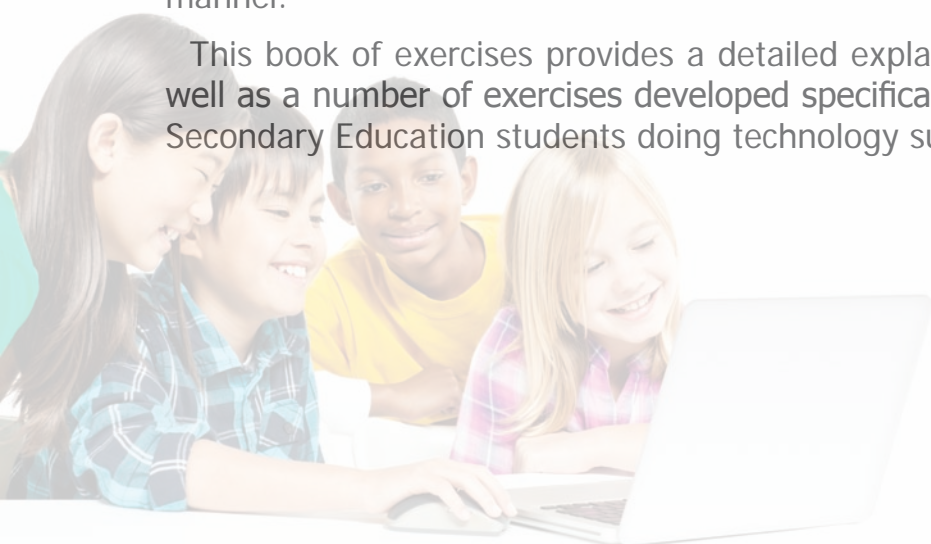
Until now, processors have not moved. Things are changing. One of the fundamental elements in the world of mobile robotics is software. The main difference in the development of programs for these robots compared to programs that are run in a personal computer lies in their interaction with the environment. In PC applications, the environment does not change at random and therefore decision-making tasks and, consequently, the programs are simplified. On the other hand, when executing commands within an application for a minirobot, we do not usually know beforehand what the result will be and therefore the algorithms must cater for a much broader range of possibilities, some of which may even be unexpected.

mOways are tools designed specifically for teaching and research. Their aim is to provide teaching centres with an introduction to the world of autonomous robotics.

The main aim of mOway is to be a useful tool both for newcomers to the world of robotics as well as for those who already have some experience and wish to develop applications of collaborative robotics.

mOway seeks to create an interest in this new and exciting branch of engineering, through the exercises contained in this manual, rapidly and in an entertaining manner.

This book of exercises provides a detailed explanation of how mOway works as well as a number of exercises developed specifically for its application in teaching Secondary Education students doing technology subjects.



2. What is mOway?

mOway is a small, programmable autonomous robot designed mainly for practical applications of mobile robotics.

mOway has provided the perfect hardware platform for those who wish to take their first steps in the world of mobile robots and for those who have already worked with robots and wish to develop more complex applications.

The mOway robot is provided with a series of sensors that help it operate in a real environment. It is also equipped with two motors that enables it to travel over the ground. All these peripheral devices are connected to a microcontroller responsible for controlling the robot.

This small robot can also be fitted with a number of additional options through an expansion bus. It can be connected, for example, to a wireless communications module, a video camera, a prototype card or any other device considered necessary in order to carry out a task.



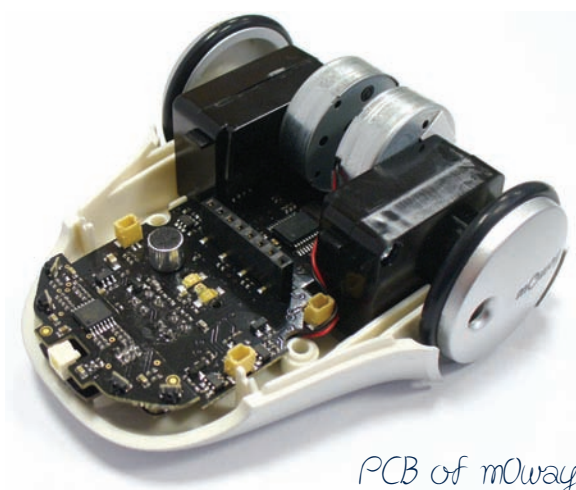
mOway robot

EmOway's external design is very compact, enabling it to move around with a great agility and without catching on any corner. Being as small as a cell phone, it has earned the nickname of "pocket robot".

mOway is a perfect tool for those who wish to learn about robotics and those who wish to teach others about this subject. Users will be surprised by the speed of their progress, even if it is the first time they have come into contact with mobile robots.

Inside a mOway we will find the following items:

1. Processor
2. Drive system
3. Set of sensors and indicators
4. Power supply system



PCB of mOway

2.1 Processor

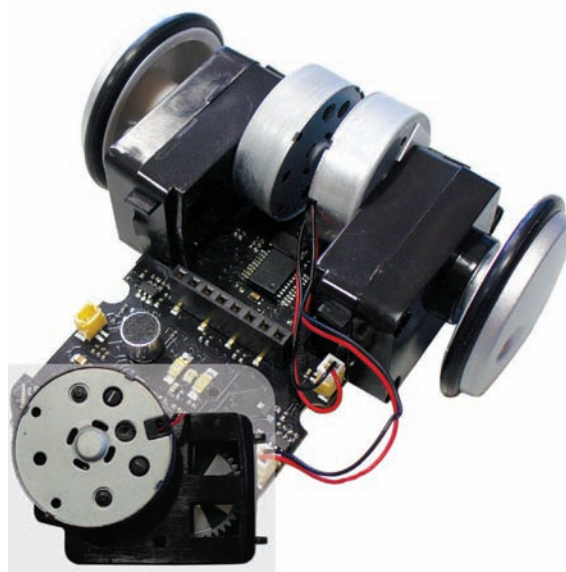
mOway robots are controlled by a 4Mhz PIC18f86J50 microcontroller manufactured by Microchip Technology. We will load the PC-developed program in this microcontroller. All the peripheral devices distributed by the robot are connected to its input/output ports. Some of these require a digital input or output, others an analogue input or output while others are controlled via one of the I2C /SPI communications buses.

2.2 Drive system

mOway robots have a dual servo-motor unit which enables them to move. The drive system is partly electronic and partly mechanical. The electronic part is mainly responsible for controlling the speed of the motors and the mechanical part provides mOway with sufficient power enabling the robot to move in different environments.

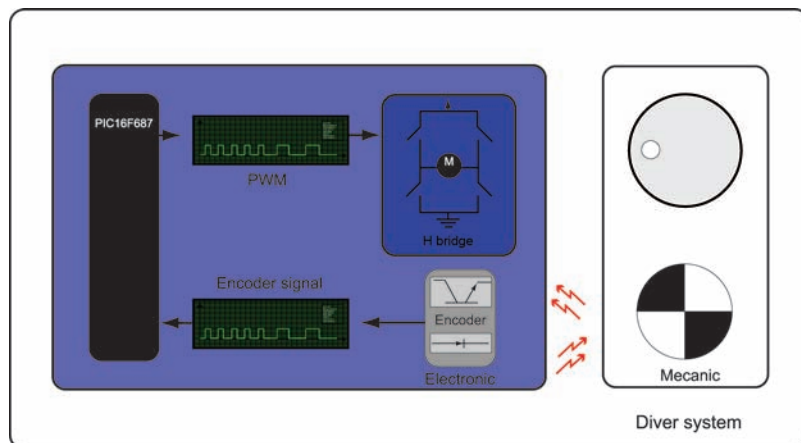
The servo-motor unit has a number of different functionalities:

1. Speed control: This controls the speed of each motor separately.
2. Time control: Controls the movement commands time with a precision of 100 ms.
3. Distance travelled control: Controls the distance travelled by each motor with a precision of 1 mm.
4. General odometer: Counts the distance travelled from the beginning of commands.
5. Angle control: Controls the angle when mOway is rotated.



Motor system with encoder

Speed is controlled by means of a closed loop control thanks to an encoder signal (sensors designed to measure the speed and travelled distance of the motors). Wheel rotation is monitored by means of an encoder placed on one of the gears of the system and an infrared sensor. The microcontroller analyses the signal and operates the motors. In this way, mOway can control its speed and the distance travelled.

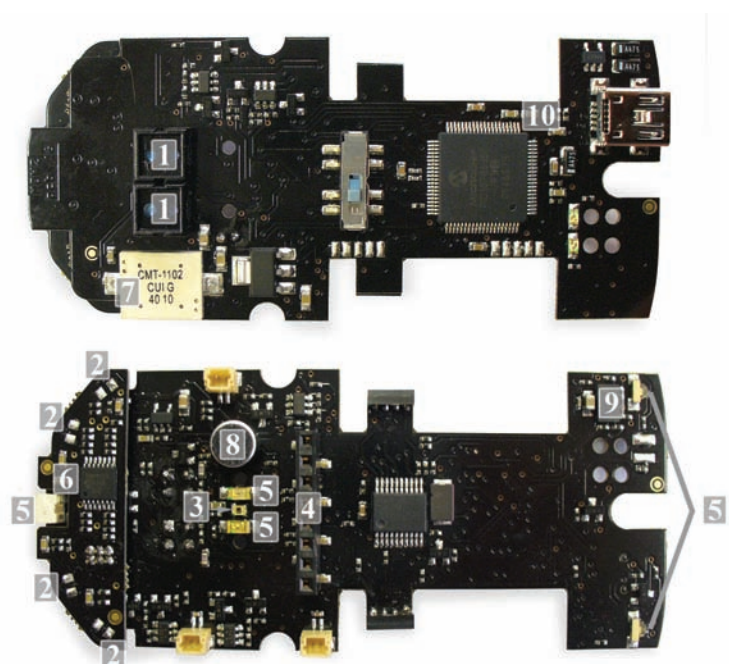


Motor control

2.3 Set of sensors and indicators

This unit consists of a number of different sensors and luminous indicators connected to mOway's microprocessor through which the robot interacts with the outside world:

1. Two line sensors
2. Four obstacle detection sensors
3. Light sensor
4. One expansion connector
5. 4 LEDs
6. Temperature sensor
7. Speaker
8. Microphone
9. Accelerometer
10. Battery meter



Sensors and indicators

2.3.1 Line Sensors

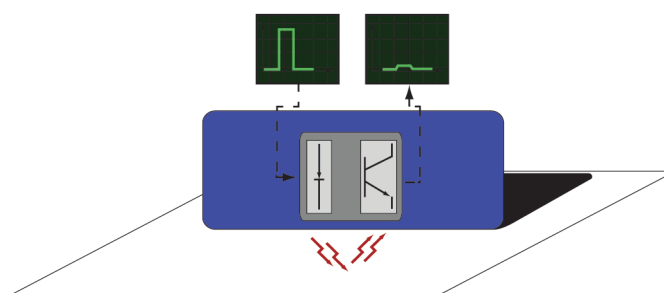
Line sensors are two reflection optocouplers mounted on the lower front part of the robot. These use reflected infrared light to detect the shade of colour of the floor where the robot is standing.

Both these sensors are connected to two of the analogue ports of the microcontroller, enabling this to detect strong contrasts in the floor, such as white lines on a black background, and to differentiate between different shades of colour.

The Vishay CNY70 sensor consists of a compact structure in which the light emitting source and the detector are arranged in the same direction in order to detect the light reflected on the floor with the use of infrared beams.

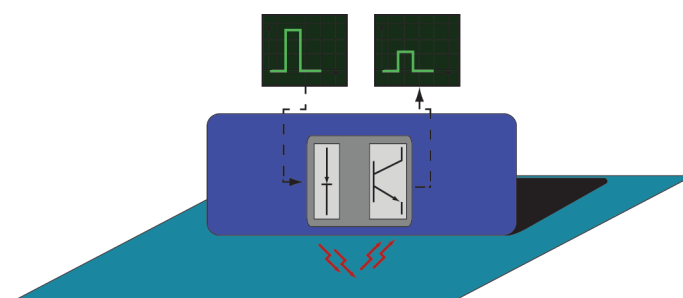
The following images show the three possible cases:

- **Light surface:** The white surface causes all the infrared light to reflect and therefore we obtain a low voltage at the output from the transistor in common mode.



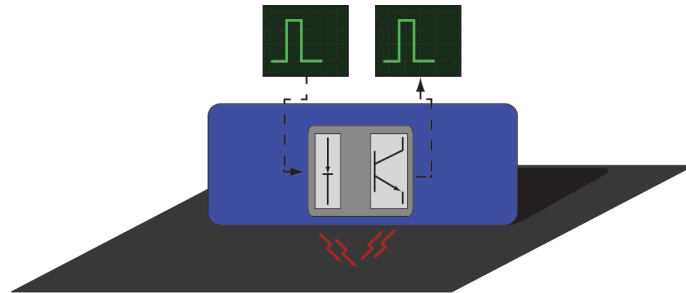
Line sensor on a light surface

- **Coloured surface:** The coloured surface causes the emitted light to reflect, obtaining an intermediate voltage at the input of the analogue channel of the microcontroller. In this way it is easy to identify colours.



Line sensor on a coloured surface

- **Dark surface:** The dark surface causes very little light to reflect, producing a very high voltage at the sensor output.



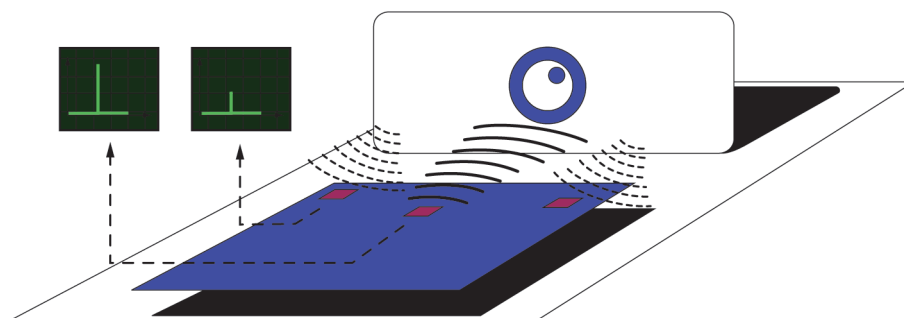
Line sensor on a dark surface

2.3.2 Obstacle detection sensors

As in the case of line sensors, obstacle detection sensors also use infrared light to detect objects located in front of mOway. The front of the mOway consists of an infrared filter, which blocks part of the infrared component of ambient light. The sensor consists of an infrared light source (KPA3010-F3C made by Kingbright) and four receivers fitted to both front ends of mOway.

The output of the Sharp PT100F0MP receivers is connected to the analogue inputs of the microcontroller. Therefore, it not only can detect the presence of an object (digital mode) but also measure the distance to the object (analogue mode).

The sensor operates in a similar way to the line sensor. The light emitter generates a 70-microsecond pulse which, if an obstacle is present, is reflected against this and captured by the receiver using a filtering and amplifying stage. Once the signal has been processed electronically, the microcontroller can measure this as a digital or analogue input.



Obstacle detection sensor

2.3.3 Light sensor

This sensor allows mOway to determine the light intensity that enters through the opening placed on the front of the robot. Thanks to its position, it can locate the light source and act accordingly.

The output of the APDS9100 sensor is connected to an analogue port of the microcontroller so that just by reading the ADC we can determine the light intensity level and whether this has increased or decreased since the last reading.

2.3.4 Expansion connector

This connector allows mOway to be connected to commercial modules or any electronic circuit that the user might require.

Available expansion modules are the mOway Camera Board, the mOway WiFi module and the RF module. The RF module allows mOway to communicate with other robots of the same type and with a PC. This will be used on the Exercise VII of this manual.

2.3.5 Front LED

The front LED is a white LED placed on the front of the robot. The OSRAM LW A6SG LED is connected to a digital output of the microcontroller.

2.3.6 Brake LED

The brake LED is an indicator which is connected to a digital output of the microcontroller.

2.3.7 Upper bicolour LED

This double indicator shares the same opening in the upper part of the robot as the light sensor. These are connected to two digital outputs of the microcontroller. It is important to realise that as they are near to the light sensor these must be turned off when it is necessary to read the light intensity.

2.3.8 Free pad

The mOway PCB is equipped with a Pad, which is only accessible when the robot is opened. It is placed on the rear, next to the brake LED, to which the user may connect electronic circuits.

2.3.9 Temperature sensor

mOway is equipped with a Murata NTC thermistor temperature meter, which is a semiconductor whose variable electrical resistance decreases as the temperature rises.

The thermistor is connected to an analogue input of the microcontroller. With a simple reading of the ADC we can determine the current temperature and whether this has increased or decreased since the last reading.

2.3.10 Speaker

The CUI INC CMT-1102 speaker, connected to the robot microcontroller, is capable of reproducing tones from 250 Hz to 5.6 kHz in 100-ms even intervals.

2.3.11 Microphone

The CUI INC CMC-5042PF-AC microphone allows the robot to detect sounds from 100 Hz to 20 KHZ.

The microphone output is connected to an analogue input of the microcontroller so that it is not only capable of detecting whether there is sound or not (digital mode), but with a simple reading of the ADC, it is capable of detecting the intensity with which that sound is arriving (analogue mode).

2.3.12 Accelerometer

An accelerometer is a device that measures acceleration and the forces induced by gravity, movement and turn. By measuring the X, Y, Z coordinates, the Freescale Semiconductor MMA7455L accelerometer allows us to determine whether mOway is in the correct position, upside down or lying on its side. Moreover, it allows us to determine whether the robot has suffered any impact or has fallen.

2.3.13 Battery level

In order to power the robot, it is equipped with a rechargeable LiPo cell. To ensure that the microcontroller operates correctly, the battery is connected to one of its analogue inputs via a signal adaptor. In this way, by reading the ADC, we can determine the remaining battery level.

2.4 Power supply system

The battery used in the mOway robot is of rechargeable Polymer Lithium and is placed inside the robot.

The battery is recharged via the USB port of any computer connected directly to the MINI-USB-B port of the robot itself. It is not necessary to wait for the battery to discharge completely in order to insert it; this can be done at any time, as this kind of battery is not affected by memory effect. Their small size, lightness and flexibility make these batteries the perfect power source for mOway.

The average duration of the battery is 1 hour and 30 minutes although this depends largely on the active sensors and the amount of time the motors are in operation. However, thanks to the battery level function, we can determine the robot's charge level at all times. Charging time is approximately 2 hours.

3. Technology in Secondary Education

The mOway robot is a teaching tool applicable to the Obligatory Secondary Education Technology course. The Didactic Units in which mOway is used are Control and Robotics and Computer Control.

In the Control and Robotics Area, students will gain practical knowledge about what a robot is, the parts it contains, its operation and control. In the Computer Control unit, students are given a grounding in computer control, its fundamental characteristics. Exercises in programming applications using flow diagrams are also included.

The skills developed with mOway are as follows:

- **Self-reliance and personal initiative:** Students are faced with a new task: programming a robot and controlling its actions. The aim of the different activities proposed is to enable students to tackle these new challenges without fear of making a mistake.
- **Group work:** Group work is a basic aspect of today's society, especially in the development and design of new projects. With teamwork, a commitment to fulfilling tasks and a respect for the opinions of others is encouraged.
- **Technology and digital skills:** Students will understand the importance of programming and digital systems and will observe the actual results of a program developed on a PC.
- **Logic and reasoning:** Through the intuitive software and the robot, which is an extremely practical tool, students learn the meaning of the flow diagram, its conditions and variables, and are capable of developing their own projects as well as detecting their successes and mistakes.
- **Results:** From the very beginning, students will obtain results, understand programming and see what they are capable of doing without having to wait, which will motivate them to go on learning.
- **Preparation for the Future:** Learning to program with mOway establishes the basis for future learning in several areas of science and technology and the students will remember easily what they learned in a simple and enjoyable manner.

Some of the evaluation criteria that can be worked on with the mOway robot are as follows:

- Understand the basic operation of the electronic elements of a robot.
- Understand how the main types of sensors operate: light, infrared, line, etc.
- Design a robot that acts in accordance with the values provided by its sensors:

light, obstacles, line, etc.

- Learn about the programming tools of a system through the computer.
- Prepare flow diagrams.
- Design programs that control the inputs and outputs of a robot.
- Open loop and closed loop control. Design of a program in closed loop.

3.1 How to work with mOway

1. First steps in robotics and programming: Knowing and understanding at a practical level the electronics of a robot, the operation of its different sensors and how to program these.

- We will start working with mOway, by identifying its different sensors and features.
- Students will become familiar with MowayWorld software, identifying the modules (processes), conditions and variables. With only a couple of modules they will have their first program and will be able to control the movement of the robot. They will be capable of designing ever more complex flow diagrams, adding different modules and conditions.
- When they download the program in the robot, they will check whether this is performed exactly as ordered and if not, will be able to easily find the module that caused the error.
- We will be able to do exercises such as:
 - The movement of a robot.
 - Follow the line.
 - Follow a maze.
 - Follow the light.
 - Enclosing a robot.
 - ...
- With these first exercises, students will learn to program and a teacher will have successfully completed this objective.

2. Collaborative Robotics: To know and learn about radio-frequency modules and their operation. Prepare programs for several mOways to communicate together or for mOway to communicate with a PC.

- The process of learning mOway continues and once we have understood its electronics and programming, we will introduce radio-frequency modules.

This module allows mOway to communicate with others of its kind (up to 254) and with the PC. The radiofrequency module is connected via the expansion slot situated on top of the mOway robots that are to communicate with each other.

- Following the same mOway programming learned with MowayWorld, we will add a new Communicate module, and we will begin to use variables in order to prepare the corresponding programs.
- We will be able to do more advanced exercises such as:
 - The copy: a robot will reproduce the same movements as the robot with which it is in communication.
 - Relay races.
 - Send orders to another robot.
 - Control mOway from a PC.
 - Find the balancing point.
 - ...
- With these exercises, students learn to prepare more complex programs and to do collaborative exercises between several robots and the teacher will have successfully completed the established objective.

3. **Expansion module:** Design an electronic circuit, add it to mOway and program it.

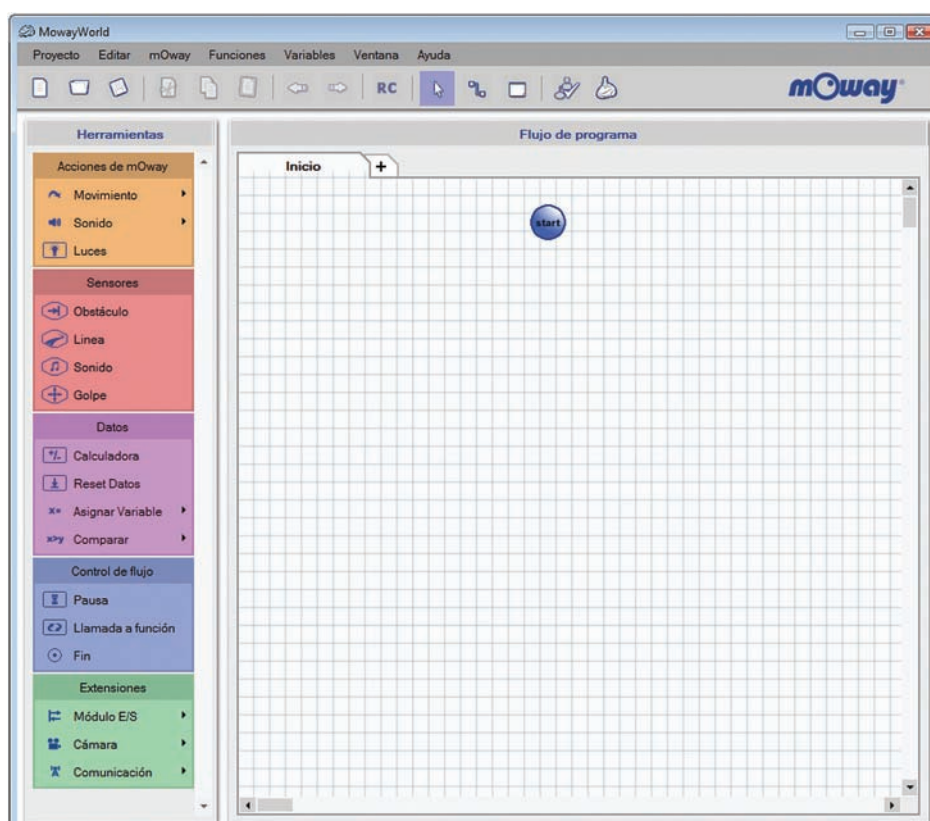
- In order to increase mOway's possibilities, we include the expansion Kit. This allows students to create their own electronic devices in order to add to mOway anything that they might imagine, thus making their robot different from the rest.
- In order to develop this electronic circuit, we will use the mOway Expansion Kit, consisting of two perforated plates on which the student will add his own electronic devices to the robot.
- Once we have created our electronic circuit, we will connect it to mOway by means of the communications bus of the expansion slot, allowing us to program the difference elements that have been added.
- Students will be able to set up a circuit with:
 - Different sensors, such as light, temperature.
 - Displays with 7 segments, LCD, voice synthesisers, actuators, etc.
 - The mOway robot will be able to read all of them.
- With these exercises, students learn to create their own electronics and with prior programming knowledge, program any sensor they may have added and the teacher will have successfully completed the established objective.

3.2 Flow diagrams. MowayWorld

Flow diagrams are used to represent and describe an algorithm graphically. When we speak about an algorithm in programming, we refer to a succession of defined and ordered instructions to carry out an activity in a series of successive steps.

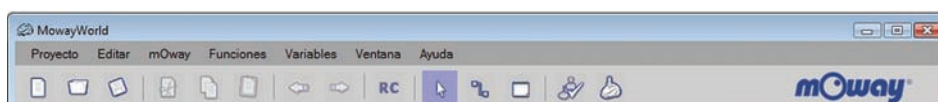
In our case, through flow diagrams we will describe and represent a set of instructions so that the mOway robot performs a specific activity such as following a line, escape from a maze, etc.

The image below shows the MowayWorld workspace. It is very user friendly, so that even a beginner user can program mOway robot in a few minutes. Below different parts of the workspace are described.



MowayWorld Workspace

MowayWorld toolbar allows the user to manage the project, edit the flowchart, create variables, program mOway robot, and so on. On the next chapter some of these options will be described.

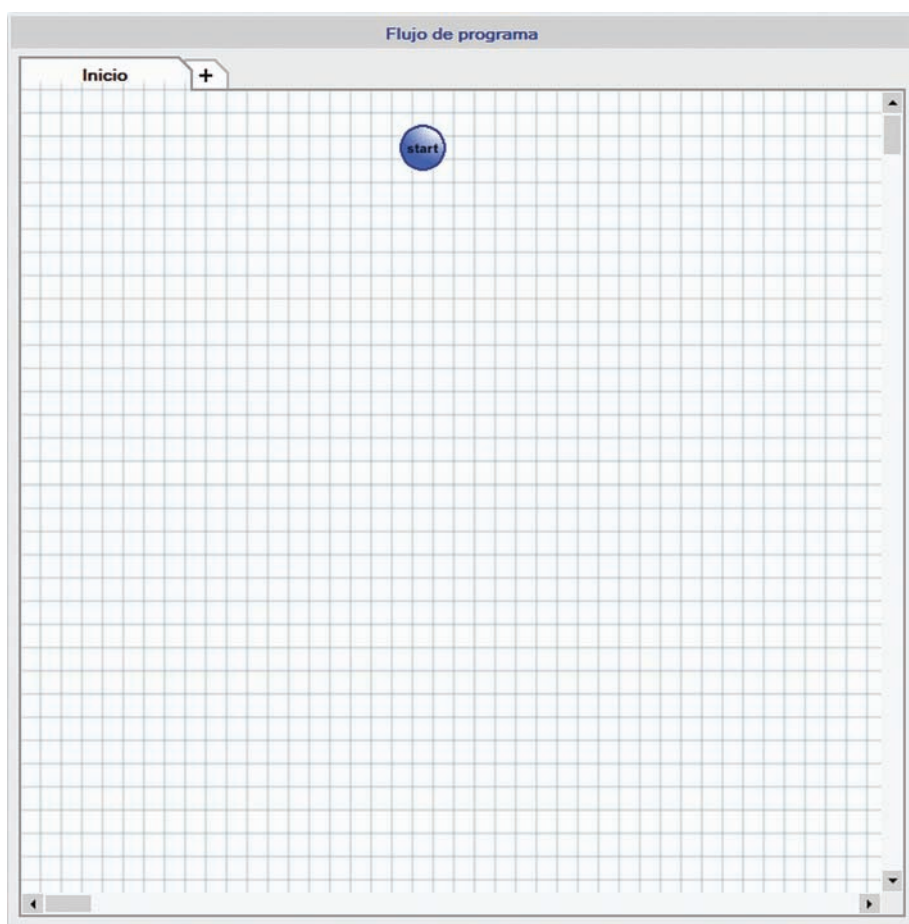


MowayWorld Toolbar

The **Flowchart window** is where modules are placed and connected to develop the program. When a new project is created, this window is empty except for the Start point. In flow diagrams we always have a "starting point" from which we will begin the program and from which the "flow" of the program will begin.

On the left side of the workspace is the Toolbox section. Here there are all the modules to control mOway, such as movement actions, checking sensors, communication, and so on.

Modules are grouped by their type of function (Actions, Sensors, Data, Flowchart Control and Expansion connector). The module functions will be described in the next chapter.



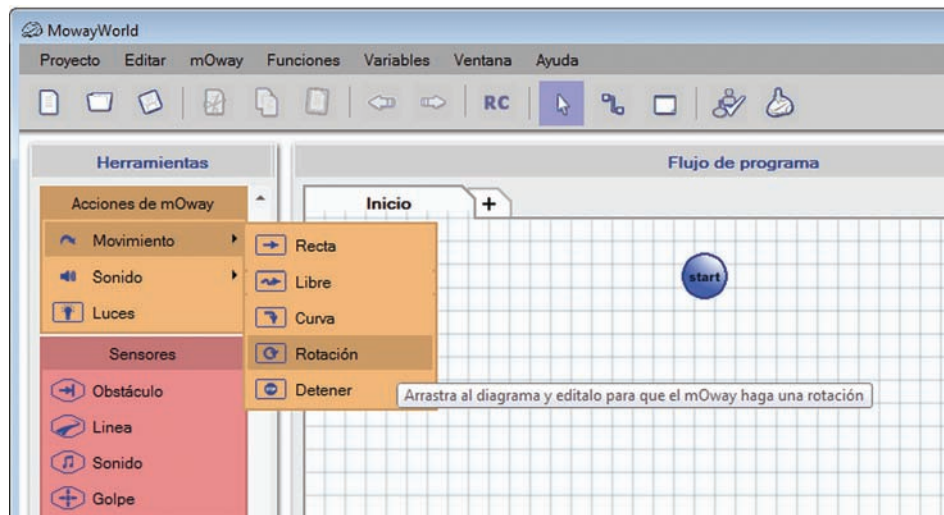
Flowchart window



Toolbox section

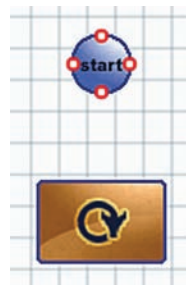
For example, if we wanted to turn the LEDs on, we have to place the mouse over the Lights module in "mOway Actions" menu. Then we drag and drop it to the Flowchart window. If we wanted to move the mOway, we place the mouse over "Movement" menu in "mOway Actions" menu. Then all the movement modules

appear and we choose the one we want.



Movement menu in mOway actions menu

Once the program modules have been placed, we must join them sequentially by means of arrows. In order to place an arrow, place the cursor over the first module and four red and white dots will appear.



Cursor over the module

Click on the lower mark and drag the cursor to the next module, until marks appears.



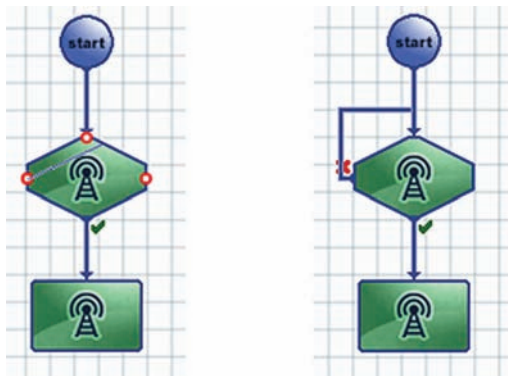
The cursor to the next module

Then, join the arrow to the upper mark of this module.



Arrow mark

If we wanted to create a loop, the arrow can be joined from one side of the block to the top of the same block. Here is an example:



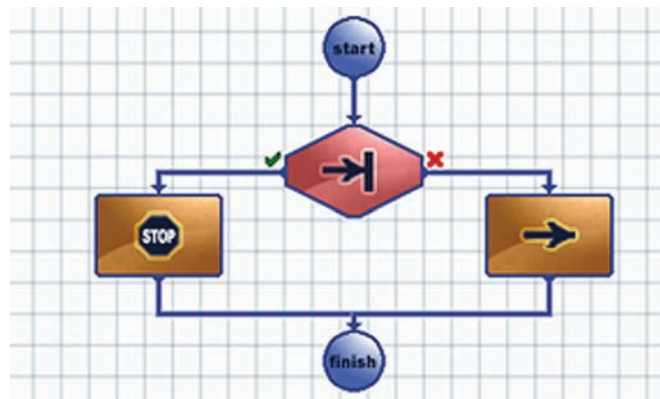
Arrow drawing for a loop

There are two possible types of modules within a flow diagram: **Processes** and **Conditions**. The Processes are instructions that execute a robot task and would not vary the program flow. They are square-shaped blocks with an input path and an output path. Examples of Processes are as follows: straight movement, turn on a LED, perform an addition, etc.



Example of processes

Conditions are instructions that diversify the program flow depending on whether its result is true or false. They are oval-shaped blocks. Examples of conditions are as follows: check whether there is an obstacle, verify whether a variable is greater or lower than a value, etc.



Example of condition

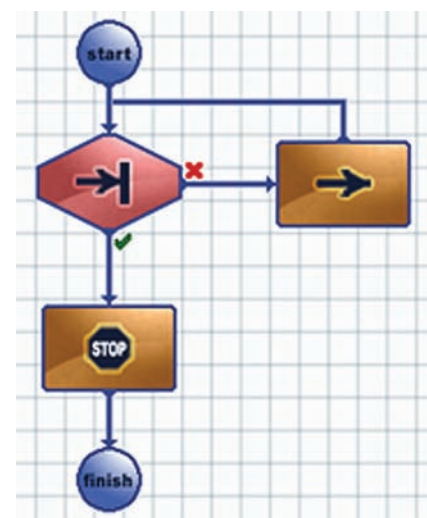
In the conditions it is observed how the program flow is divided between the “true” path (green mark) and the “false” path (red cross) in accordance with the instruction declared in the condition.

Previous flowcharts have a Finish point. The block connected to this point will be the last one of the program, and then it will finish. The Finish point is not always used, for example in some programs with a loop.

When in a condition one of the program flows returns to the initial condition, we create a “loop”. A loop or cycle, in programming, is an instruction performed repeatedly until the conditions assigned to the loop cease to be complied with.

In the drawing, the false program flow returns over an instruction and then over the same condition. Therefore, we repeat the same instruction until the condition is true. In this case, the program flow will continue until the second instruction.

Once programming with flow diagrams has been described, we will start to do exercises with MowayWorld.



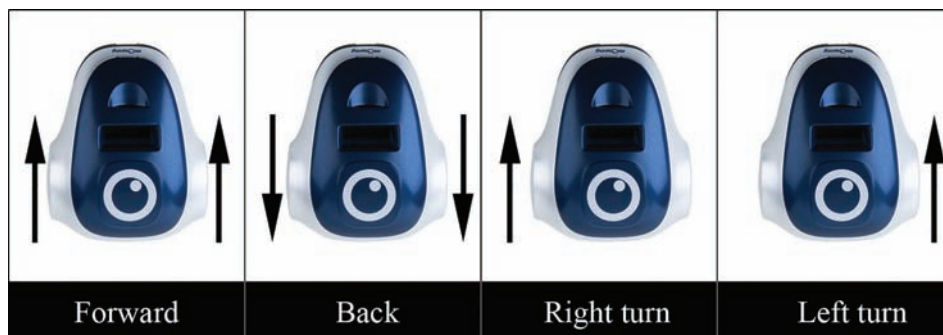
Example of loop

4. Exercises

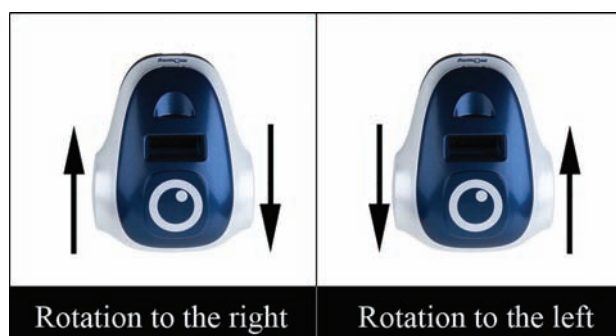
4.1 Exercise I. Movement of a robot

In the first exercise we are going to learn about the movements of the robot. The mOway robot has two motors with independent closed loop control connected to each one of the two wheels. By controlling the turning direction of the motors we can make the robot move forward, turn or move back.

The following table shows the behaviour of the robot in accordance with the turning direction of the motors.



We can also make the robot rotate if we reverse the turning direction of a motor with respect the other. In this way, the robot will turn on its centre.



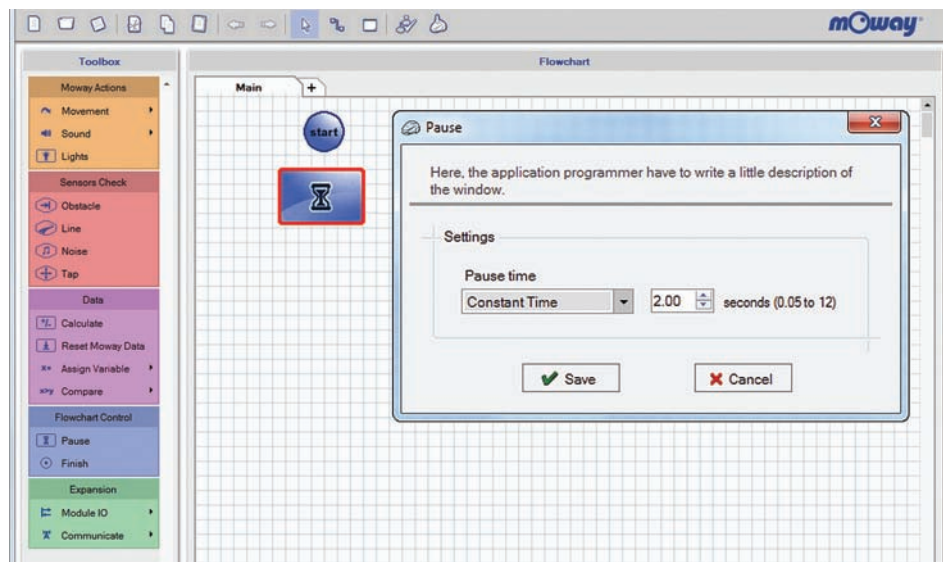
If we modify the motor speed in the above movements, we can achieve infinite possibilities in the movement of the mOway robot, it being possible to plot circumferences, spirals, curves, etc.

In order to do this exercise, we will use the mOway robot with its programming interface MowayWorld based on flow diagrams. In this first exercise, we will only use processes and therefore the program flow will be the only thing we will use, without conditions.

The aim of this first program will be to make a robot move forward in a straight line for 2 seconds, turn on its own axis 90 degrees to the left and then move forward 10 cm. To do this, we will use the Movement modules in mOway Actions panel.

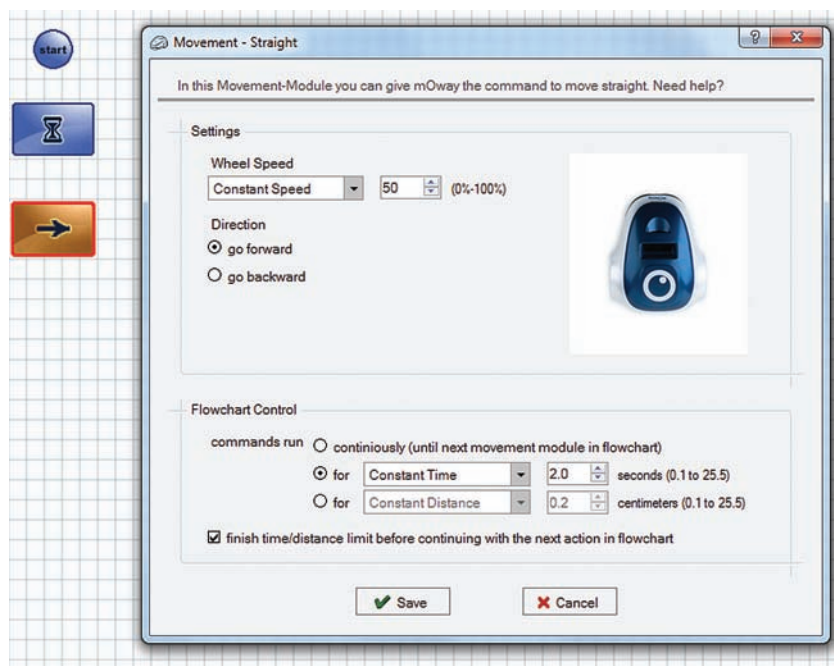
In order to place a module, drag the mouse cursor to the Modules Panel on the left side of the window and select the type of module. Then drag and drop to the Flowchart Editor (white squared panel). By double-clicking on the module, we can gain access to its configuration.

The first module used is located on "Flowchart Control → Pause". Select the dwell time to 2 seconds and push "Save".

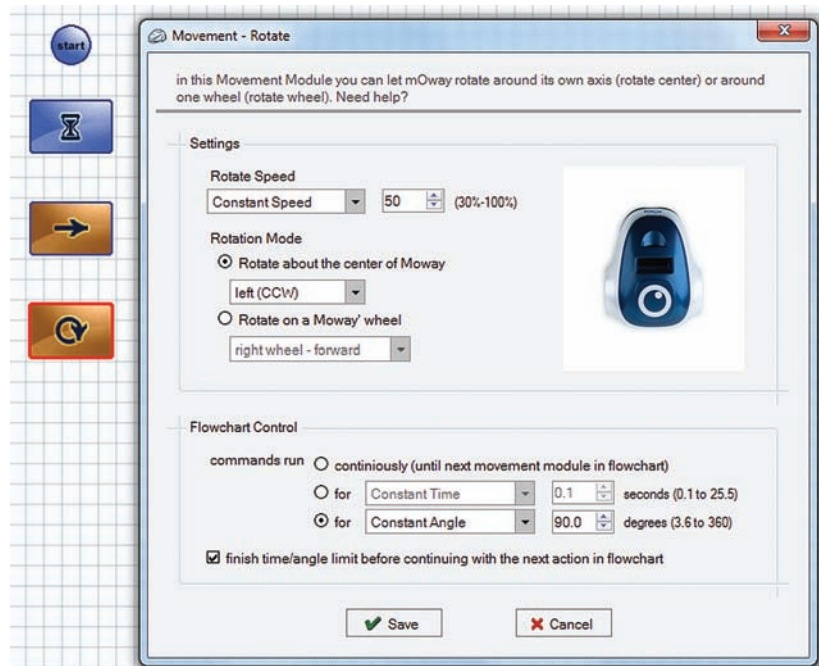


The second module will be a forward movement in a straight line with a duration of two seconds. Place a new module from "mOway Actions → Movement → Straight". The moving speed will be 50. Select the option "commands run for Constant Time" to 2 seconds.

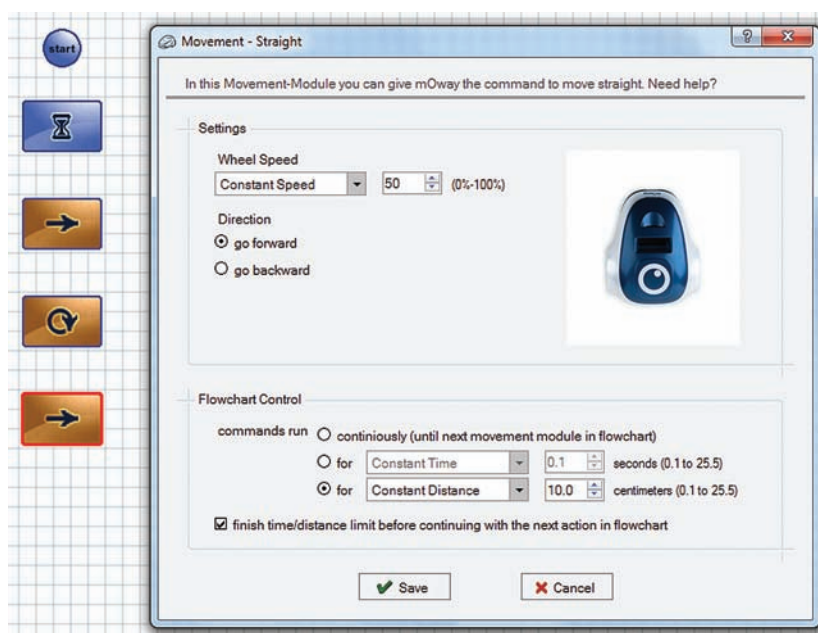
It is important to select the option "finish time/distance limit before continuing with the next action in flowchart" so that the program does not continue to run to the next module until it has completed the 2-second movement.



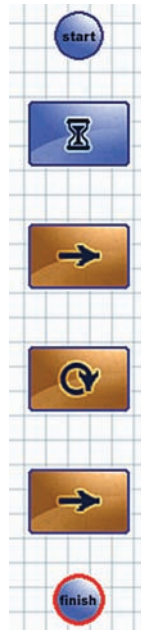
The next step will be the 90-degree turn. To do this, place a new module from “mOway Actions → Movement → Rotate”. The diagram below shows the configuration of the module.



Once the robot has turned 90 degrees to the left, insert the movement forward module of 10 cm. The illustration below shows the configuration of the module.



The program end is set by placing a Finish module, which is located in “Flowchart Control” panel.



Once the program modules have been placed, we join them sequentially by means of arrows. In order to place the arrows, place the mouse over the first module, click on the red and white marks and drag to the next module. For further information about placing arrows, please refer to the previous chapter.



Joining all the modules, the flowchart will look like this.

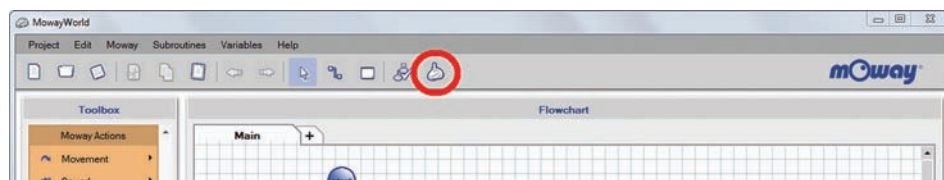


In order to connect the mOway robot to the PC, do this through the USB port located at the rear of the robot as shown in the enclosed illustration.



When the diagram has been completed, click the "Program mOway" button (shown on the image below) with the mOway robot connected. Once programed, unplug the robot and the program will begin to run.

This concludes the first exercise. In this exercise we have learnt a number of concepts relating to the creation of flow diagrams and the programing of robot movements.



Based on the project, we can make improvements in order to learn how to program the robot:

- **Práctica 4.1.1.** Change the turning angle.
- **Práctica 4.1.2.** Change the speed of the movements.
- **Práctica 4.1.3.** Rotate on one wheel instead of on the centre. In this way, one wheel will remain immobile while the other turns.

What we have learnt



How the robot can **move** forwards, backwards and turn depending on the turning direction of both wheels.



How to make a simple **flow diagram**.



How to **program** the robot.



New challenges

- Change the turning angle in order to make a 180° turn.
- Try to change the speed of the movements.
- Turn the robot on one of the wheels instead of on the centre.



Check what you have learnt

1. The robot turns on its centre when:
 - a) One wheel moves forwards and the other backwards
 - b) One wheel moves forwards and the other is stopped
 - c) Both wheels move forward
2. What value must we put in the "Pause" block if we want to wait 5 seconds?
 - a) 0,05
 - b) 0,50
 - c) 5,00

4.2 Exercise II. Conditions. The enclosure

In exercise II, the Condition concept is introduced in a flow diagram. A condition is an operation that has an input path and two output paths: one if the condition is true and the other if the condition is false.

In order to practice conditions, we will use infrared line sensors. The aim of this activity is to program a robot to advance continuously until it finds a black line. In this case, it will make a 126-degree turn and continue on its way.

To do this exercise, we will use the mOway track. The robot will be placed within the black line and will remain enclosed.



If we want to detect marks in the floor we will have to use the line sensors. The electronic component used is a CNY70 fitted with an infrared emitter and detector. The infrared emitter is a LED diode and the receiver is a phototransistor, in other words a transistor that conducts current in accordance with the light that falls on its base.

This type of sensor is often used in bar code scanning applications and motor speed gauges. One of its applications in consumer products is in mobile phones with large screens. In this case, the centre is located next to the screen in order to detect when we bring the telephone close to our ear in order to speak. When this occurs, the sensor detects this movement and the mobile turns off the screen in order to save battery.



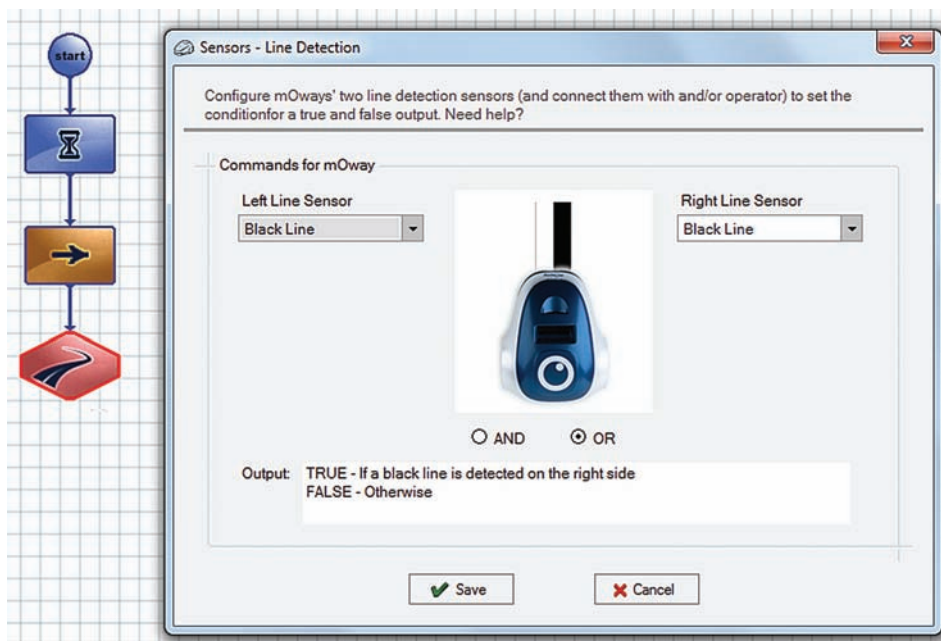
In this program, we will use the sensors in digital mode (black or white). However, the mOway robot can also make analogue readings to discriminate between different levels of grey.

The dark surface causes very little light to reflect, producing a very high voltage at the sensor output. If the surface is light, the amount of light is high and therefore the transistor is maintained in saturation mode with a low output voltage level.

Firstly, insert a 2-second pause module as in the Exercise. Secondly, we will indicate the default action, which will be an indefinite straight, forward movement at a speed of 50. The movement is not limited as we will be the one to modify the movement when a black line is detected.

The next step will be to establish the program condition. To do this, drag and drop the "Sensors Check → Line" module. By double-clicking, we can access to the configuration of this module. As we want to select black lines, select the colour "Black" for both sensors.

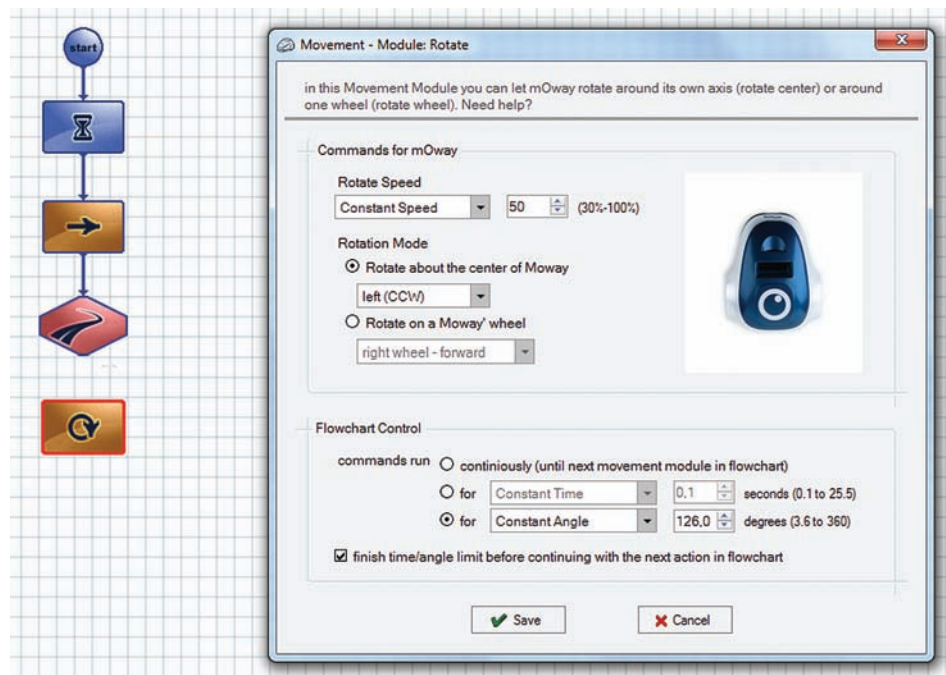
Some Sensor Check modules allow to choose between AND and OR options. With AND selected, the condition will be true if all the sensors perform the condition. With OR selected, the condition will be true if at least one of the sensors perform the condition. In this case, if one of the line sensors (left OR right) performs the condition (in other words, detects the black line), the condition is true (black line has been detected) and mOway has to turn.



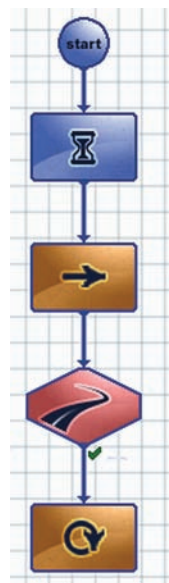
There are two output paths for conditions, one if the condition is true and another if it is false. In the condition explained before, if one of the line sensors detects a black colour, the path taken by the program will be the one for the true condition and if not, it will take the false condition path. Therefore, the true path will take to a 126-degree turn module. On the other hand, if the result is false, we must continue to move straight forward.

When the robot detects the black line, it must rotate in order to stay into the circle. Therefore, the true path will take to a 126-degree turn module. On the other hand, if the result is false, we must continue to move straight forward.

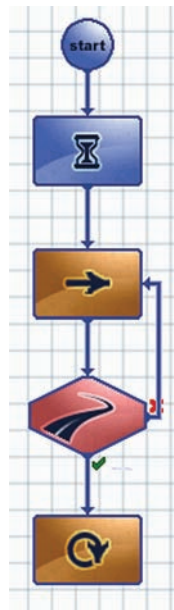
So that, we insert the following 126-degree rotation module. The necessary configuration can be found in the following figure.



In order to complete the program we must join the different modules and conditions. When we join the condition module with the next module, the first arrow that appears is the “true” condition, and it is represented with a green mark. Join this arrow to the “Rotation” module, as this is the required action if the robot observes a black line.

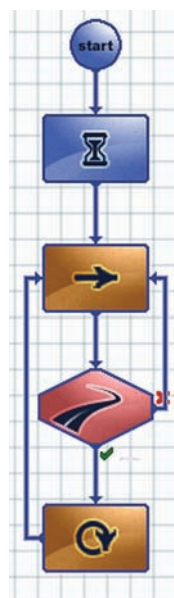


Drag another arrow from the condition module again and join it this time to the previous “Movement” module. This arrow is the “false” condition, and it is represented by a red cross. If the robot does not see a black line, we want it to continue advancing straight forward.



Finally, join the "Rotation" module to the "Movement" module, as once the robot has completed its turn it will continue moving straight forward.

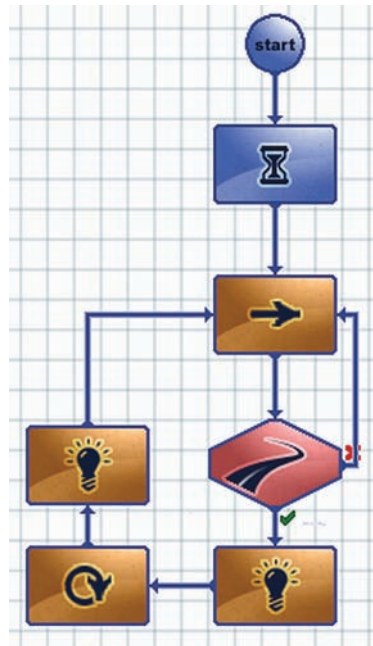
In this exercise, the "loop" concept is introduced. This is a part of the program that is repeated in sequence as long as a condition is met. In this case, the loop is infinite, as the program is repeated indefinitely. For this reason, we have not placed a program "Finish".



Based on the project, we can make improvements in order to learn how to program the robot:

- **Práctica 4.2.1.** Change the turning angle.
- **Práctica 4.2.2** Change the speed of the movements.
- **Práctica 4.2.3** Turn on the luminous indicators when the line is found and turn them off when the turn has been completed.

To do this, we must include a Lights module “mOway Actions → Lights” and configure this in order to turn on the LEDs before beginning the turn, then rotate and once the rotation has been completed, turn them off. Lights module also allows to blink the LEDs.



• **Práctica 4.2.4.** Change the lines for walls and use the obstacle sensors. In this way, we will design a robot that moves forward until it sees an obstacle. At that moment, it will turn around.

In order to carry out this program we will use the logic used before but this time in reverse. We will program the robot so that while no obstacles can be seen it continues straight and if an obstacle appears (the condition that there are no obstacles is false) it rotates. The AND operation must be checked.



The truth table of “ p AND q ”:

p	q	AND
V	V	V
V	F	F
F	V	F
F	F	F

The truth table of “ p OR q ”:

p	q	OR
V	V	V
V	F	V
F	V	V
F	F	F

What we have learnt



How mOway's **line sensors** work.



What a **conditional block** is.



What difference there is between **"AND"** and **"OR"** operations.



What a **loop** is.



New challenges

- Try to change the turning angle.
- Make the robot turn on the LEDs when it detects the black line.
- Change the lines for walls and use the obstacle sensors. In this way, we will make the robot move forward until it sees an obstacle. At that moment, it will turn around.



Check what you have learnt

1. When the sensors work in digital mode, what colours does it detect?
 - a) Different levels of gray
 - b) White and black
2. If we want the robot turn only when its two sensors detect the black line, how would we have to configure the "Line" block?
 - a) With "AND" option
 - b) With "OR" option
3. Why a conditional block has two outputs?

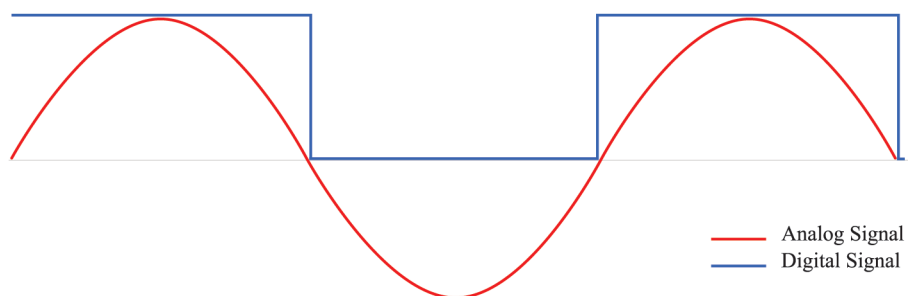
4.3 Exercise III. The light sensor

Having learnt about the operation of the line sensor and the obstacle sensor in the previous exercise, we are now going to introduce the ambient light sensor. This sensor indicates the amount of light (from 0 % to 100%) received through an opening in the form of a half moon on top of the chassis. As this points forward, it can determine where the source of light is located and act accordingly.

The sensor is an APDS9102 photodiode. The photodiode is a PN joint (diode) whose behaviour varies in accordance with the amount that falls on it. These components are widely used in consumer products. For example, mobile phones and portable computers often include a photodiode to detect the ambient light and adjust screen brightness.



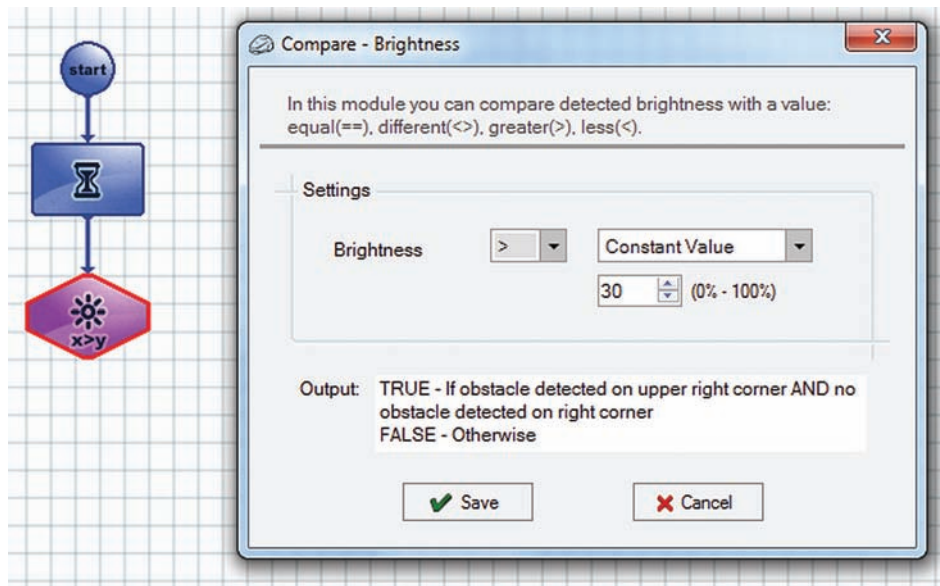
The mOway light sensor is an analogue sensor. An analogue signal is a type of continuous signal which is available in amplitude and period, which is generated by an electromagnetic phenomenon. In the following diagram, we can see the difference between the digital signals that we have used before (black or white line, obstacle yes/no) and the analogue signal of the light sensor.



The aim of this third exercise will be to reproduce a well-known children's game. We will design a robot that will remain still when the lights are on. When the light is turned off, the robot will start to move and the red LEDs at the top will blink until the light is turned on again.

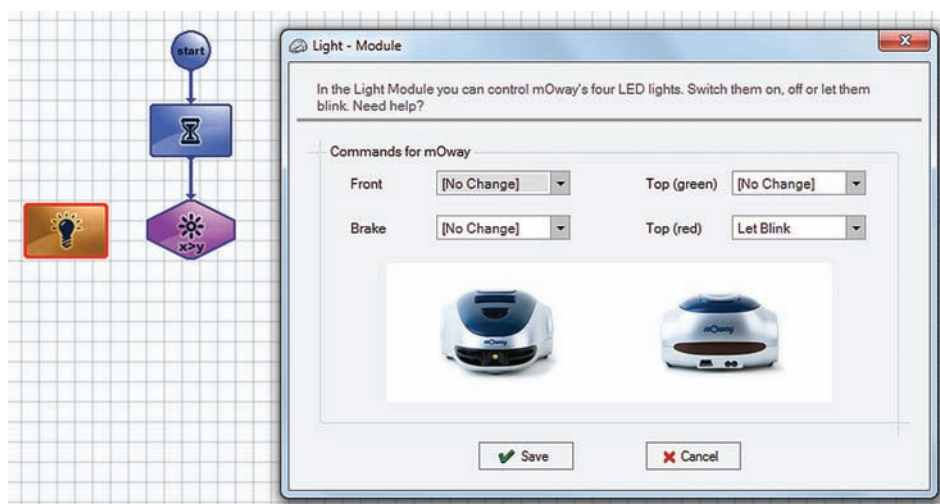
To begin the program, insert a 2-second pause. Secondly, insert the condition of this exercise. The condition used will be "Sensor comparison". This condition module is located in "Data → Compare → Brightness".

We are going to compare the luminosity sensor to check whether it exceeds a certain threshold. In this exercise, a threshold value of 30 has been placed. Nevertheless, this value may vary in accordance with the environment.

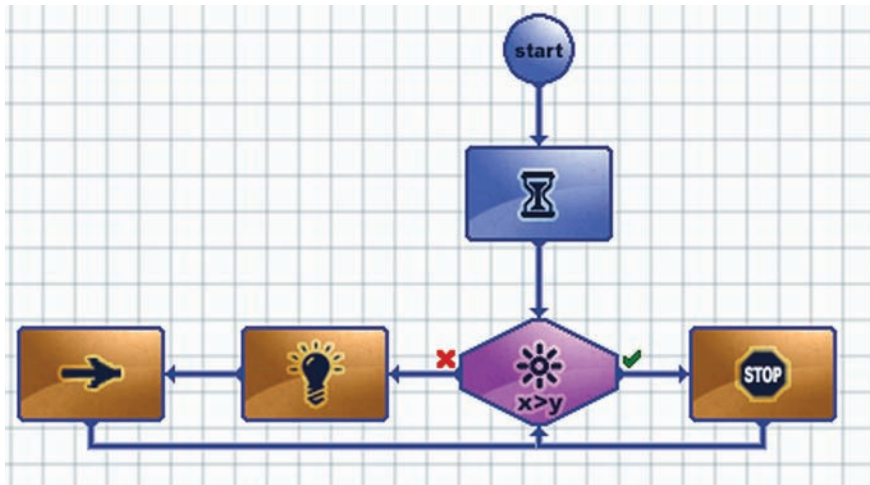


We have configured the condition to continue on the true path when the light sensor value exceeds 30%. In this case, the robot will remain motionless while it waits for the conditions to change. Therefore, we will introduce a Movement Stop module.

In the false path, if the luminosity is under 30 (without light), we will place two modules, one to make the upper red LED blink and the other with a continuous forward movement at a speed of 100. In order to configure the blinking operation of the LED, we select the action "LEDs Diodes" as shown in the figure below.



Once the modules and conditions have been entered, we join the program with arrows as shown in the figure below.



Based on this, we can make improvements in order to learn more about programming the robot:

- **Exercise 4.3.1.** Change the speed of movement and the luminosity threshold.
- **Exercise 4.3.2.** Make the green LED blink while the robot is at a standstill.
- **Exercise 4.3.3.** Change the straight movement into a zigzag movement.

What we have learnt



What a **light sensor** is.



What an **analog signal** is.



How to **compare** the value of one sensor with another value.



New challenges

- Change the luminosity threshold, in other words, the value that is compared to the sensor.
- Try to make the green LED blinks when the robot is stopped.
- Try to make the robot move forwards in zigzag.



Check what you have learnt

1. An analogue signal can have different values within a range, while a digital signal can have only two values.
 - a) True
 - b) False
2. The light sensor is a:
 - a) Digital sensor
 - b) Analog sensor
3. How do we configure the "Luminosity" block to check whether the light level detected is less than 50%?
 - a) Luminosity ≥ 50
 - b) Luminosity < 50
 - c) Luminosity $< > 50$

4.4 Exercise IV. Line Tracker

In the line tracking exercise, we will make use of the movement conditions and modules as in the previous activities. However, in this exercise we will learn about “nested conditions”, in other words, conditions in which an output path leads us to another condition.

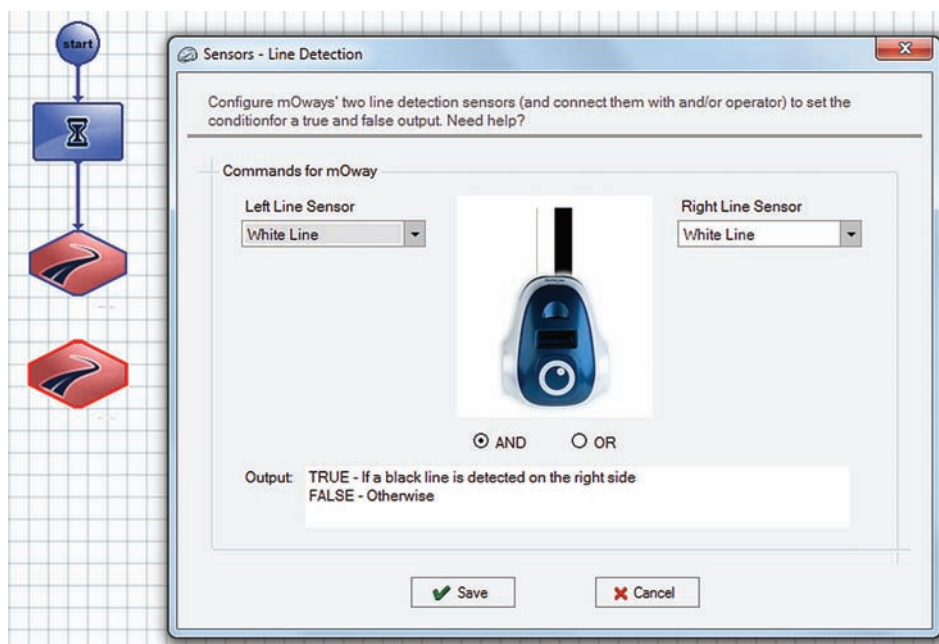
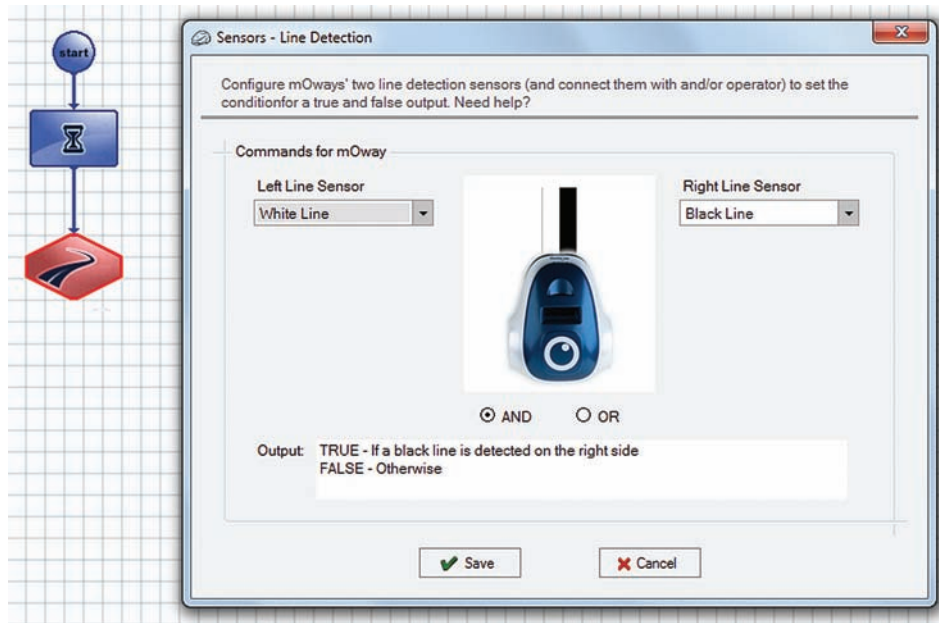
We can use the mOway track used in Exercise II as the environment for doing this exercise. In order to develop a program that follows the line, we must try to make the robot advance along the line of contrast between the white area and the black line. In this way, one of the line sensors will be on the white area and the other on the black line. In this case, we will try to place mOway's left-hand sensor on the white area and the right-hand sensor on the black area.

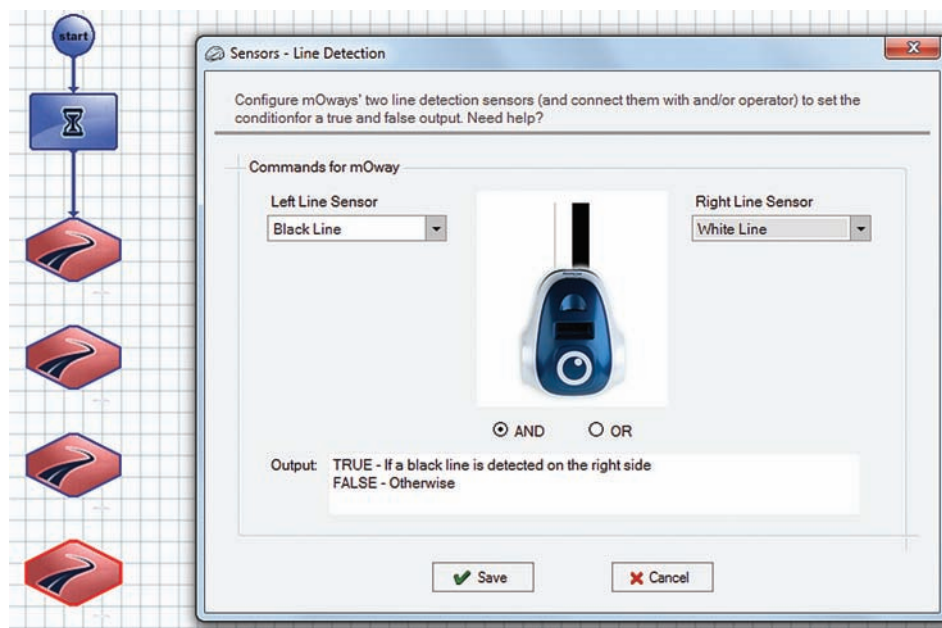
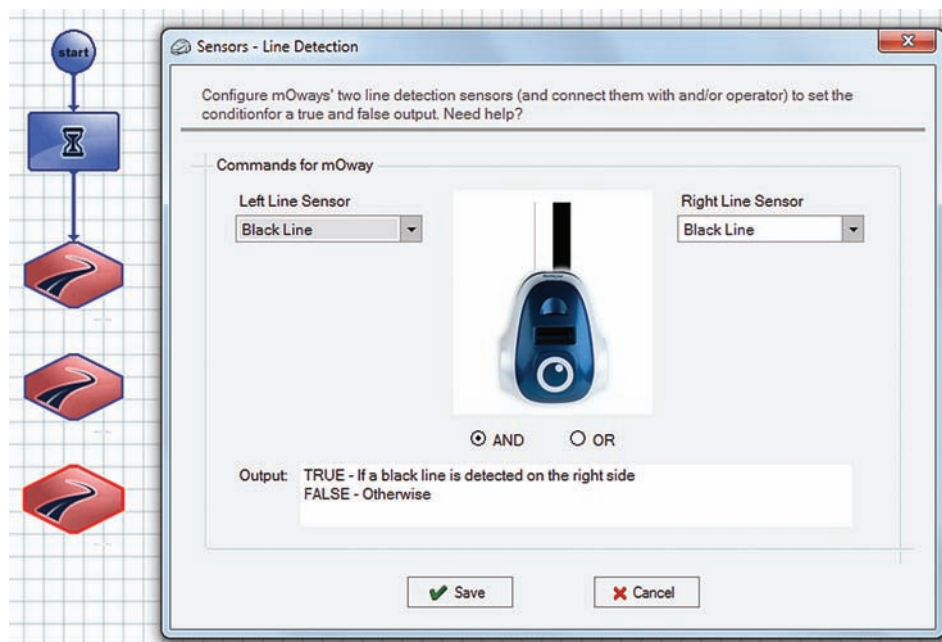


According to the value of the line sensors, we will have four different statuses and according to the status of the sensors, the robot will have to carry out a different action.

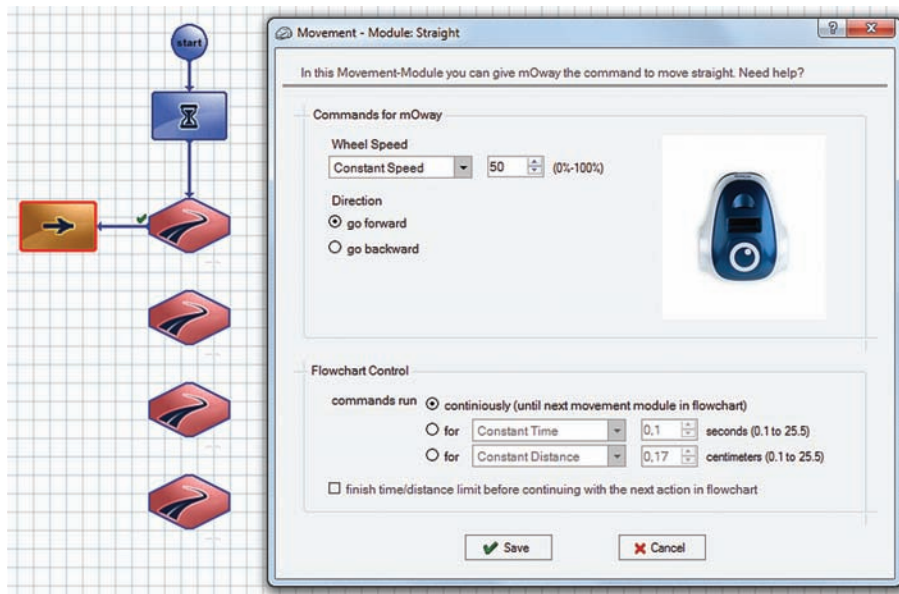
Left sensor	Right Sensor	Action
White	Black	Continue straight. We are in the correct position
White	White	Turn right. We have deviated to the left
Black	Black	Left turn. We are within the line
Black	White	Left turn. We are at the other end of the line

Once the 4 statuses have been analysed, we can begin to create the diagram. The first block will be a 2-second pause module. Secondly, insert the four conditions explained in the table above. The following images show the aforementioned conditions.





Once the conditions are placed, insert the actions associated with each of these. In the first place and according to the table above, the action to be taken will be a movement straight forward. In the second condition (white, white) we will turn to the right. In the remaining conditions we will turn to the left. The configuration of each module is shown in the images below.

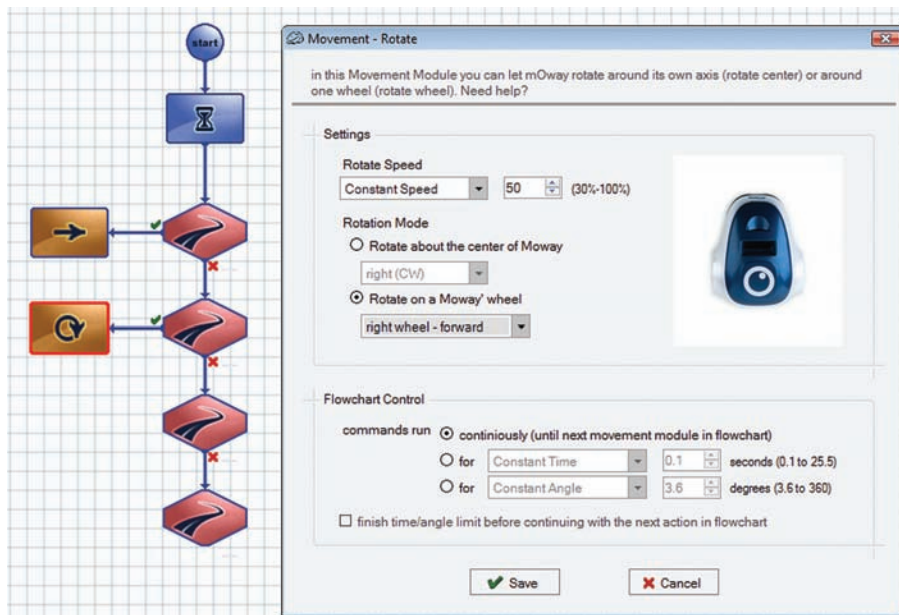


The flowchart on the left shows a sequence starting with a 'start' node, followed by a delay block, then a series of four red diamond-shaped decision blocks. The first decision block has a green checkmark and is connected to a blue arrow block. The subsequent three decision blocks have red X marks.

The 'Movement - Module: Straight' dialog box is open. It contains the following settings:

- Commands for mOway:**
 - Wheel Speed: Constant Speed, 50 (0%-100%)
 - Direction: ☒ go forward, ☐ go backward
- Flowchart Control:**
 - commands run: ☒ continuously (until next movement module in flowchart)
 - ☐ for Constant Time, 0.1 seconds (0.1 to 25.5)
 - ☐ for Constant Distance, 0.17 centimeters (0.1 to 25.5)
 - ☐ finish time/distance limit before continuing with the next action in flowchart

Buttons: Save, Cancel

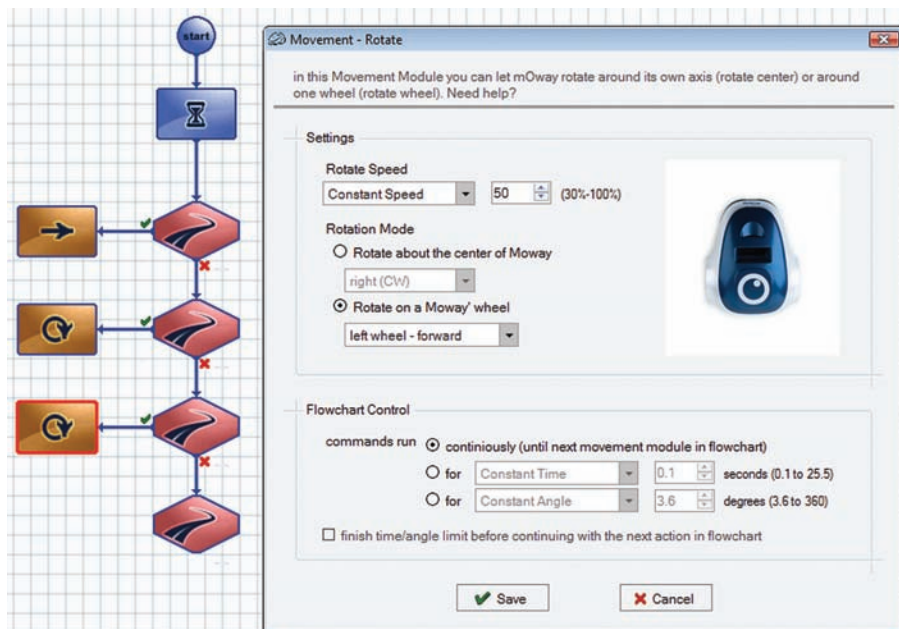


The flowchart on the left shows a sequence starting with a 'start' node, followed by a delay block, then a series of four red diamond-shaped decision blocks. The first decision block has a green checkmark and is connected to a blue arrow block. The second decision block has a green checkmark and is connected to a blue circular arrow block. The subsequent two decision blocks have red X marks.

The 'Movement - Rotate' dialog box is open. It contains the following settings:

- Settings:**
 - Rotate Speed: Constant Speed, 50 (30%-100%)
 - Rotation Mode: ☐ Rotate about the center of Moway, ☒ Rotate on a Moway' wheel
 - right (CW) dropdown menu
 - right wheel - forward dropdown menu
- Flowchart Control:**
 - commands run: ☒ continuously (until next movement module in flowchart)
 - ☐ for Constant Time, 0.1 seconds (0.1 to 25.5)
 - ☐ for Constant Angle, 3.6 degrees (3.6 to 360)
 - ☐ finish time/angle limit before continuing with the next action in flowchart

Buttons: Save, Cancel

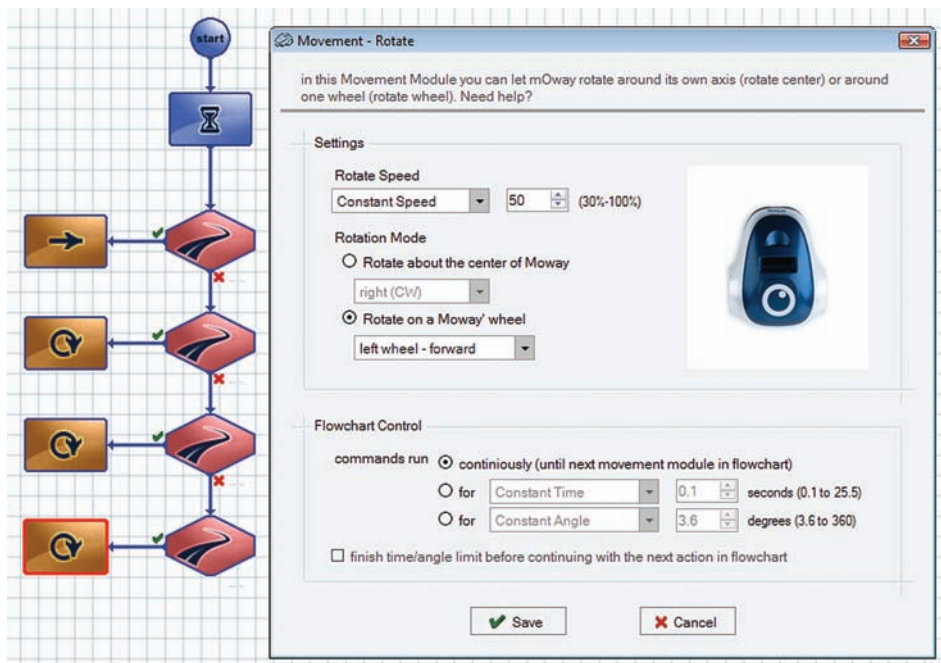


The flowchart on the left shows a sequence starting with a 'start' node, followed by a delay block, then a series of four red diamond-shaped decision blocks. The first decision block has a green checkmark and is connected to a blue arrow block. The second decision block has a green checkmark and is connected to a blue circular arrow block. The third decision block has a green checkmark and is connected to a blue circular arrow block. The fourth decision block has a red X mark.

The 'Movement - Rotate' dialog box is open. It contains the following settings:

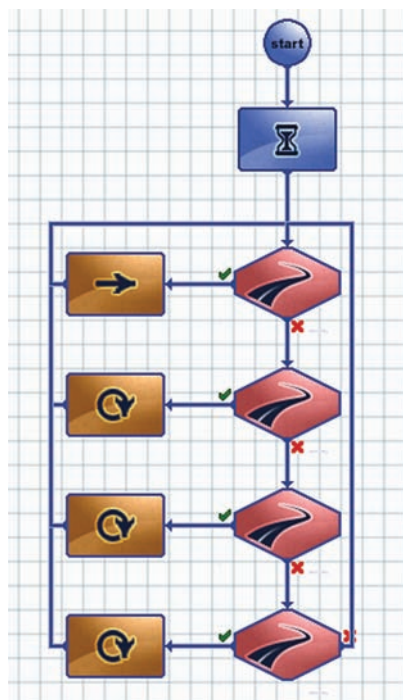
- Settings:**
 - Rotate Speed: Constant Speed, 50 (30%-100%)
 - Rotation Mode: ☐ Rotate about the center of Moway, ☒ Rotate on a Moway' wheel
 - right (CW) dropdown menu
 - left wheel - forward dropdown menu
- Flowchart Control:**
 - commands run: ☒ continuously (until next movement module in flowchart)
 - ☐ for Constant Time, 0.1 seconds (0.1 to 25.5)
 - ☐ for Constant Angle, 3.6 degrees (3.6 to 360)
 - ☐ finish time/angle limit before continuing with the next action in flowchart

Buttons: Save, Cancel



It should be emphasised that in this program, all rotations are made on the wheel and not on the centre. Rotating on the centre is the same as moving forward with one motor and moving back with the other. If we were to use this rotation, the robot would not move forward along the line and would remain in the same place.

Now it only remains to join the different modules and conditions with the arrows. The true path of the conditions will be joined to the movement module associated with each of these. The output path of the modules will return to the beginning of the loop. This can be done joining each movement module with the first conditional module. The false path will be joined to the next condition. The last condition will have a return path to the first one.



A partir del proyecto se pueden realizar mejoras en el programa para profundizar en la programación del robot:

Based on the project, we can make improvements in order to learn a lot more about how to program the robot:

- **Exercise 4.4.1.** Change the speed of the movements.
- **Exercise 4.4.2.** Modify movements according to the route. Example: smoother movements (use of the curve module) if the track does not have many angles.
- **Exercise 4.4.3.** Turn on the lower right hand or left hand red LEDs when the robot detects a specific value in the sensors.

What we have learnt



What **nested conditions** are.



How to **follow a black line** thanks to mOway's two line sensors.



New challenges

- Change the speed of the movements to get a faster line follower.
- change the rotating mode to the centre of the robot and see what happens. Does it improve or worsen the operation?

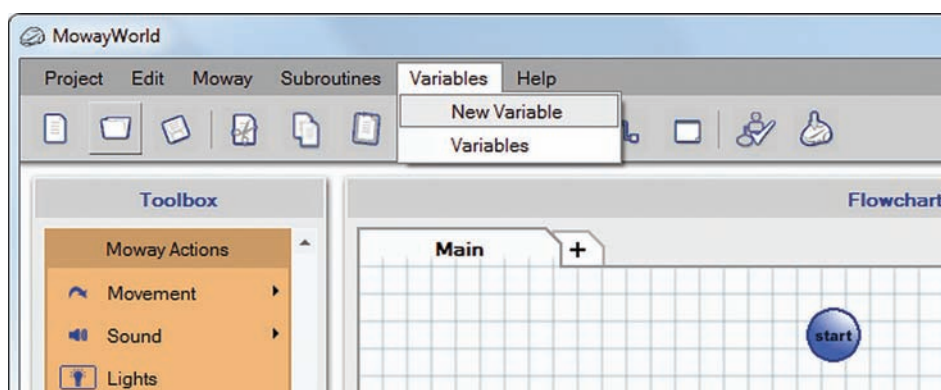


Check what you have learnt

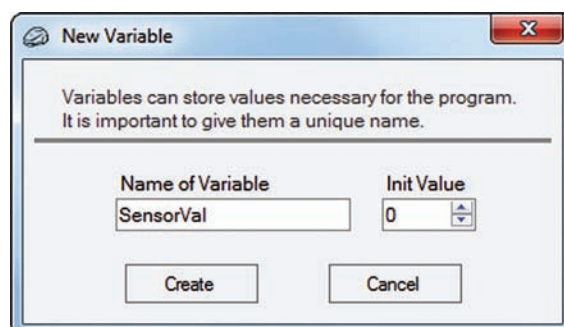
1. With which rotating mode is the robot capable of moving forwards while it turns?
 - a) Rotation on one wheel
 - b) Rotation on the center
2. Would the line follower operate using only one line sensor?

4.5 Exercise V. Variables

We are now going to begin to work with variables. A variable is a piece of program data that may change value while the program is running. These variables can be created from the toolbar.



A continuación se debe declarar un nombre y un valor inicial para esta variable.



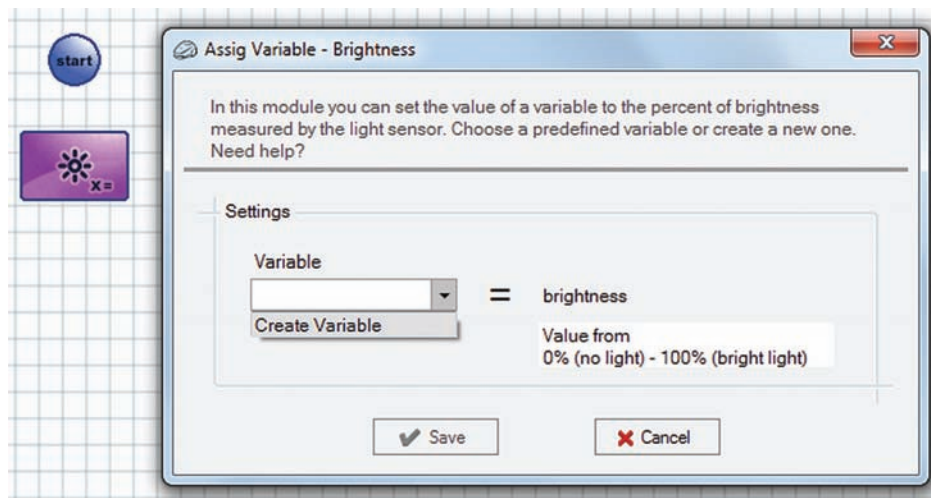
In addition, some modules allow to create a variable directly from the module. This option will be used on this exercise.

A variable may be used to store the value of a sensor. For example, we can store the value of the light and temperature sensor. We can also use the line or obstacle sensors, which we have used before in digital mode, as analogue sensors. In the case of obstacle sensors, we can measure the distance the object is away from us and in this way check whether we are moving away or moving closer to the objects.

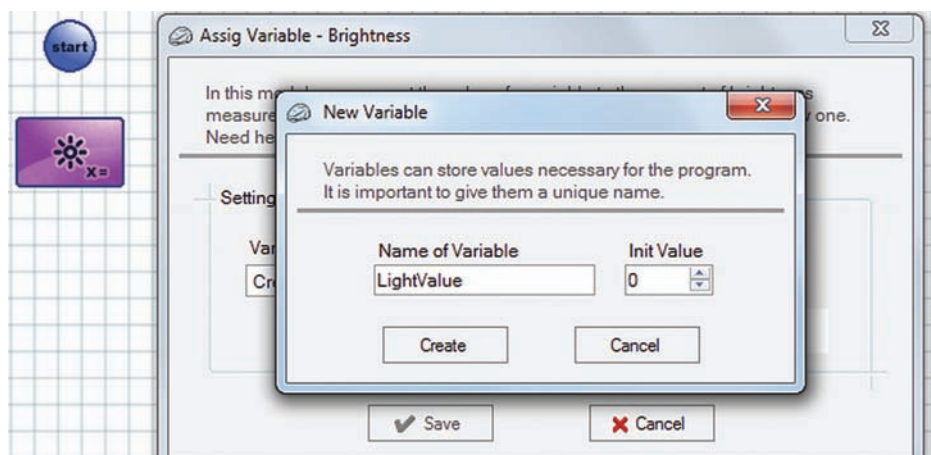
Variables can also be used as movement parameters. For example, we can configure the speed of a movement with the value of a variable.

By combining the storage of the value of a sensor together with the use of a variable as the parameter of a module, we will develop a program that moves the robot forward more quickly or more slowly in accordance with the ambient light. For example, when the robot enters a tunnel with less light it reduces its speed to prevent accidents.

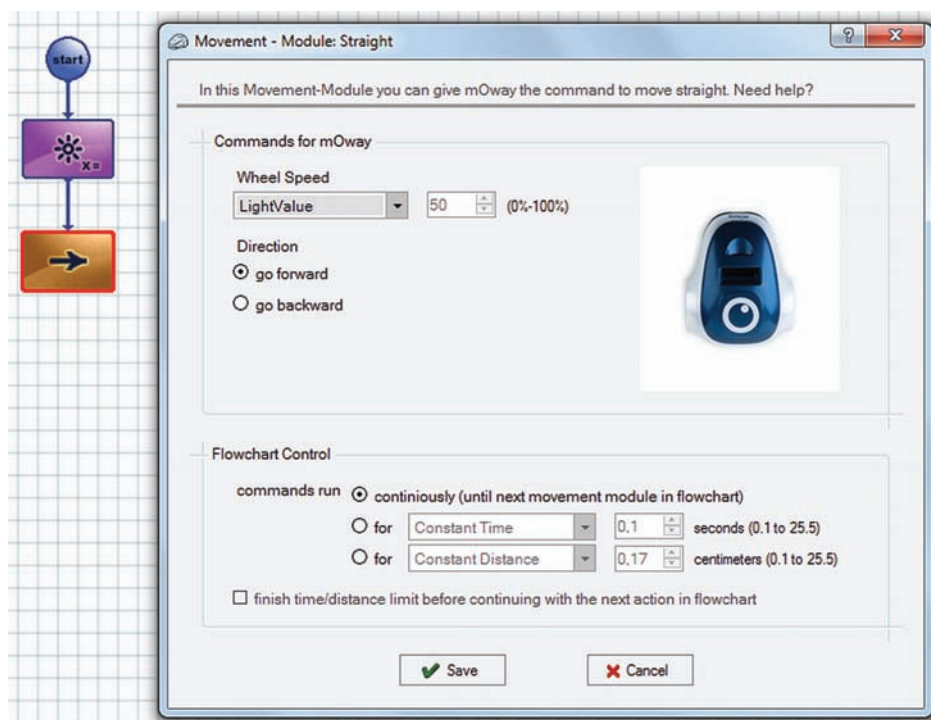
The program starts assigning the light sensor value to a variable. So that, we use the module "Data → Assign Variable → Brightness", and we choose the "Create Variable" option.



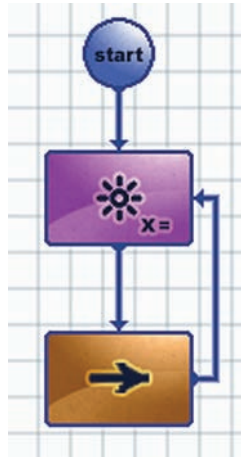
We name the variable "LightValue", which will store the light sensor value.



Then, we assign this variable value to the speed of the robot. This is achieved by selecting the "LightValue" variable in the "Wheel Speed" option of a Movement Straight module.



We need to check continuously the light sensor value in order to change the speed of mOway every time the light of the room varies. This is why the program runs in a loop.



- **Exercise 4.5.1.** Turn on the front led when the light sensor value is less than a certain value, using the "Ddata → Compare → Brightness" module.

What we have learnt



What a **variable** is and what it can be useful for.



How to **assign the value** of a sensor to a variable.



How to use a variable to **modify the speed** of the movement.



New challenges

- It turns on the robot's front LED when it is dark.



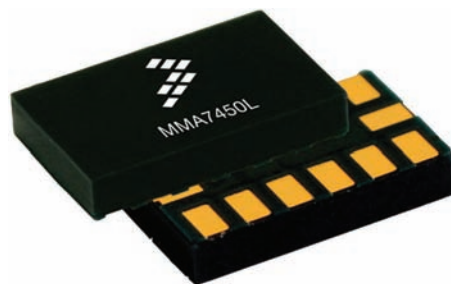
Check what you have learnt

1. If the programme were not a loop, in other words, if it finished in an "End" block, would the value of the sensor change?
2. Why is the use of a variable necessary in this programme?
 - a) In order to detect the level of luminosity and move forwards if it is dark
 - b) In order to be able to vary the rate of advance depending on the luminosity law.

4.6 Exercise VI. Accelerometers. Touch Parking

In this exercise we will use the accelerometer built into the mOway robot. The Freescale MMA7455L is a three axis accelerometer with I2C/SPI communications. Its acceleration detection range is from 2g to 8g (g = acceleration of gravity 9.8 m/s^2).

The operating principle of MEMS technology devices, accelerometers and inclinometers is based on natural convection thermal transfer. These devices measure internal changes in the heat transfer caused by the acceleration, offering significant advantages over the use of a traditional solid structure based on mechanical elements.



By measuring accelerations, we can measure the action of gravity to determine whether the robot is inclined or on a flat floor. We can also measure sharp speed variations such as those caused by an external impact or bump.

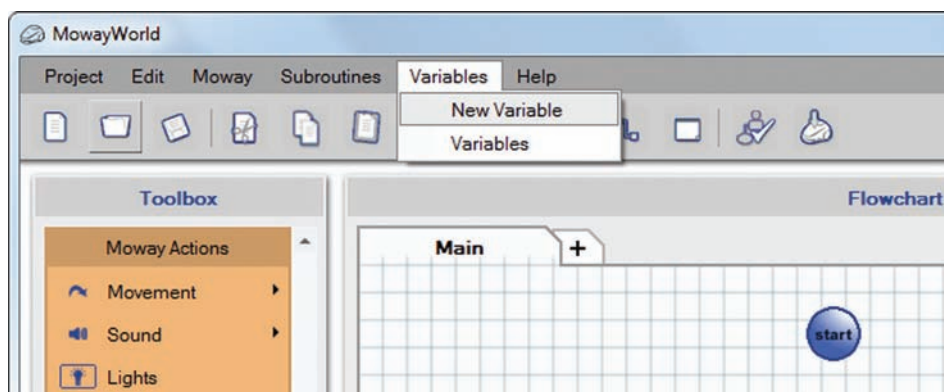


The different axes in which we can measure accelerations and their sign are shown in the illustration. The accelerometer returns an analogue signal on each axis. An acceleration of value 0 corresponds to a value of 127 in the MowayWorld variable. Values under 127 will correspond to negative accelerations and those of over 127 to positive accelerations.

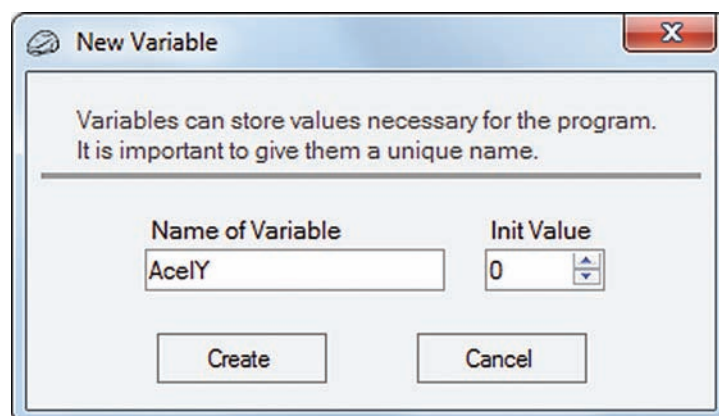
In this exercise we are going to use the accelerometer to detect accelerations in the y axis in order to detect an impact on the rear of the robot while it moves backwards. While mOway is moving backwards at a constant speed, its acceleration will be approximately zero. When the robot impacts, it experiences a positive acceleration in the y axis. This acceleration is positive because there is a variation from negative speed (backwards) to a zero value speed (when robot bumps), so that, the speed increases.

We will design a program that uses the front obstacle sensors to move backwards when we find an obstacle in front and the accelerometer to detect impacts at the rear.

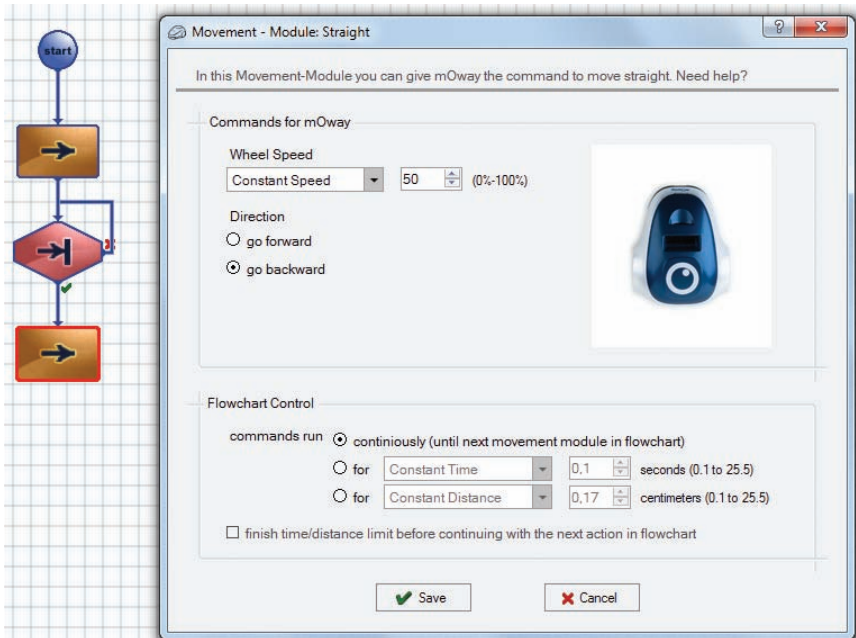
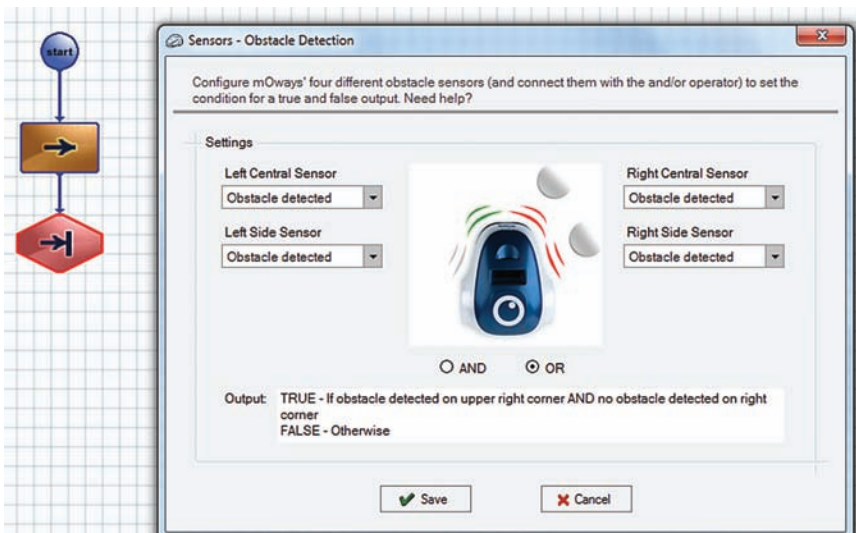
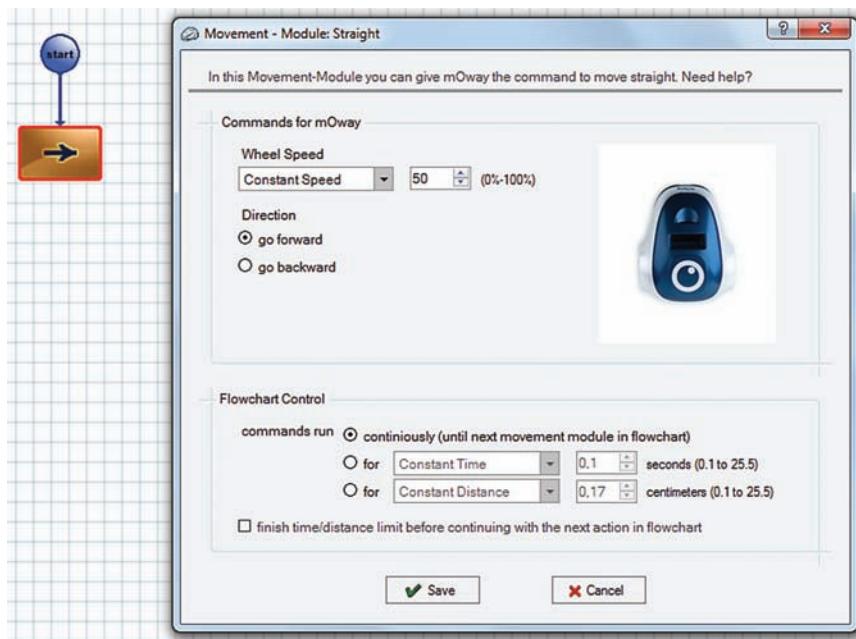
A new variable can be created from toolbar, as it is shown in the next image. In addition, some of the modules allow creating a new variable from them.



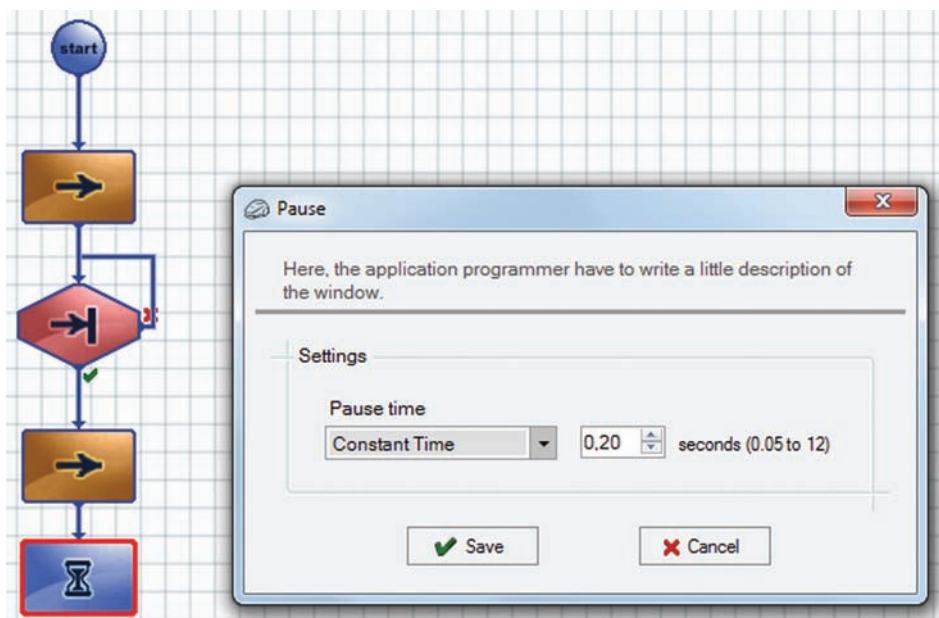
El nombre de la variable es "AcelY".



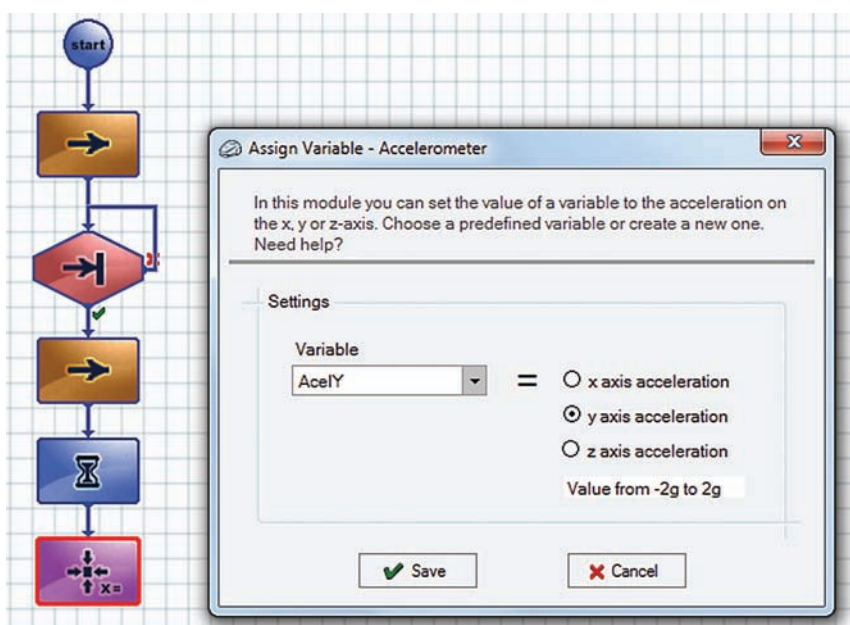
The program starts with mOway going straight forward. If the obstacle sensors detect an obstacle, the direction changes, so the robot goes backwards. This part of the diagram is shown below.

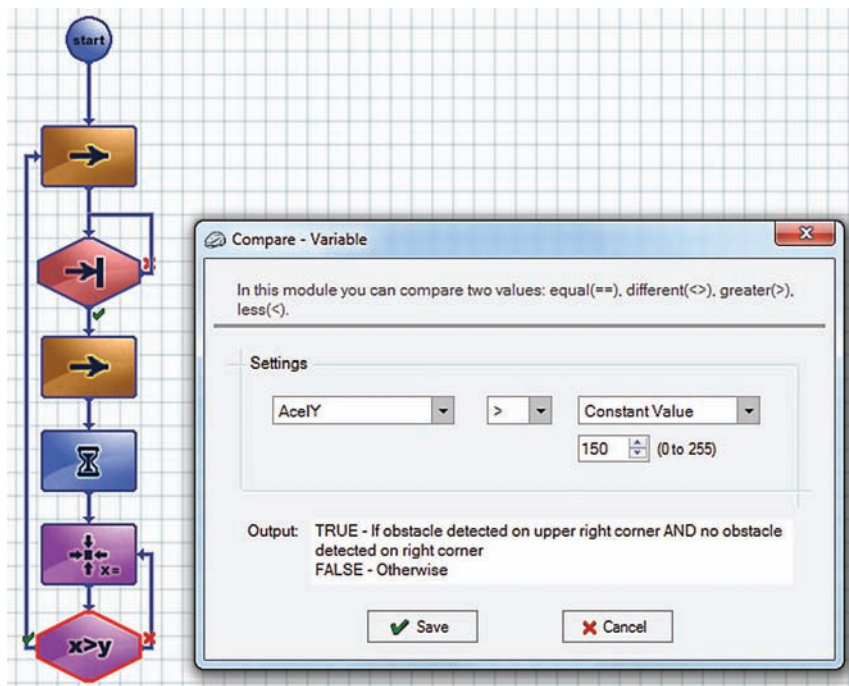


Una vez que cambiamos de dirección introducimos una pausa de 200 ms para que las variaciones de velocidad asociadas al cambio de dirección no se confundan con el choque del robot.



In order to detect the impact we have chosen a value of 150, which is the equivalent of 23 in the acceleration of the y axis. This value will depend on the robot and on the obstacles chosen and must be adjusted.

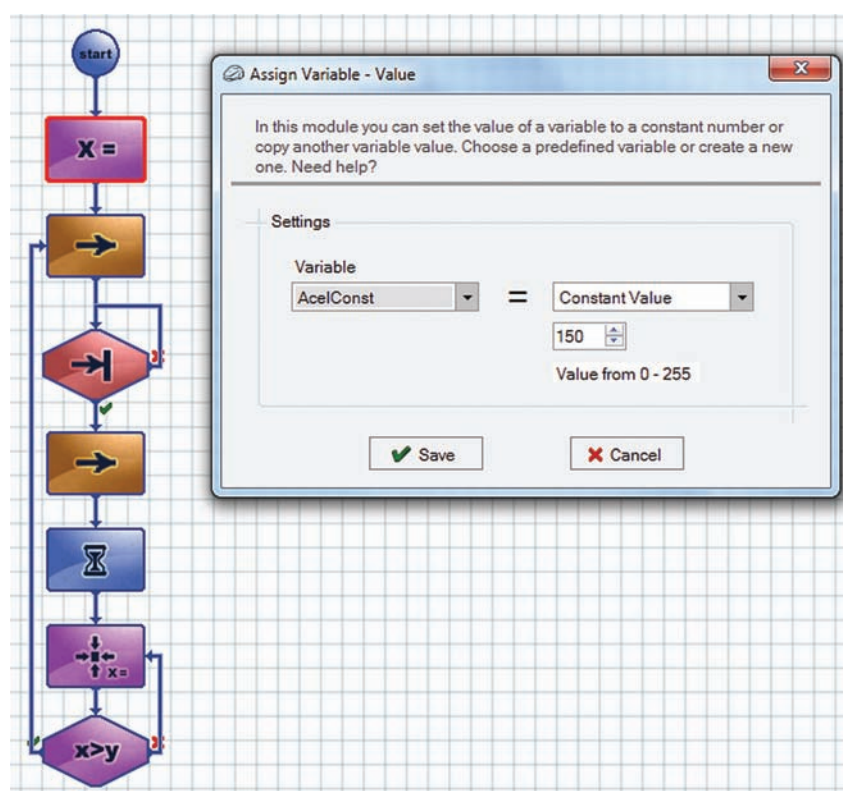




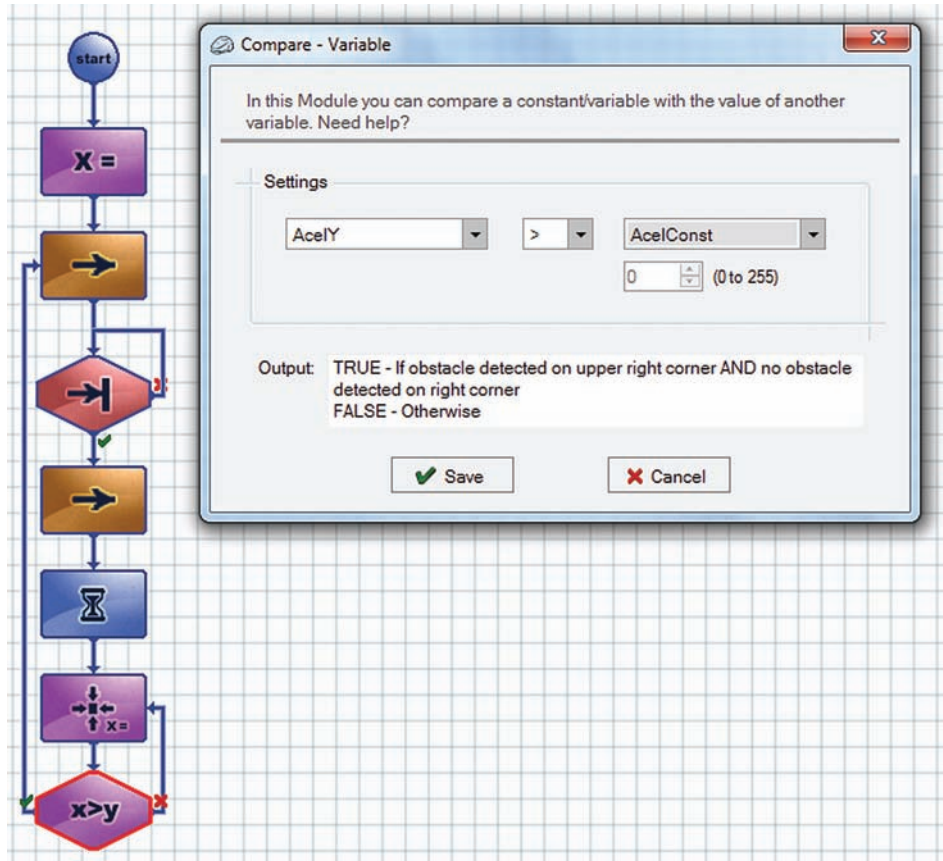
This program will make the robot move forward and backward alternatively in a sequential manner.

For a more flexible program, it is useful to include an Assign Value module, located in "Data → Assign Variable → Value". This allows to assign a constant value to the value in the Compare module, which makes easier the process of adjustment in case it was necessary.

This new variable can be created in the module mentioned before. In this example, a variable called "AcelConst" has been created with a value of 150.



Then we can compare the value of the acceleration sensor with this value. If it has to be adjusted, it can be done in the first module (Assign Value module).



- **Exercise 4.6.1.** Modify the program to use the accelerometer also in the case of obstacles in front.

What we have learnt



What an **accelerometer** is.



How to detect whether the robot has crashed.



New challenges

- Modify the programme to use the accelerometer for head-on crashes too, instead of using the obstacle sensors.



Check what you have learnt

1. What accelerometer value corresponds zero acceleration (0g)?
 - a) 0
 - b) 127
 - c) 150
2. Is the 200ms pause necessary?
 - a) Yes, to prevent the acceleration caused by the change of direction from being detected by the accelerometer
 - b) It is not necessary, the programme would operate in the same way
3. In order to detect the crash, would it be convenient to use the "Impact" block?
 - a) Yes, because it would detect the impact on crashing
 - b) No, because this block detects crashes on any of the axes and in this case we are only interested in the "y" axis

4.7 Exercise VII. The Copy

In this exercise we are going to introduce the use of the mOway Radio-frequency Module. This module allows the mOway robot to communicate with other robots and with a PC.

We will start by using the program described in Exercise II: The enclosure. The aim of this exercise is to use two mOway robots in such a way that one is enclosed within a space and the other repeats the movements of the first one.

mOway robot 1 will be placed on a white surface delimited by a black line and second robot will be placed near the first robot but outside the enclosed space, as shown in the following photograph.



The mOway robot will move forward in a straight line until it meets the black line. In this case, it will notify the other mOway by radio-frequency that it has found a line and will turn 126 degrees. Once it has completed the movement it will continue straight. The second robot will move forward in a straight line until it receives the order to turn by radio-frequency. In this case, it will turn 126 degrees and continue moving straight forward.

In order to do radio-frequency exercises it is necessary to use the RF expansion module. This module is plugged into the top of the two mOway robots, which we will use in the exercise as shown in the following diagram.

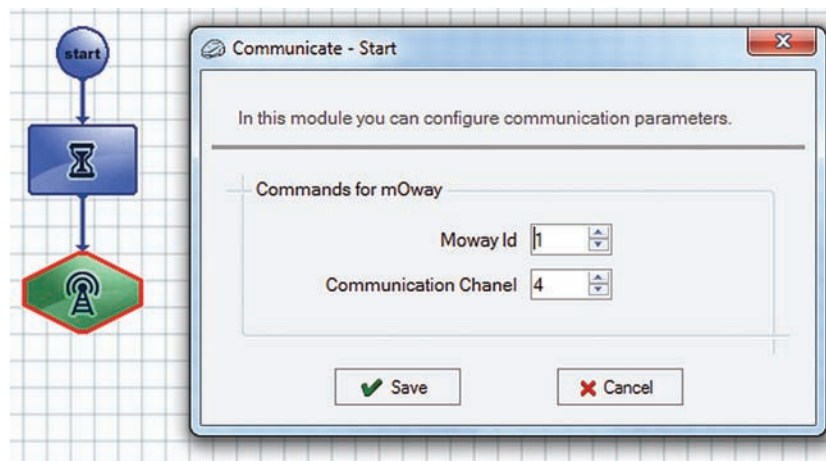
Students have no trouble working with RF modules as is it only necessary to introduce a few concepts such as channel and address. The channel in RF Communications with RF modules is similar to the radio transmitter concept. If we want to hear a particular radio station is necessary to tune to the frequency of the station in question. Therefore, if we want to robots to speak with each other, they must be on the same channel. In other words they must "tune to the same frequency". The channel is configured using the "RF Start" module.



The address concept is similar to the idea of a telephone number. When we want to talk to someone we dial their telephone number. In the same way, if a mOway robot wants to send information to another, it must “dial its telephone number”, in other words, its address. The mOway’s address is configured with the “RF Start” module.

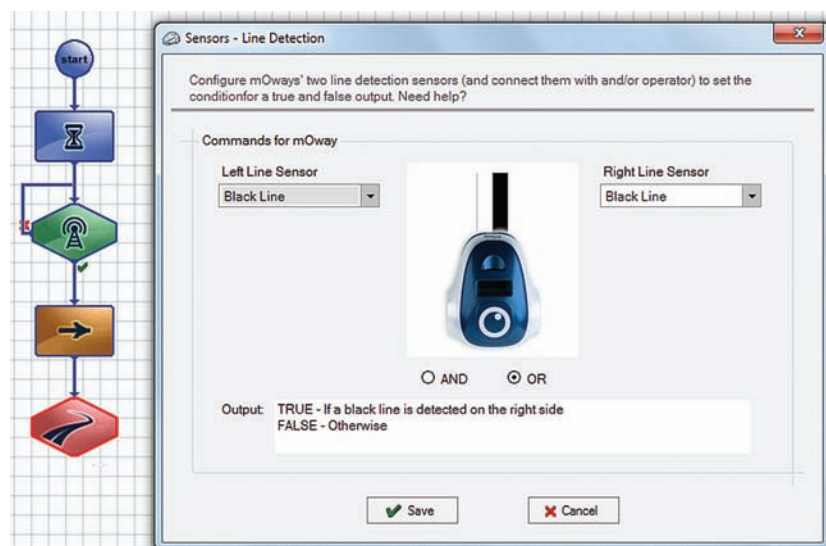
Let us begin therefore by programing the first robot. This robot will behave in the same way as the robot in Exercise II with the addition of RF communications.

Firstly, it is necessary to add the “RF Start” module. It is located in “Expansion → Communicate → Start”. In this module, we must select the channel and the address.

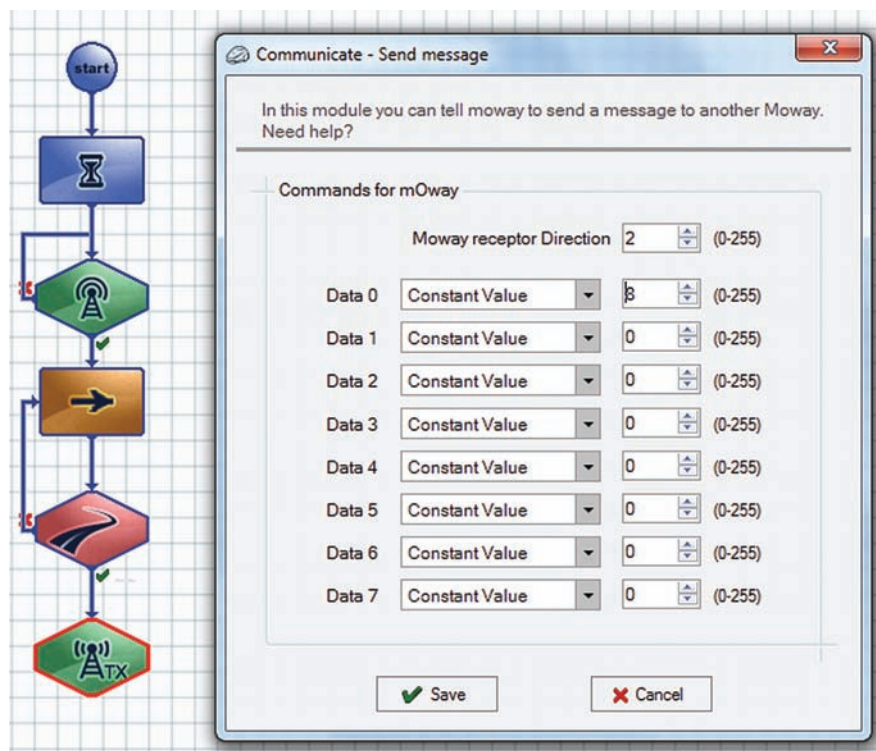


Remember that in order to enable two robots to talk to each other, they must be on the same channel and have one address. If several groups are being taught in the same classroom, it will be advisable to assign different channels to each one of the groups in order to avoid interferences. In this case, we will select channel 04 and the address 01 for the enclosed mOway robot.

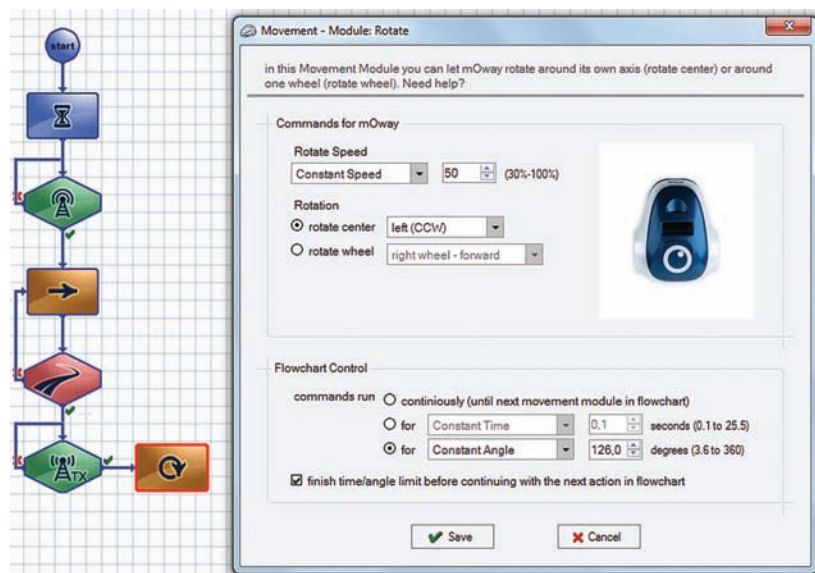
Once the radio-frequency module has been configured, we will insert the loop from exercise II. The robot moves straight forward and if its sensors detect a black line it carries out an action.



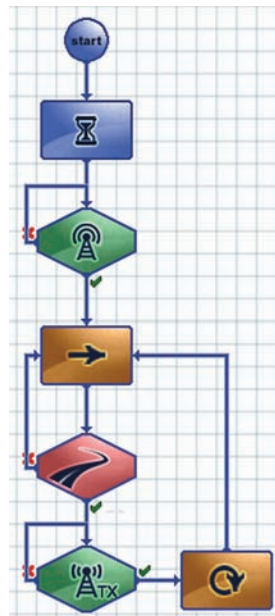
In this case we will add a transmission module, located in "Expansion → Communications → Send". This action of the robot is the one used to send information from one robot to another. This involves a condition as it is possible to send data or not, i.e., the information we send to the destination can be received or not by the recipient. Therefore, if it is not received ("false" path), we try to transmit again and if, on the other hand, the information is received ("true" path), we turn 126 degrees and continue on our way.



In the condition, the destination address is configured with the number "02" as this will be the address of the second robot. The data to be sent will be "08" which is the data we have defined to carry out the turn (in the second robot, we will analyse the data received and if we receive a "08", we will carry out the rotation).

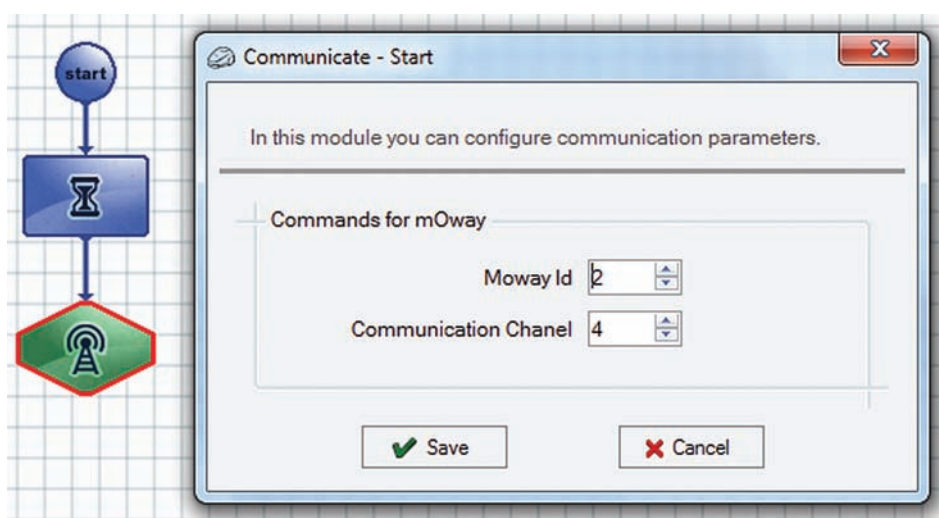


Once the program for the first robot has been completed, we will save this and download it into one of the robots.

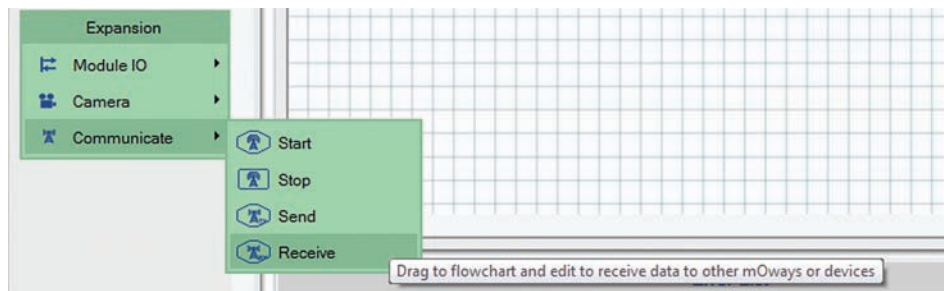


The programming for the second robot will have a similar structure. In this case, the robot will move forward in a straight line as long as it does not receive an "08" by radio-frequency. If this data is received, the robot will turn 126 degrees.

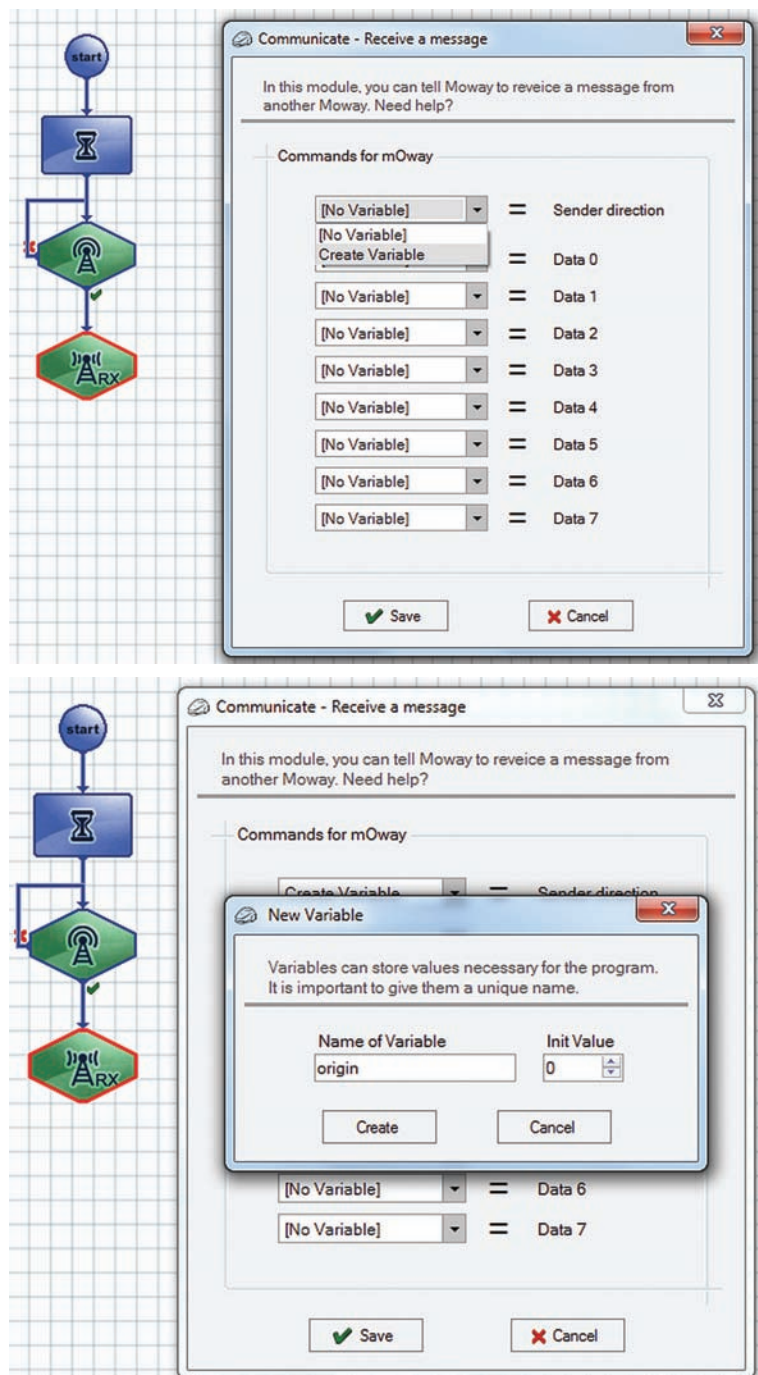
In the first step, we will configure the second robot in the "04" channel and with an address of "02".

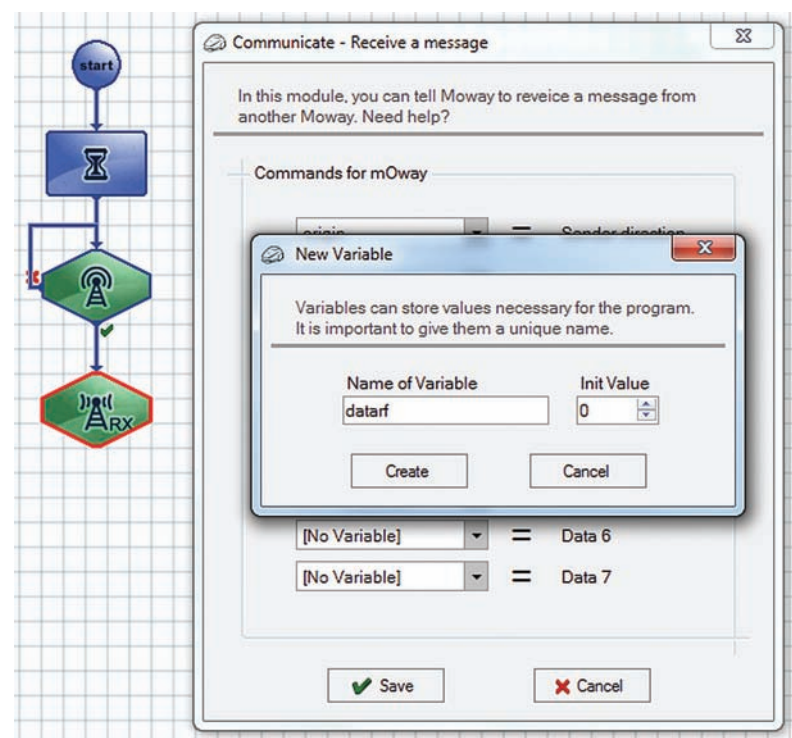
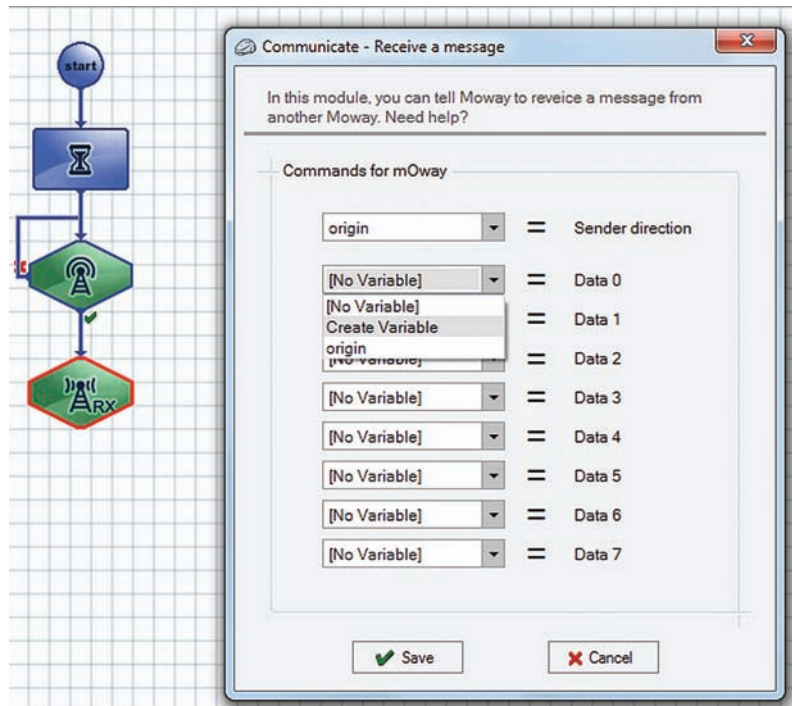


The next step will be to establish the "Receive" condition. As in the case of "Transmit RF", this action is a condition, the result of which is false if no data has been received and true if the information has been received. The Receive module is located in "Expansion → Communicate → Receive".

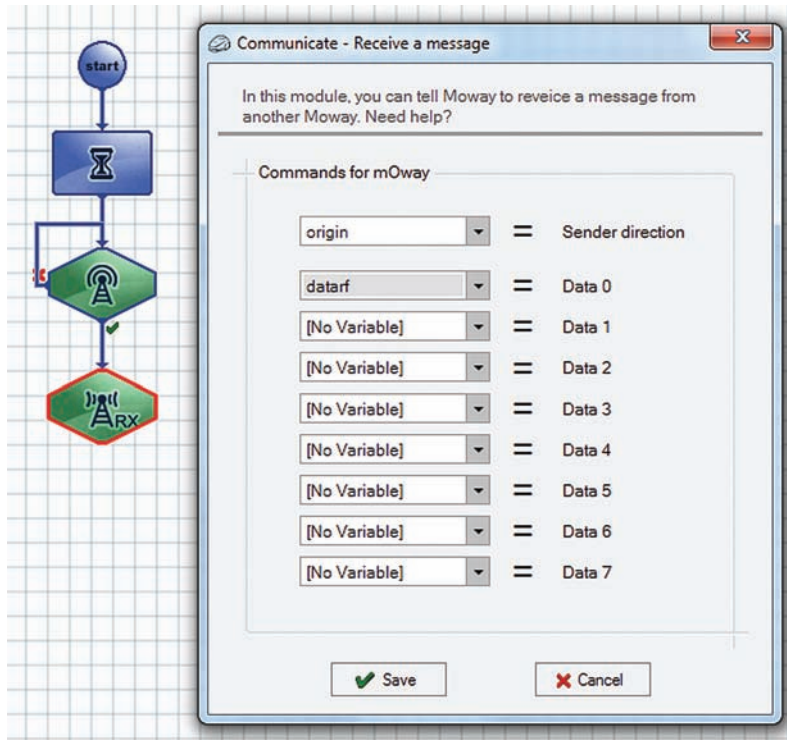


The information received must be stored somehow. In order to store this information, the "variable" concept is used. We will create two variables, "origin" and "dataarf". These can be created from Toolbar, or directly from the module, as it is shown on the images below.





Once these variables have been created, we introduce the condition "Receive" and configure this as in the diagram below, so that the address of the emitter of the information is stored in the "origin" variable and the first of the pieces of data received is stored in the "datarf" variable.

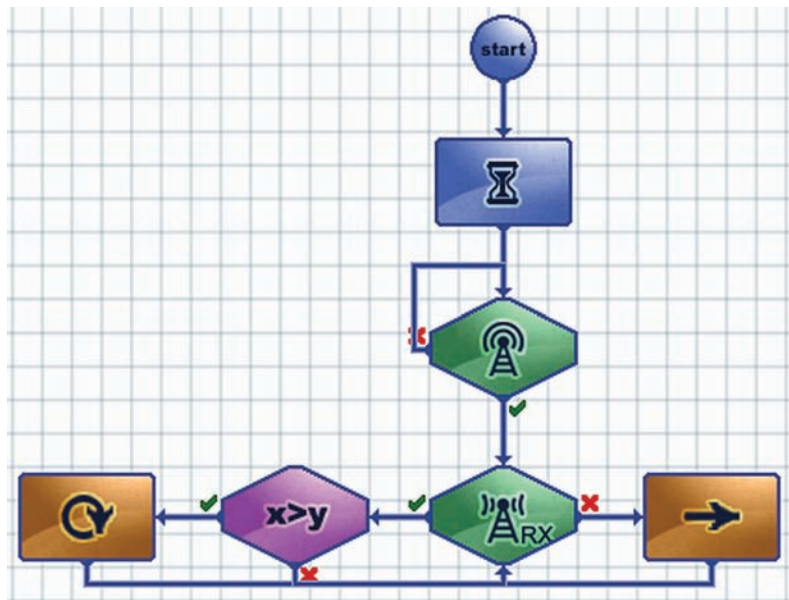


The piece of data sent by the first robot was "08". Therefore, we will check that the data received in the second robot is "08". To do this, we will use the "Compare data" condition. It is located in "Data → Compare → Compare data". We will select the "datarf" variable, the "==" condition and a constant value of 08.



If the condition is met (datarf=08) we will then carry out a rotation of 126 degrees. However, if this is false, we will then return to the beginning of the loop.

On the other hand, in the "Receive" condition, if we have not received any information then we will continue to move forward in a straight line.



Once the second program has been completed, we will store this and download it into the robot. Now we are ready to turn on the robots: robot 1 inside the space and robot 2 outside. In this exercise, we have worked with the line sensors and introduced the radio-frequency module and the use of variables.

Based on the project, we can make improvements in order to learn a lot more about how to program the robot:

- **Exercise 4.7.1.** Turn on the LEDs while the robot is turning
- **Exercise 4.7.2.** Modify the program in order to use obstacle sensors instead of line sensors.
- **Exercise 4.7.3.** Add a new "follower". Make two robots copy the movement of the first mOway.

What we have learnt



How **radio-frequency** communications operates.



What the **channel** and the **direction** are.



How to **communicate two mOway** robots by radio-frequency.



New challenges

- Turn on the LEDs while the robot is turning.
- Modify the programme in order to use obstacle sensors instead of line sensors.
- Add a new "follower". Make two robots copy the movement of the first mOway.



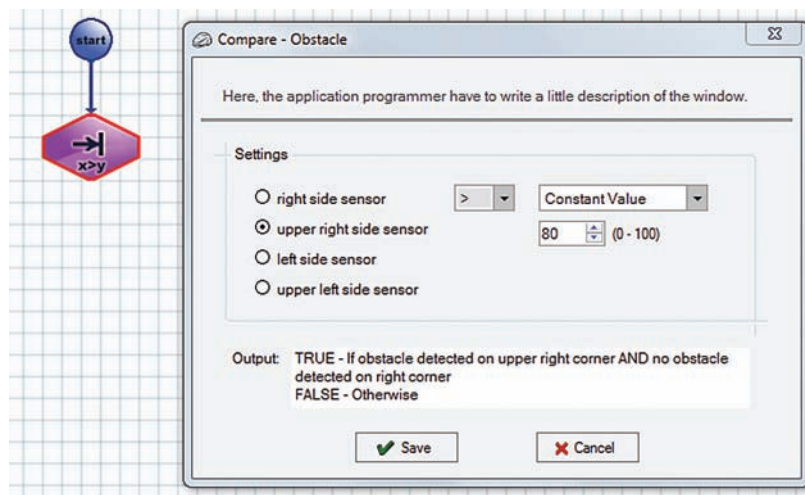
Check what you have learnt

1. How must the robots be set up to communicate by RF?
 - a) Both with the same direction and each with its own channel
 - b) Both with the same channel and each one with its own direction
 - c) Each one with its own direction and its own channel
2. Is it necessary to send the number "8" in any programme to make a turn?
 - a) Yes
 - b) No. Any number can be defined for making any action provided that both robots include this in their programme

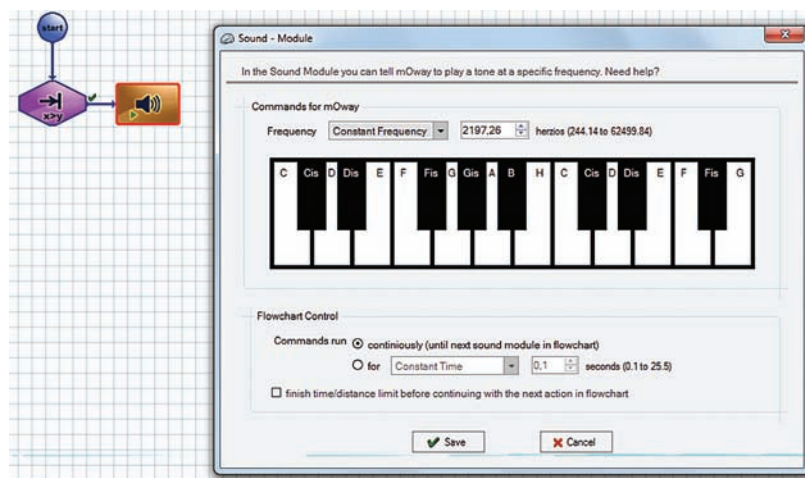
4.8 Exercise VIII. Speaker

In exercise VII, the mOway's built-in speaker is used to warn of the distance to an obstacle. The speaker allows reproducing a sound from 250 Hz to 5.6 kHz, which can be set in the Play Sound module. In this program, the closer the obstacle is, the higher the sound frequency is.

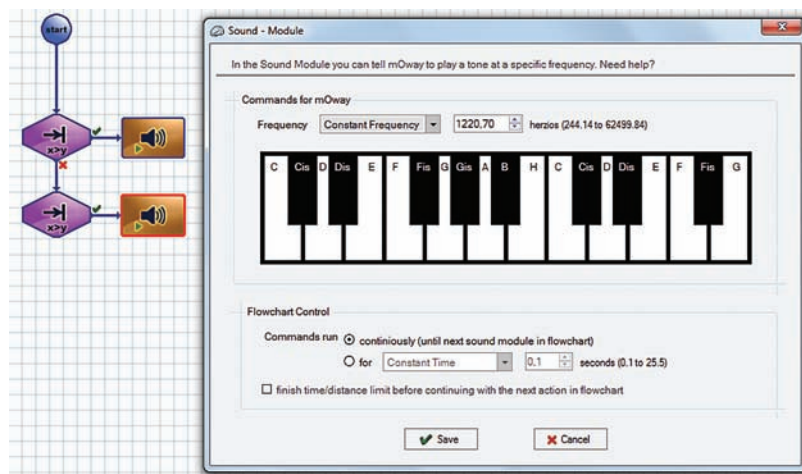
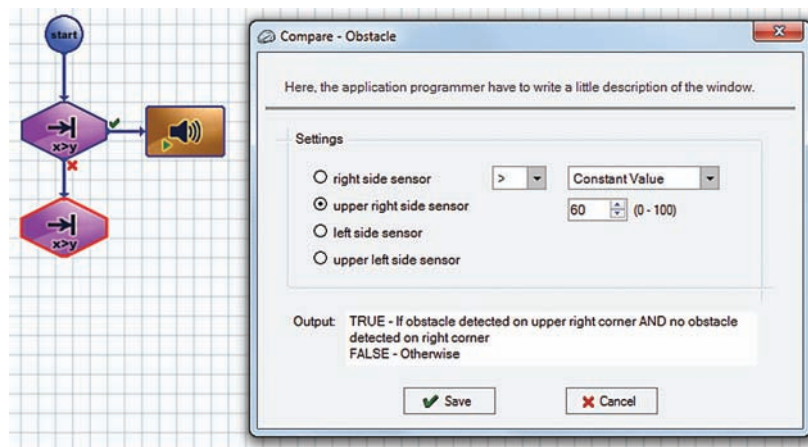
We compare the upper right obstacle sensor value with a constant value of 80.



If the value of the sensor is higher than 80, that means that the obstacle is very near and a high frequency sound is played by the speaker (for example, 2197Hz). The Speaker Sound module is located in "mOway Actions → Sound → Play".



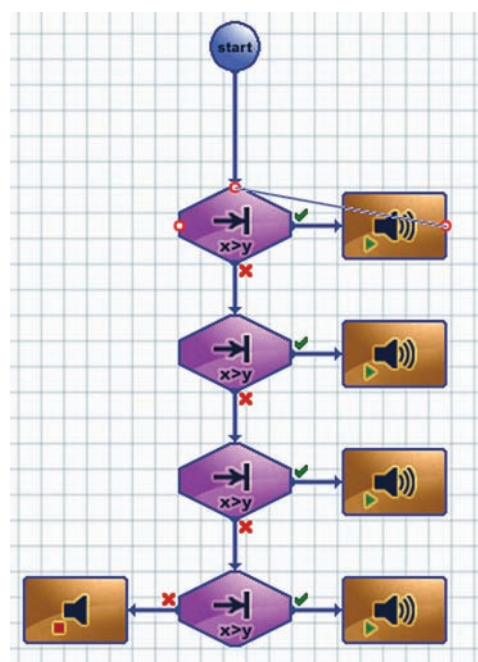
If the obstacle is not as near as the previous case (i. e., the obstacle sensor value is lower than 80 but higher than 60), the frequency of the sound will be lower (for example, 1220Hz).

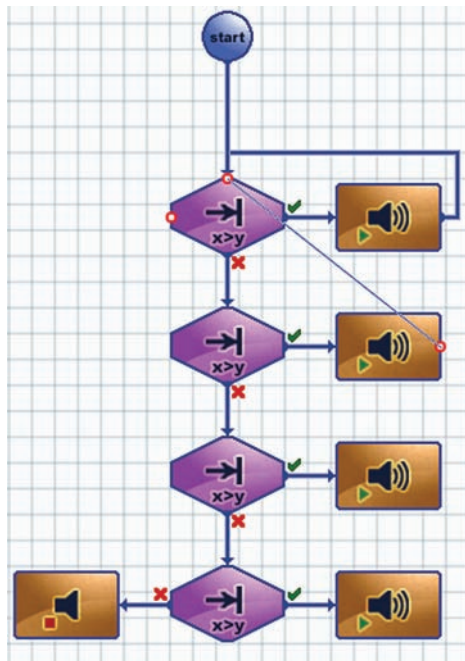


The rest of the values are explained now. If the sensor value is lower than 60 but it is higher than 40, the sound frequency is 732Hz. If the sensor value is lower than 40 but it is higher than 20, the sound frequency is 488Hz.

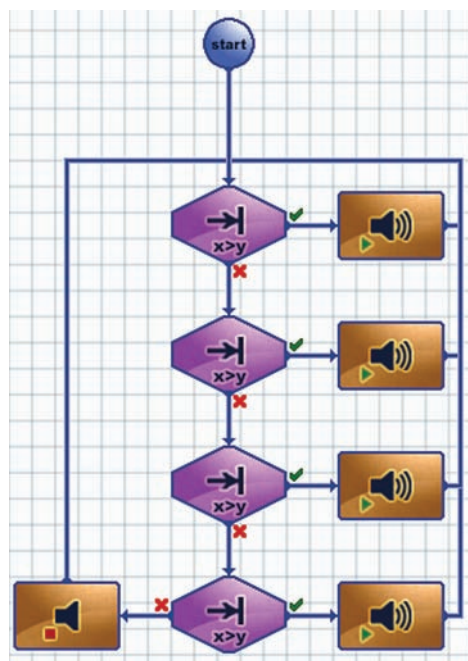
Otherwise, if the obstacle is far from the sensor, the speaker stops. The Speaker Stop module is located in "mOway Actions → Sound → Stop".

In order to set a loop, the arrows from the speaker modules have to be connected to the first conditional module. The images below show how to close the loop by means of arrows.





The complete diagram will look like this.



- **Exercise 4.8.1.** Add more resolution and frequencies to the sensor value.

What we have learnt



To detect the **distance measured** by the obstacle sensors.



To emit sounds with mOway's internal **loudspeaker**.



New challenges

- Add more resolution and frequencies to the obstacle sensor. For example, check the value of the sensor at intervals of 10.



Check what you have learnt

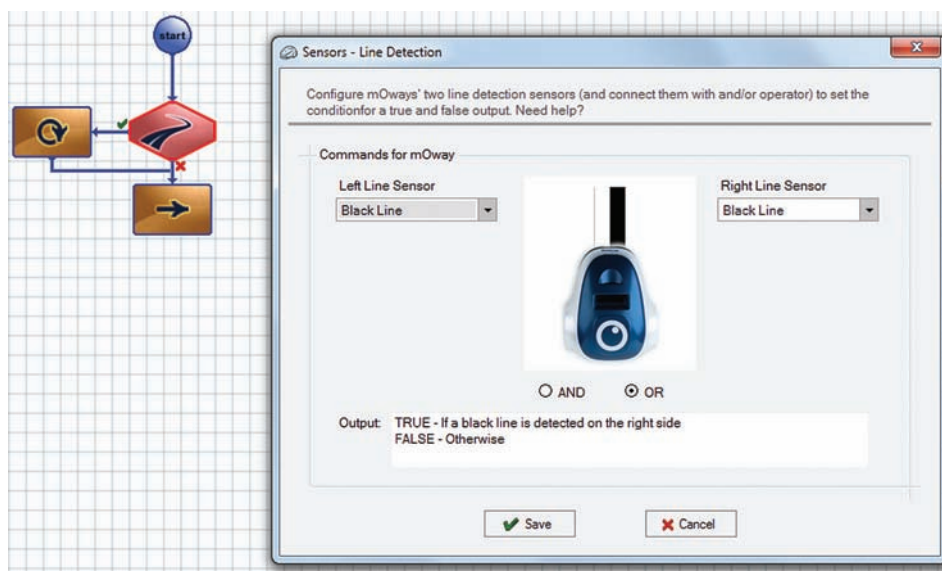
1. What is the unit of measure of the frequency?
 - a) Hertz (HZ)
 - b) Decibels (dB)
2. A sound is sharp if its frequency is:
 - a) High (many hertz)
 - b) Low (few Hertz)
3. What exactly get is obtained with the "Compare Obstacle" block?
 - a) Detect whether there is an obstacle or not
 - b) Obtain the distance at which an obstacle is found
 - c) Check the value of one of the obstacle sensors with another value
4. Which comparison is the correct one if we want the block output "Compare Obstacle" to be true when the sensor of value is equal to 20?
 - a) sensor == 20
 - b) sensor < > 20

4.9 Exercise IX. Defender / Fighter

In this exercise we will combine the "Enclosure" example of Exercise II and the use of mOway obstacle sensors. To do this exercise, we will use the mOway track. The robot will be placed within the black line circle and it will push obstacles out from the enclosed area. The robot will be enclosed into the circle.

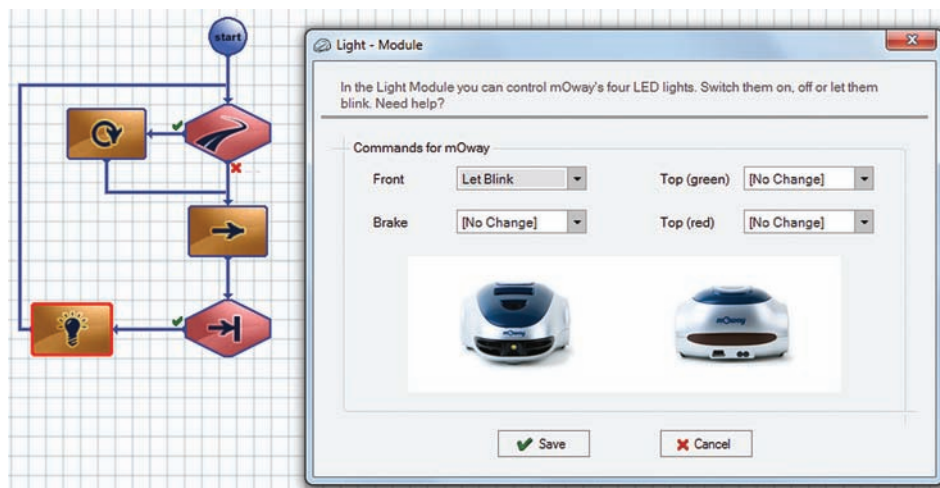
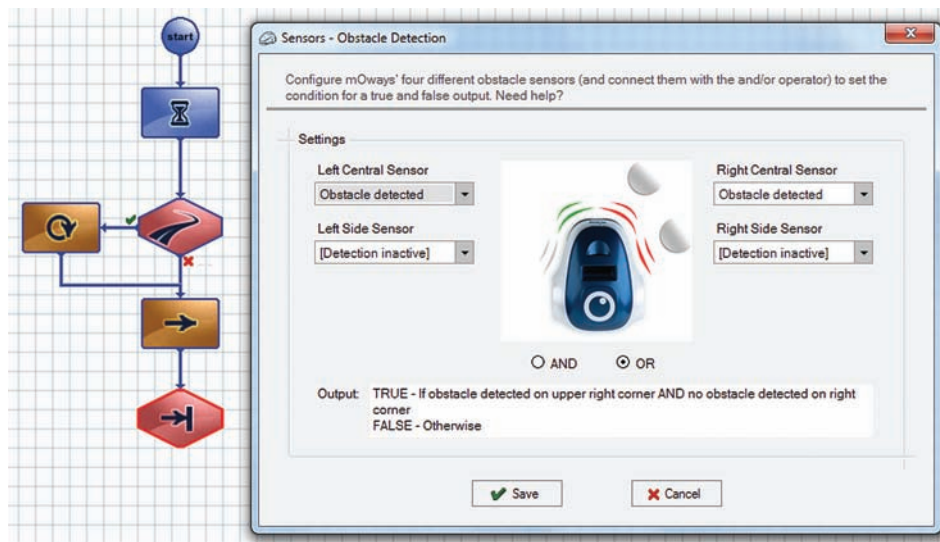


The first part of the flowchart keeps mOway robot into a black circle. The logic is basically the same as in the Exercise II (The Enclosure). Line sensors are checked to detect if the robot has reached the black line of the circle. If so ("true" condition), mOway turns 144 degrees in order to stay into the circle. If the black line is not detected ("false" condition), mOway goes straight forward.

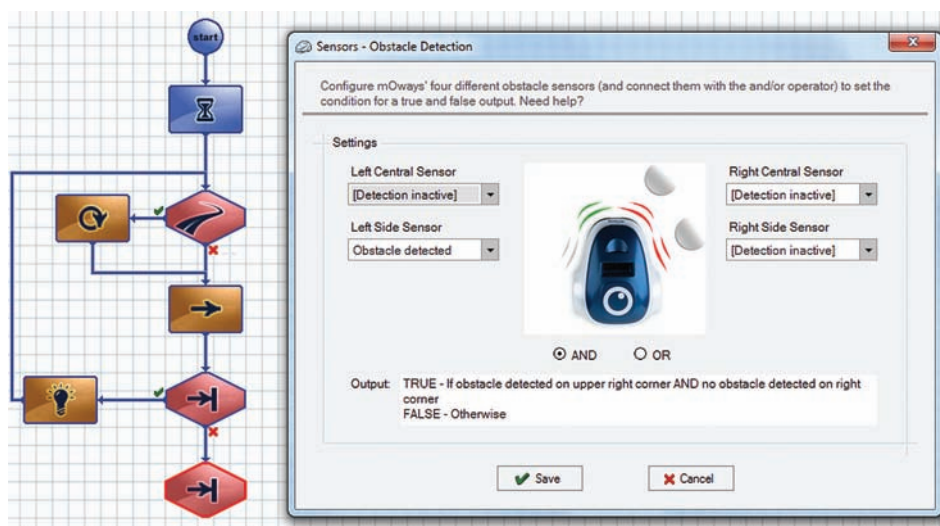


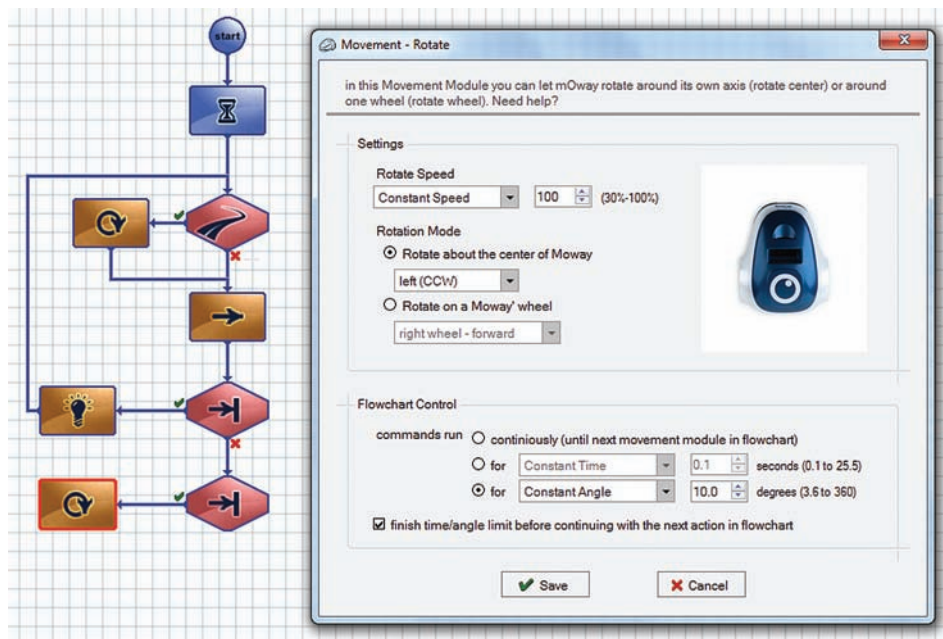
Second part of the flowchart makes the robot to detect an obstacle and to move it in order to push the obstacle out of the circle. It works as follows:

- The first conditional checks if the left central or right central sensor detects an obstacle and if so, the front LED blinks. The program will loop back to the start. So that, if mOway doesn't detect the black line, it goes straight forward towards the obstacle. In this conditional the OR option should be selected.

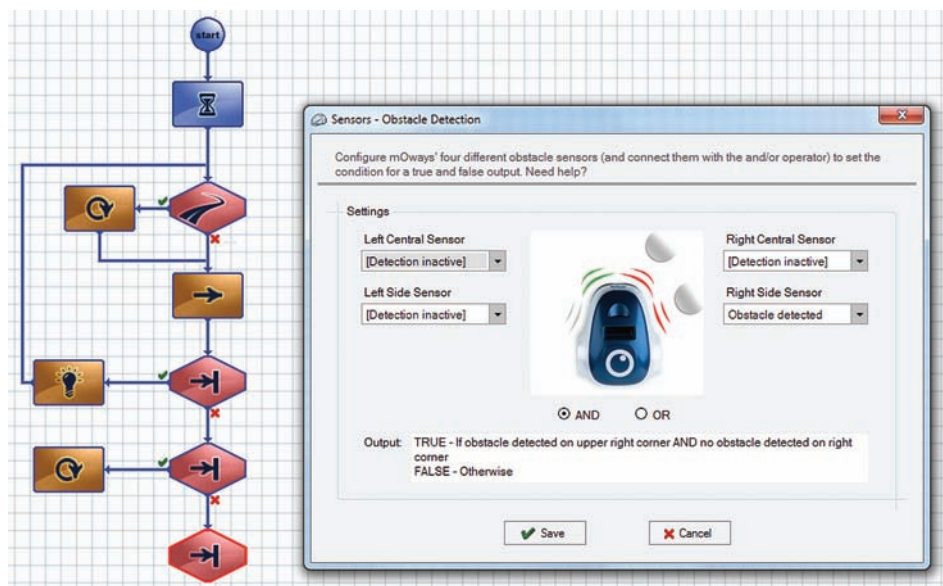


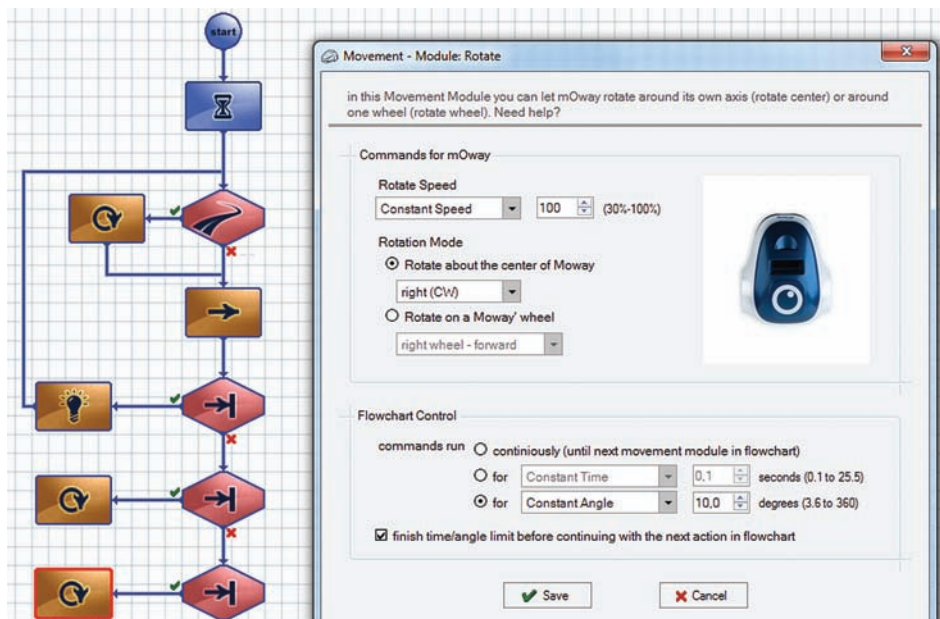
- Second part of the flowchart makes the robot to detect an obstacle and to move it in order to push the obstacle out of the circle. It works as follows:



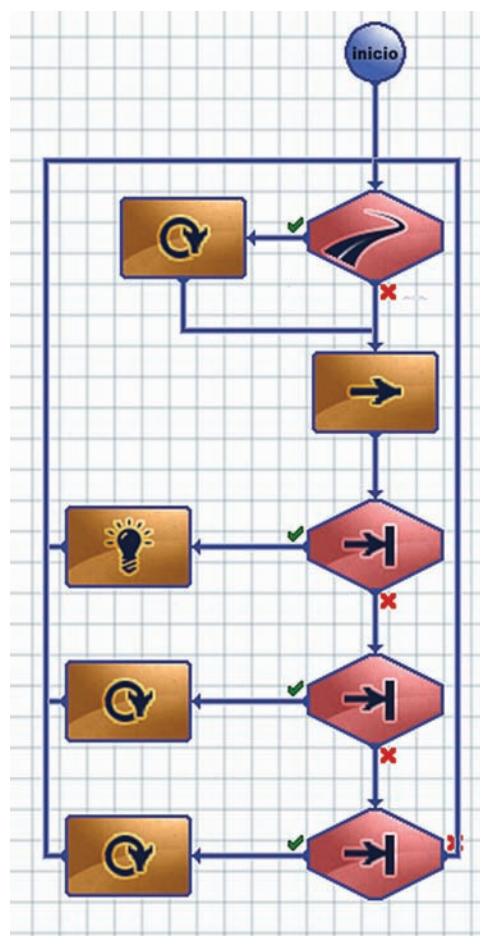


- The third obstacle conditional checks if the right sensor detects an obstacle. If so, mOway turns right for 10 degrees.





The finished diagram will look like this.



- **Exercise 4.9.1.** If two mOways are programed with this program, they will fight into a black line circle. The robot that pushes the opponent out of the circle wins the fight.

What we have learnt



Combine the **Line** and **Obstacles** blocks to do more complete exercises.



Use mOway's 4 **obstacle sensors** to find objects.



New challenges

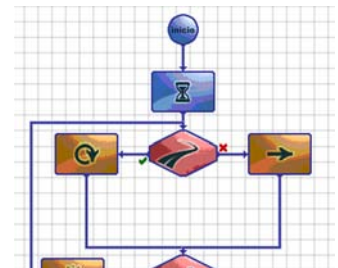
- If this programme is recorded in 2 mOways, they will fight within the black circle. The robot that pushes its opponent outside the circle wins the fight.



Check what you have learnt

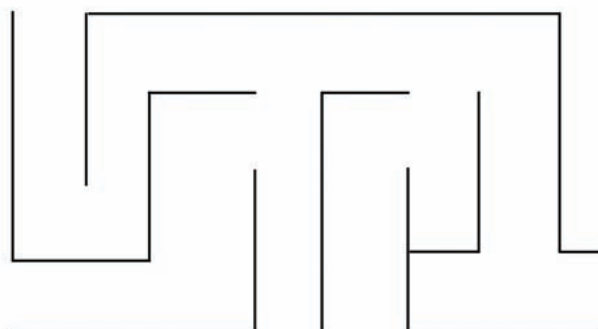
1. What would the robot do if the black line were detected in the following way?

- It would turn but it would not move forwards.
- It would move forwards but it would not turn.
- The operation is the same.

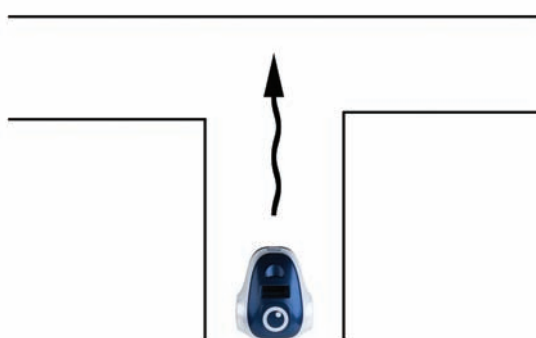


4.10 Exercise X. Maze

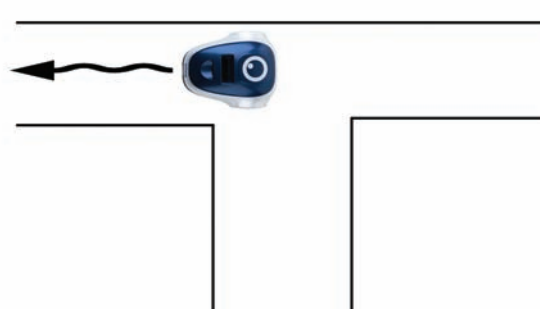
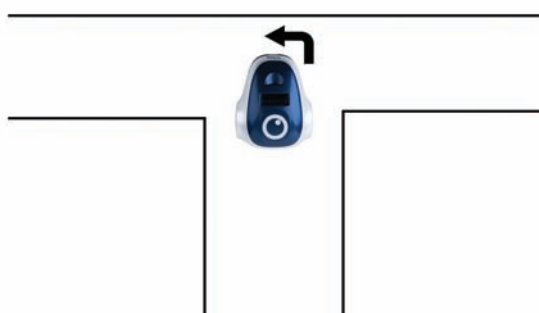
In this exercise mOway robot has to find the way out from a maze. This maze consists of straight aisles and 90 degrees turnings, made with "walls". These "walls" can be some kind of obstacle, such as cardboard sheets or boxes. An example of maze is shown below.



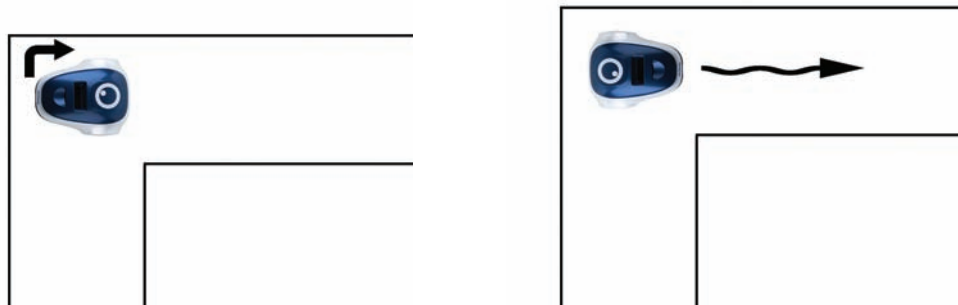
mOway robot will go forward following the walls of the maze, using the obstacle sensors on the sides to check if it is getting close to the walls. If mOway is near from the wall, it corrects its trajectory to avoid crashing against it.



When mOway detects that there is a wall in front of it, the robot firstly turns left for 90 degrees. Once it has turn, if no wall is detected the robot goes straight.



However, if it detects a wall that means that this is not the way out of the maze, so it turns right for 180 degrees and then it goes straight.

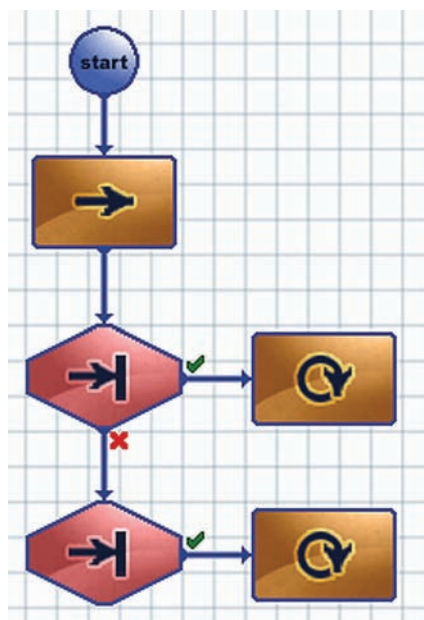


Firstly, the robot goes straight. We drag and drop a Straight module from “mOway Actions → Movement”.

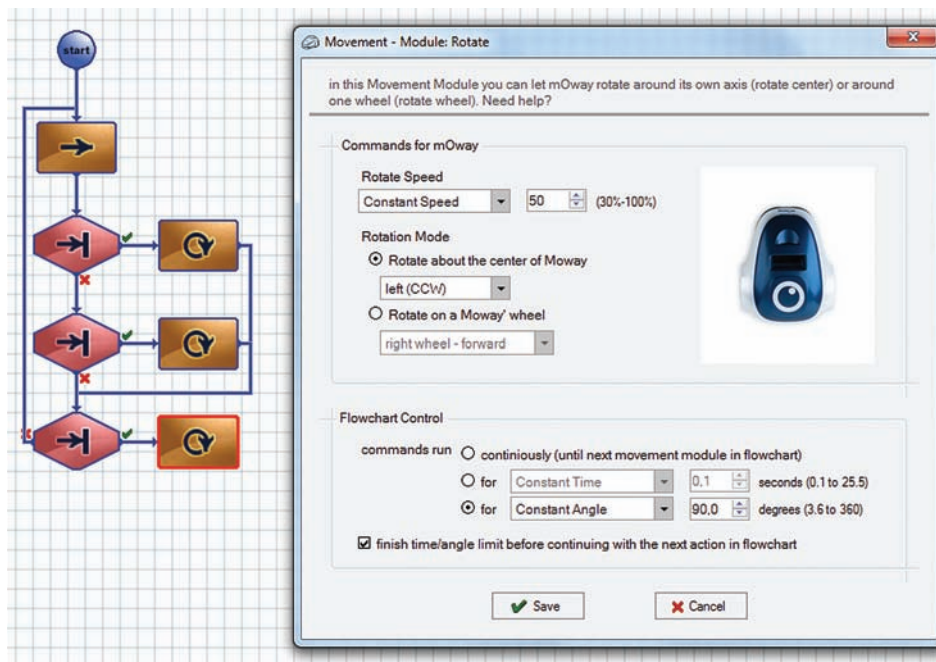


Then, it checks the side obstacle sensors to follow the maze walls. If the left sensor is active, the robot is near the left wall of the maze and it rotates right to avoid crashing against it. If the left sensor is not active, the robot checks the right sensor and if it is active, it rotates left.

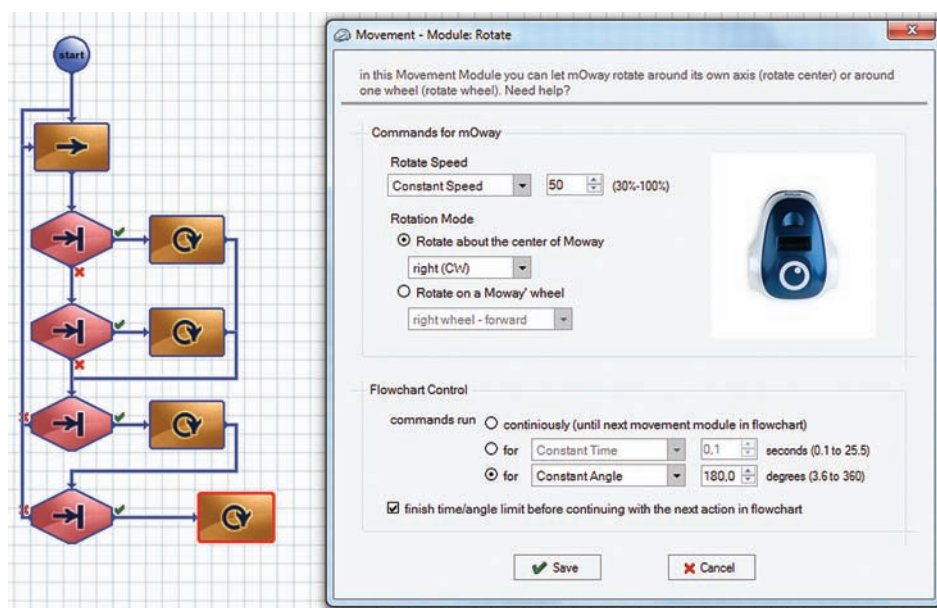
Both rotations have the “commands run continuously” option selected.



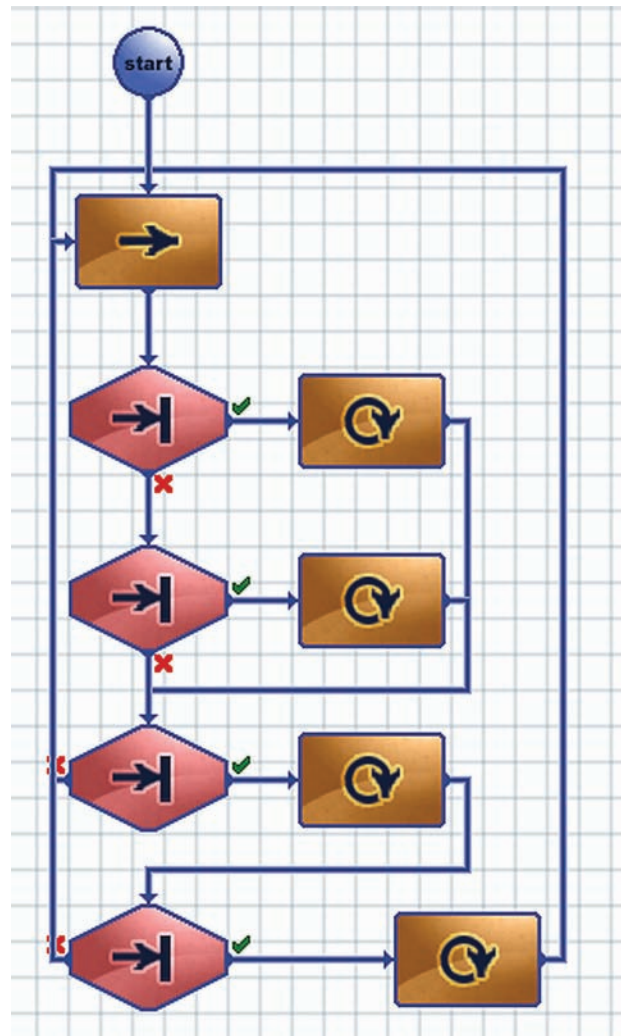
Once the side wall detection is done, mOway checks if there is a wall in front of it. This is carried out by central obstacle sensors. If one of them (left OR right central sensor) detects a wall, mOway rotates left for 90 degrees to try to find the way out. If none of the sensors detects a wall, the program loops back to the beginning and mOway goes straight until the next front wall.



If mOway has rotated left, central sensors are checked again. If none of them detects a wall, it means that this is the correct way, so the program loops back and mOway goes straight. However, if a wall is detected this means that there is no way out, so the robot rotates right for 180 degrees.



Once the mOway has found the way out, it goes straight (the program starts again).



What we have learnt



An **algorithm** to resolve labyrinths.



New challenges

- Make a different algorithm to resolve the labyrinth. For example, use one of the lateral sensors to follow a wall.



Check what you have learnt

1. Would it be possible to use this algorithm if the labyrinth were drawn with black lines instead of being walls?

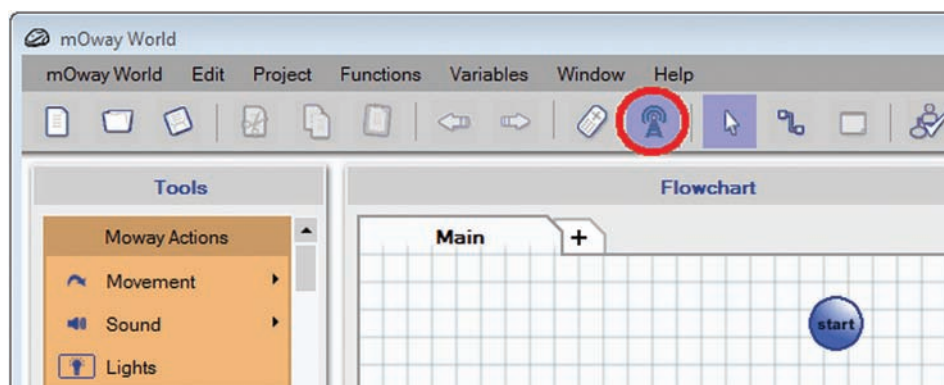
4.11 Exercise XI. Remote data station

In this exercise we are going to use the RFUSB module to enable mOway to communicate with the PC using MowayWorld.



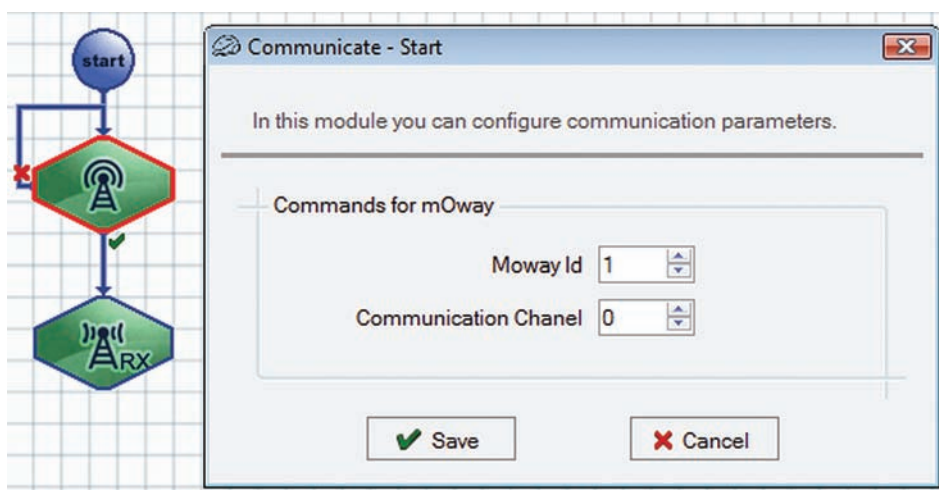
The program consists of sending a command from PC to mOway and when it is received, the robot sends the light sensor value to PC. This value is displayed on the Communications window of MowayWorld.

In the exercise VII (The copy) we used the RF module and learned the meanings of the concepts "channel" and "address". In the case of the PC, the RFUSB module operates exactly the same way as in the mOway robot. It is one more node of the network to which we must assign an address and indicate a channel in which it is going to operate. Sent and received data will be managed using the Communications window, that will appear pushing the button shown on the next image.

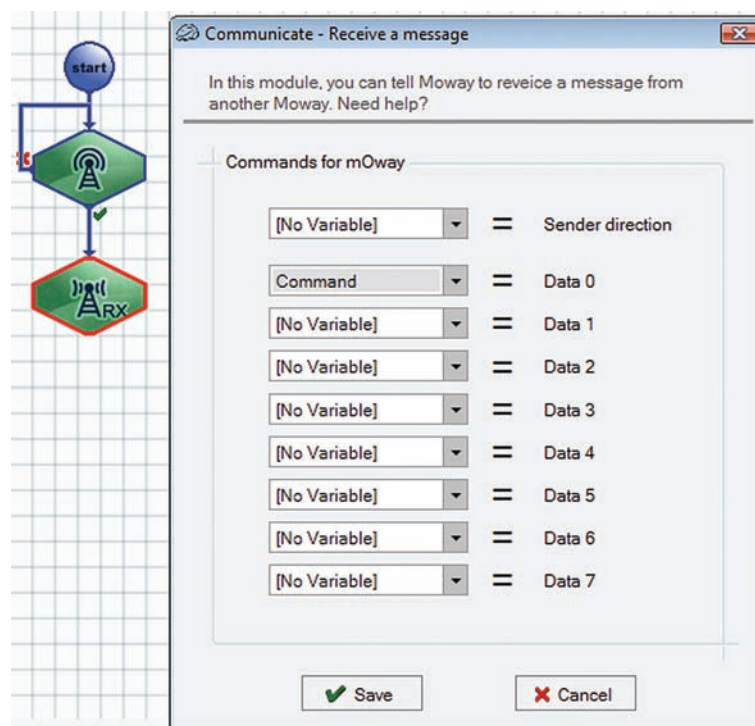
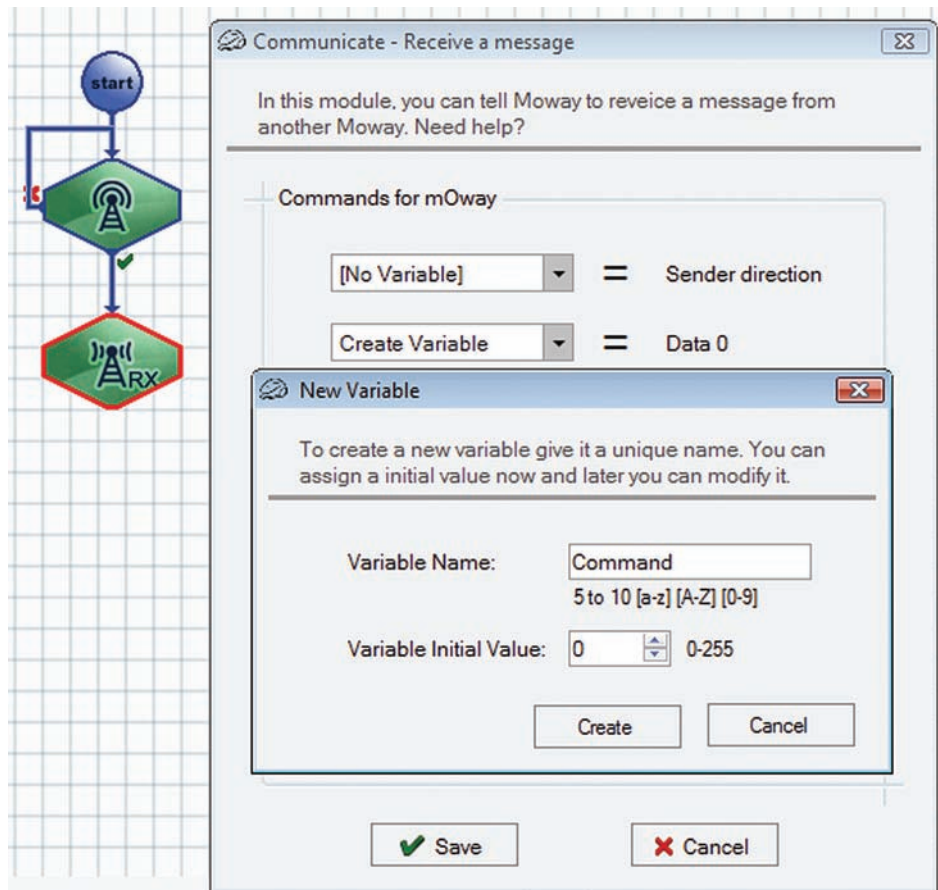


The robot will "listen" to requests from the PC, to which it will react by sending back information. If the frame sent by the PC has the chosen value to request for the light sensor value, the robot will send back the detected brightness.

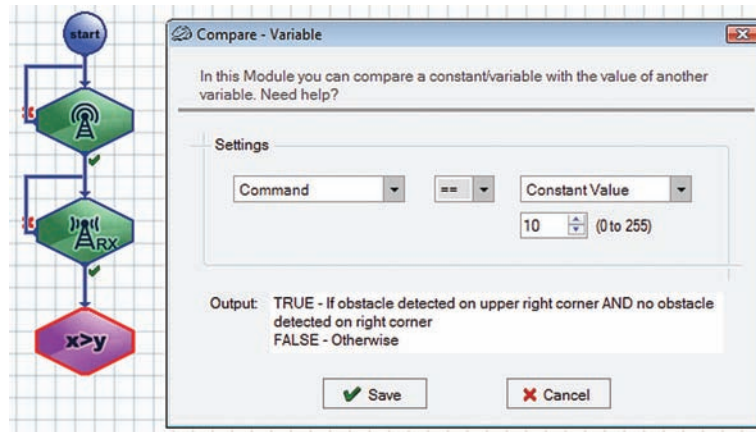
We will firstly configure the RF module of mOway with the address 1 and the channel 0 with "Expansion → Communicate → Start".



Next the RF Receive module "Expansion → Communicate → Receive" is configured to store the received data from PC in a variable named "Command". The data will be sent to PC in "Data 0" field.

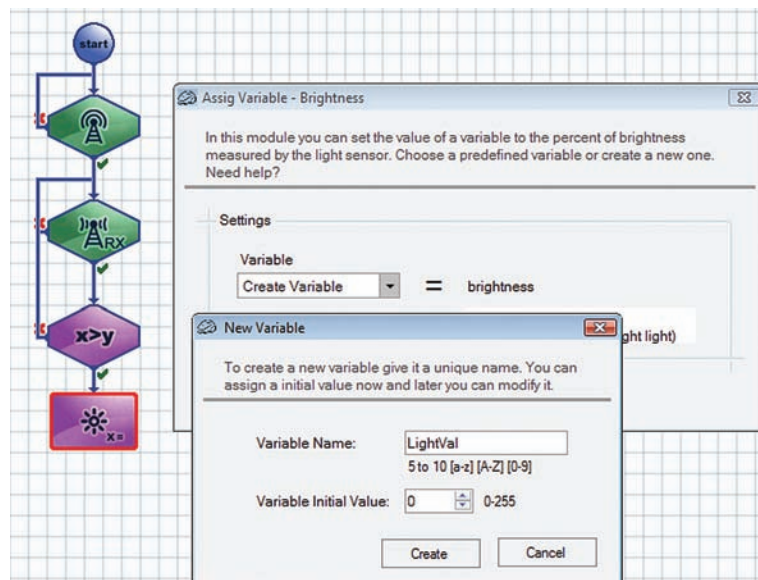


The program waits into a loop for a RF message to be received from PC. Once it is received, the condition is true and then the received data is compared to check if it is the expected command. In this case, a command with a value of 10 has been chosen (this is the value that has to be sent from PC).

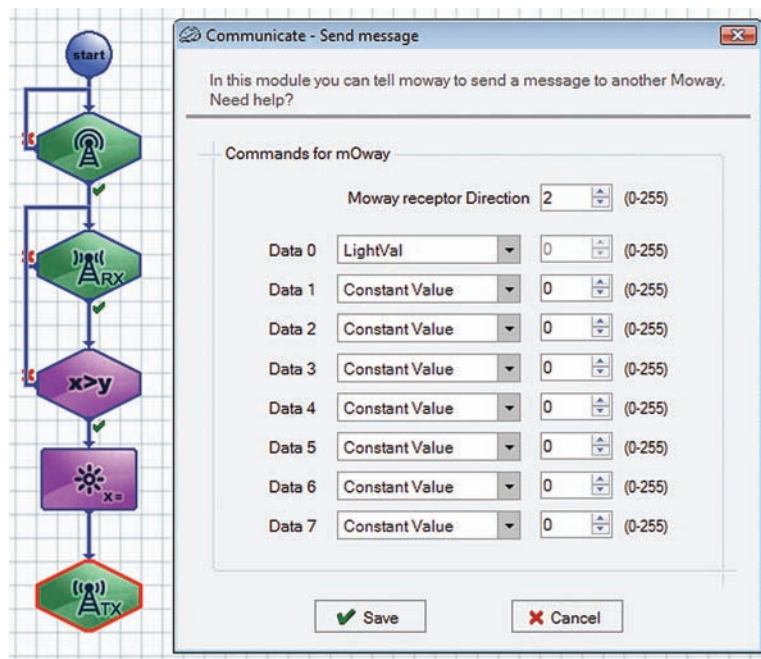


If the command is not the expected one, the program returns to receive the next RF message. However, if the received command has a value of 10, the brightness value is read. This is performed by the "Data → Assign Variable → Brightness" module.

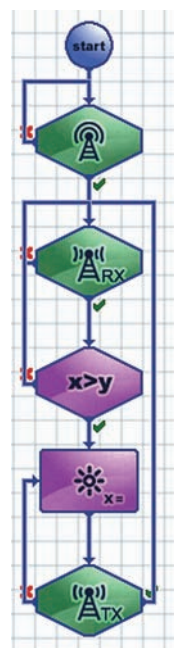
A new variable is created to store the value of the light sensor, named "LightVal".



Once the light sensor value has been stored in "LightVal" variable, this value is sent to PC by means of "Expansion → Communicate → Send" module. The RF message has to be sent to PC direction, which is 2. So that the "Receptor direction" field is assigned with a value of 2. The brightness value ("LightVal") is sent in the "Data 0" field.



If the sensor value is sent correctly to PC, the program will return to wait to receive a RF message from PC. Otherwise the light sensor is read again and mOway tries to send this value.



Once the program is done, we download the software in the robot and connect the RFUSB module to one of the PC ports. We configure the module by means of Communications window, with the address 2 and the channel 0. Then we push the Start button. The robot RF address is 1, which is the address to which PC will send the RF messages. So that, the "Receiver address" field must be 1.



Once the RF configuration of the PC has been done, a command to read the brightness value detected by mOway can be sent. A value of 10 is assigned to the field 0 of data to be sent, as it is shown in the image below.



When the Send button is pushed, the mOway receives the command and then it sends the sensor value to PC. This value is written on the last field (Data 0) of the PC received data, and it also receives the transmitter address (in this case, the robot address has a value of 1). For this example, a value of 18% of light has been read.

Based on the project, you can make improvements in order to learn a lot more about how to program the robot:

- **Exercise 4.11.1.** Turn on the robot's LEDs when it receives an order from the PC and turn these off when it responds with information.
- **Exercise 4.11.2.** Add new commands between the robot and the computer. For example, command 11 for line sensors, command 12 for obstacle sensors, etc.
- **Exercise 4.11.3.** Add movement orders to the robot. For example, command 20 to go forward, command 21 to go backward, command 22 to stop, etc.

What we have learnt



How to send messages from the robot to the computer.



How to see the RF messages in MowayWorld.



New challenges

- Turn on the robot's LEDs when an RF message is received and turn them off when the robot sends the data.
- Add new commands to read other senses For example, 11 for the line sensors, 12 for the obstacle sensors, etc.
- Add commands to move the robot. For example, 22 move forwards, 21 to move backwards, 22 to stop, etc.



Check what you have learnt

1. How much data can be sent by RF?
2. Is the configuration of the RF module different, depending on whether we are sending messages to a robot or to a computer?
 - a) Yes
 - b) No