Omniwheels X2

With this model, you will learn some initial fundamentals of programming a two-wheeled vehicle with a support wheel and the elements installed in the model.

If you have built the model using the building instructions, then you have certainly noticed some new components. You will be using the infrared sensor, 2 encoder motors as “actuators” and a new type of wheel – the Omniwheel.

Infrared sensor:

The infrared track sensor is a digital sensor used to detect a black track on a white background at a distance of 5 - 30 mm. It consists of two transmitter and receiver elements.

Omniwheel:

Important: Later, when you are completing the programming, you need to pay attention to the direction the wheels are turning. If they are turning to the left, then the wheels are turning forward, to the right means they are turning backward.

The “Omniwheels x2” model is divided into 5 programming tasks:

Task 1	Programming level 2 Drive command – Drive back and forth along a set path (time or pulse-controlled)
Task 2	Programming level 2 Travel command – travel straight for a short distance and then travel a short distance to the right or left.
Task 3	Programming level 2 Travel command – drive in a square (time or pulse controlled), drive along lines – the crossed house
Task 4	Programming level 2 Travel command – follow a black line
Task 5	Programming level 2 Travel command – seek a black line and then follow it

Controller configuration

First, start the “ROBO Pro Coding” program and then complete the controller configuration. Please see the building instructions once again for the wiring for the actuators and sensors.

You will need the following actuators and sensors:

Eingang
IR-Spursensor
IR-Spursensor

Motor
Encodermotor
Encodermotor

Zähler
Encodermotor - Zähler
Encodermotor - Zähler

Mode
Master
Name
TXT
Input
I7
IR-Spursensor
I8
IR-Spursensor
Output
Motor
M1
Encodermotor
M2
Encodermotor
Servomotor
Counter / Digital Input
C1
Encodermotor M1 Zähler
C2
Encodermotor M2 Zähler
I2C
USB1

Task 1

In the first task, the goal is for the vehicle to drive forward for a specific length of time, then drive back. Then, the program should stop. To do so, both wheels need to be activated at the same time. You are also in learning level 2.

You can insert the program into the specified endless loop.

Insert this command at the docking point.

Use your mouse to click the + – symbol in the command. By doing so, you will expand the command so that the motor at output TXT_M2 is synchronised with the motor at output TXT_M1 and will turn in the same direction at a speed of 512 (this is full speed).

Add a wait time of 3 seconds, then the command to stop both motors.

Here as well, click the + – symbol and TXT_M2 is added to the command. Then the second motor will be stopped as well.

To keep the program from running continuously, add the command “Break the loop” from the “Loops” block. Then you can send the program to the model and try it out. The model should drive forwards for 3 seconds.

Programmstart
dauerhaft wiederholen
mache
setze Motor TXT_M1 Geschwindigkeit links 512
sync mit Motor TXT_M2 Richtung links
warte s 3
stoppe Motor TXT_M1
sync mit Motor TXT_M2
die Schleife abbrechen

To cause the model to drive in the other direction after a 3-second wait time, simply copy the travel command and insert it after the stop command for the motors, with an additional wait time of 3 seconds. Here, you only need to change the direction of the two motors.

The model should stop again after a run time of 3 seconds. Insert the wait command and copy the stop command to the appropriate point in the program.

This part is now finished, and you can save the program on your computer.

Simply name it

“Omniwheels_x2_forward_backward_time”

In the next example, you will be learning some fundamentals on driving along a certain path using the pulse control. You will need to consider a few things first.

If you have read the information for the encoder motor, you will know that the encoder motor has an impulse wheel.

63.9 impulses are generated for one turn of the motor shaft. We simply call this number I_{shaft} = 63.9. Next, consider the model. There is a toothed gear with 10 teeth on the shaft. This drives a toothed gear with 20 teeth on the wheel axle.

This means you have a translation from fast to slow, at a ratio of 10 teeth to 20 teeth. Therefore, the ratio of the number of teeth Z on the toothed gears is the translation ratio i. The number is assigned based on the “direction of force”, meaning the driving wheel (motor shaft) is Z_1 and the driven wheel (wheel shaft) is Z_2.

Therefore, our formula for the drive is:

$$i = \frac{Z1}{Z2} \quad \text{or} \quad i = \frac{10}{20} \quad i = 0.5$$

Now, you have to transfer this value to the number of pulses. As we said before, the number of pulses on the motor shaft is I_{shaft} = 63.9.

I_{axle} therefore is I_{axle} x 0.5. Therefore, for one rotation 63.9 x 2 = 127.8 pulses are counted by the pulse counter on the TXT 4.0 Controller. This value is fixed for all models, since the toothed gears do not change.

Another fixed variable is the diameter of the Omniwheels. This is 60 mm.

The illustration shows the distance in mm when the wheel completes one revolution. But how do we determine the distance? To do so, use the mathematical formula to calculate the circumference of the wheel.

The formula is: U=d x 3.14 (Pi). Therefore, the distance travelled by the wheel during one revolution is 60 mm x 3.14 = 188.40 mm. We can use 188 mm in the following calculations.

The model should drive forwards and then stop. This means you need to calculate the impulses (I_{travel}) required to travel this distance. S_{rev} stands for the distance for one revolution, or 188 mm, I_{rev} for the impulses per revolution, here 127.80.

Calculate how many impulses you need for 1 millimetre of travel distance.

$$I_{imp/mm} = I_{rev} / S_{rev} = 128/188 = 0{,}68 \text{ imp/mm (purely mathematical)}$$

Noting the tolerances and dimensions of the wheels, this value should be calculated at **0.72**. You can use this value for all distance calculations.

Multiply the value by the distance travelled.

$$0{,}72 \text{ imp/mm} \times 200 \text{ mm} = 144 \text{ imp}$$

Delete all commands from the program, then start by inserting a new command “Set motor – Speed – Increment” from the “Motor” block.

Since the two motors should turn at the same time, you can activate “+” to expand the command.

Insert a mathematical expression in place of the value for increment. Select the command “Is the total of two numbers” from the “Maths” block. Insert the command “One number” at both blank points from the same block.

Instead of the “0”, insert the value I= 0.72. Then change the + symbol to x and change the 2nd entry to the distance to be travelled (200 for 20 cm). **Use a period as a decimal point!**

Add a command from the “Util” block “Wait until” after the block, then the command “Once motor --- reached position” from the “Motor” block. Then the loop should be stopped.

Util
warte bis

Motor
hat Motor - Position erreicht

Programmstart
dauerhaft wiederholen
mache
setze Motor TXT_M1 Geschwindigkeit links
250
Schrittweite
0.72 × 200
sync mit Motor TXT_M2 Richtung links
warte bis hat Motor TXT_M1 Position erreicht
die Schleife abbrechen

Schleifen

die Schleife abbrechen

Test the subprogram.

You will certainly see that the vehicle does not drive straight ahead, depending on the characteristics of the track it is on. This is partially because of the slip factor of the track, and because the entire torque is applied to the wheels when the vehicle starts.

Simply reduce the speed to 200.

Add an impulse-controlled reverse manoeuvre to the program. To do so, copy the relevant commands into the program. Change the direction of the wheels here from “left” to “right”.

Then, save the program and test the program function. If everything works correctly, the vehicle will travel forward 200 mm and then 200 mm back, then stop.

OmniwheelsX2_forward_backward_dist

Task 2

You can adopt the Controller setting for the next task. Start a new project and use a new project name, such as:

Omniwheels_x2_forward_rot90_dist

First, the vehicle should travel a certain distance (200mm) forward. You are familiar with this command sequence from the first task.

The vehicle will stop and should then turn 90° to the right.

To do so, the left motor needs to maintain the same direction, while the right motor needs to change direction.

The pivot point for the wheels is the centre of the support wheel.

Here as well, you must control the wheels by entering a certain number of impulses.

640 impulses are required to turn the model

once.

In the next step of your calculations, you will put the impulses in a ratio with an angle.

First, calculate how many impulses you need for the vehicle to turn 1°.

The formula is: $I_{360°}/360° = 640/360 = 1.77$. Based on the tolerances and dimensions of the

Omniwheels, you can use the **value 1.66**.

You can calculate the number of impulses by multiplying this number by the required angle.

Example: The vehicle should turn 90° – this corresponds to 1.66 x 90 = 149.4 or 149 impulses.

Integrate the command sequence to execute a 90° turn to the right into your program. Set the motor speed to 250.

You can use the same formula from the “Maths” block. However, the fixed value is “1.66”.
You can go ahead and test out the subprogram. Here, you were trying to program a curve to the right or left.

Task 3

Omniwheels_x2_square

In this task, we will make the vehicle drive in a square. Do not delete the commands from the program; instead, drag them to the right and out of the program. We will need them again later.

You will need to create a so-called “variable” for the task, which you will name “counter_rotation90°”. To do so, open the command “Create variable” from the “Variables” block.

An input window will appear where you can enter the relevant name.

End the entry by clicking “OK”.

Then open the “Variables” block. ROBO Pro Coding provides you with 3 commands.

First, drag the command “set counter_rotation90°” into the program start. Get the command “if the total is ...”, which you are already familiar with, from the “Maths” bock, along with the command “one number” twice, then add them at the docking point.

Change the “+” symbol to the “x” symbol. Replace the first 0 with the value “1.66” and the second 0 with the angular value “90°”.

It is important that you enter a period as a decimal point.

Now, create a second variable for the distance the vehicle should travel straight ahead. Name it “counter_dist”.

Now, add another command from the “Loops” group. Since a square has 4 sides, you can use a loop command that is completed four times. Insert it into the program. Change the repetition to “4”.

Then, add the command sequence for the motor controller. Simply drag the commands back into your program.

As the last step of programming, insert the two variables “counter_dist” and “counter_rotation90°” from the “Variables” block. They replace the two increment values.

This task is then completed. Save the program.

Things are getting more interesting now; you're becoming a real programming pro! Now that you have driven your vehicle along set distances and curves, we will tackle a task using an

infrared sensor. When integrated into the program, this allows the vehicle to follow a black line.

Task 4

Start a new program and name it

Omniwheels_x2_linefollower

You have already inserted the IR sensor into the controller configuration.

Ensure you are using the “I7” and “I8” connections.

You will use the course included in the building set for the test trips later on.
First, let’s learn some important information on the IR sensor. What is the switching behaviour?
You can easily determine this using the interface test.

Start the test using the command

The test window will open.
Set your model with the IR sensor on the black line so that both sensors are on the line.

In the interface test, the input values are displayed as “0”.

Now, push your model so that one IR sensor is outside of the line. Push it to the left. The value of “I8” switches to “1”.

Try it with the second IR sensor as well.
Push the model to the right. The value of “I7” switches to “1”.

When both IR sensors are outside of the line, both input values will be “1”.

First, insert a “continuous repetition” loop into the program start.

Add the command “If – then – otherwise” from the “Logic” block.

First, set up one condition: if the value at I7 and I8 is zero, both motors should turn to the left with a speed of 300.

Define this speed in a variable named “speed_fast”. Create the variable.

Add the command “set speed_fast” from the “Variables” group. Add a “number” from the “Maths” block, and change the content to “512”.

Create a second variable named “speed_slow” and insert it after the last command. Change the value to “200”.

Insert both variables before the endless loop.

When you click the arrow beside the variable name, a selection window will open. The current variables are displayed there. You can also select a variable here.

Insert the condition to query both IR sensors in the “if” query. To do so, you will first need the command “is true, if both ...” from the “Logic” group.

Insert the command “if IR track sensor ...” from the “Input” group at both empty points.

Change the value for I7 into the second query for I8.

You need to run both motors with the pre-defined speed of “speed_fast” for the “then” area. Both wheels will turn to the left. To do so, use the command

from the “motor” group.

Add the “+” symbol to the command and insert the variable “speed_fast” for the speed. In the “otherwise” area, you need to switch off both motors. To do so, use the “Stop motor” command from the “motor” group. Here as well, add the “+” symbol to the command for the second motor.

This section is now complete, and you can test out the subprogram. To do so, place the model on the black line. Place it so that both IR sensors are over the line. The vehicle will travel along the line until it comes to a curve, then stop because neither of the IR sensors measure the value “0”.

Now, expand your program by clicking the “+” symbol for “if” twice. The expansion is required to query whether there is an IR sensor outside of the line. The program should then control the vehicle so that both IR sensors are back over the line.

Then, copy the query for the IR sensors to both free points in “otherwise if”.

Change the value of sensor I7 to “1” and the value of sensor I8 to “0”.

In the second query, reverse the two values – first “0” and then “1”.

Drag the command

from the “motor” group to the “then” area twice into the program. Change the speed to the variables “speed_fast” and “speed_slow”. Also change the direction of rotation and

designations for the motors.
This task is then completed, and you can test it out. If the vehicle or one of the IR sensors leaves the line, the second or third query will readjust the motors and it will continue travelling along the line.

Task 5

But what happens if neither of the sensors are on the black line? You need to adjust your program so that the model looks for the black line in this case. To do so, first query the two IR sensors to determine whether they are on a white surface. Do so using the command:

Insert two variables before this, or create them first. You will need one variable named “linelost”
and one variable named “counter”.

Insert the command “true” at the two docking points from the “Logic” block, and change it to “false”, as well as the command “one number” from the “Maths” block.

First, drag all commands to the right and out of the main program – except for the continuous loop.

Insert a query “if – then – otherwise”. “If” will query the variable “linelost”. If the vehicle has left the line, then a command block follows under “then” that queries the two IR sensors and searches for the line. To do so, the model will turn around a specified angle and travel a specified distance. This process is repeated until it finds the line. Then the program will continue with the “otherwise” command block. You can insert this command block into your main program from the right once again.

Delete the commands from the “otherwise” query from the inserted block. These are then replaced by two commands at the end of the program.

Save your subprogram under the name

Onmiwheels_x2_linefollower_linesearch

on your computer.

Insert another logic query after “linelost” into the query. Copy the IR sensor query to the “if” query and change the status to “1”.

Now, you also need a logic query at the “then” input. Here, first query the variable “counter” to determine if the value from the formula is “less than/equal to – 1.66x360”.

In the “then” area, insert the command “set motor” from the “Motor” group twice. M1 will turn to the left, M2 to the right, and the speed will be determined based on the variables “speed fast”. The model will now turn around its centre point. After a wait time of 100 ms, the counter will be set to the value for counter input C1.

The model should travel forwards for a certain distance under “otherwise”. Drag a motor command to the free point, then add the “+” symbol to it. Change the speed by inserting the variables “speed_fast” and the increment by the mathematical formula. Then, insert the wait command “if motor has reached position”. Then the “counter” variable is set to “0”. This search for the line is repeated until one of the IR sensors has the value “0”.

If this is the case, the free point “otherwise” still needs to be defined. The variable “linelost” needs to be set to “false” here. Drag a new command “output...” into your program from the “Text” block. When you start the program later on, the defined text will be displayed here in the so-called “console” in the bottom part of the screen. Enter the text “line found” in the two quotation marks.

The entire block is now complete, and you can make the following changes.

The query “otherwise” is not yet defined in the second block. Copy the two commands from the “otherwise” query of the first block here. Change the name of “output” to “line lost” and set the variable “linelost” to “true”.

Save the program and test it out.
Now all tasks are programmed, and you can take on the next model.