

**13 Motor A is off For**

**12 As long as the red container contains less than 1**

**14 Then MotorA runs forwards until**

**15 The rotation sensor3 is 24**

**16 Add "1" to the yellow container**

**17 Play note B**

**18 B is off, End program**

active forward wheel, in this case ( right) it's the **C wheel**, write it down.

Repeat the turn 3 times

First test

2<sup>nd</sup> test

3<sup>rd</sup> test

Add together

Divide by 3

Divide by 1.12

|  |
|--|
|  |
|  |
|  |
|  |
|  |
|  |

This is the rotation duration you will use

in you program for that specific distance

Making a left 90°turn is made in the same manner b ut reversing the direction of the wheels.  
i.e. C is stopped and B is now forwards and you check the rotation reading for the B wheel.

---

Using this method you can place your robot on the First LEGO League mat and measure the degrees for any size turn that you need to make and accurately control it using the programming software and use your rotations measurements in programs.

If you need to control a lifting arm using a B motor you can apply this technique here too.

---

### Using containers to control motors on/off

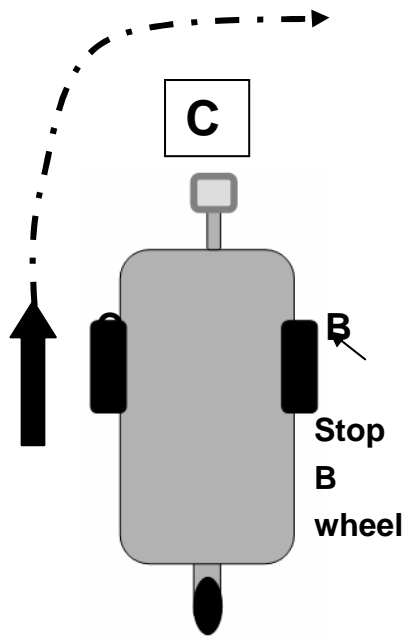
One other factor you need to consider is keeping the opposite wheel off for the exact time that the drive wheel is turning. You can use CONTAINERS in the program to trigger motors on/off

This program causes the robot to make a right turn and then a left turn.

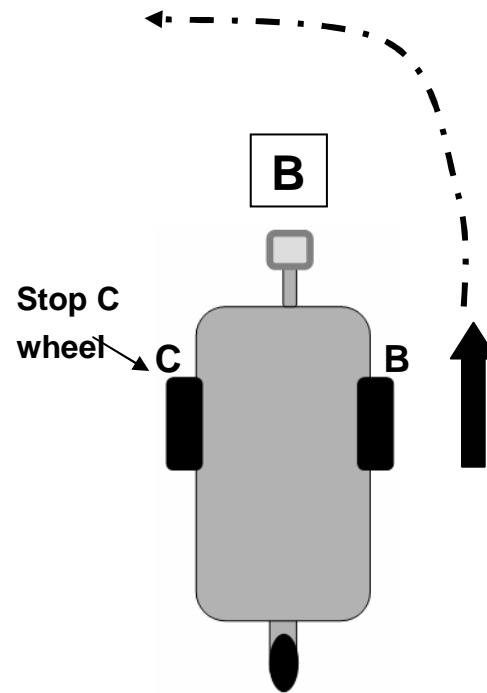
It uses the rotation setting of 24 to cause the driving wheel to travel approx 28 cm.. This will be an arc on a circle because the opposite wheel is off.

The program uses containers rather than timers to keep one motor off while the opposite motor is running. This is more accurate than timers

TURN RIGHT



TURN LEFT



To control how far the robot turns you will need to control the rotations setting for the forward wheel

For a left turn this is the C wheel

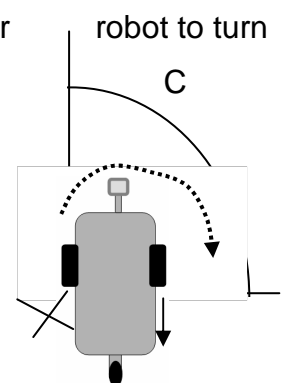
For a right turn this is the B wheel

### **Right Turn 90°**

Use the VIEW method outlined on pages 4-5 . Manually manipulate your right by placing a finger on the B wheel and another on the wheel.

Keep the B wheel still, it acts as a pivot and move the C wheel forwards at the same time until your robot has made a physical 90° turn

Check the ROTATIONS reading in the VIEW screen for the



|                |   |
|----------------|---|
| Distance in cm | Divide distance by<br>1.12<br>= Rotations |
| Example 24     | 21.42 (round<br>up/down)                  |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |
|                |   |

## Using rotations to program an exact turn

### Turning

As the **ROTATIONS** reading/setting does not mean that your robot can turn a specific degrees/rotations turn, the reading refers to the rotation of the axle with 360° broken up into 16 points on a circle, which then causes the rotation of the wheel ( standard wheel circumference is 18cm) **you will still need to program the turn itself**

By setting the STEERING bar completely Left for a LEFT turn or completely right for a RIGHT turn the robot will execute a tight turn pivoting from P

3 times taking 3 degrees readings for each motor in the VIEW area , add then average.  
 Now use this degrees number in your program. Download, run.

**Motor A**

**Motor C**

**Sensor 1**

**Sensor 3**

|                     |  |  |  |  |  |
|---------------------|--|--|--|--|--|
|                     |  |  |  |  |  |
| First run           |  |  |  |  |  |
| 2 <sup>nd</sup> run |  |  |  |  |  |
| 3 <sup>rd</sup> run |  |  |  |  |  |
| Add together        |  |  |  |  |  |
| Divide by 3         |  |  |  |  |  |
| Divide by 1.12      |  |  |  |  |  |

Which method was more accurate, the experimental method or the mathematical method?  
 Why do you think this was so?

|  |
|--|
|  |
|  |
|  |

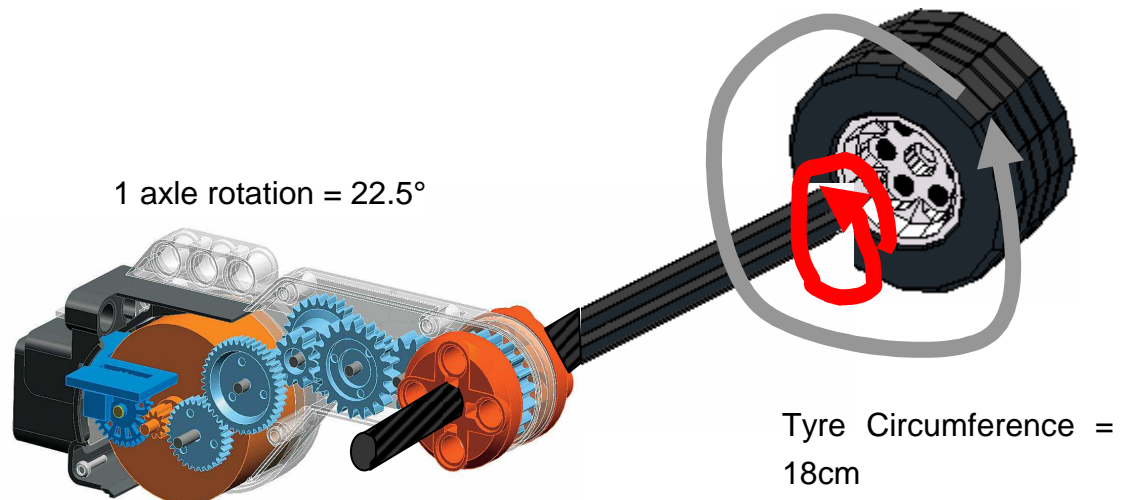
**Using the mathematical method – Making a distance**

**Programming tool**

We can now use this knowledge to make a rotations programming tool. Once we measure the length of the straight lines on the mat that we need to move along we can use our tool to provide the exact degrees we need for our program.

Make an Excel reference chart ( or a hand-made table – see next page) for degrees settings i.e the distance that the robot travels with standard 18cm tyres.

1 axle rotation =  $22.5^\circ$  =  $1/16^{\text{th}}$  rotation of the wheel =  $1.12\text{cm}^*$



$\therefore$  we know that if the axle rotates  $360^\circ$ , the wheel will turn a full rotation and move 18cm along the surface, this is a rotation setting of 16

### The Formula:

For a specific distance that you want to travel – divide the distance by 1.12 and round up if necessary. This is the number of rotations you need to program for.

### Try it out

Let's assume we want the robot to travel exactly 45cm.

How many 1.12's in 45 ?

So , for the robot to travel 45cm we will need to program the robot to move 40.17 rotations ( round down in this case to a whole number = 40)

Let's program this into our robot and test it.

## Programming using Degrees

The actual programming process for the both the experimenting and mathematical methods for duration are the same, it's only the way you decide what rotations you need that is different.

### TEST 2

Using the experimental method outlined on pages 4 & 5 , push the robot along a 45cm line

## 8 Calculate how many degrees for that specific distance

|                     |  |
|---------------------|--|
| First run           |  |
| 2 <sup>nd</sup> run |  |
| 3 <sup>rd</sup> run |  |
| Add together        |  |
| Divide by 3         |  |
| Divide by 1.12      |  |

This is the rotation duration you will use  
in you program for that specific distance

9 Repeat this process for every straight line path you need to make on the mat. Make sure you write down your degrees readings as you will need them when you start programming.

***We will deal with making turns as a separate issue so stick to straight lines for the time being.***

The manual experimentation process has some inherent weaknesses that can reduce accuracy, what do you think these weaknesses are?

|  |
|--|
|  |
|  |
|  |


## Method 2 – The Mathematical method.

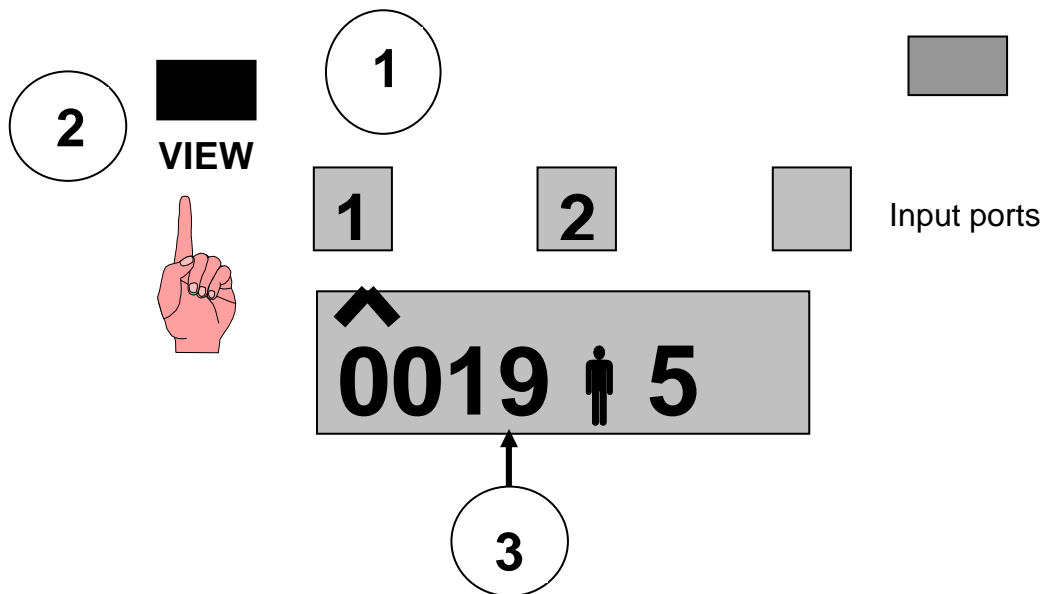
The second method involves doing some basic math's but it has the potential to be more accurate than the manual method.

*What we know about the basic robot:*

## robot tell you the degrees.

Firstly you will need to load a program into your robot that tells it that it has 2 rotation sensors connected to input ports 1 and 3. Load and run the program example above. The LCD panel on top of the RCX brick is able to give you a read out from the sensors.

1. Plug in a light sensor on **INPUT 2**
2. Press the VIEW button until the  is pointing to **INPUT port 2**
3. The LCD screen should show you a reading from the light sensor In the illustration below, the light sensor plugged in on input no 2 is reading 57

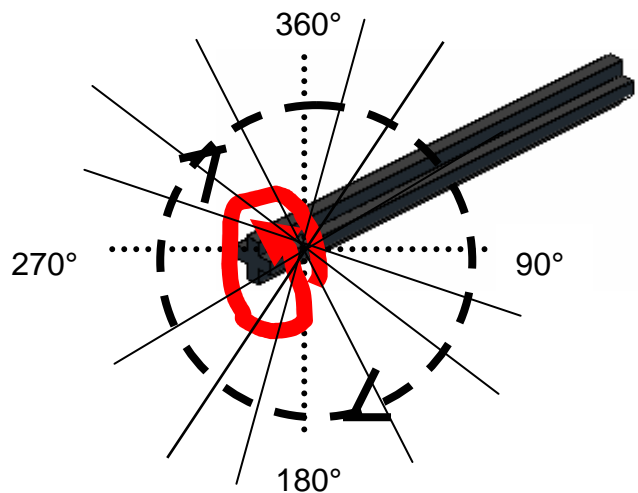


Push your robot along the First LEGO League mat so that it's wheels freely rotate. Start exactly where you want the robot to start from and stop exactly where the robot should stop. Take a few readings from each sensor and average the readings to get some idea of the rotations you will need to program into your robot,

**VIEW** the readings from input ports 1 and then 3 for the 2 rotation sensors.

## ROTATIONS

**ROTATIONS** = the number of degrees of rotations of the **AXLE**, in turn this setting will result in a specific linear distance traveled by the robot.



*ROTATIONS settings will not result in a specific degrees or rotations of turn by the physical robot, ie a 1 rotations setting will not result in a full circle turn by the robot, rather a 1 rotations turn of the axle resulting in the wheel rotating 22.5° and causing the robot to move forwards or backwards (use negative numbers) a specific distance, in this case 1.12cm. We can accurately calculate( rotational movement resulting in linear movement, like a trundle wheel) .*

There are several ways of using the rotations sensor to program your robot. Here we will look at two basic methods:

**1 Experimentation , directly reading the degrees sensor using the VIEW function on the robot**

**And**

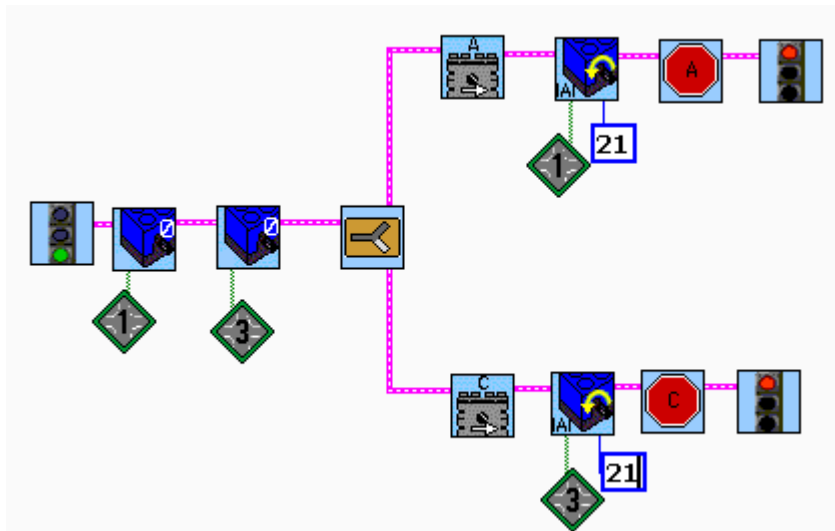
**2 Mathematically calculating the degrees settings using a simple formula.**

Try both methods and decide which method works the best for you.

**Method 1 – For STRAIGHT LINES - Experimentation: let the**

| Tyre Circumference 18cm |                                |
|-------------------------|--------------------------------|
| Rotation Point          | Linear distance traveled in cm |
| 1                       | 1.12                           |
| 2                       | 2.24                           |
| 3                       | 3.36                           |
| 4                       | 4.48                           |
| 5                       | 5.60                           |
| 6                       | 6.72                           |
| 7                       | 7.84                           |
| 8                       | 8.96                           |
| 9                       | 10.08                          |
| 10                      | 11.20                          |
| 11                      | 12.32                          |
| 12                      | 13.44                          |
| 13                      | 14.56                          |
| 14                      | 15.68                          |
| 15                      | 16.80                          |
| 16                      | 17.92                          |
| 17                      | 19.04                          |
| 18                      | 20.16                          |
| 19                      | 21.28                          |
| etc                     | etc                            |

You can then measure the distance you need your robot to travel and use these calculations within your program to cause your robot to travel accurately in increments of 1.12cm.



The program sample above zeros both rotation sensors. It then uses a fork to cause both motors to run forward at the same time at the same speed for 21 points on each rotation sensor ( sensors plugged in on ports 1 and 3) . Both motors should then stop. Theoretically both motors should run the exact same distance in rotation points therefore the robot should

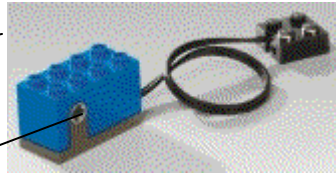
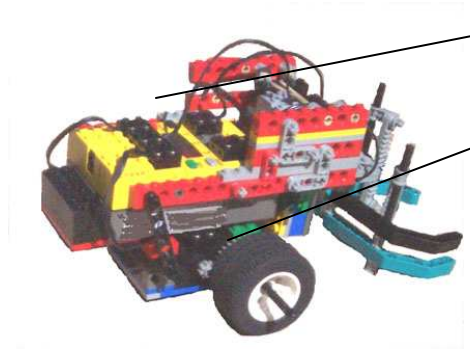
travel straight for 23.52 cm. Copy the program in Inventor Level 4 and test it



## First LEGO League Primer

### D DEGREES- exact control of distance & turns

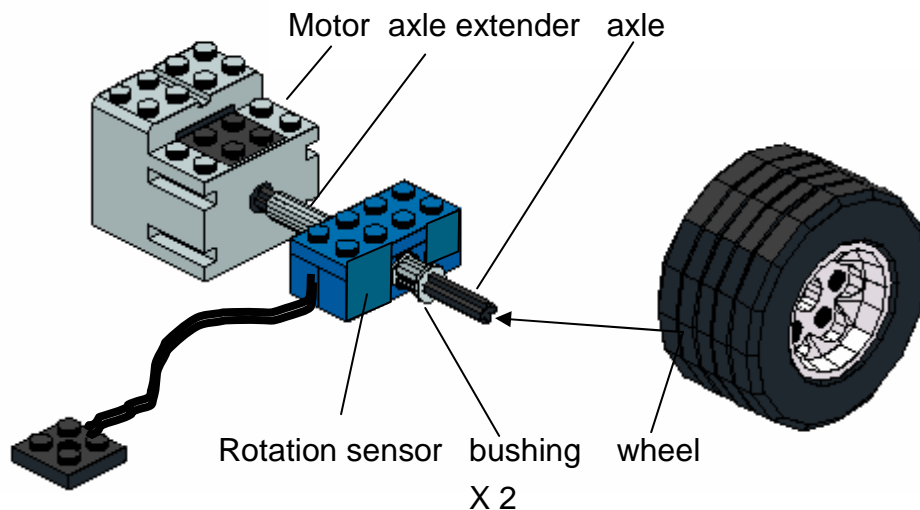
Successfully completing First LEGO League challenges relies foremost on being able to find the object in question, positioning the robot exactly on the mat so that it can perform a task.



The LEGO RCX robot does not have an in-built sensor that makes this process

possible. You will need to connect one rotation sensor to the axle coming off each of the drive motors of your robot.

The sensor has a hole through its centre where the axle is inserted.



The standard balloon tyres measure around 18 cm in circumference

The distance traveled by the robot when rotation measurements are used as a duration setting are dependent on the circumference of the tyres fitted.

The rotation sensor measures 16 points on a circle.

Therefore the axle will rotate  $22.5^\circ$  per point.

1 full rotation = 16

This will result in the robots wheels traveling a linear distance that is relative to the size of the tyre. In the case of an 18cm wheel this will be  $18\text{cm} \div 16 \text{ rotation points} = 1.12\text{cm per rotation point}$ .

