

This is a large worksheet you will need to break it up into sections -

STUDENT WORKSHEETS START ON PAGE 11

TEACHERS NOTES

DEGREES- exact control of distance & turns

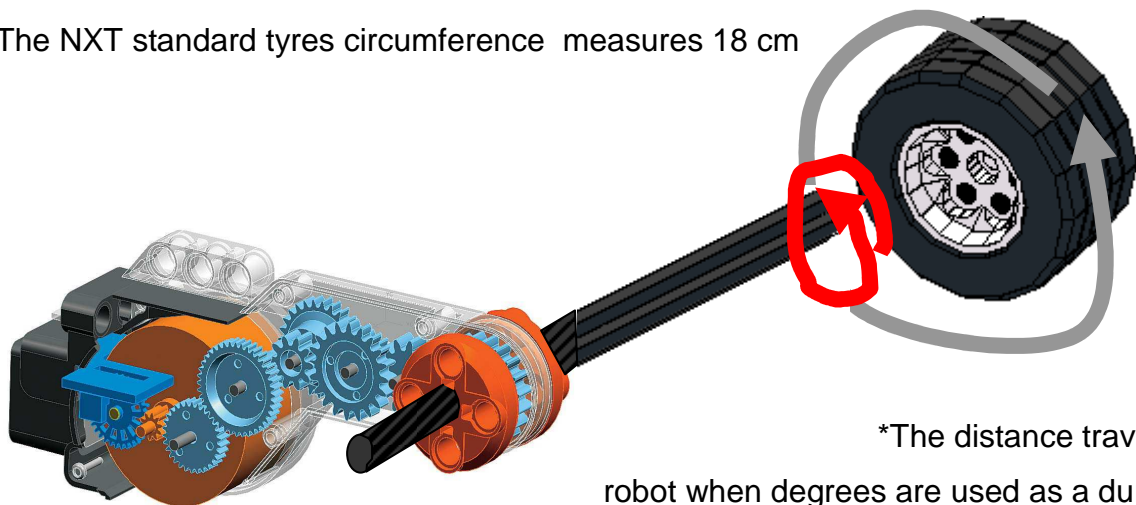
The ability to exactly position a robot where it needs to go to is a major part of robotic control. For example successfully completing First LEGO League challenges relies foremost on being able to find the object in question by positioning the robot exactly on the mat inside home base (registration) so that it can travel the exact distance required to the object perform a task.

The LEGO NXT robot provides an in-built sensor that makes this process possible. Inside each motor is a rotation sensor that you as the programmer can control

FOR THE NXT ROBOT MOTOR DEGREES = the number of degrees of rotation of the AXLE, in turn this setting will result in the wheel (and tyre) turning and causing the robot to move a specific linear distance .

DEGREES settings will not result in a specific degrees of turn by the physical robot, ie a 90° setting will not result in a 90° turn by the robot, rather a 90° turn of the axle resulting in the wheel rotating 90° and causing the robot to move forward or backwards a specific distance that we can accurately calculate (rotational movement resulting in linear movement, like a trundle wheel) .

The NXT standard tyres circumference measures 18 cm



*The distance traveled by the robot when degrees are used as a duration

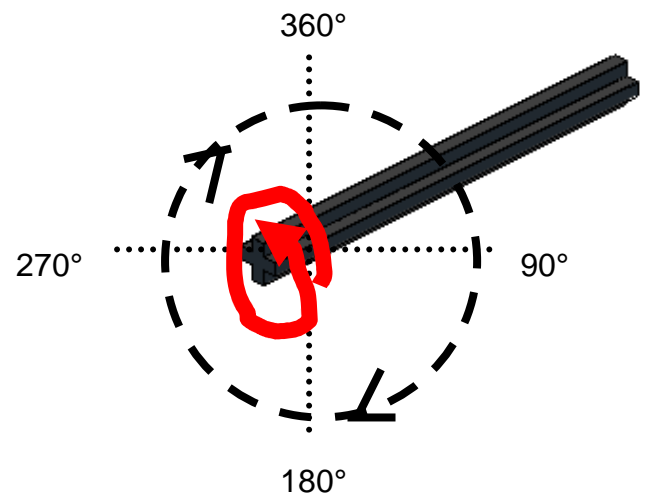
setting are dependent on the circumference of the tyres fitted. 18cm is the circumference of the standard balloon style tyres

DEGREES – ROTATIONS ?

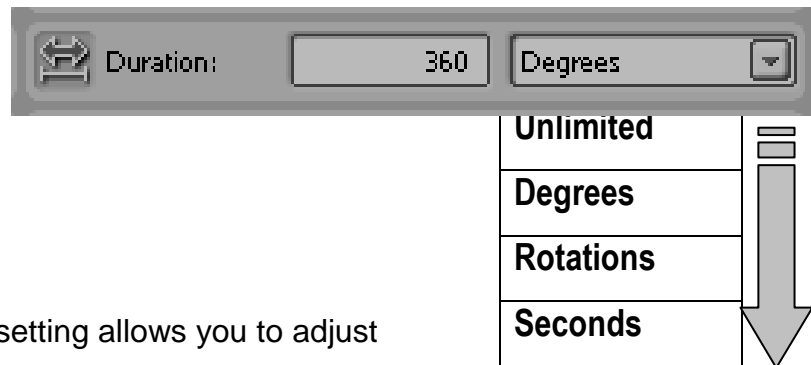


D

Both ROTATIONS and DEGREES reference the in-built encoder in the NXT motors . Before you start programming the NXT on a computer you need to experiment to see exactly how many degrees = what distance, you can then use the programming software to control the distance your robot travels exactly



The NXT Programming software illustrating the DURATION : Degrees option



The **DURATION** setting allows you to adjust

HOW LONG the motor will run for this **translates to how far your robot** will actually travel

Wait For (Duration) selections in the pull down menu:

Unlimited – Select this is you are using a switch (IF) block / Wait for block and sensors.

The action is limited by the sensor parameters you set following the MOVE icon

Degrees – Degrees of rotation of the axle. This references the in-built sensor in the motors

Rotations – One rotation of the axle. This also references the in-built sensor in the motors.

USING DEGREES TO TRAVEL EXACT DISTANCES

There are several ways of using the degrees sensor to program your robot to move exact distances in straight lines; for turning exact degrees of turn by the physical robot the method is a little more complicated.

We will explore these basic methods in the following worksheets , there is no need to do all these worksheets one after the other.

Basic methods:

1 Mathematically calculating the degrees settings (for forward movement) using a simple formulas (a and b)

1a Logical Mathematical Method (assumes all robots motors & tyres are exactly the same)

Theoretically the mathematical method outlined above should be consistent and accurate for all robots. In practice I have found individual differences in the robots motors that make programming using mathematical calculations less accurate than they should be. Some motors have “slop” inside the mechanisms that cause variations.

1b Calculating and exact “key” for each robot i.e. exactly how far does my robot travel per degree?

The most accurate method is to perform a similar test that calculates the exact performance of your robot, how far it actually moves in mm per degree, this will give you a “key” for programming distances using degrees that you can use for all distances.

This is the method we will use for *FIRST* LEGO League straight lines

2 Experimentation (not so accurate) pushing the robot and directly reading the degrees sensor using the VIEW function on the robot (for straight lines and turns).

In practical terms this option introduces a plethora of variables that influence results and is the least reliable. (However, to start programming controlled spins before exploring more complex mathematical formulas relating to the length of arc on a circle, I will use “view” despite it’s inherent inaccuracies)

This is the method we will use for *FIRST* LEGO League turns

AND

For controlled accurate spins : This is not a new method but rather an application of 1b the “KEY “ method and then processing it via the mathematical formula for the length of the arc on the circle

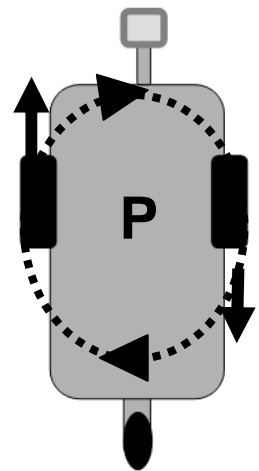
3 Mathematically calculating the degrees settings for controlled robot spin turns using an additional formula and the robot key.

Terminology:

SPIN – I define a spin where the robot having 2 drive wheels and a rear skid/dolly wheel causes the NXT brick to rotate around a centre point that is located in the middle of the two drive wheels.

P = centre of circle →

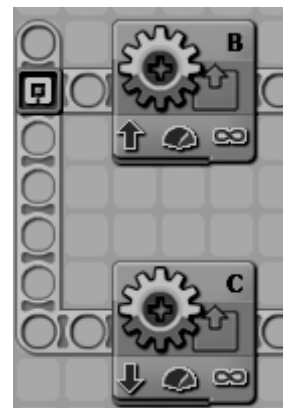
The robot does not actually move it's forward position when it spins.



This is the tightest circle the robot can execute and is programmed in the NXTg software **COMMON PALETTE** (basic restricted palette) by sliding the steering bar all the way to the left or the right:



In the **COMPLETE** palette you have the option of controlling the motors independently in a number of ways but the end result is still a spin

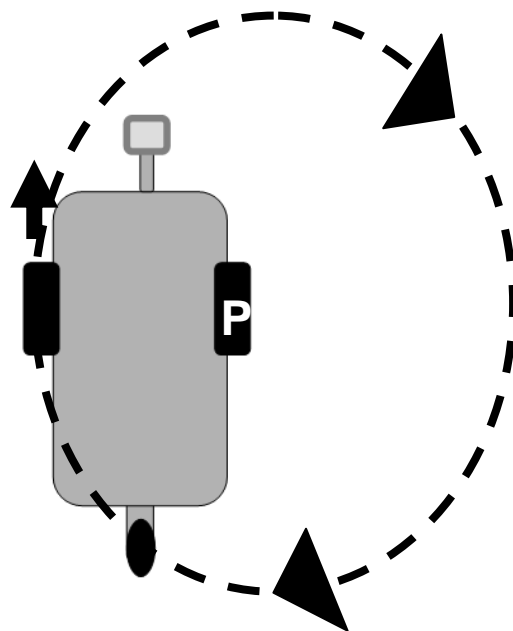


This programming results in one drive wheel moving forward and one drive wheel moving backwards.

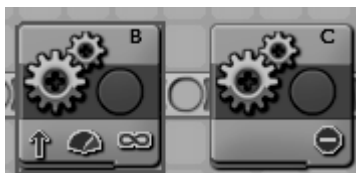
PIVOT - When the robot pivots it executes a larger circle and pivots around one drive wheel. The pivot wheel is stopped and the other wheel drives forward.

For a pivot you need to program each drive motor separately rather than use the B and C MOVE block.

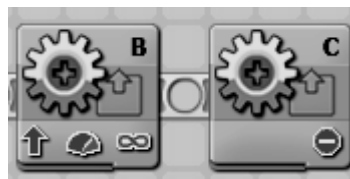
This can be done using either the COMMON PALETTE or the COMPLETE PALETTE



Common Palette



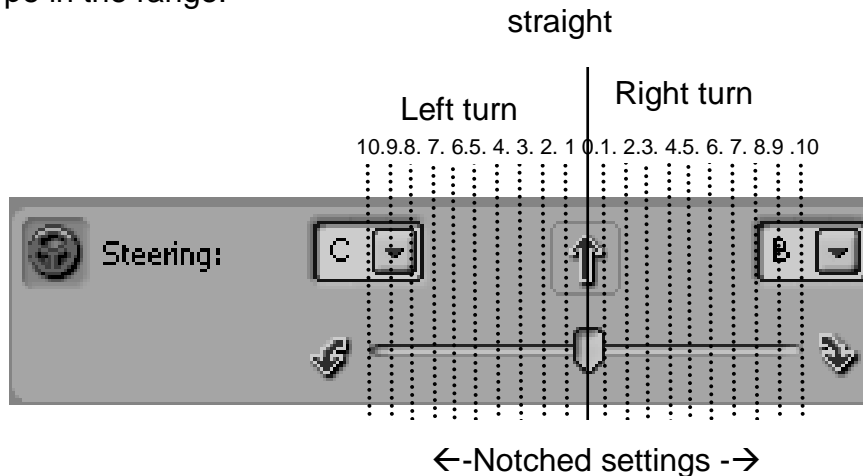
←Complete palette V



ARC – When the robot executes an arc(either forwards or backwards) it makes turn while progressing either forwards or backwards with a range of arc/circle sizes. Both the inside and outside wheels of the robot execute an arc and both wheels of the robot move forward. The programmer controls the amount of power (and therefore speed) of each motor initially in the COMMON PALETTE by using the steering configuration of NXTg programming.

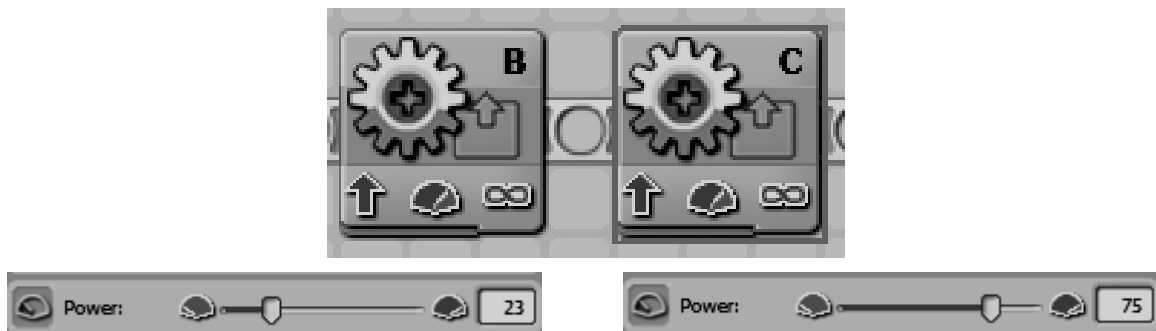
The steering slider in the NXT configuration panel for the MOVE icon has 10 distinct “notched” stop-points Left and Right (excluding straight ahead) , each stop-point delivers a certain sized circle/arc (here we will just deal with the generic basic robot, other “builds” of robots with different distances between the drive wheels)

We already know that in positions 10 left and 10 right the robots wheels will actually switch to one motor forward and one motor backwards (a pivot turn) as this is the most acute circle type in the range.



The steering slider simplifies the programming behind programming an arc by automatically changing the amount of power (therefore speed) delivered in a ration to the 2 drive motors, usually B and C .

In the COMPLETE palette you are able to have a much wider range of arc control as you have 100 power settings available for each motor



[For a more complete investigation on arc turns click here](#)

Method 1 – The Mathematical method.

a – Logical Math's

This method involves doing some basic math's but it has the potential to be more accurate than the manually viewing the degrees as it reduces variables.

What we know about the basic robot:

| | |
|-------------------------------|------------------------------------------|
| 1 axle rotation = 360° | = 1 rotation of the wheel = 18cm* |
|-------------------------------|------------------------------------------|

| |
|---------------------------------------------------------------------------------------------------------------------|
| ∴ we know that if the axle rotates 360°, the wheel will turn a full rotation and move 18cm along the surface |
|---------------------------------------------------------------------------------------------------------------------|

The Formula:

Convert 18cm to 180mm so that we are using 100'ths units for both degrees and distance.

Divide 180mm by 360°

| |
|-----|
| 900 |
|-----|

The robot should travel exactly 0.50mm per degree

Try it out – 0.50mm per degree

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

This is 450mm

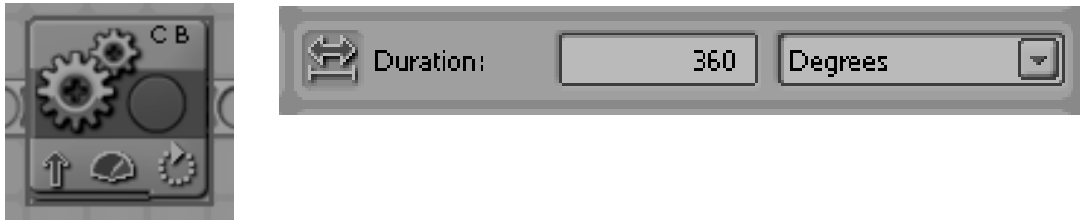
How many 0.50's in 450 ? (or just double 450)

So , for the robot to travel 45cm we will need to program the robot to move 900 degrees.

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

Programming with NXTg Software using Degrees

The **DEGREES** setting gives the programmer very precise control of the robot without variables caused by battery power fluctuations and terrain. **ROTATION** is less precise as it is a “chunkier” measurement that we won’t use here.



TEST 1

Your test program will have a MOVE icon controlling motors B and C, forward for 900 degrees



Your robot will break straight after this , not coast.

Program this, download into your robot and check if it moves exactly 45cm. Run the program at least 3 times and accurately measure the distance in mm that your robot traveled. Measure the distance travelled from the centre of the axle (this is where the sensor is measuring from) to the stop point centre of axle. Record your results here :

| Run | Distance in mm |
|-----|----------------|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |

You will probably find that your robot actually traveled a different distance, a little more or less than 45cm

Why do you think this happened?

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

The most accurate method is to perform a similar test that calculates the exact performance of your robot, how far it actually moves in mm per degree, this will give you a “key” for programming distances using degrees that you can use for all distances.

Programming tool

We can now use this knowledge to make a degrees 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 following) for degrees settings i.e the distance that the robot travels with standard 18cm tyres.

| | | |
|----------------|----------------|-------------------------------------------------|
| Distance in cm | Distance in mm | Divide by 0.50 or Multiply by 2 = Degrees |
|----------------|----------------|-------------------------------------------------|

| Example 24 | 240 | 480 |
|------------|-----|-----|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

1b Individual “Key” method



The most accurate method of controlling your robot’s distance traveled is to perform a similar test to 1a that calculates the exact performance of your robot, how far it actually moves in mm per degree, this will give you a “key” for programming distances using degrees that you can use for all distances. This key will be specific to your particular robot and if you change robots or the wheels / tyres or build of the robot you will need to repeat this process.

TEST 2

Make a program that drives your robot forward for a **DURATION of 900°**. Your test



program will have a MOVE icon controlling motors B and C, forward for

900 degrees

Your robot will break straight after this , not coast.

Program this, download into your robot, run the program and measure exactly in mm how far it travels. . Measure the distance traveled from the centre of the axle (this is where the sensor is measuring from) to the stop point centre of axle.

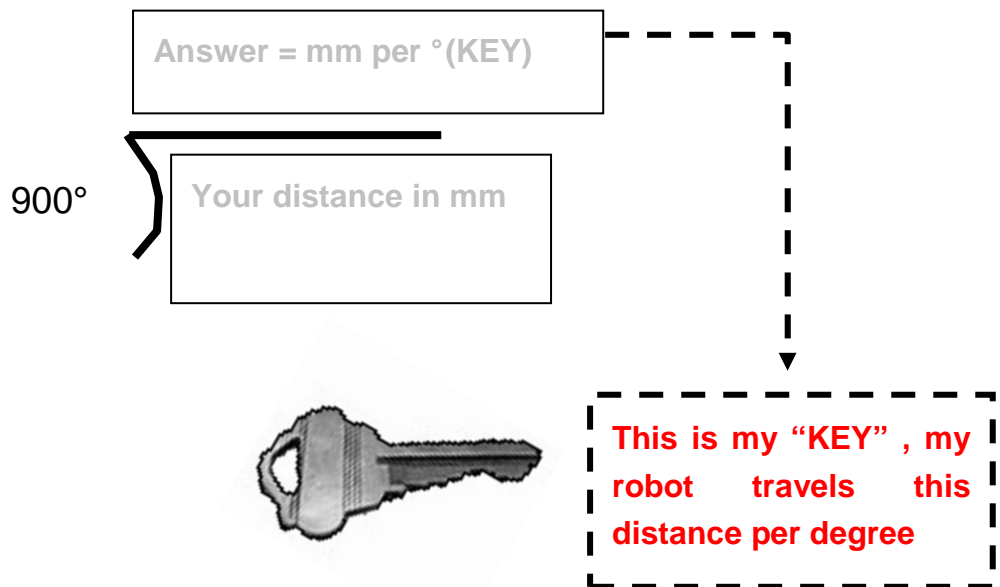
Repeat this procedure at least 4 times and average the results.

Record your results here :

| Run no | Distance in cm | Distance in mm |
|-----------------------------|----------------|----------------|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| Total | + | + |
| Divide by no of runs | ÷ | ÷ |

We will now use this result to calculate the Key

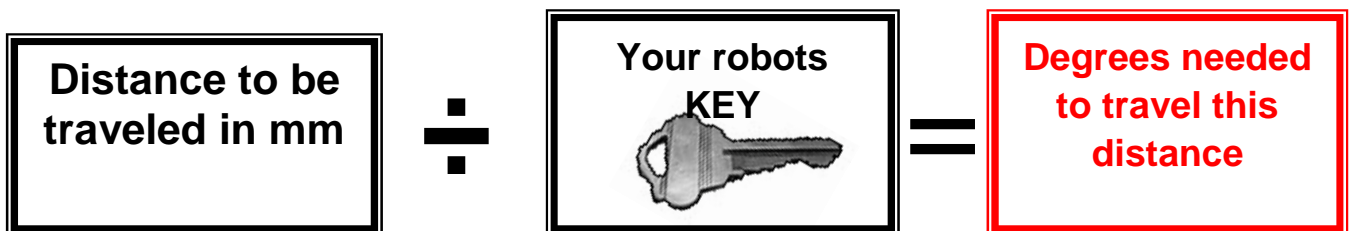
Now divide the distance in mm in the square above by 900°



You should get an answer that is within a range of .40 to .55, if you don't, repeat your tests, math's and measurements

How the "key" works. We now know exactly how far your robot travels in mm per degree. If you need the robot to travel a distance (straight only , curves are more difficult) .

Whenever we need the robot to travel a distance we convert the distance into mm and then divide the distance by the key.



Example

Let's assume we want the robot to travel exactly 57cm.


This is 570mm

How many 0.42's in 570 ? 1357

So , for the robot to travel 57 cm we will need to program the robot to move 1357 degrees.


 Duration: Degrees 


Use your robot's key to calculate, program and test these distances:

| Distance in cm | Distance in mm | Divide by your "KEY" = Degrees  |
|----------------|----------------|--------------------------------------------------------------------------------------------------------------------------|
| 92 | | |
| 74 | | |
| 120 | | |
| 68 | | |
| 54 | | |

Using the mathematical method – Making a distance Programming tool

We can now use this knowledge to make a degrees programming tool. Once we measure the length of the straight lines we wish to travel 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.


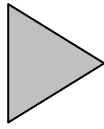



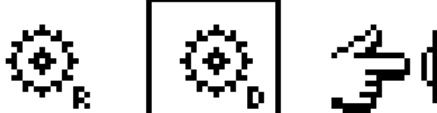
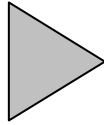

| Distance in cm | Distance in mm | Divide by your "KEY" = Degrees  |
|----------------|----------------|----------------------------------------------------------------------------------------------------------------------------|
| | | |
| | | |
| | | |
| | | |
| | | |


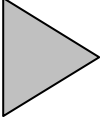




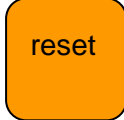





Method 2 – For STRAIGHT LINES - Experimentation: let the robot tell you the degrees.



Push your robot along the First LEGO League mat (or any surface) so that it's wheels freely rotate. Make this distance accurately from the middle of the axle at the starting point to the middle of the robots axle at the end point. Start exactly where you want the robot to start from (axle) and stop exactly where the robot should stop (axle).

At the same time set the **VIEW degrees function** on the robot. You **MUST** nominate either **C** or **B** motors (not A) if you are working with a basic robot.

Here's how to read the degrees :

| VIEW - MOTOR DEGREES | | | |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| 1 |  | SCROLL  RIGHT X2 | SCROLL to VIEW |
| 2 | <div style="border-bottom: 1px solid black; padding-bottom: 5px; margin-bottom: 5px;"> USB NXT  </div>  |  | VIEW YES |
| 3 | Select Motor degrees  | SCROLL  RIGHT X10  | SCROLL MOTOR DEGREES YES |

| | | | |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| 4 | <p>Select Port A</p>  | <p>SCROLL</p>  <p>RIGHT</p> | <p>SCROLL</p> |
| 5 | <p>Select Port B</p>  |  | <p>PORT B or C SELECT</p> |
| 6 | <p>USB NXT </p> <hr/>  | <p>Look</p>  | <p>Shows Degrees at 0 Reset to 0 – press the orange button</p> |
| 7 | <p>USB NXT </p> <hr/>  | <p>Look – write down the degrees reading</p> | <p>Push the robot over the required path so that C or B wheels rotate- Write down the degrees reading</p> |
| 8 | <p>USB NXT </p> <hr/>  | <p>Repeat #5 two more times</p>  <p>Write down the readings</p> | <p>Reset to 0 – press the orange button. Repeat</p> |

| | | | |
|----------|----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------|
| 9 | Select Motor degrees  |  | STOP GO BACK |
|----------|----------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------|

10 Read how many degrees for that specific distance

| | |
|---------------------|--|
| First run | |
| 2 nd run | |
| 3 rd run | |
| Add together – | |
| Divide by | |

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

11 Repeat this process for every straight line path you need to make .

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 next 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?

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

Test 3 - Using the experimental method outlined above , push the robot along a 45cm line 3 times taking 3 degrees readings in the VIEW area , add then average. Now use this degrees number in your program. Download, run.

| | |
|---------------------|--|
| First run | |
| 2 nd run | |
| 3 rd run | |
| Add together | |
| Divide by 3 | |

This is the degrees duration you will use in you program for 45cm

Program your robot to travel DURATION the exact degrees you read and decided on for the 45mm distance. Download, run

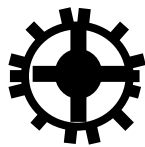
How accurate was this method?

Which method was more accurate, mathematical method 1a , 1b or 2 experimental method?

Why do you think this was so?

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

Using degrees to program an exact turn by the physical robot



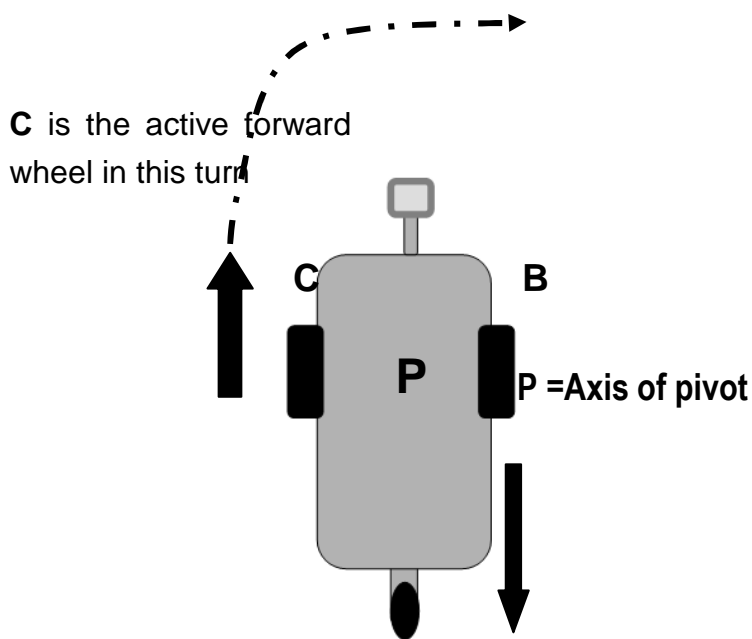
D

Executing Exact Spin Turns

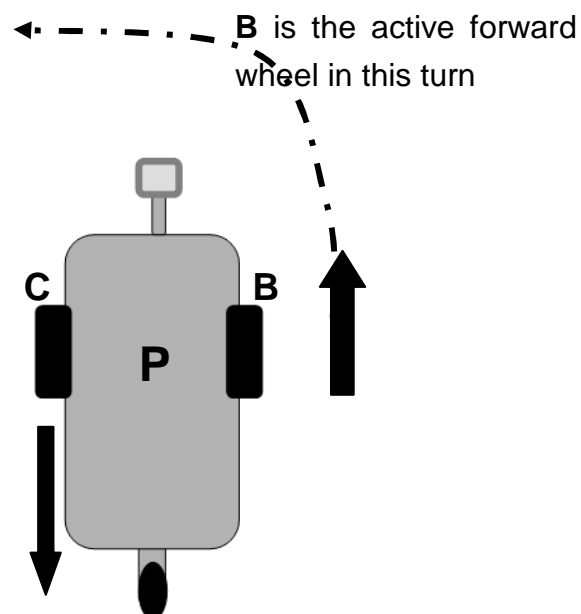
As we now know that the **DEGREES** reading/setting does not mean that your robot can turn a specific degrees turn, the reading refers to the degrees of rotation of the axle which then causes the rotation of the wheel **you will still need to program the turn itself**

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

TURN RIGHT



TURN LEFT



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

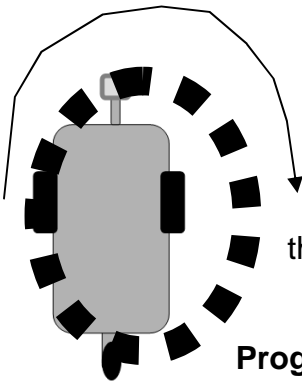
For a right turn this is the C wheel

For a left turn this is the B wheel

Again there are several methods for calculating exactly how many degrees You need to program the forward wheel (axle) to rotate to cause the robot to travel an exact distance. Two of these methods are variations on the straight line methods outlined previously, the third involves a different mathematical formula.

We will use a 90° turn as our test turn

The Mathematical Method



TEST 4 - For this investigation we will need to use/make a few tools , we will use a paint stamp pad to load the robots tyres with paint so that when it executes a full circle spin the tyres will print the circumference of the circle on white paper.

Program the robot to execute a full circle spin. Use a calculated guess for DURATION –degrees. This is unimportant at this point as we just want the robot to make a full spin circle (a little more is ok too) so that we can measure it e.g.



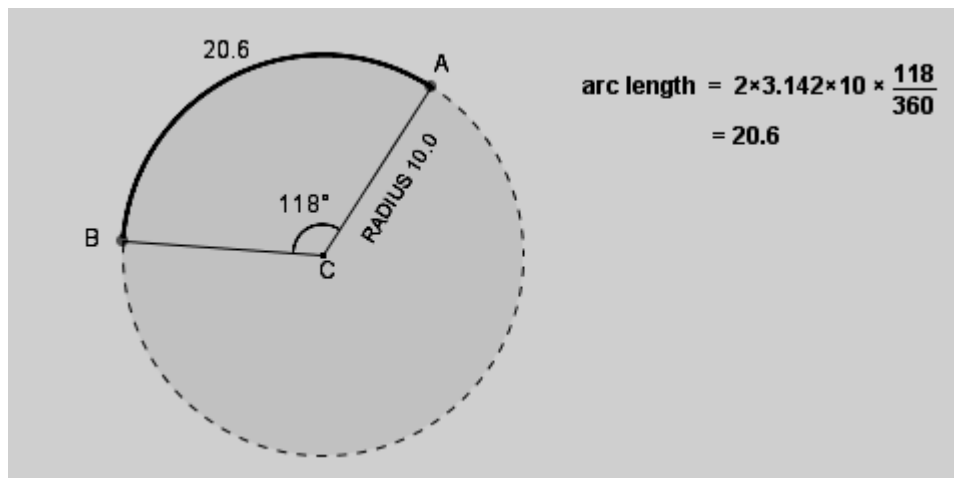
We use the steering set completely to the left or right to standardize the circumference (size) of the circle our robot executes to simplify our calculations .

Download the program, load the robots tyres with paint and run the program with the robot on a white sheet of paper.

Measure the diameter of the circle that the robot made (outside of tyre mark) and then calculate the radius .

For the robot to execute a specific angle turn (central angle) we can calculate the length of the arc for that turn (the distance the forward/outside wheel needs to travel) and then use the “Key” we calculated for our robot to determine exactly how many degrees we need to program for DURATION

e.g.



Source :

<http://www.mathopenref.com/arclength.html>

“The arc length is the measure of the distance along the curved line making up the arc. It is longer than the straight line distance between its endpoints (which would be a [chord](#))

The formula the arc measure is:

$$\text{arc length} = 2\pi R \left(\frac{C}{360} \right)$$

where:
 C is the [central angle](#) of the arc in degrees
 R is the [radius](#) of the arc
 π is [Pi](#), approximately 3.142

$2\pi R$ is the circumference of the whole circle, so the formula simply reduces this by the ratio of the arc angle to a full angle (360).

TEST 5 - Using the RADIUS for the robots circle , calculate the length of the arc for the robot to make a 90°

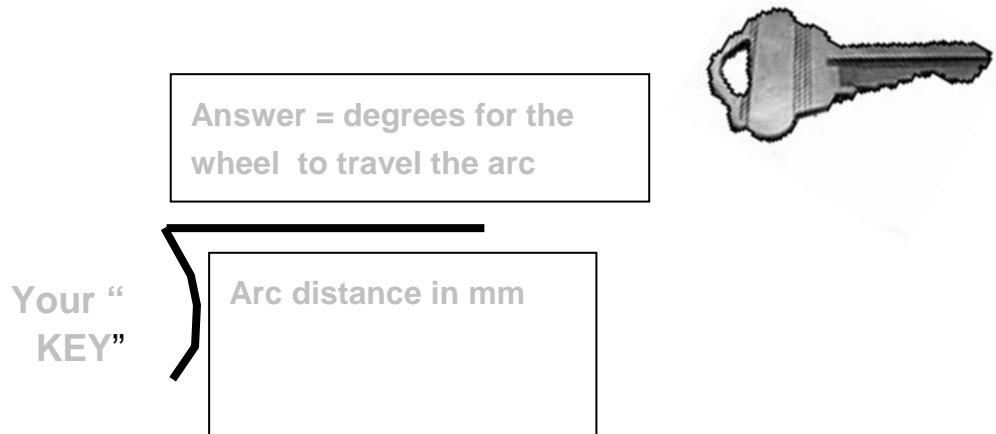
C central angle is 90°

R The RADIUS of my circle was

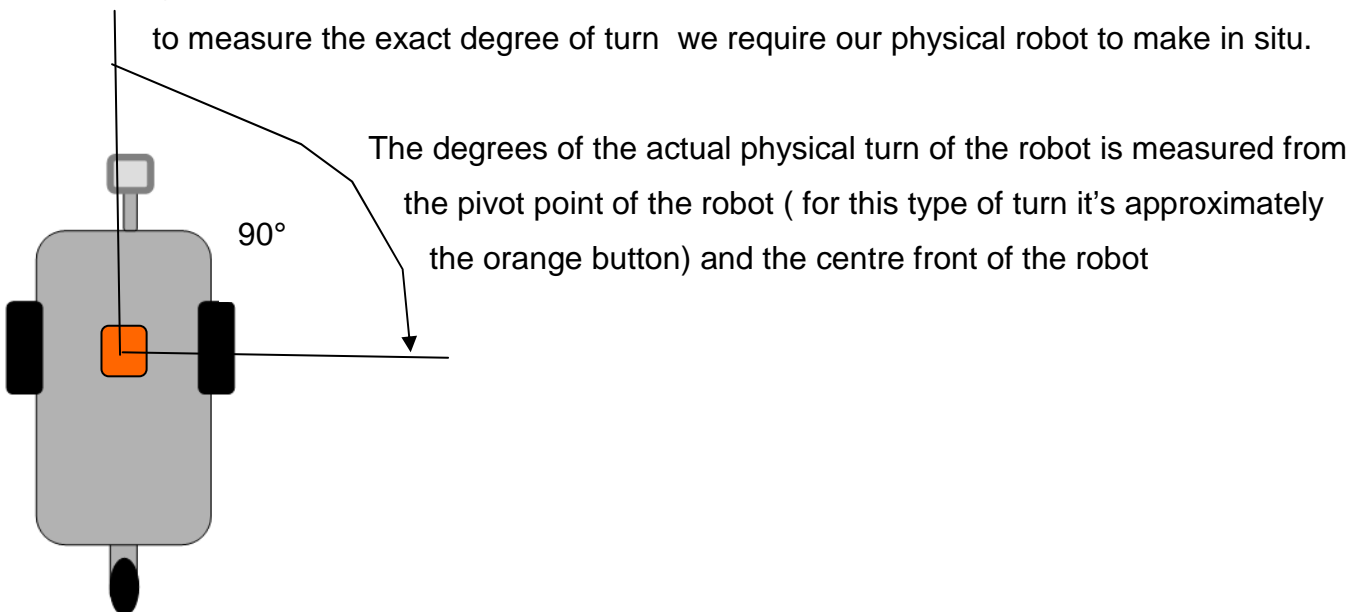
Arc length = 2π $\left[\frac{90}{360} \right]$

Now we have the length of the arc we can use the “Key” for mm per degrees to calculate how many degrees the robots wheel has to rotate for the robot s forward wheel to cover the particular distance.

Now divide your selected distance in mm by 90°



For angles other than multiples of 90° we will make a simple protractor that will assist us to measure the exact degree of turn we require our physical robot to make in situ.



The Manual “View “ Method

Right Turn 90°

TEST 5- Use the VIEW method outlined on pages 7-9. Manually manipulate your robot to turn right by placing a finger on the B wheel and another on the C wheel.

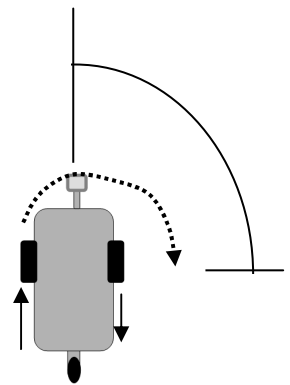
Gently move the B wheel backwards and the C wheel forwards at the same time until your robot has made a physical 90° turn

Check the DEGREES reading in the VIEW screen for the active forward wheel, in this case (right) it's the **C wheel**, write it down.

Repeat the turn 3 times

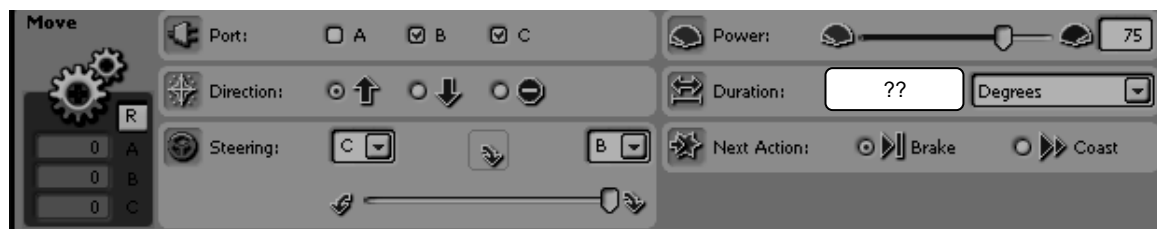
| | |
|---------------------|--|
| First run | |
| 2 nd run | |
| 3 rd run | |
| Add together | |
| Divide by 3 | |

This is the degrees duration you will use in your program for a 90° right turn



Now use the NXT programming software and key in your

number of degrees, download and check that the robot makes an accurate 90° turn



Making a left 90° turn is made in the same manner but reversing the direction of the wheels.

i.e. C is backwards and B is now forwards and you check the degrees reading for the B

wheel.

**Download
without 'Run'**

Teachers Notes

Programming using degrees and manual methods is only as accurate as the measurements that the students take. It is vital that students recognise the need to measure accurately and note exactly where they are measuring from and to, when using degrees settings the distance should be taken from the centre of the axle of the motor you are using.

In practice students using the manual method tend to be less accurate because of their procedure for manipulating the wheels of the robot to take readings. The surface and dexterity play a major role here.

The logical mathematical formula technique relies on every robot being exactly the same and performing the same way. This is not so, each robot will have some minor variation to the "rule". The further a robot has to travel the more this variation will amplify so that the robot becomes more and more inaccurate over longer distances. This is why the individual "Key" method is recommended as the most accurate way of controlling a robot's motors (unless you are relying on other sensor inputs for distance)

Your CD contains a sample of an Excel spread sheet programming in degrees tool.