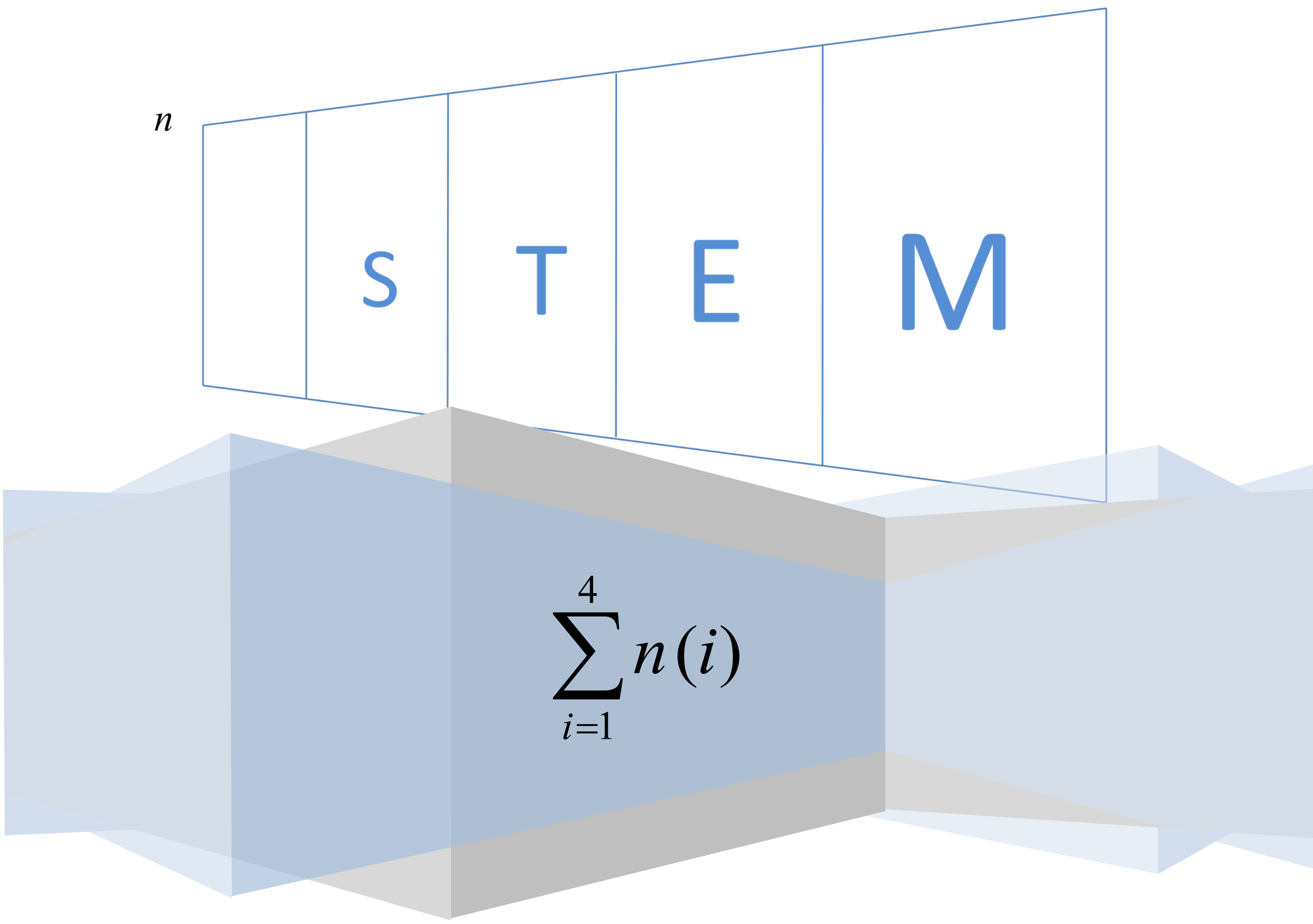
RoboMath - Learning K-12 Math with LEGO® NXT-G Programming Language

CJ Chung, Ph.D.

Lawrence Technological University
Department of Mathematics and Computer Science
Southfield, Michigan, USA



Version: 1.0 October 2011

ISBN: TBR

Printed in the United States of America

Preface

This eBook is intended to be used for K-12 teachers who would like to introduce autonomous robotics in 5th ~ 12th grade classes. This eBook will also be a valuable resource for coaches, mentors, and teams for Robofest® that requires the use of math.

This eBook is designed to support Problem-Based Learning (PBL) method. PBL is a student-centered pedagogy in which students learn about a subject in the context of complex, multifaceted, and realistic problems. Working in groups, students identify what they already know, what they need to know, and how and where to access new information that may lead to resolution of the problem. The role of the instructor is that of facilitator of learning who provides appropriate scaffolding of that process by (for example), asking probing questions, providing appropriate resources, and leading class discussions, as well as designing student assessments (wikipedia.org).

After introducing some basic concepts, this eBook provides the following different types of problems for groups of students to solve.

- (Class) Missions: they are intended to be explained and demonstrated by the instructor/teacher/mentor in class. Source code will be provided to the students.
- (Class) Exercises: It is recommended to be done by students in class. Source code will be provided to the students, after.
- Challenges: These are for PBL. Usually they are parts of Robofest® Games that require multiple hours to solve the problem. Can be homework assignments. There can be multiple ways to solve the problems and usually no solution will be given by the author.

Source codes of sample programs in *.rbt file format to introduce basic concepts, class Missions and some Exercises in this eBook can be obtained in a zipped file from a server. Where to get it? Please send the author an email at chung@LTU.edu. Bug reports as well as suggestions should be sent to the email address. A Power Point version of the book can be obtained too, if you contact the author.

Acknowledgement

First of all, I would like to thank all the Robofest participants who encouraged me to start and continue Robofest (www.robofest.net). Dr. David Bindschadler, Math and Computer Science Department Chair, has been fully supporting Robofest since the inception in 1999. Dr. H.P. Moore, Dean of College of Arts and Sciences, gave me all the support in various ways right after she came to Lawrence Tech as Dean of Arts and Sciences. I would like to thank Dr. Chris Cartwright, Associate Professor of Math and Robofest Program Manager, for his comments and for teaching some contents of this book during RoboMath summer camps since 2010. Taiga Sato, Robofest Lab Assistant, built a prototype robot arm to measure the height of the box and implemented ideas to solve the problem.

50+% of the profits from this book will be donated to Lawrence Tech University's Robofest program, one of the most affordable robotics competition programs in the nation.

Robofest is an annual autonomous robotics competition focusing on learning STEM (Science, Engineering, Technology and Math) and ICT (Information and Communication Technologies) for students in grades 5 - 12 and college students. Robofest challenges teams of students to design, build, and program robots to compete in the following categories in two age divisions (Senior and Junior):

- **Game Competition** - A team of students competes to accomplish robotics missions using fully autonomous robots. Robofest game specifically targets participants' math skills.
- **Exhibition** - Each team has complete freedom to show off any creative computer programmed robotics R&D project.
- **RoboFashion & Dance Show** **-** A team of robots (two are recommended) will use the whole track (stage) to show off their costume, walk (driving), and performing dancing motions. (4 - 12th grade students)
- **Vision Centric Challenge** (Sr. high school and college students - Associate Event)
- **nVn RoboSumo** (Sr. high school and college students - Associate Event)
- **RoboParade** (4 - 12th grade students - Associate Event)

Find more information at www.robofest.net .

Table of Contents

1 Introduction 8

2 Introduction to NXT-G Programming 11

2.1 Loop – DO UNTIL (Termination Condition) 11

2.2 Nested Loop 12

2.3 Decision Making 12

2.4 Event Driven Programming 13

2.5 Parallel Sequence Beams (PSB) 13

2.6 My Blocks 14

2.7 Variables and Data Connections 14

2.8 Parameterized My Blocks 15

2.9 Custom Blocks 16

3 Learning Elementary Math with Basic Missions 18

3.1 Mission – Let the TriBot Go straight 30cm exactly, and Stop 18

3.2 Mission – Let the TriBot Go straight *X* cm exactly, and Stop 20

3.3 Mission - Turn 30 degrees 21

3.4 Mission - SPIN 90 degrees 24

3.5 Mission – Measure a line length 26

3.5.1 Zigzag line following 26

3.5.2 Line following using a custom block made by Steve Hassenplug 27

3.5.3 Measure a line length while following a line 28

3.6 Mission – Stop at the midpoint of a line 29

4 Numbers, Operations, Variables, and Algebra 33

4.1 Proportional Sound 33

4.2 0+1+2+3+…9 =? 33

4.3 More Exercises using Variables 34

4.4 POW: X to the Y power function Block made by Michael Gasperi 34

4.5 Random Numbers 34

4.6 Geometric Progression 36

4.7 Mission (Counting): Stop at the 4th black line 37

4.8 Mission: Count # of touch sensor clicks (bumps) for 5 seconds 39

5 Geometry 41
5.1 Square Root using the Math Block (Not available in V1.x) 41
5.2 Exercise on Circles 41
5.3 Exercise on Squares 42
5.4 Exercise on Right Triangles 42
5.5 To measure a degree of an angle 43
5.6 Exercise: To find opposite side *a* 46
5.7 Exercise: To find theta 47
5.8 Challenge: Find the angle theta 48
5.9 Challenge: Find the distance *d* 49
5.10 Challenge: Find the length of d, triangulation 49
5.11 Angle Challenge 50
6 Probability Games 51
6.1 Even or Odd Game (Mission) 51
6.2 Dice Roll Game (Mission) 51
7 Data Analysis 53
7.1 Mission: Go straight for 2 seconds using different power levels 53
7.2 Mission: NXT-G Data Logging 55
8 Logic / Boolean Logic 57
8.1 Watching two sensors 57
8.2 Watching three sensors 57
8.3 Negation of Conjunction 58
9 Science 60
9.1 Calculate velocity of an object using a sonar sensor 60
9.2 Speed Meter 60
9.3 Triathlon Challenges 61
10 More Challenges and Projects 62
10.1 Centering 62
10.2 Locate at z 62
10.3 Go to (x, y) point 63
10.4 Draw y=*ax*+*b* *approximately* on the 1st quadrant of the Cartesian plane 64
10.5 Drawing a right triangle – Robofest 2009 Game 64

10.6 BTOS (Block the Oil Spill) – Robofest 2011 Game 65
10.7 Measure the height of a wall 66
10.8 Measure the volume of a box 67
10.9 Measure the volume of a cylinder 68
10.10 Follow right side of a black line. Stop at the end of the line 68
10.11 Make a Collision 68
10.12 Funnel Math 68
References and Credits 70

1

1 Introduction

We believe computer programming is a powerful learning tool for children (Papert, 1980). Robots first appeared in U.S. classrooms for educational purposes over 20 years ago. More recently, there have been many informal learning environments using computerized autonomous robotics such as after-school programs and competitions. Using autonomous robotics in formal and informal learning environments has been shown to improve math and science learning (Weiss, 2004), as well as critical thinking and problem solving skills (Wagner, 1998).

However, the current problems of Lego robotics competitions for K-12 STEM Education can be identified as the following:

- The learning of math concepts is not emphasized (except some Robofest games in the past years).
- Some popular Robotics competitions can be completed without using sensors other than built-in rotation sensors. Some teams got perfect scores without using Light, Sonar, or Touch sensors; dead reckoning has been better strategy for some popular robotics competitions.
- Robofest discourages dead reckoning. However, the robots are still simple re-active machines. At this point, Robofest does not require the use of variables.

It is desirable to consider the following factors when competition games are designed.

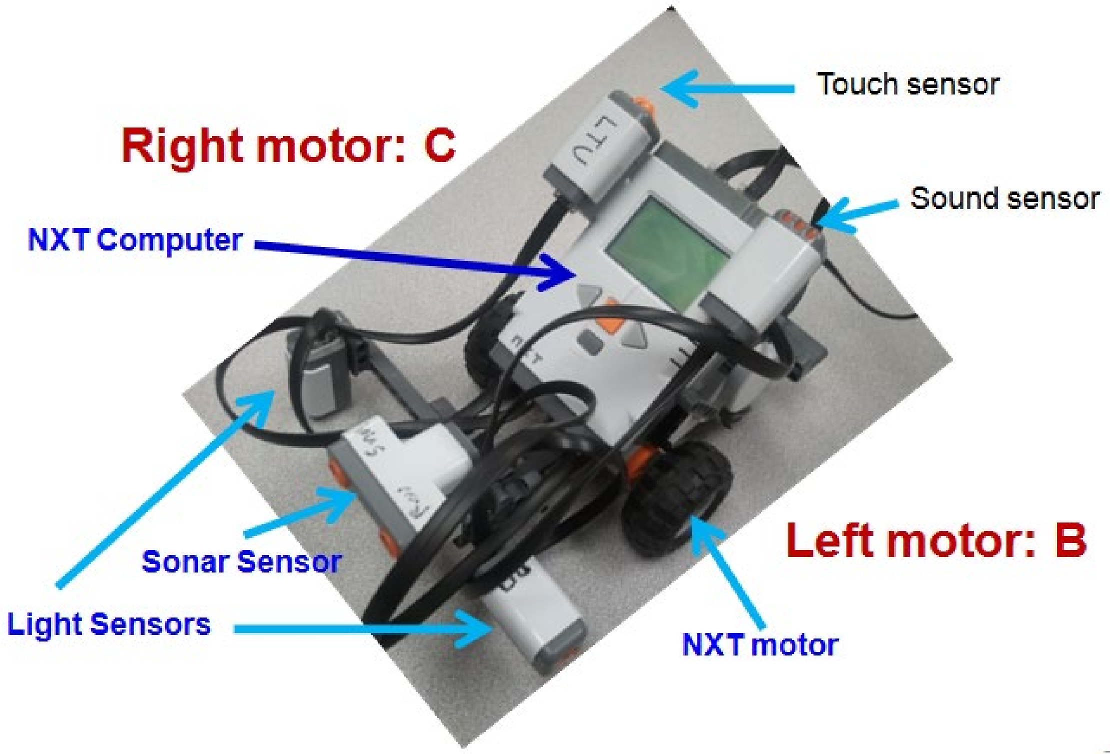
- Requiring the use of explicit math
- Applying math
- Use of variables and introduce the concept of state machines
- Requiring various sensors
- Displaying calculated or measured numbers on the LCD panel or sending the number to host computer via wireless communication such as Bluetooth

The objective of this book is to provide ideas and examples for teachers who would like to use Lego NXT and NXT-G programming languages to let young students learn more about mathematics in a fun way by doing (solving problems). Possible math topics to learn through hands-on Lego robotics in K-12 classrooms consist of the following:

Numbers, Operations, and Functions	Ratios & Proportions Scale Power Square root Algebra Series
Geometry	Radius Diameter Perimeter Circumference Length Area Volume Angles Radian Pythagoras theorem Trigonometry
Measurement	Calibration Counting Unit conversions
Data Analysis	Data logging Min/Max Average Regression Graphing
Probability	Random Numbers Conditional Probability (*)
Logic	Set Theory Boolean Logic Propositional Logic
Science	Speed Velocity Acceleration Gearing and Gear Ratio* Power/Energy* Motion Friction

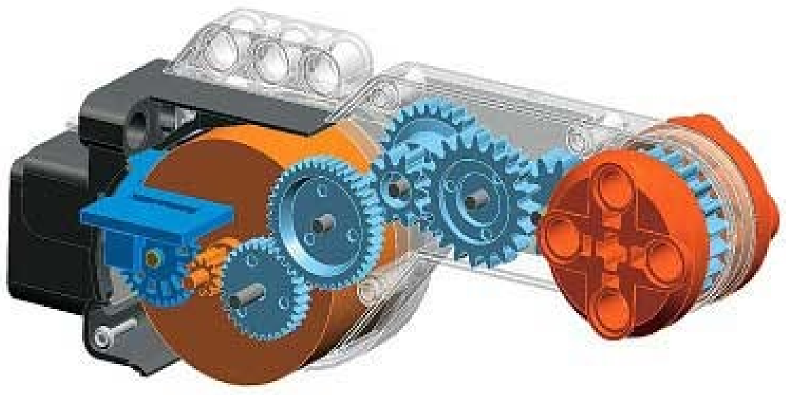
(*) Not covered in this eBook, yet

Throughout this book, the following modified “TriBot” Lego NXT robot is used.



- Left Motor connects to **B**
- Right Motor connects to **C**
- Light sensor connects to port no. **1** (For the edge following, you may need to extend more to the right)
- Touch sensor connects to port no. **2** (You may need an additional light sensor for some examples in this eBook)
- Sound sensor connects to port no. **3**
- Sonar sensor connects to port no. **4** (Additional Light sensor can be using No 4. Instead of Sonar for some challenges.)

Note that a NXT motor is a geared servo motor with built-in rotation sensor.



2

2 Introduction to NXT-G Programming

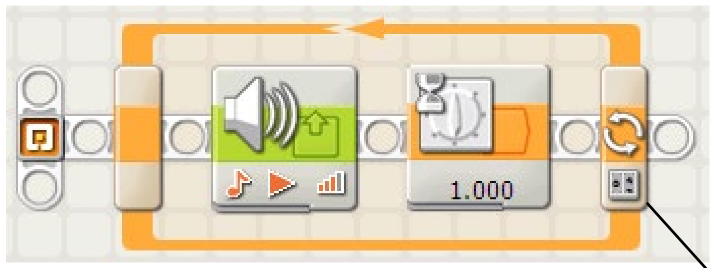
NXT-G is the programming software that comes bundled with the NXT Lego robotics kit. The software is based on National Instrument's LabVIEW® and provides a visual programming language for writing simple programs and downloading them to the NXT Brick. This means that rather than requiring users to write lines of text code, they can instead use flowchart like "blocks" to design (drag & drop) their program.

The programs in the book have been tested with version **2.1**. The NXT Firmware version used was **v1.31**. We will also use custom NXT-G blocks downloaded from the web.

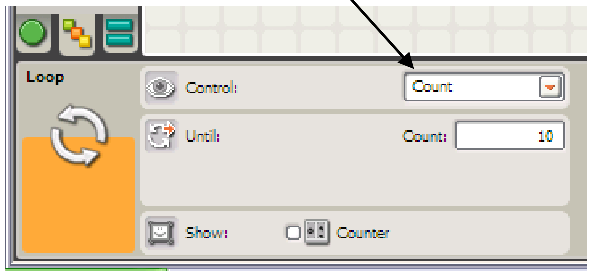
The purpose of this book is not for learning NXT-G, but for investigating ways to let students learn math with hands-on robotics programming. Therefore it is assumed that you know some basics of NXT-G.

2.1 Loop – DO UNTIL (Termination Condition)

How many beeps can you hear from the following program?



(numBeeps0.rbt)



Loop

Control: Count

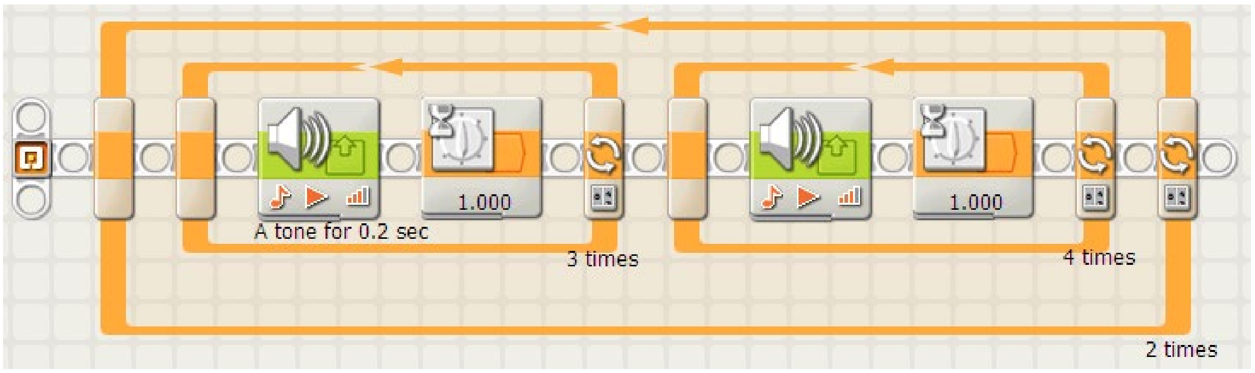
Until: Count: 10

Show: ☐ Counter

Yes, 10. But note that the counter begins with zero.   0, 1, 2, 3, 4, 5, 6, 7, 8, 9

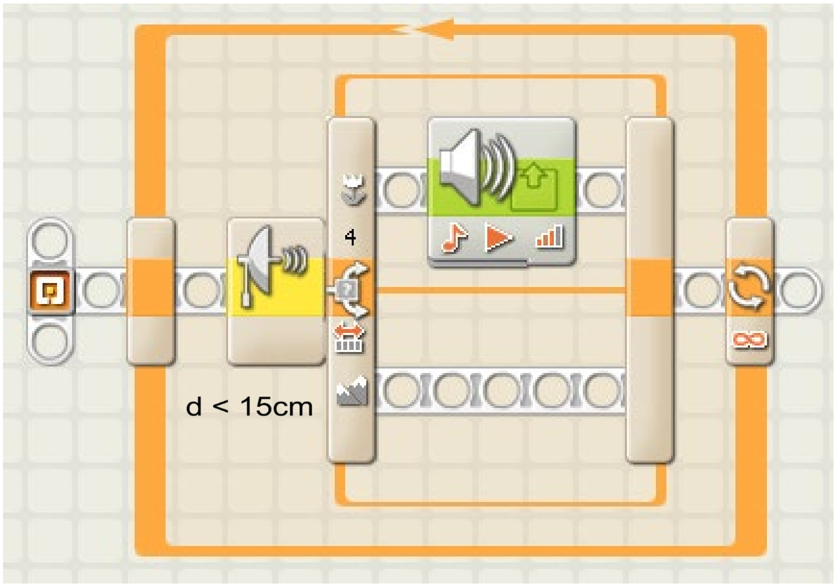
2.2 Nested Loop

How many beeps can you hear? Please review and discuss about distributive law.



(numBeeps1.rbt)

2.3 Decision Making

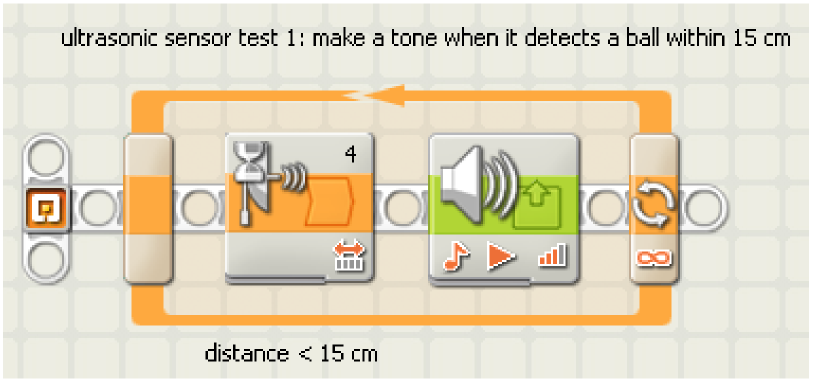


(usTst2.rbt)

Explain the behavior of the above program. Discuss range, equality, inequality, and logic.

2.4 Event Driven Programming

The program in 2.3 (usTst2.rbt) keeps checking the data from the sonar sensor to see if the sensor detects an object located less than 15 cm away or not. The same program can be re-written using so-called event-driven programming method.

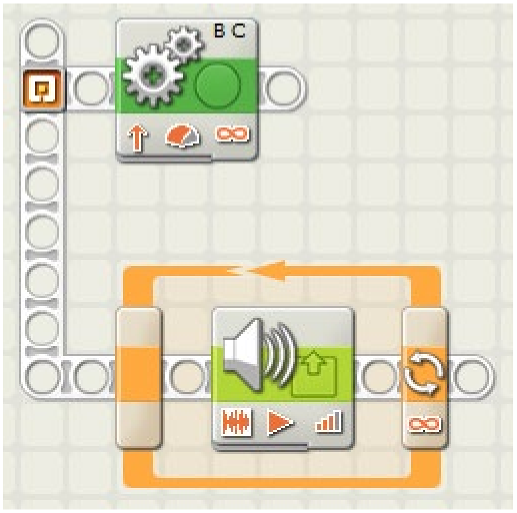

ultrasonic sensor test 1: make a tone when it detects a ball within 15 cm
4
distance < 15 cm

(usTst1.rbt)

Note that the loop can be counted instead of the "forever (∞)" condition. However, if the loop termination condition is based on a sensor watcher, this will not work. See (usTst11.rbt). Why is that? When the sonar sensor watcher is blocked and waiting for an object to come close, there is no way to reach the other sensor watcher to be executed.

Exercises: Let the robot move forward for a second when a clap is heard. Make two programs for this task, one using data-driven paradigm and the other using event-driven paradigm. (See clapMove0.rbt *and* clapMove.rbt)

2.5 Parallel Sequence Beams (PSB)



(psb.rbt)

How do you connect a parallel sequence beam?

- Move the mouse on the beam.
- Press shift. You will then see a spool.
- Drag and connect (click or double click).

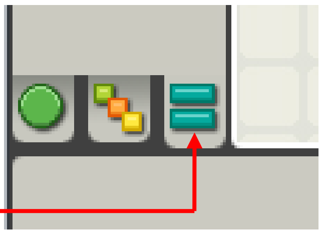
2.6 My Blocks

My Blocks are used for the following purposes:

- To wrap a procedure into a package
- To do the same thing from different places; to reuse code; to make code non-redundant
- To divide a large task into smaller meaningful modules (provide structure)
- To hide complex details
- Parameterized block = function

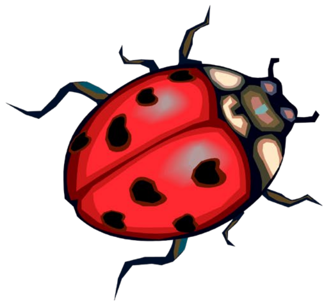
Creating MyBlocks

- Start with working code
- Highlight blocks to include
- Choose the “Make a New My Block” in the “Edit” menu
- Name the MyBlock
- Describe the MyBlock
- Build icon(s)
- MyBlock replaces the selected Blocks
- It is now available in the ‘custom’ palette



2.7 Variables and Data Connections

Why are variables are needed?



Reactive Machine

v.s.

State Machine

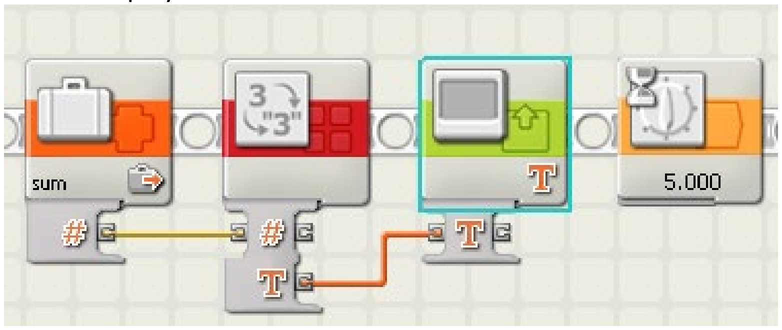
Variables are memory containers that hold numeric values. They enable the NXT to collect and store information. You can change the value during your program execution. Variables are used to pass values to parameterized MyBlocks.

How do you create a variable?

- EDIT | Define Variables

- Create
- Name the variable (as meaningful as possible)
- Select a data type
  - Number
  - Text
  - Logic (True / False)

How to display variables:



(displayVar.rbt)

The number needs to be converted to text. Then the text will be sent to the LCD display block to be displayed for 5 seconds before the termination of the program. Without the timer block at the end, you will not be able to see the displayed text since the program terminates so quickly.

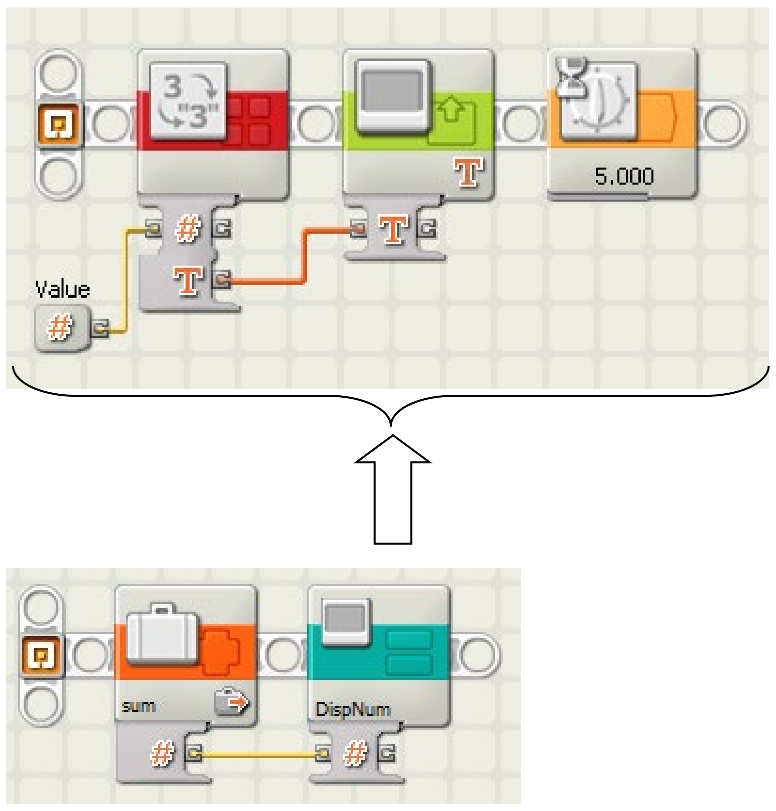
Data Wiring

- Data wires carry information between programming blocks.
- Many blocks require a data wire. For example, the output from a Random block.
- Open a block's data hub - by clicking the tab at the lower left edge of the block
- Drawing out a data wire: The cursor will change shape when it hovers over or is near a data plug. If you then press the mouse button and drag to the right, a data wire will "unroll" that can be connected to a plug on another block's data hub.
- To delete a data wire that stretches from left to right between two data plugs, click on the **right** plug.
- Data wire colors
  - wires carrying number data are colored yellow
  - wires carrying logic (true/false) data are colored green
  - wires carrying text data are colored orange
- If you try to connect a data wire to a plug of the wrong data type, the data wire will be "broken" and colored gray. You will not be able to download your program.
- If you click on a broken wire you can read why it is broken in the small help window.

2.8 Parameterized My Blocks

Let's create a generalized my block to display any variable. How to make parameterized MyBlocks:

- Introduce variables to the working code
- Highlight blocks to include only the working code part **without variables**
- Select the "Create My Block" in Edit menu
- The created My Block will have a formal parameter value to be connected to the right connector



2.9 Custom Blocks

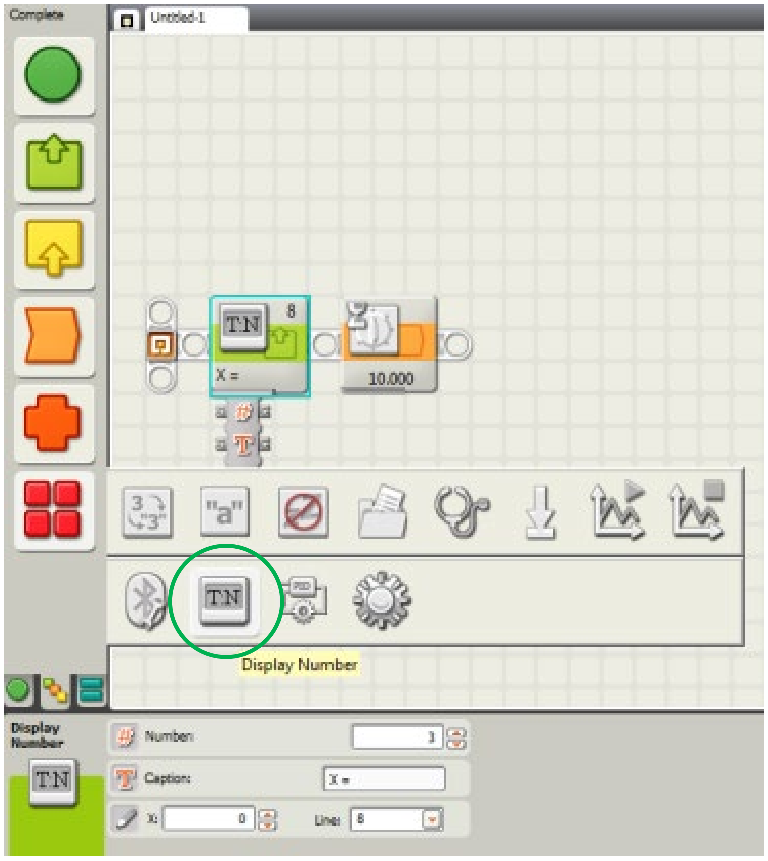
You will find custom made blocks on the web that will be loaded directly into your NXT-G software. For NXT-G V1.0, you will need to make sure the [Dynamic Block Update]() is installed. Those blocks are made using usually [NI's LabVIEW NXT Toolkit]().

Go to: [http://www.teamhassenplug.org/NXT/NXTGAdditions.html]() and download "Display Number Block v2.0" zip file made by Steve Hassenplug. (Or directly, [http://www.teamhassenplug.org/NXT/DisplayNumber.zip]())

To install the new NXT-G blocks:

- Unzip the files on your local machine (remember the folder where you unzip files)
- Run NXT-G
- Tools -> NXT-G Block Import and Export…
  - Click on the Browse button and Select the folder you extracted the files to and Click OK.
  - Select the "block" you need to install
  - Select the Advanced Palette
  - Click on Import Button
- Answer 'Yes' to any 'Replace ...' popup that might show up

"X = 3" will be displayed at line 8 (bottom line on the LCD) if you run the following program with the custom made NXT-G block.



Number: 3

Caption: X =

X: 0 Line: 8

(testDispBlock.rbt)

3

3 Learning Elementary Math with Basic Missions

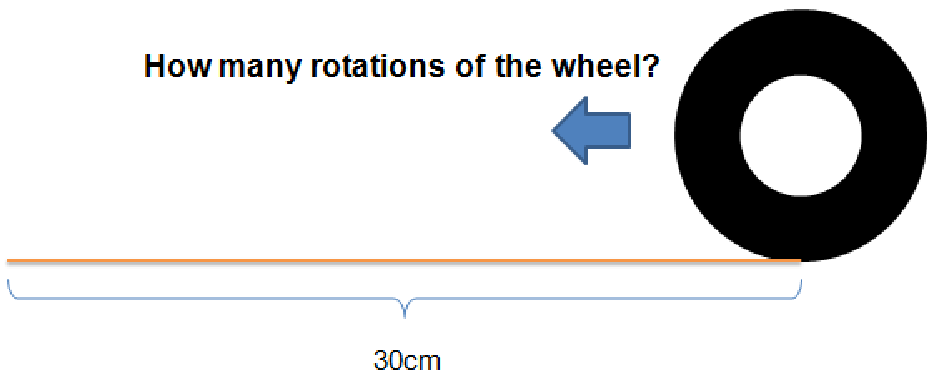
Learning objectives of this chapter are numbers, operations, functions, ratios, proportional logic, and basic geometry.

3.1 Mission – Let the TriBot Go straight 30cm exactly, and Stop

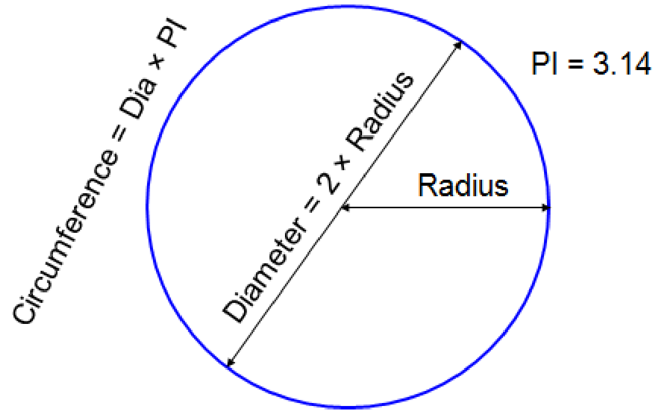
A simple idea can be to let it go forward for *x* seconds using a timer. Then the question is how you can find *x*? Trial and errors will take long time. More trouble is it will not work when battery becomes weaker or new batteries are installed. If a timer is used, the travel distance is dependent on the battery voltage. For a given time, with fresh batteries, it will go further.

Drive manually and view the rotation sensor value. If the travel distance is changed, you need to redo this step. Not a general solution.

Then what will be a better solution? Let's **<u>do the math</u>** to find exact number of rotations needed as depicted in the following picture.



Let's Review circle basics:



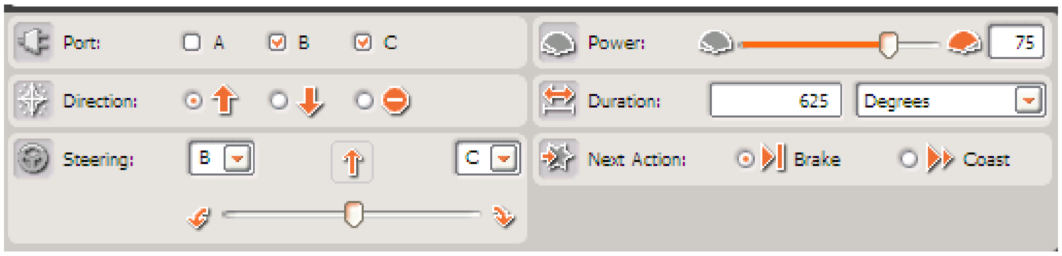
To calculate number of rotations needed to travel 30cm exactly, we need to measure:

- Diameter of the wheel: 5.5cm
- Circumference of the wheel: 5.5x3.14=17.27cm

How many rotations? 30 / 17.27 = 1.737116

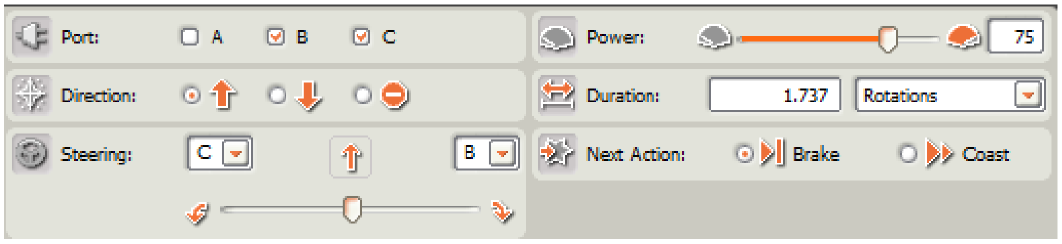
How many degrees? 1.737116*360=625

(go30cm.rbt)



Port: A [✓ B] [✓ C]
Direction: [✓ ⇧] ⇩ ⊖
Steering: B C
Power: 75
Duration: 625 Degrees
Next Action: [✓ Brake] Coast

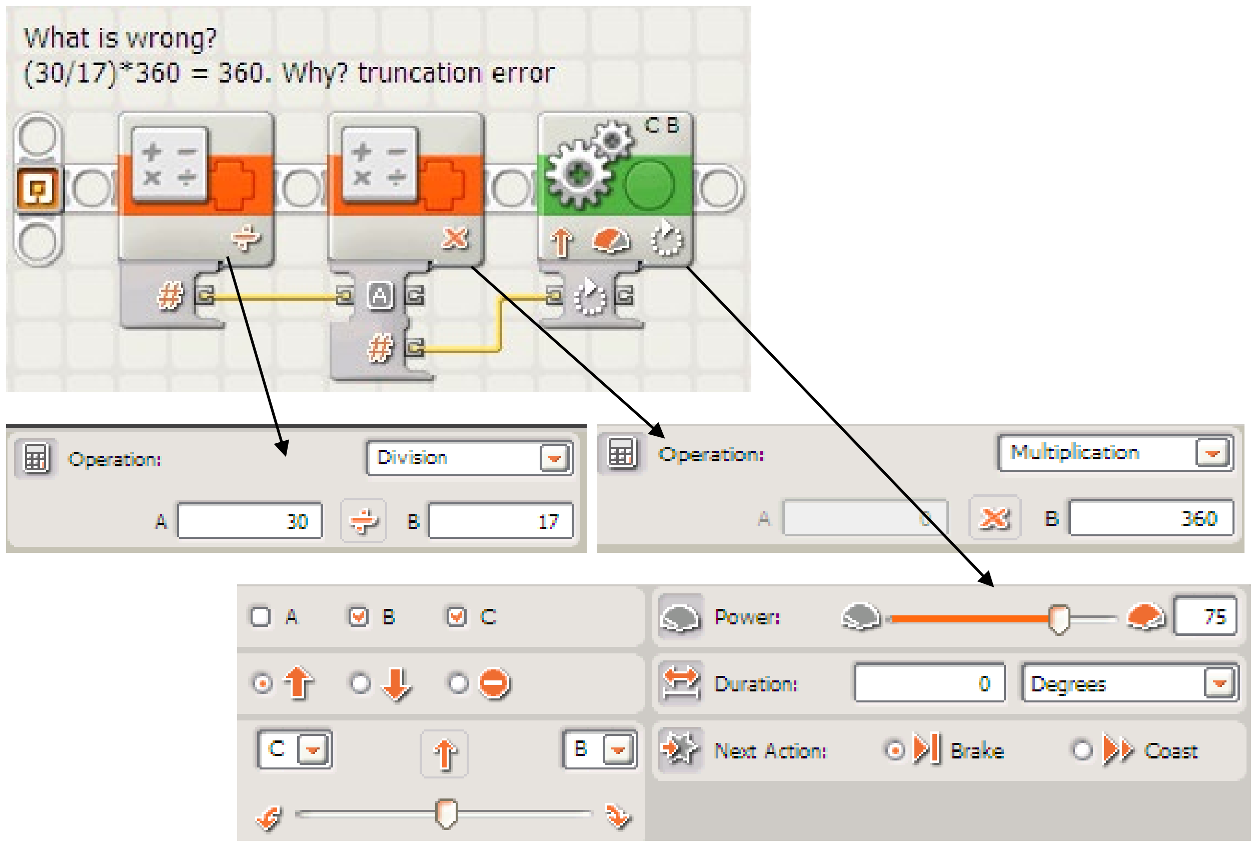
or



Port: A [✓ B] [✓ C]
Direction: [✓ ⇧] ⇩ ⊖
Steering: C B
Power: 75
Duration: 1.737 Rotations
Next Action: [✓ Brake] Coast

3.2 Mission – Let the TriBot Go straight *X* cm exactly, and Stop

What is wrong with the following (goXcm.rbt) program?



What is wrong?
(30/17)*360 = 360. Why? truncation error

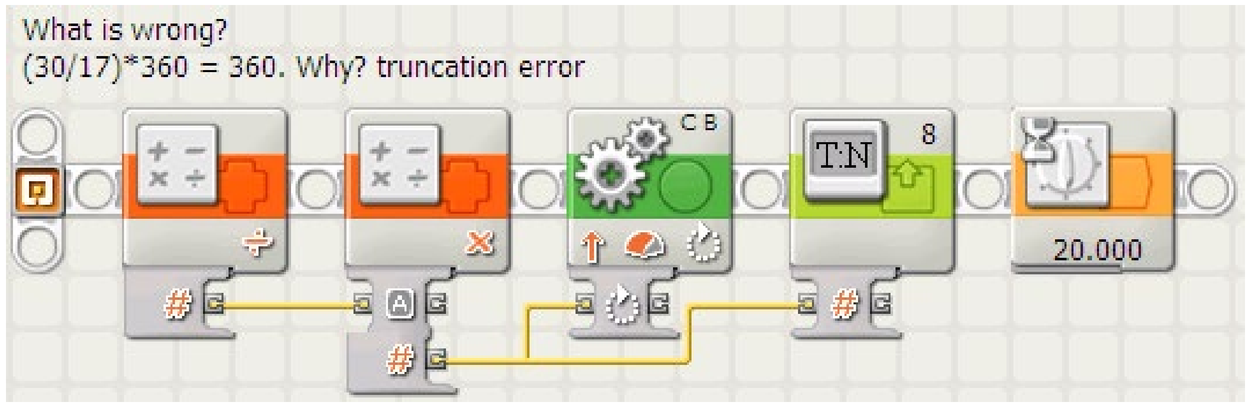
Operation: Division
A 30 ÷ B 17

Operation: Multiplication
A ✕ B 360

☐ A ☑ B ☑ C

Power: 75
Duration: 0 Degrees
Next Action: [✓ Brake] Coast

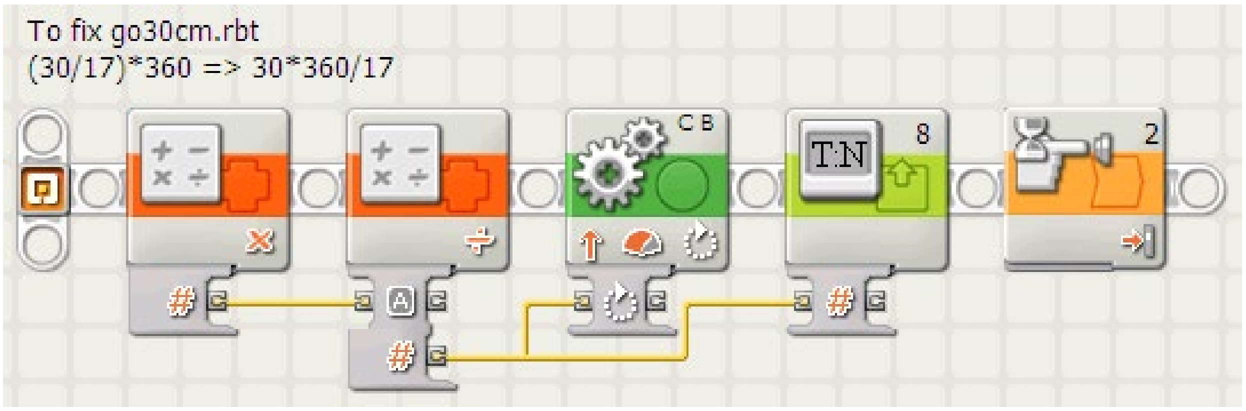
Let's display the calculated output number.



What is wrong?
(30/17)*360 = 360. Why? truncation error

(goXcm1.rbt)

"360" will be displayed for 20 seconds before ending the program. To fix the problem, the following goXcm2.rbt program changes the order of calculations.


To fix go30cm.rbt
(30/17)*360 => 30*360/17

(goXcm2.rbt)

"625" will be displayed until touch sensor is pressed.

Review Question: How many centimeters can the TriBot go forward with 1000 degrees? Estimate the distance by calculation then compare it with the actual experiment.

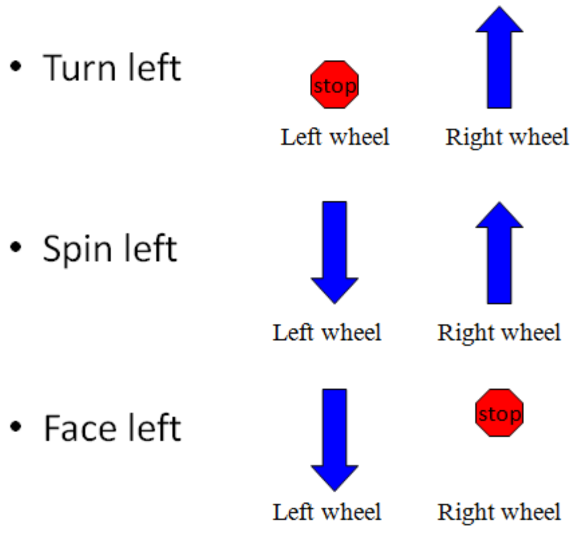
Wheel travel distance = numRotations * diameterOfWheel * pi

= 1000/360 * 5.5 * pi

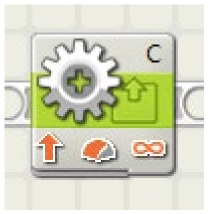
Exercise: Go forward exactly 0.5 meter (50cm)

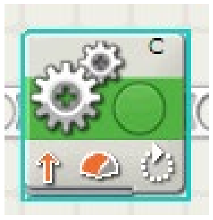
3.3 Mission - Turn 30 degrees

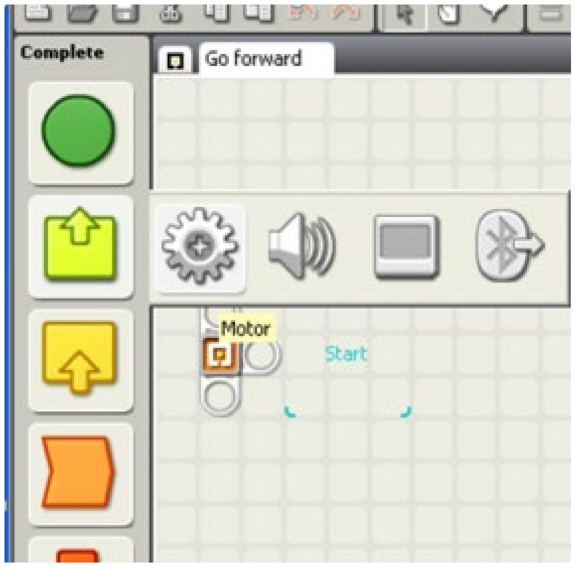
How to making turns, spins, and faces is explained in the following diagram:



Move vs. Motor Blocks: Move blocks don't allow for precise turns because one wheel is in neutral.


Motor Block

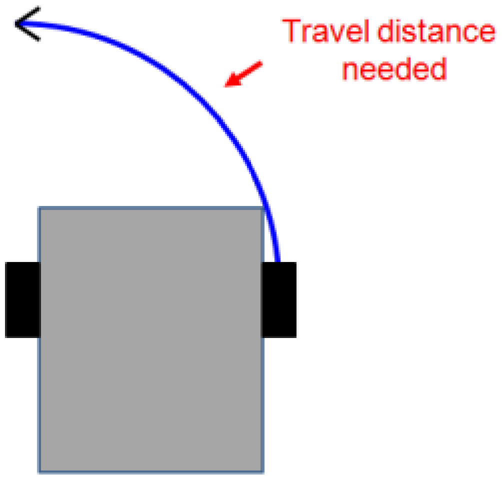

Move Block

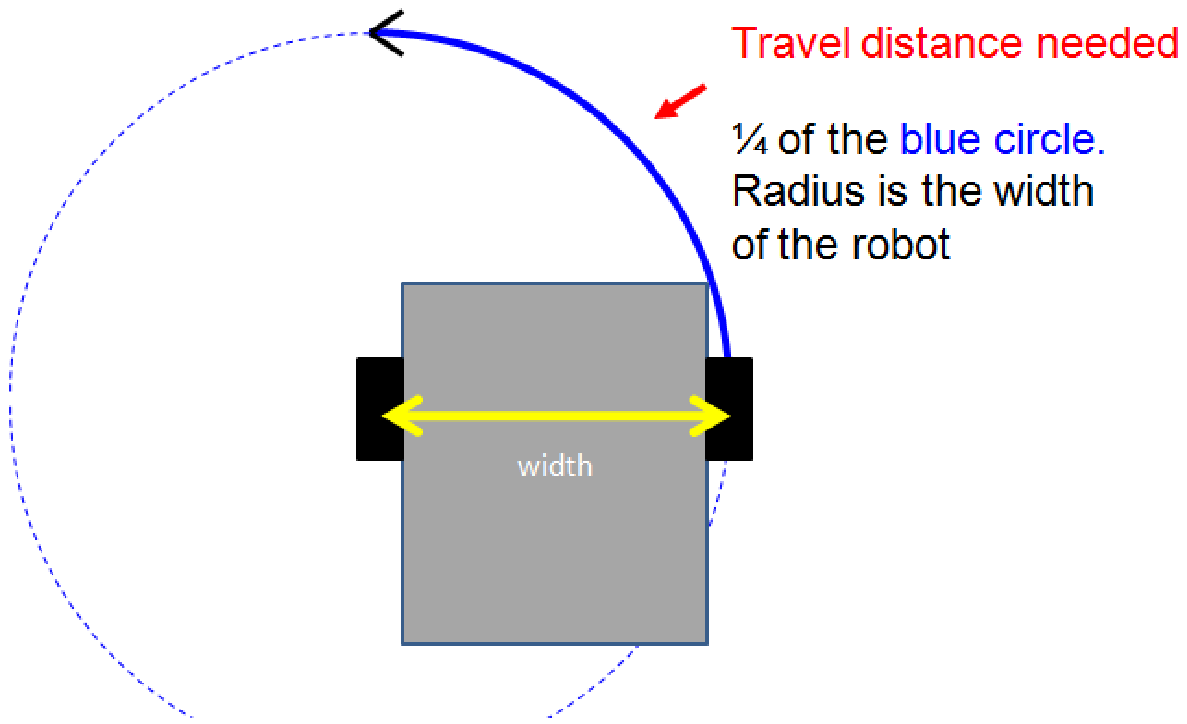


Where to get the Motor block

How can we make it turn 90 degrees?

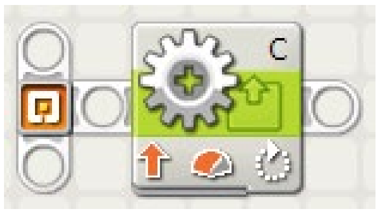
- Power one motor for a certain time using a timer and stop. To find the right timer duration by trial and errors.
- Drive by hand and view the rotation sensor value. If the angle to turn is different, you need to redo this step. Not a general solution.
- Let's do the math again to find a general formula.

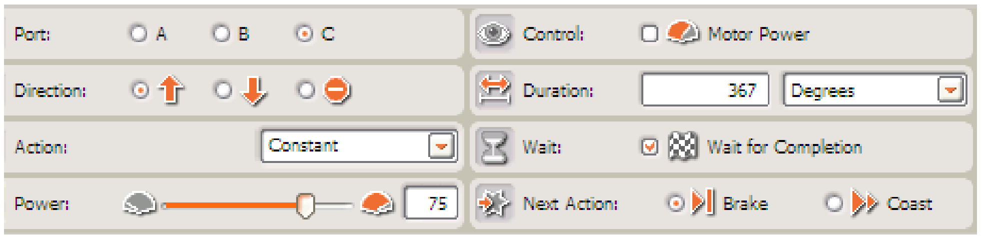




- Width of the robot: 11.2 cm
- Diameter of the wheel: 5.5 cm
- Circumference of the wheel: 5.5cm x pi = 17.27

Travel distance needed = (11.2 x 2 x pi) / 4 = 17.584 cm
Number of rotations needed = 17.584 / 17.27 = 1.01818
Or number of degrees needed = 1.081818 * 360 = 367



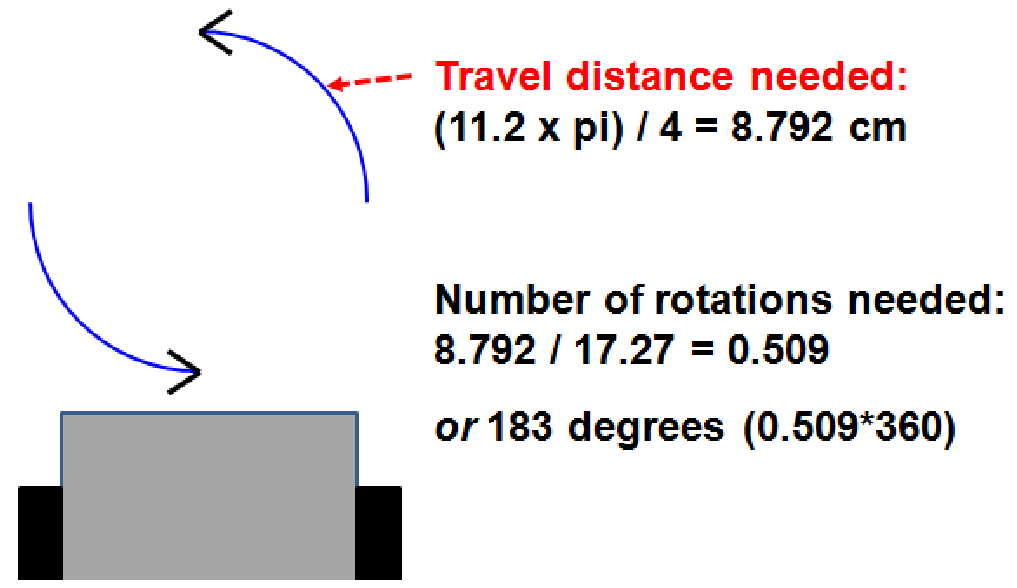


Port: ( ) A ( ) B [✓ C]
Direction: [✓ ↑] ( ) ↓ ( ) ⊖
Action: Constant
Power: 75
Control: ☐ Motor Power
Duration: 367 Degrees
Wait: ☑ Wait for Completion
Next Action: [✓ Brake] ( ) Coast

(Turn90deg.rbt)

3.4 Mission - SPIN 90 degrees

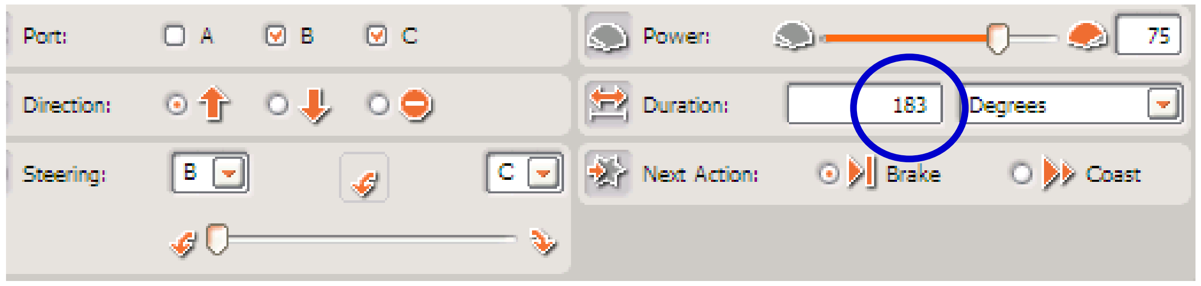
- Width of the robot: 11.2 cm (diameter of the circle)
- Diameter of the wheel: 5.5 cm
- Circumference of the wheel: 5.5cm x pi = 17.27



Travel distance needed:
(11.2 x pi) / 4 = 8.792 cm

Number of rotations needed:
8.792 / 17.27 = 0.509

***or* 183 degrees (0.509*360)**



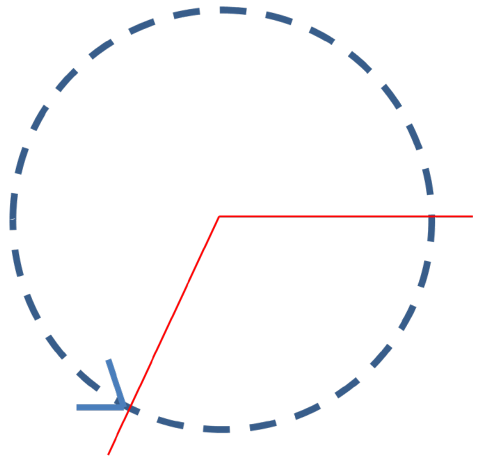
Port: A [ ] B [✓] C [✓]
Direction: [✓ forward] backward stop
Steering: B C
Power: 75
Duration: 183 Degrees
Next Action: [✓ Brake] Coast

(Spin90deg.rbt) using Move/Steering

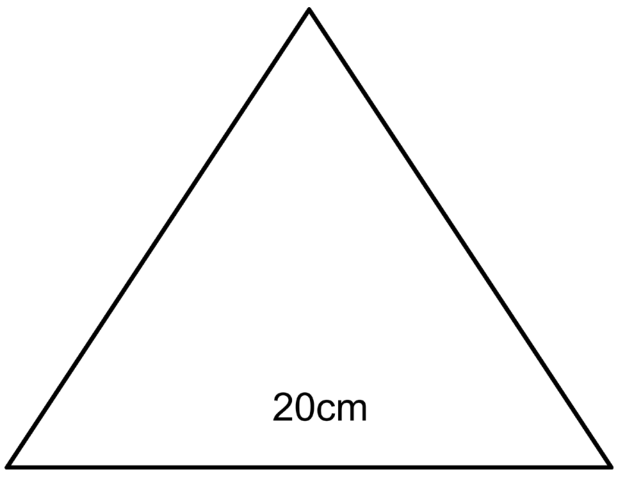
Exercise: spin π/3 (radian)

Review Question: If we use 500 degrees for the Move block in the previous spin90deg.rbt program, what is the degree the robot is spinning?

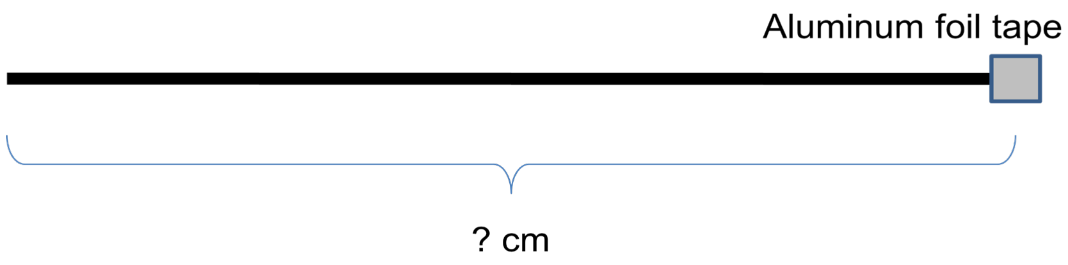
- Wheel travel distance = (500/360 rotations) * 17.27 cm = 23.99 cm
- Circumference of the spin circle with 11.2 as the diameter = 35.17 cm
- 23.99/35.17 = 0.68 circles
- => 246 degrees



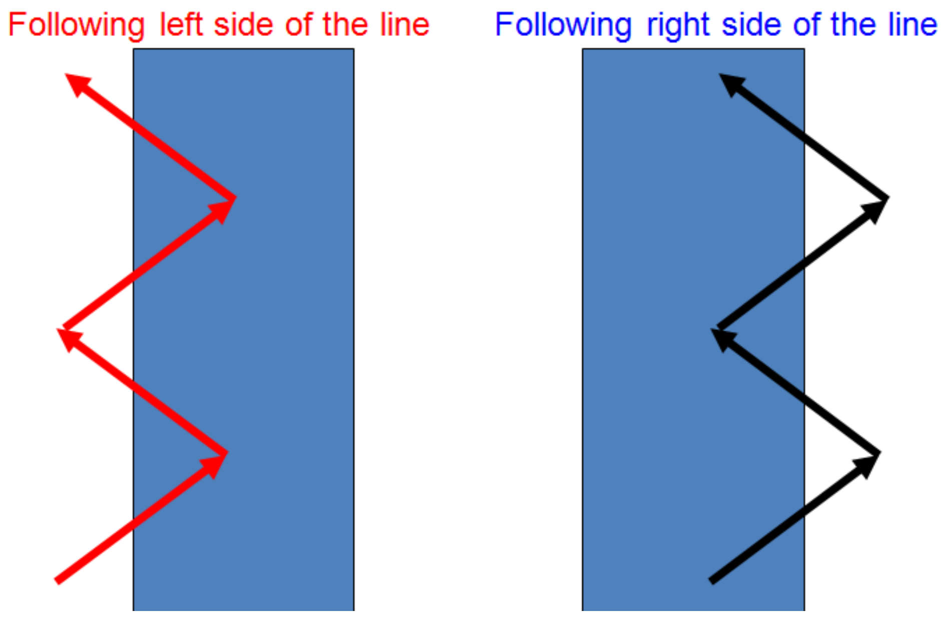
Challenge: Build a (new) robot to hold a pen at the center of the axle point. Draw the following equilateral triangle on a paper. (Robofest 2009 Game unknown problem)

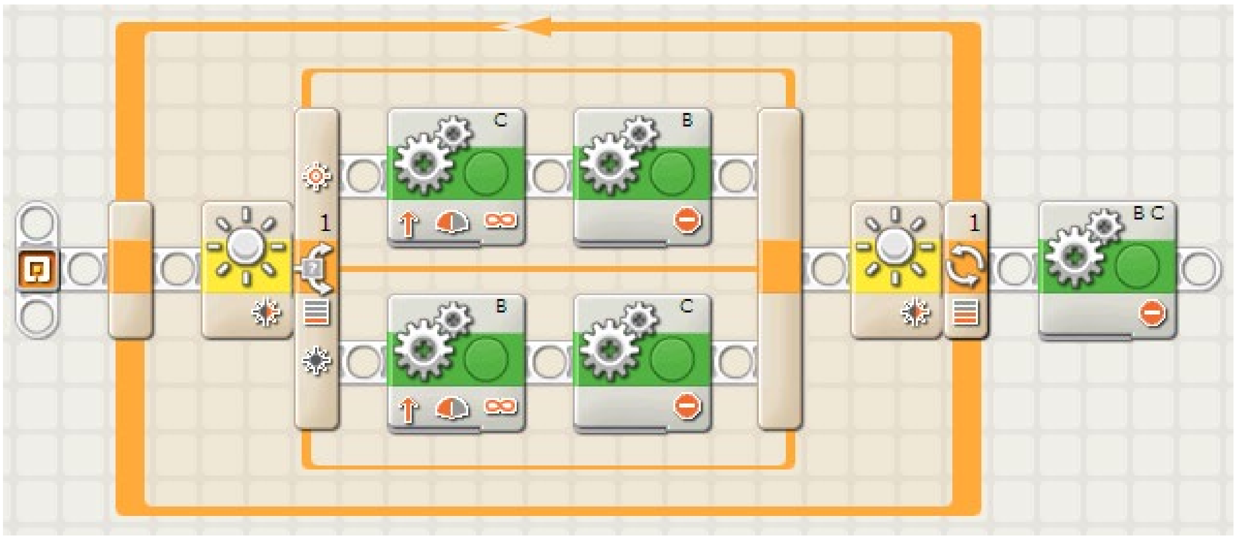


3.5 Mission – Measure a line length



3.5.1 Zigzag line following


Following left side of the line
Following right side of the line

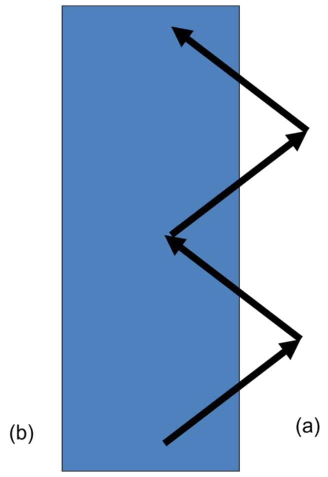


(LineF.rbt)

This is a rudimentary line following program to follow the right side of the black line on the bright surface. This program is using "coast" to stop a motor. Try "**brake**" to see the differences.

Question 1: Can we start the robot on the right side of the black line on bright area, (a)? yes

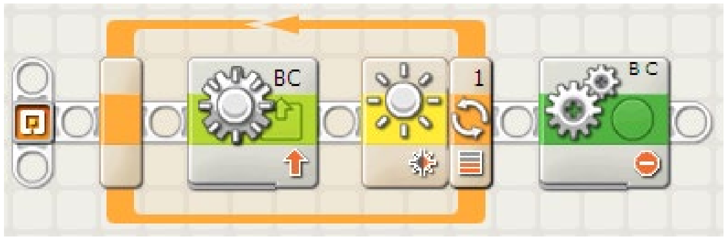
Question 2: Can we start the robot on the left side of the black line, (b)? No.

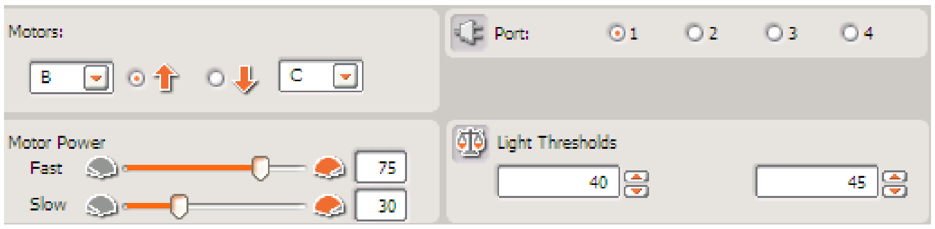


3.5.2 Line following using a custom block made by Steve Hassenplug

Download and install "Line Following Block" from http://www.teamhassenplug.org/NXT/NXTGAdditions.html . (Or directly at http://www.teamhassenplug.org/NXT/LineFollowingBlock.zip)

This block should be placed in loop, and it will set the two motors to follow the **right side edge of a black line**, based on one light sensor value. [Slow Motor Power] is the speed the inside motor will drive, when the robot is turning. [Fast Motor Power] is the speed of the outside motor, when the robot is turning, and the speed of both motors, when the robot is directly on the edge of the line. The [Light Thresholds] define the light and dark thresholds for the line. Tests show this single block will execute about 2 to 3 times as fast as the same code in NXT-G.



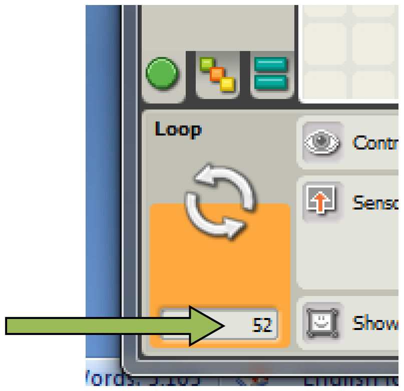


Motors: B ◉↑ ○↓ C
Port: ◉1 ○2 ○3 ○4
Motor Power
Fast 75
Slow 30
Light Thresholds 40 45

(LineFB.rbt) Line Following Using the custom NXT-G block

To program, we must know the data values sensors are getting from the environment. How to get light sensor data values?

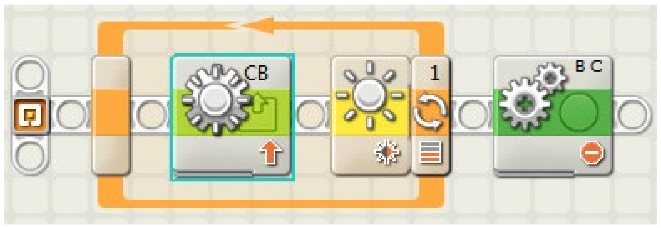
- Use NXT "View" function
- NXT-G offers a Live Update feature that lets you see what your robot is seeing. Do not forget to <u>*click on the sensor block*</u> to show the sensor values.

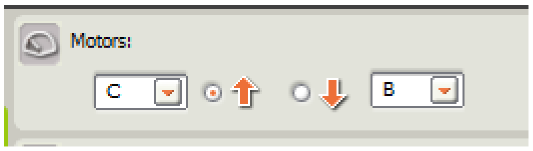


Tips using this custom block:

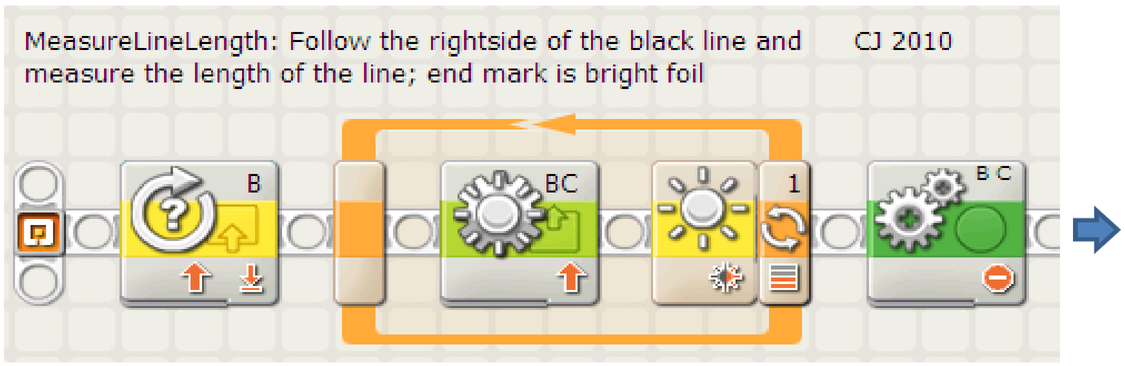
- If there is a sharp curve like 90 degrees, use zero for Slow.
- For Light Thresholds: Measure values for darkest (d) and brightest (b).
  - Use d+2 as left threshold value
  - Use b-2 as right threshold value

<u>**Question**</u>: Can we use the custom block to follow left side of the black line?
Answer: The idea is to change the turning behavior of the robot. For example, if it sees black turn left, instead of turning right. So change the motor connections: left to right and right to left. This can be achieved by changing the physical connector wires to the motor, or change the program as shown below.

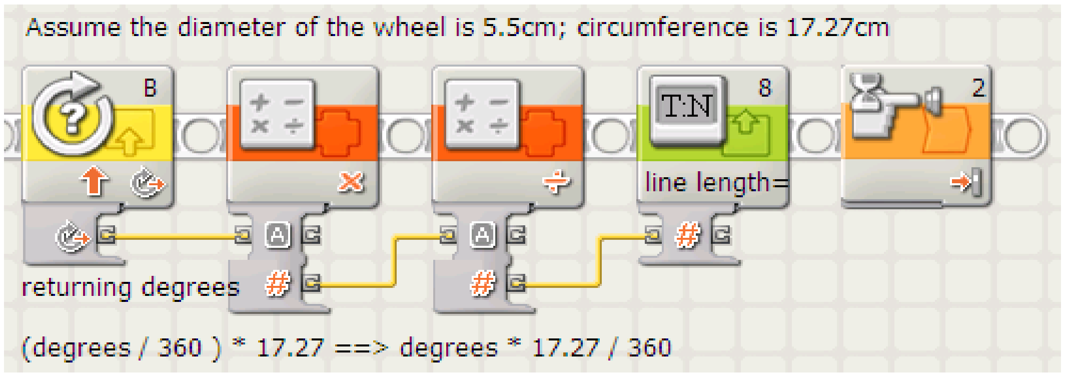



Motors: C B

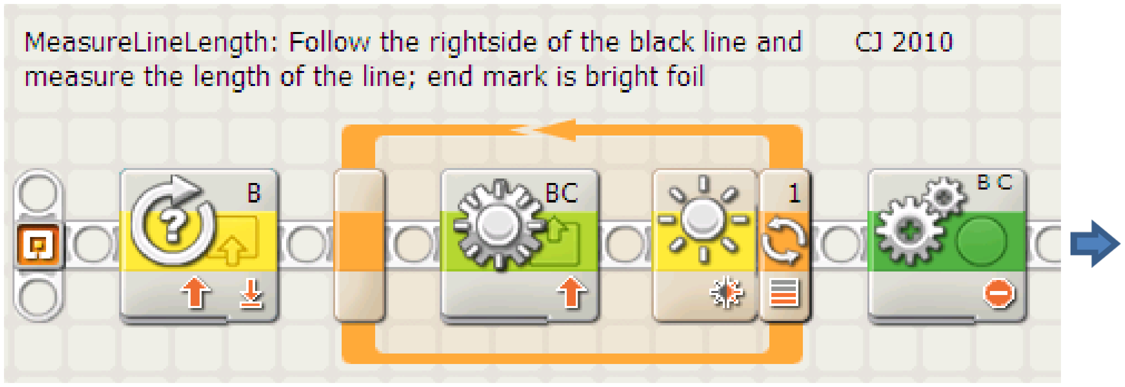
3.5.3 Measure a line length while following a line

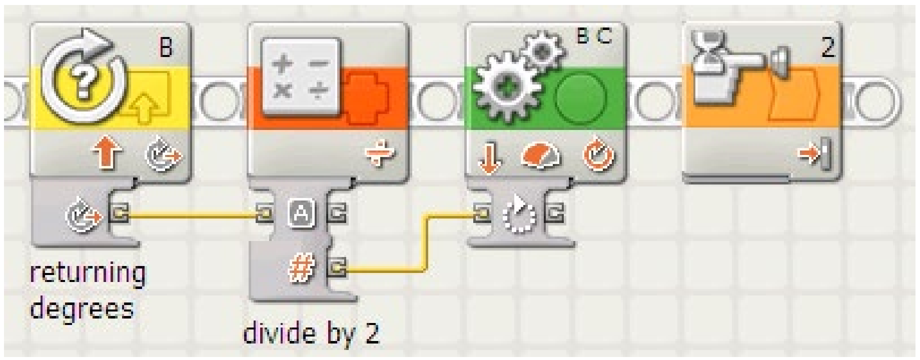

MeasureLineLength: Follow the rightside of the black line and measure the length of the line; end mark is bright foil

CJ 2010


Assume the diameter of the wheel is 5.5cm; circumference is 17.27cm
B
8
2
line length=
returning degrees
(degrees / 360) * 17.27 ==> degrees * 17.27 / 360
(LineLength.rbt)

3.6 Mission – Stop at the midpoint of a line

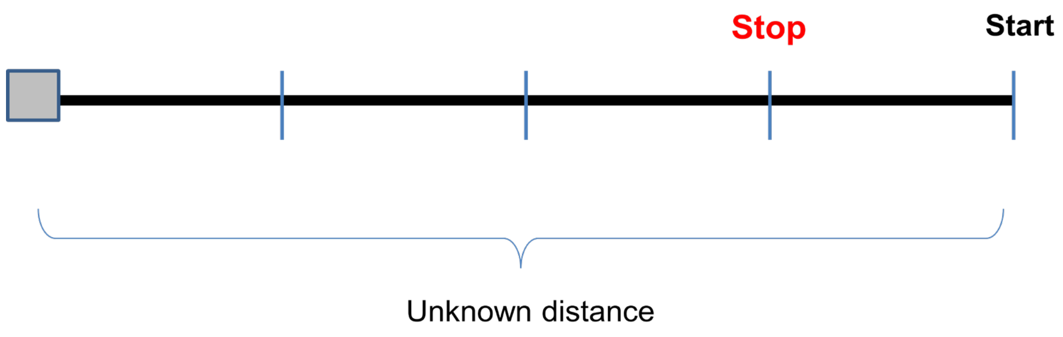

MeasureLineLength: Follow the rightside of the black line and measure the length of the line; end mark is bright foil
CJ 2010
B
BC
1
B C



(lineCenter.rbt)

Ideas to improve: Reverse line following is possible without changing the sensor location.

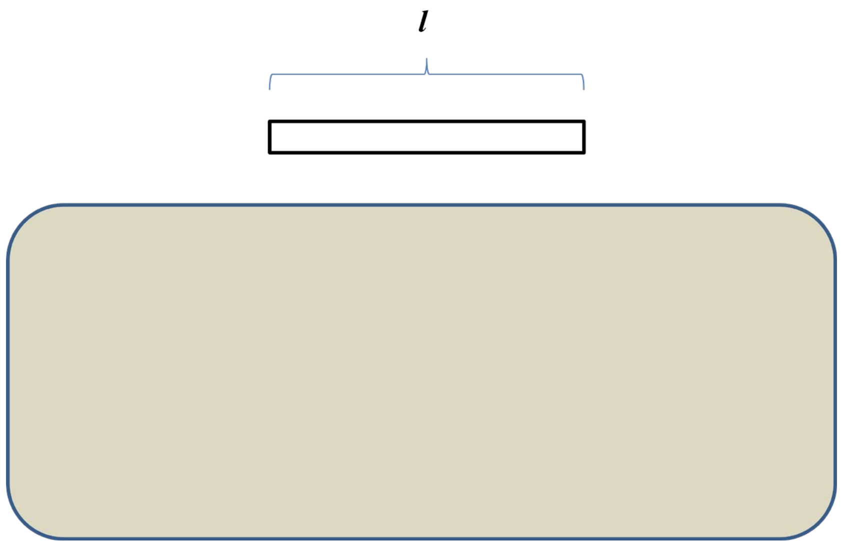
Exercise: Stop at ¾ point of a line. (See Line3-4th.rbt)



Exercise 1: Follow the edge of a table clockwise. Stop at the foil tape. (Hint: This is like following right side of a black line)

Exercise 2: Follow the edge of a table counter-clockwise. Stop at the foil tape. (Hint: This is like following left side of a black line)

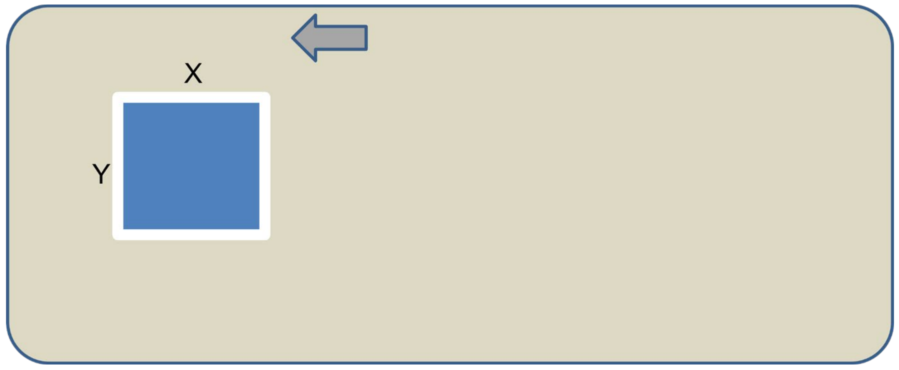
Exercise 3: Follow the edge of a table. Measure the length of a wall located outside the table. The wall has two shiny aluminum foil tapes at both ends. (A part of Robofest 2011 BTOS game)



Recommended steps to solve the above exercise:

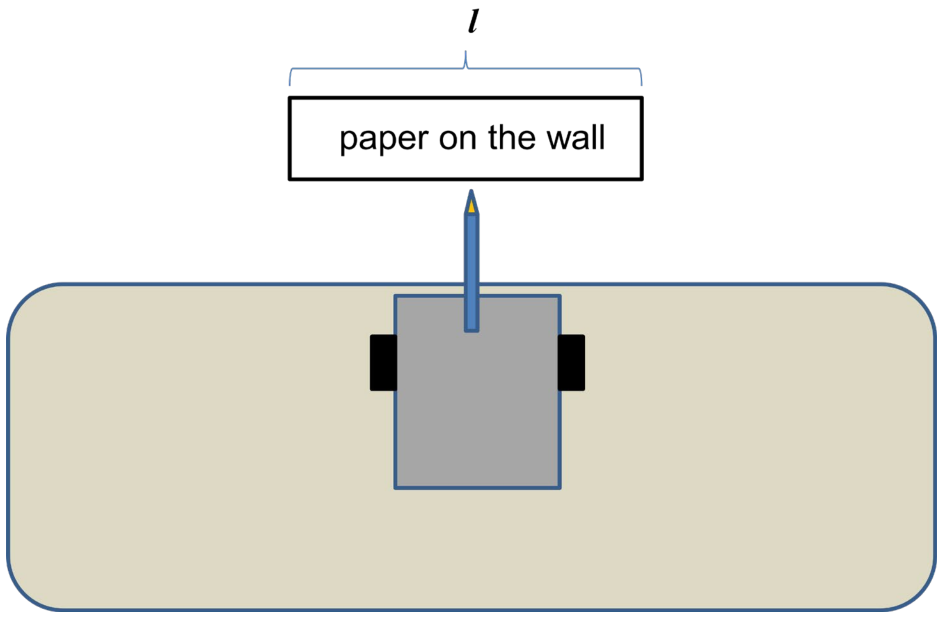
1. To follow the edge of the table
2. Stop at the start of the box
3. Stop at the end of the box
4. Display the length of the box

Exercise: Follow the edge of a table. Measure the length of the wall X of a box on the table and display it in cm. The color of the wall is white and it does not have the two shiny aluminum foil tapes at both ends.

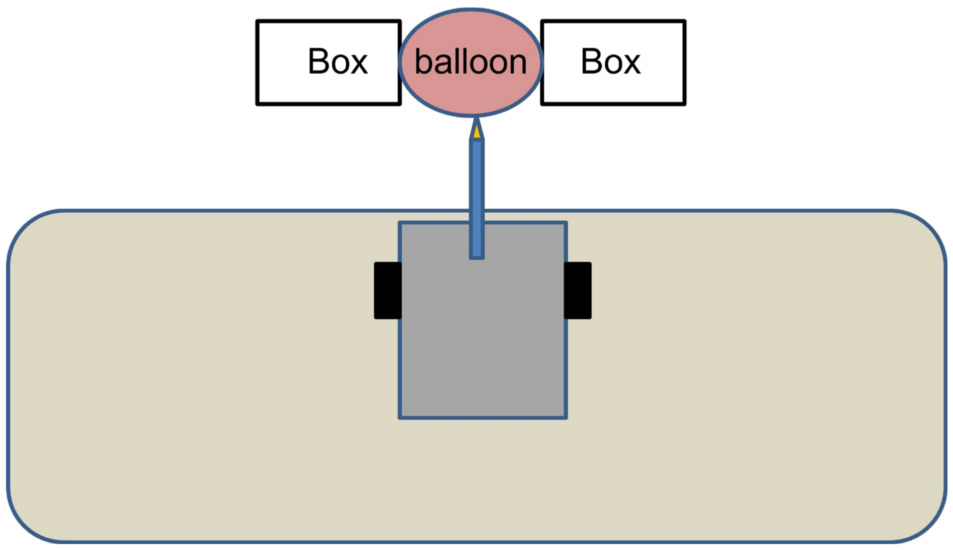


Challenge: Follow the edge of a table. Measure the length of the wall X and y of a box on the table as shown above. Display the area of the box in cm. The color of the wall is white and it does not have the two shiny aluminum foil tapes at both ends.

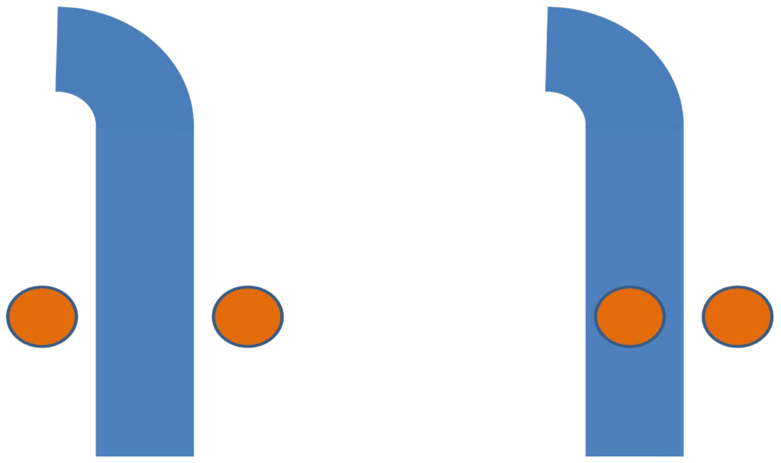
Challenge: Attach a pen to the center of the robot. Follow the edge of a table. Measure the length of the paper located outside the table. Rotate carefully. Mark the center point by driving forward using the pen.



Challenge: Attach a push pin to a long Lego piece then mount it to the center of the robot. Follow the edge of a table. Measure the length of the object connected outside the table. A balloon is placed in the center. Rotate carefully. Pop the balloon by driving forward using a push pin.



Challenge: We can improve the line following by introducing additional light sensor as shown below. Make two programs to follow a black line (1) when the black line is in between two light sensors (2) when left light sensor is on the black line and right light sensor is on the bright surface.



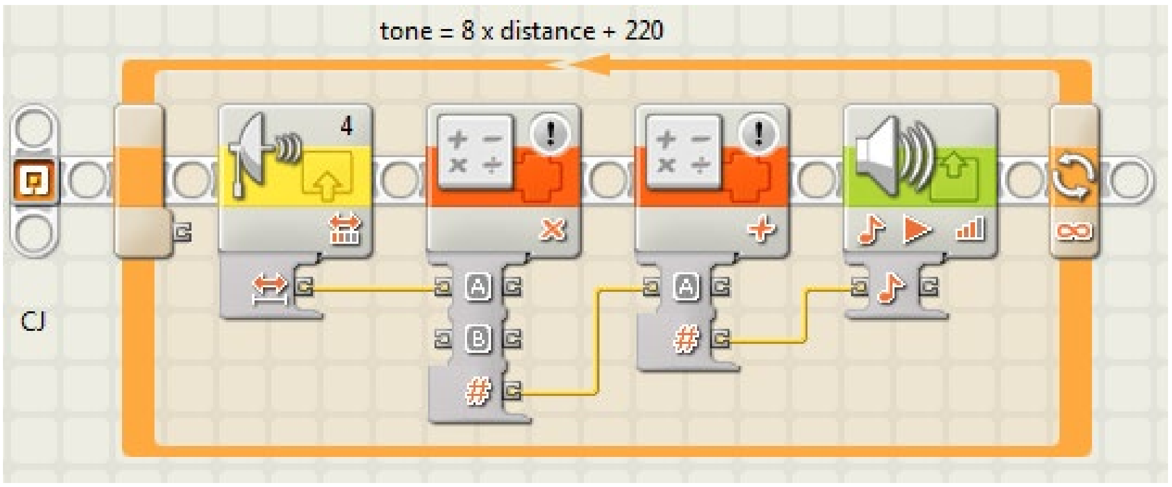
Challenge: (Dynamic threshold) Start the robot when the light sensor is on the black line. Store the light sensor value on the black line in a variable. Add 8 to the variable. Use the variable as a threshold value, i.e. if the current light sensor value is less than the threshold, regard it as black; otherwise white.

4

4 Numbers, Operations, Variables, and Algebra

4.1 Proportional Sound

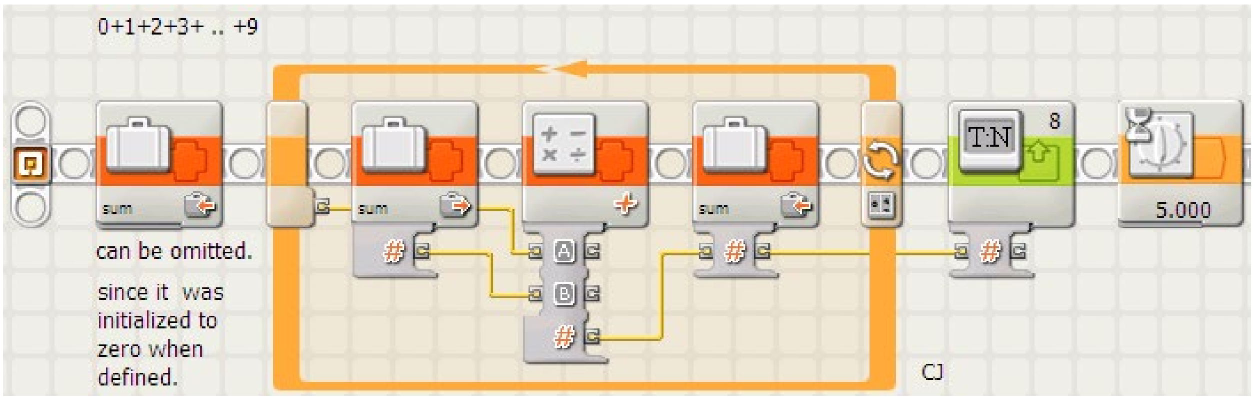
- Low A: 220
- High C: 1047



(proportionalSND.rbt)

4.2 0+1+2+3+…9 =?

$$\sum_{i=1}^{9} i$$

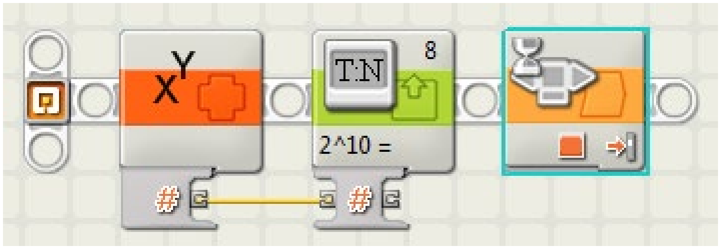


4.3 More Exercises using Variables

- Count up from 1 till touch sensor is pressed (DisplaySec.rbt)
- Count down from 10 to 1 (countdown.rbt)
- -1 -2 -3 -4 -5 -6 -7 -8 -9 = ? (negativeSum.rbt)
- 1+3+5+ … +99 (solution: sumOdds.rbt)
- 10! = 10 x 9 x 8 x … x 2 (factorial.rbt)
- 2^9 (2power9.rbt)
- To display numbers that are multiples of 3 from 3 until a touch sensor is pressed: 3, 6, 9, 12, 15, …

4.4 POW: X to the Y power function Block made by Michael Gasperi

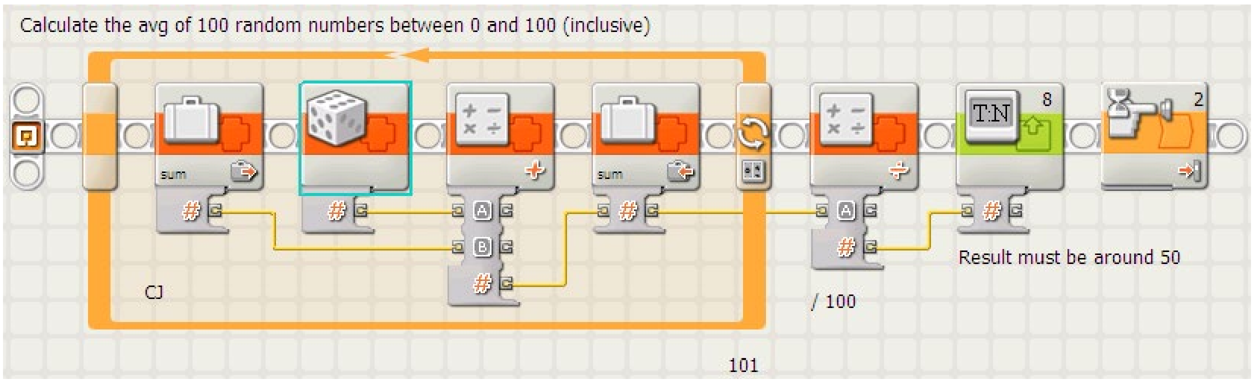
Download: http://www.extremenxt.com/pow.zip and install the block.

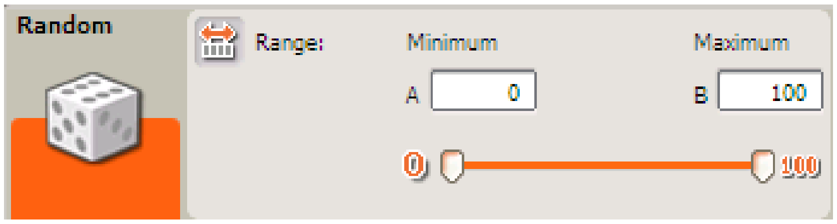


(pow.rbt)

4.5 Random Numbers

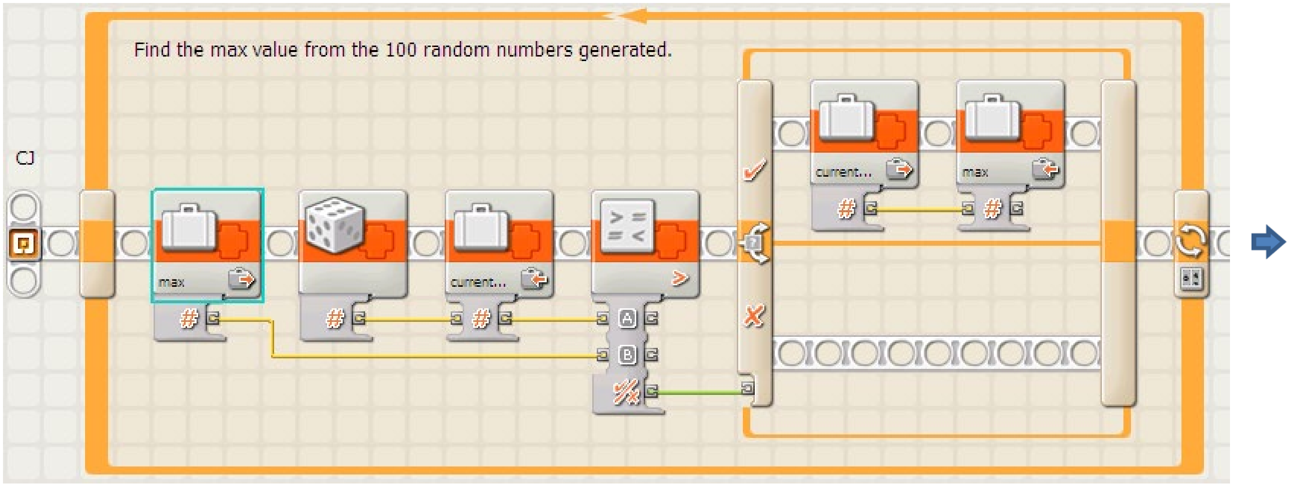
Mission: Calculate the Average of 100 random numbers between 0 and 100.

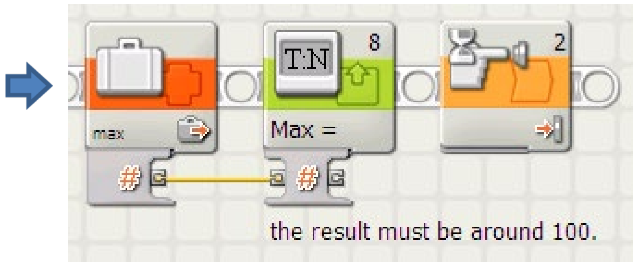

Calculate the avg of 100 random numbers between 0 and 100 (inclusive)
sum
sum
8
2
Result must be around 50
CJ
/ 100
101



(AvgRand.rbt)

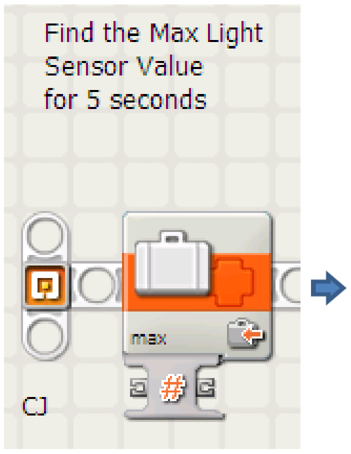
Mission: Find the max value from the 100 random numbers generated.

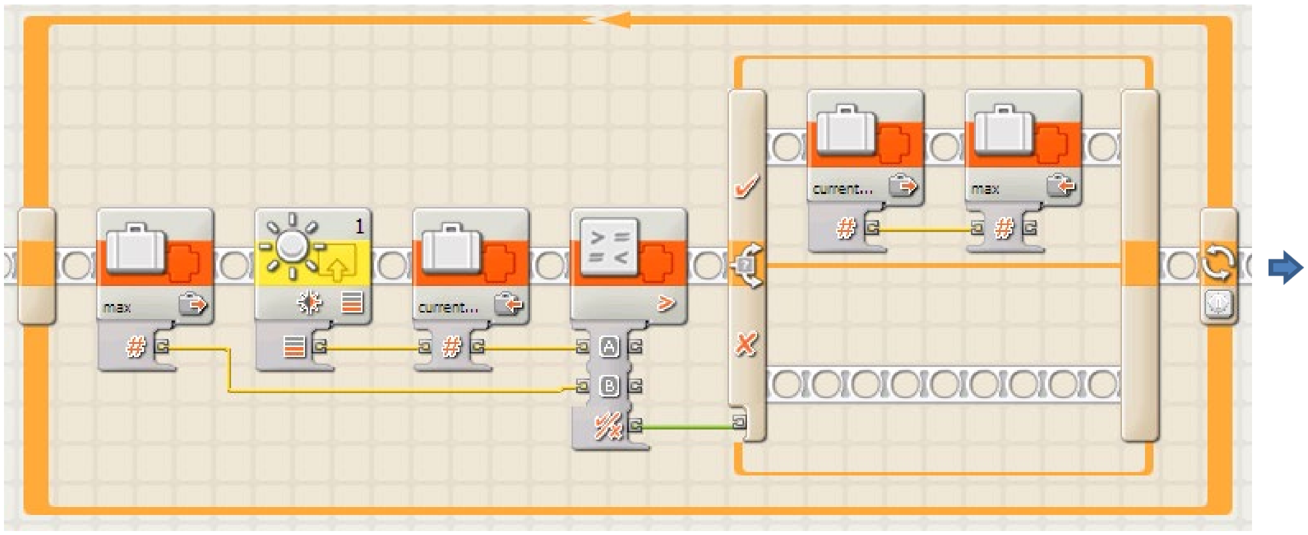


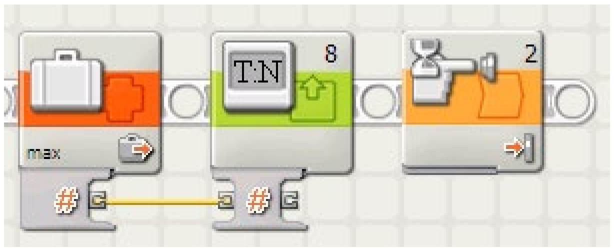


(MaxRand.rbt)

Mission: Find the maximum light sensor value for 5 seconds.







4.6 Geometric Progression

Can you guess the next number: 8, 4, 2, 1, **?**

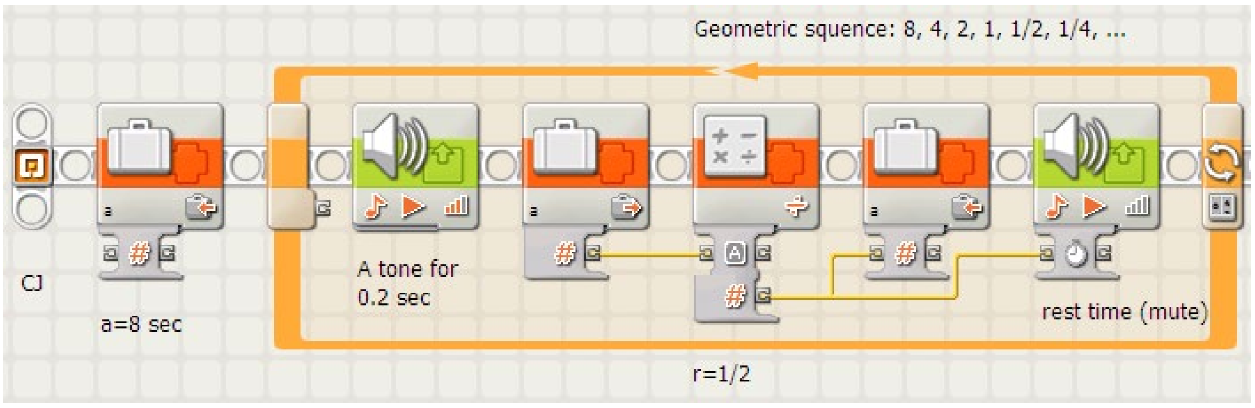
Geometric Sequence

- 8, 4, 2, 1, ½, ¼, …
- Initial value a = 8

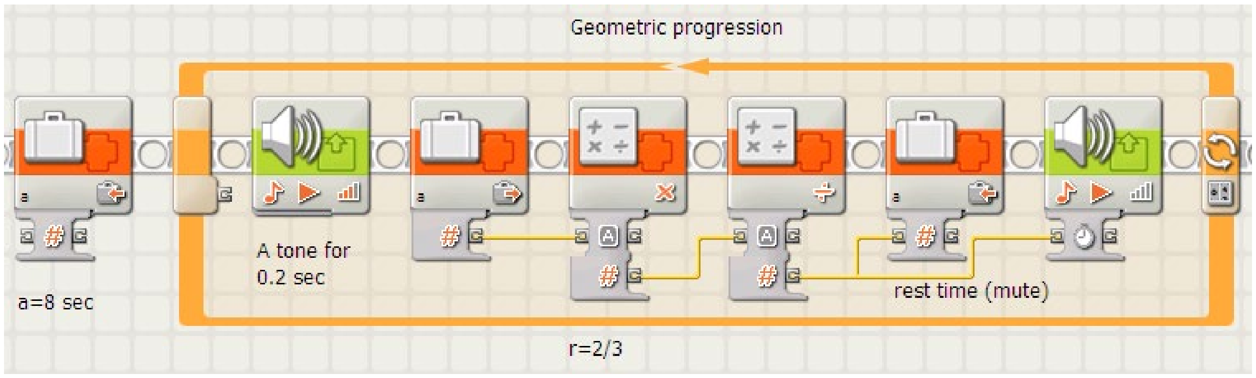
- Ratio r = ½

$$a_n = a \cdot r^{n-1}$$
$$a_n = r \cdot a_{n-1} \text{ for every integer } n \geq 1$$

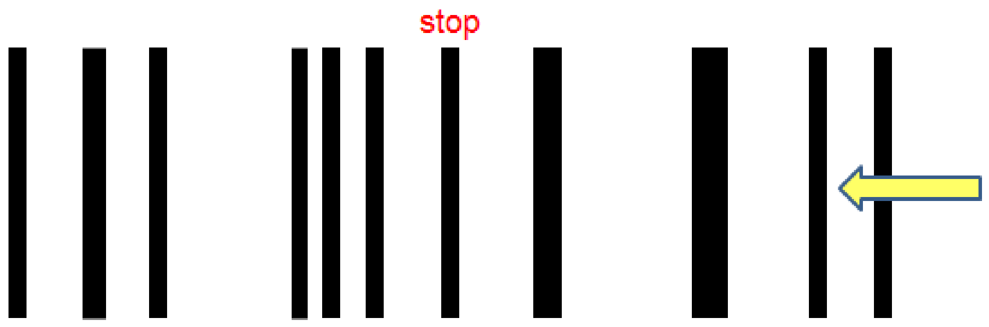
Mission: Play a tone using a geometric sequence duration – geoSeq.rbt



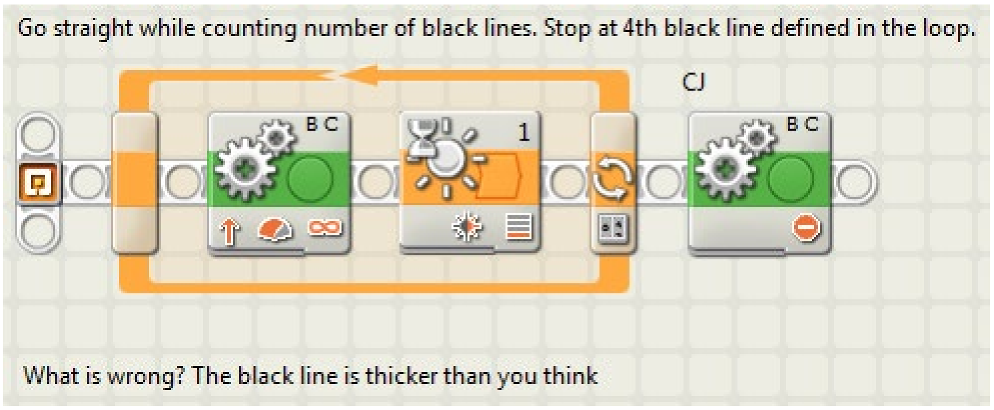
Mission: Play a tone using a geometric sequence duration (geoSeq2.rbt)



4.7 Mission (Counting): Stop at the 4th black line



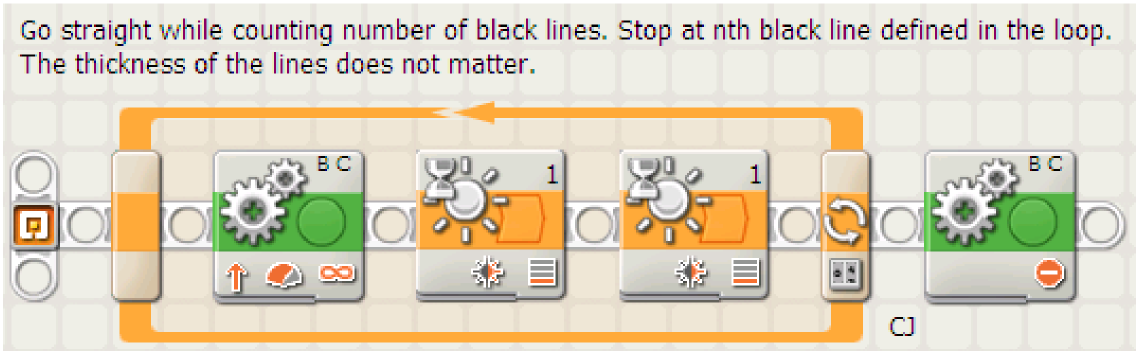
Note that some lines are thicker than others…


Go straight while counting number of black lines. Stop at 4th black line defined in the loop.

What is wrong? The black line is thicker than you think

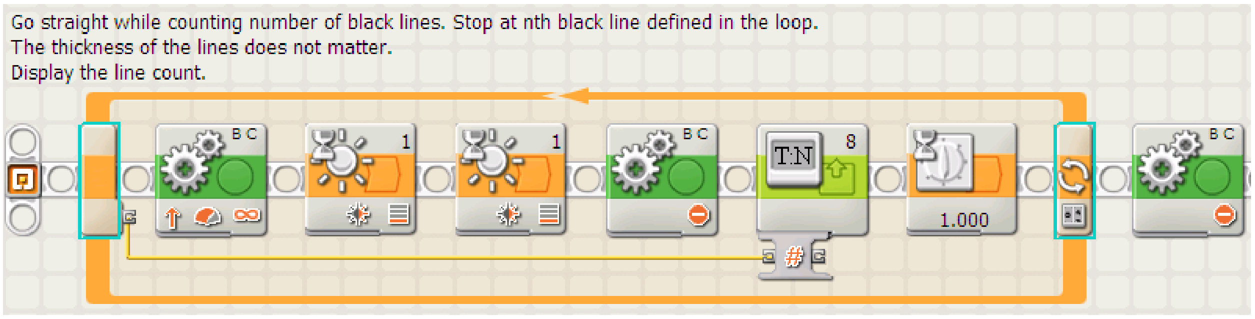
(stopXthL1.rbt)

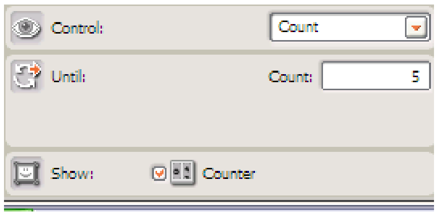
What is wrong with this program? The black line is thicker than you think; (or the program execution speed is much faster than you think)


Go straight while counting number of black lines. Stop at nth black line defined in the loop.
The thickness of the lines does not matter.

(stopXthL2.rbt)

stopXth2.rbt program has two wait light sensor blocks. The first one waits until dark (less than 40). The second one waits until bright (greater than 45). Now let's display a number when it detects each line. The following program will display numbers from zero till 4.

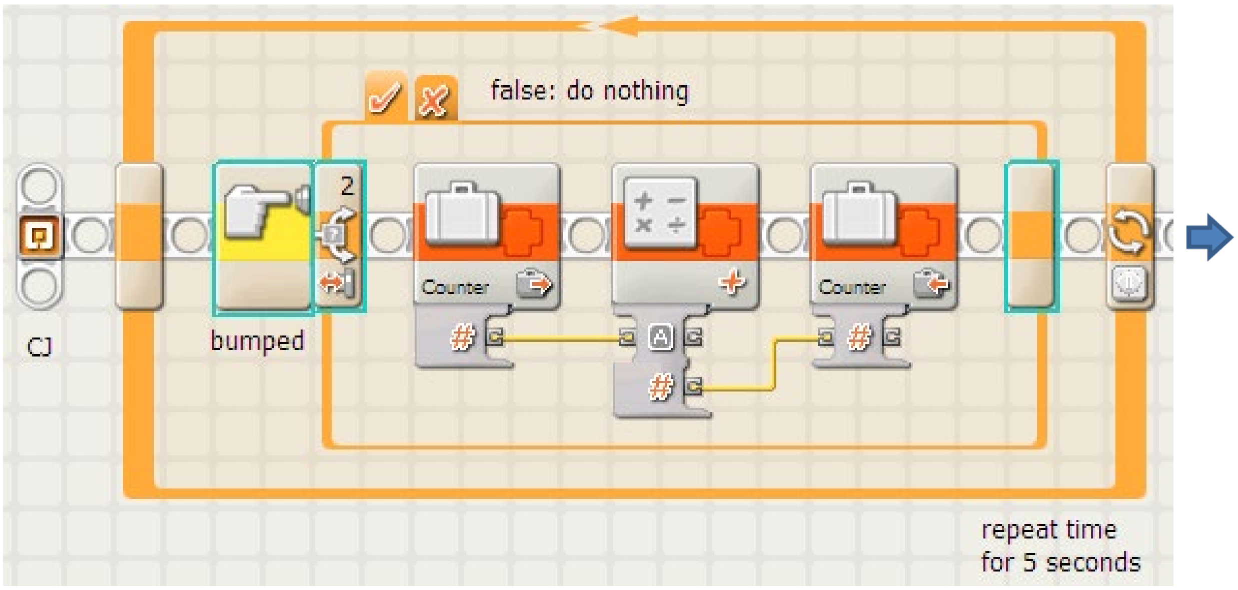

Go straight while counting number of black lines. Stop at nth black line defined in the loop.
The thickness of the lines does not matter.
Display the line count.

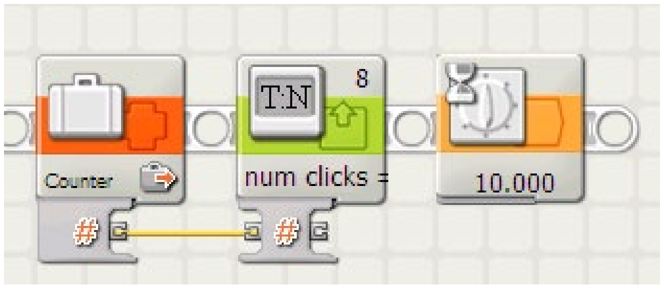

Control: Count
Until: Count: 5
Show: ☑ Counter

(stopXthL3.rbt)

Exercise: Revise this program to display 1 … 5.

4.8 Mission: Count # of touch sensor clicks (bumps) for 5 seconds





(countTouch.rbt)

Class tips: Do a small competition to find who the clicking champion in the class is!

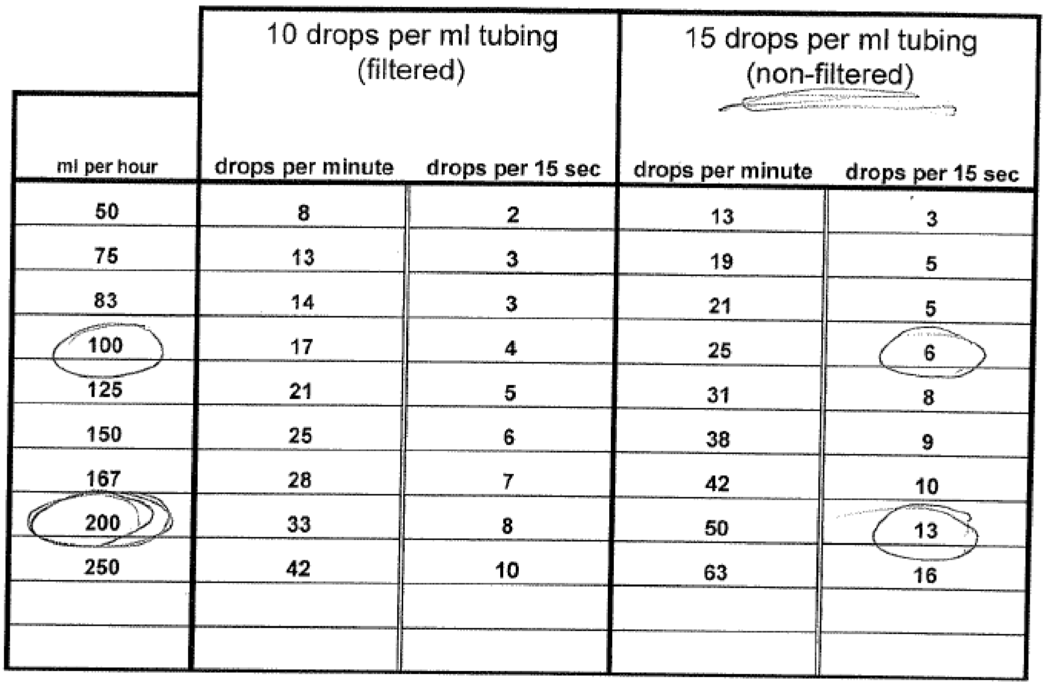
Challenge: Detect double claps

- Whenever it hears double claps (within 0.7 seconds), move forward 0.5 rotations.
- Use a variable and loop “time”
- Difficulty level of this challenge is high. Solution doubleClaps.rbt will be provided.

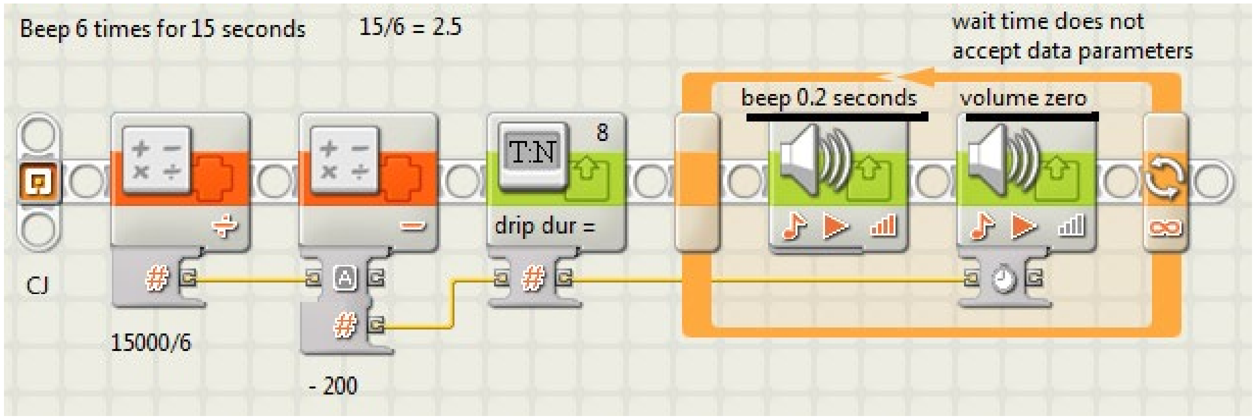
Exercise: Beep 6 times for 15 seconds

- Needed for Home intravenous (IV) antibiotic therapy using gravity method to accurately measure the number of antibiotic drops
- Make the program more general for other IV medications. See the table below.

Gravity Drip Calculator



	10 drops per ml tubing (filtered)		15 drops per ml tubing (non-filtered)	
ml per hour	drops per minute	drops per 15 sec	drops per minute	drops per 15 sec
50	8	2	13	3
75	13	3	19	5
83	14	3	21	5
100	17	4	25	6
125	21	5	31	8
150	25	6	38	9
167	28	7	42	10
200	33	8	50	13
250	42	10	63	16



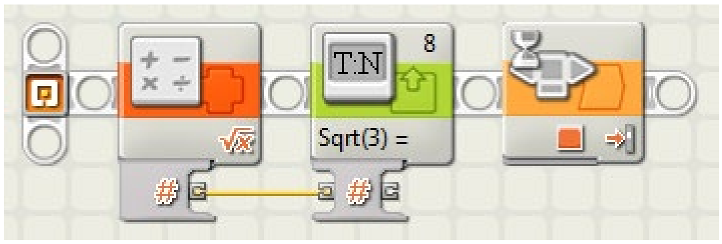
Beep 6 times for 15 seconds
15/6 = 2.5
wait time does not accept data parameters
beep 0.2 seconds
volume zero
8
drip dur =
CJ
15000/6
- 200

(countIVdrops2.rbt)

5

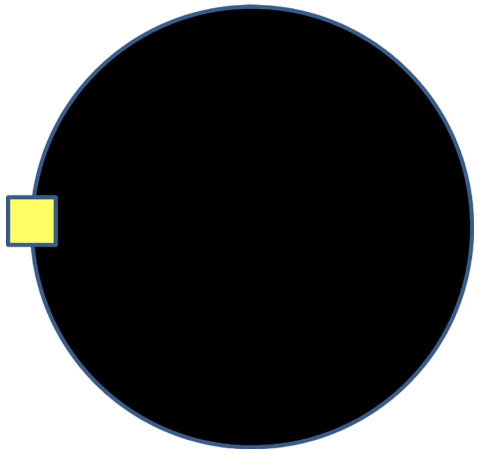
5 Geometry

5.1 Square Root using the Math Block (Not available in V1.x)



(sqrt.rbt)

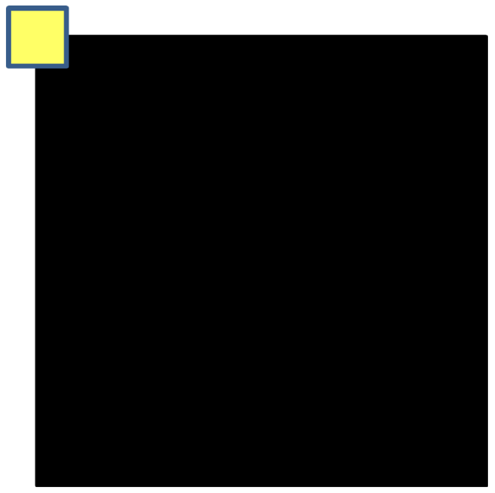
5.2 Exercise on Circles



A square aluminum foil tape is located on the edge of black circle shape as seen above. Calculate the following after measuring estimated circumference of the circle in cm:

- Estimated diameter of the circle
- Estimated radius of the circle
- Estimated area of the circle in cm^2

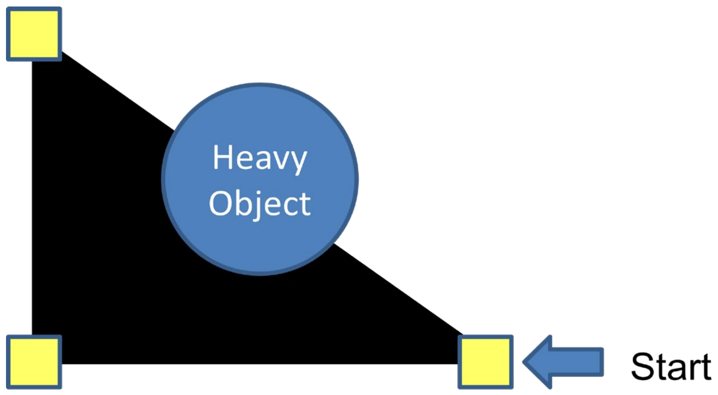
5.3 Exercise on Squares



A square aluminum foil tape is located at the corner of the black square shape as seen above. Calculate the following after measuring perimeter of the square in cm

- length of a side
- area of the square in cm^2
- diagonal length of the square

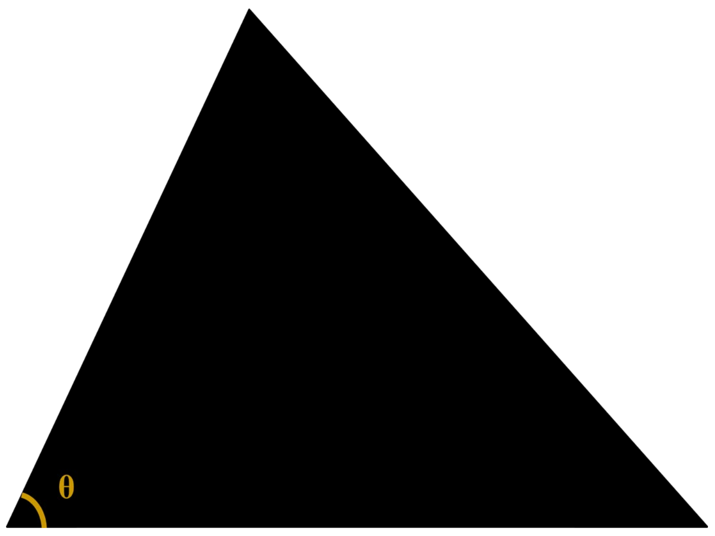
5.4 Exercise on Right Triangles



Find the following:

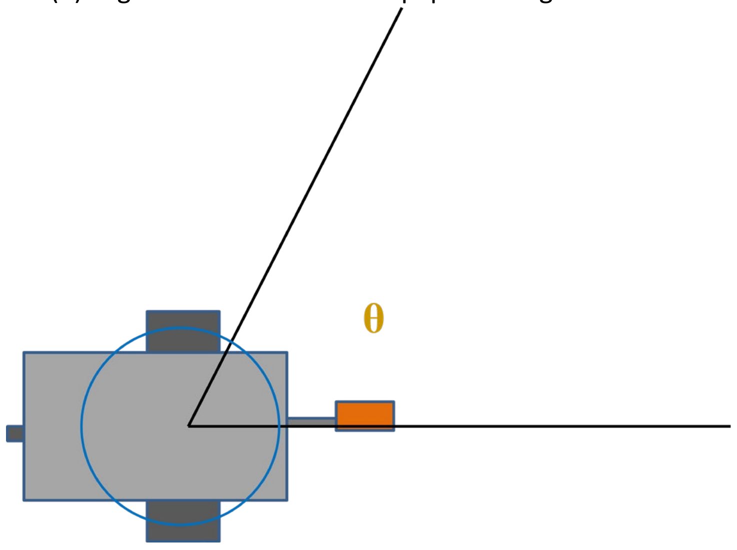
- Length of the hypotenuse in cm
- Area of the triangle in cm^2

5.5 To measure a degree of an angle

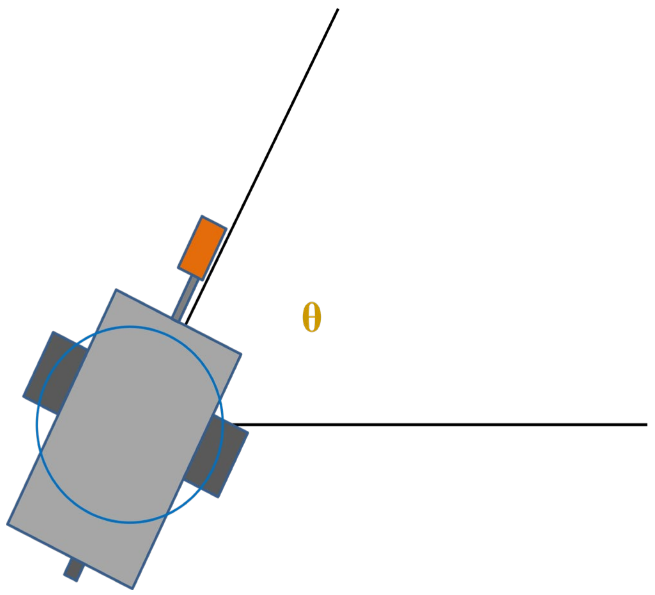


<u>Idea</u>

(1) Light sensor on the black paper to begin.

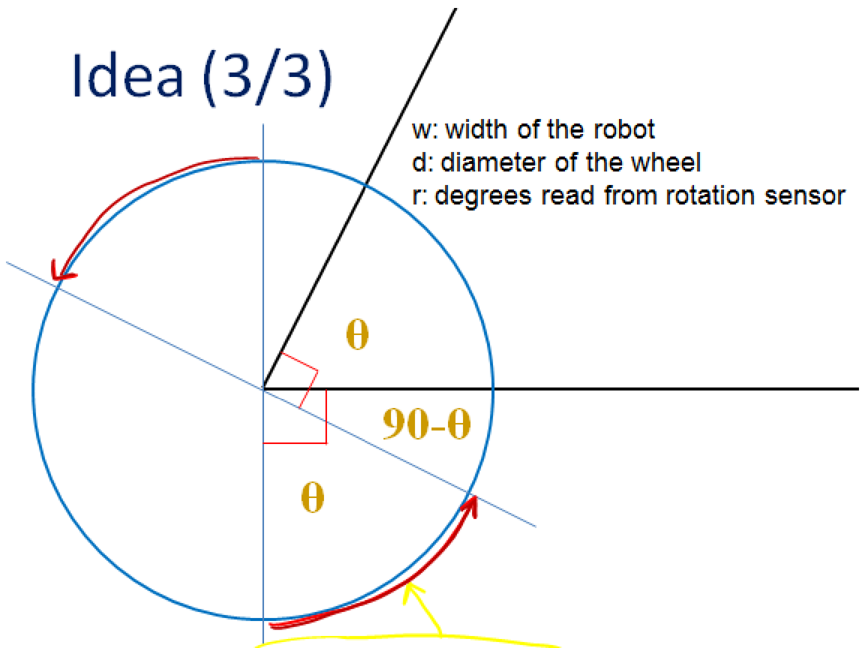


(2) Spin left until Light sensor detects white floor color. The robot must have spinned θ degrees.



Idea (3/3)

w: width of the robot
d: diameter of the wheel
r: degrees read from rotation sensor

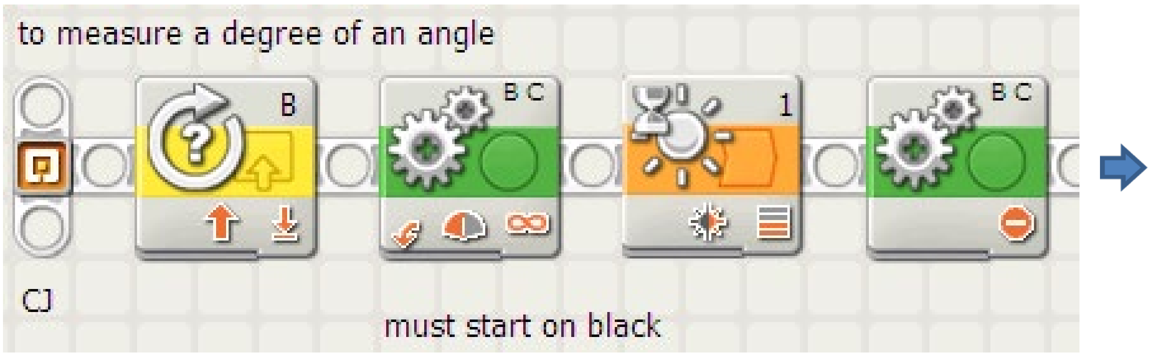


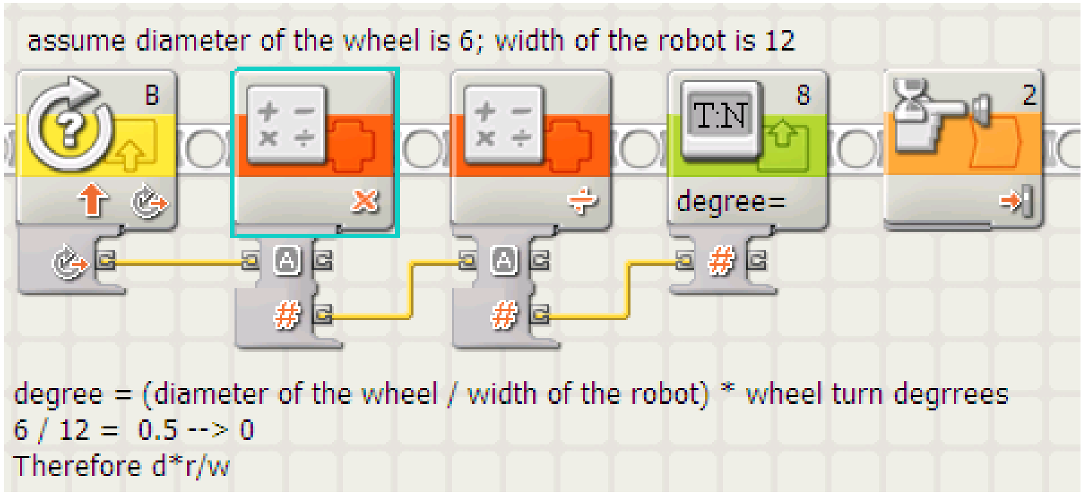
Wheel travel distance = w x pi x (θ / 360)

Circumference of the wheel to travel the distance = d x pi x (r / 360)

w·θ = d·r

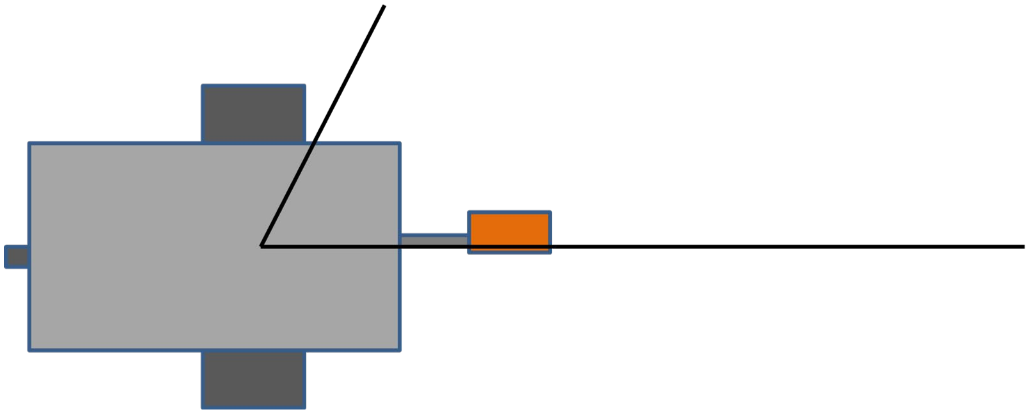
Therefore, θ = d·r / w



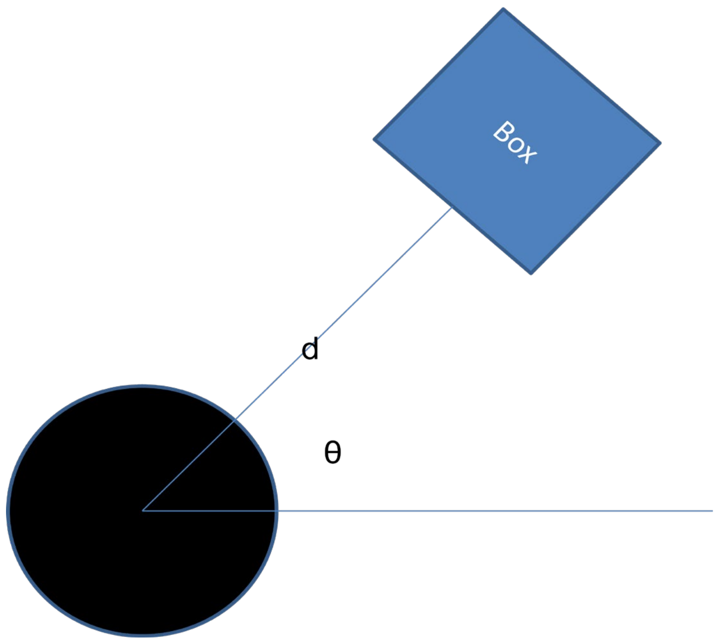

assume diameter of the wheel is 6; width of the robot is 12
B
8
2
degree=
degree = (diameter of the wheel / width of the robot) * wheel turn degrrees
6 / 12 = 0.5 --> 0
Therefore d*r/w

Ways to calculate the degree of an angle more precisely:

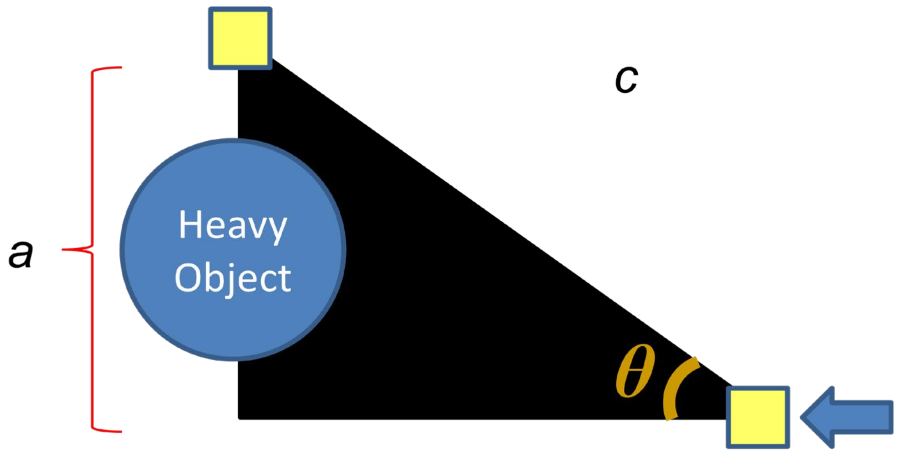
- Use two light sensors
- Double checking
- Truncation error? 5.5 => 55, 11.2 => 112
- Make sure to place the robot at the right position



Challenge: The robot must face east in the beginning. The degree θ and distance d will be given. Shoot a ball into the box.



5.6 Exercise: To find opposite side *a*



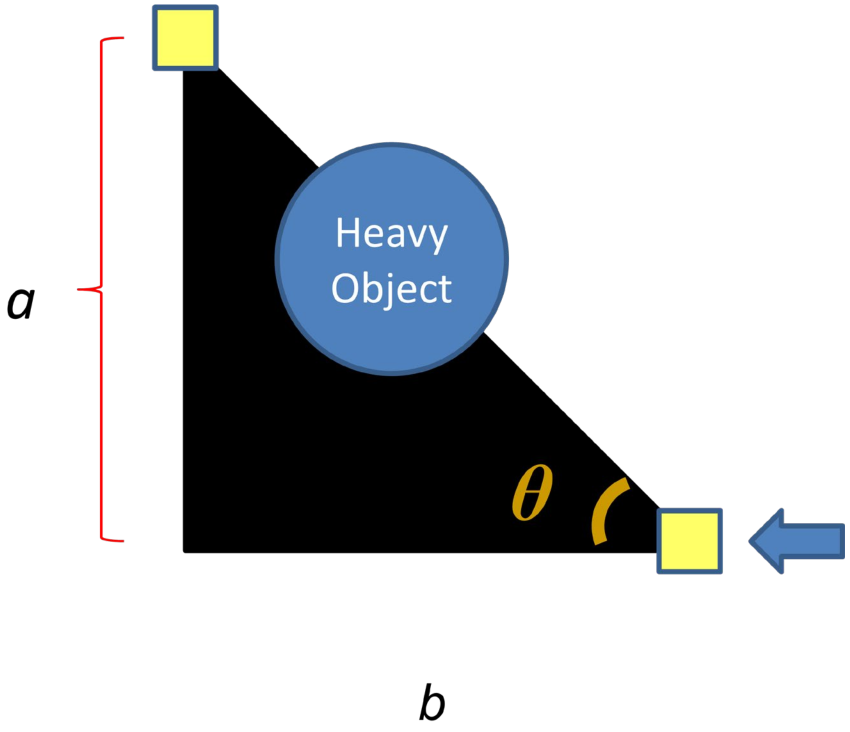
$$\sin\theta = \frac{a}{c}$$

$$a = \sin\theta \cdot c$$

1. Measure the degree of the angle
2. Measure length *c*
3. Calculate *a*

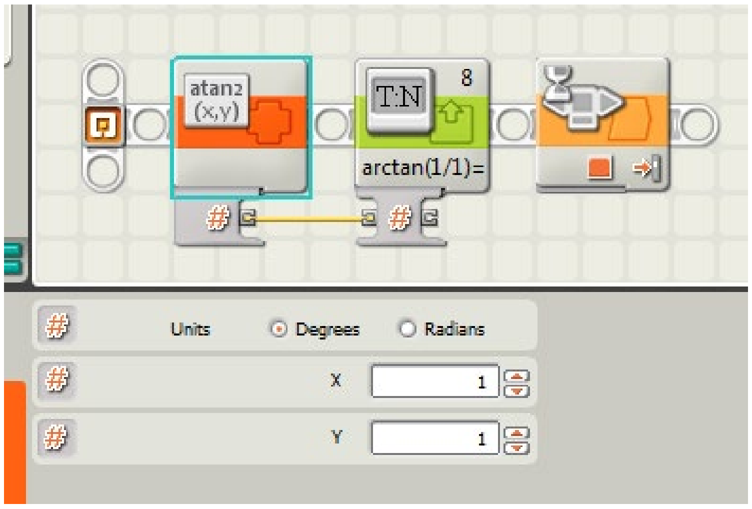
Where to get the sin, cos, and arctangent function blocks? Go to:
http://www.hitechnic.com/blog/uncategorized/new-hitechnic-blocks-for-sin-cos-and-atan2/

5.7 Exercise: To find theta



$$\tan\theta = \frac{a}{b}$$
$$\theta = \arctan\left(a/b\right)$$

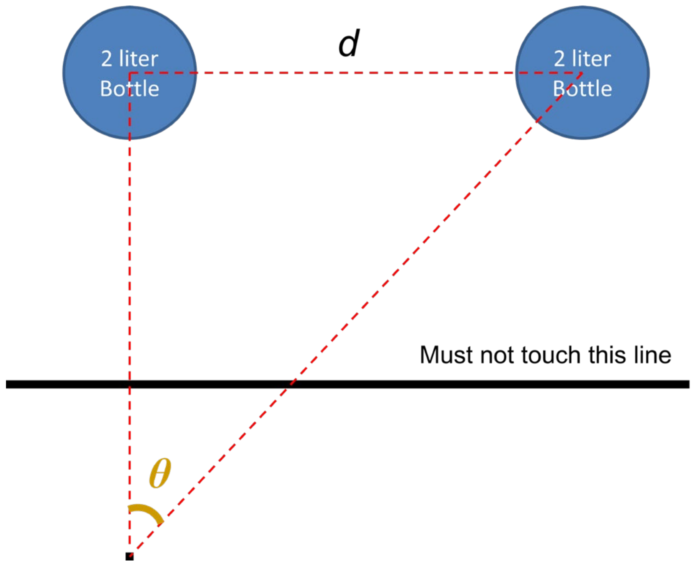
Here is a test program for the arctan custom block by Hitechnic.



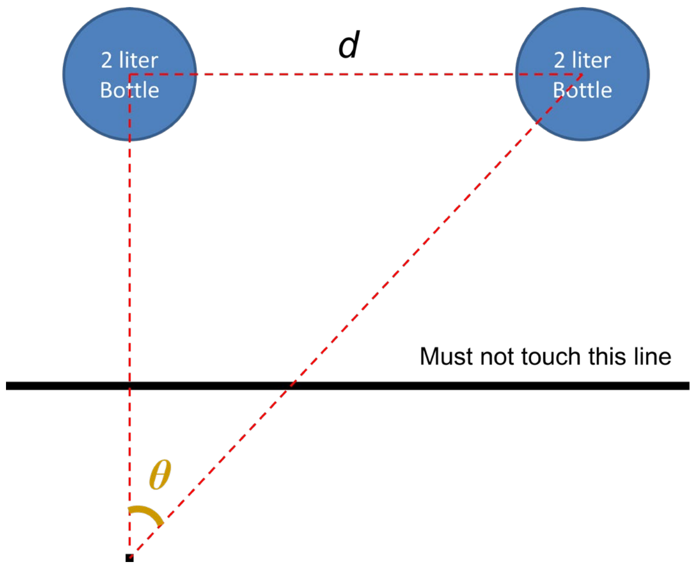
Units [✓ Degrees] Radians
X 1
Y 1

(arctan0.rbt)

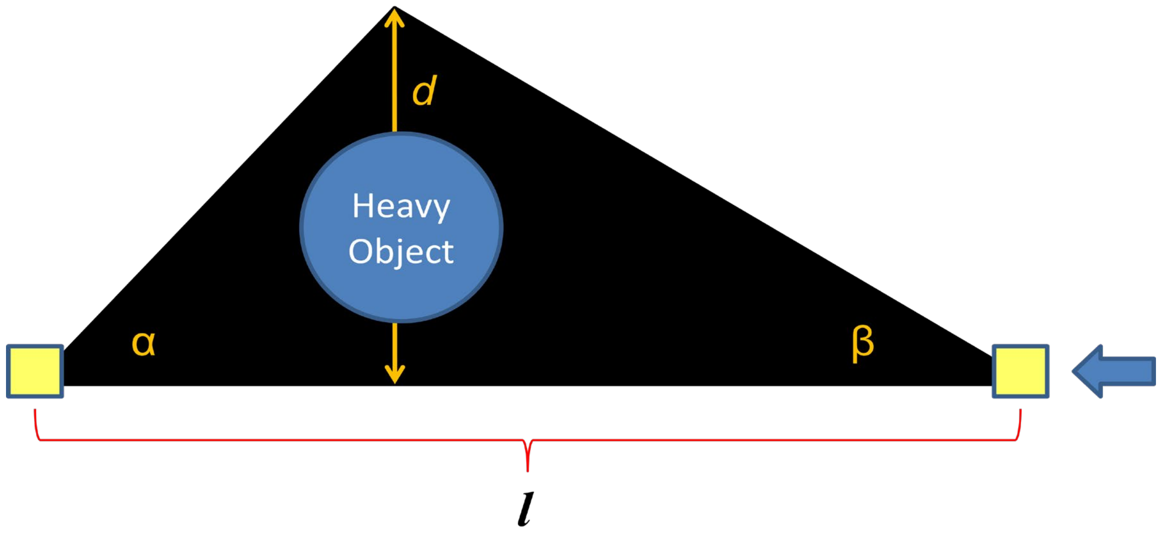
5.8 Challenge: Find the angle theta



5.9 Challenge: Find the distance *d*



5.10 Challenge: Find the length of d, triangulation



We assume that the robot cannot directly measure the distance d, because it cannot move the heavy object.

$$l = \frac{d}{\tan\alpha} + \frac{d}{\tan\beta}$$

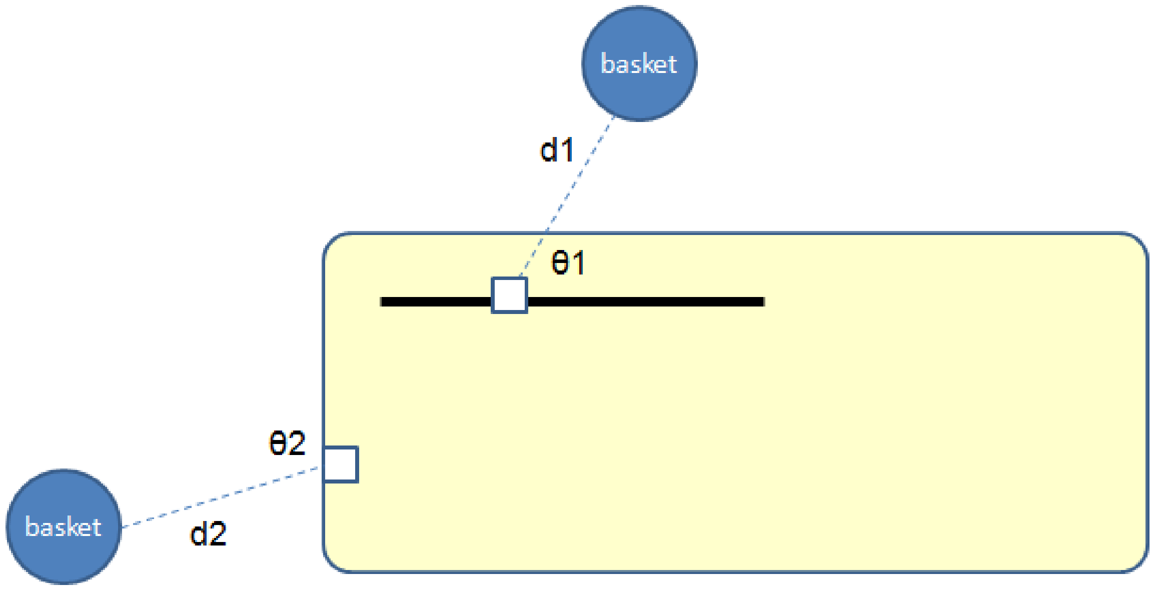
$$d = l \Big/ \left(\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}\right)$$

$$d = \frac{l \cdot \sin\alpha \cdot \sin\beta}{\sin(\alpha+\beta)}$$

5.11 Angle Challenge

Shoot balls into baskets placed at unknown location. Angle and distance information will be given on the day of the competition. Standard Lego NXT balls are too heavy to throw. Table tennis balls are recommended to use.

Learning objectives: logic, ratio, proportion, circles, angles, math operations, measuring, data analysis and trigonometry

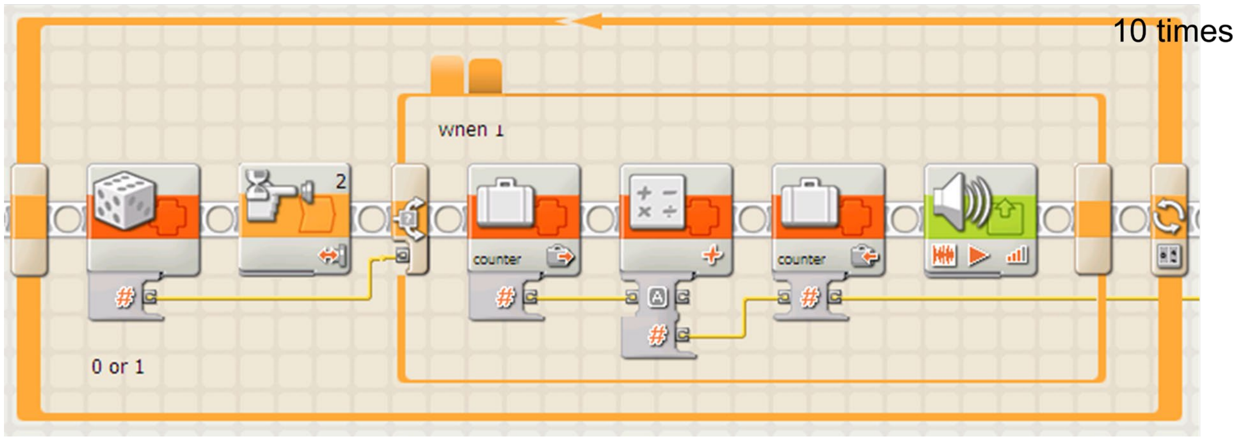


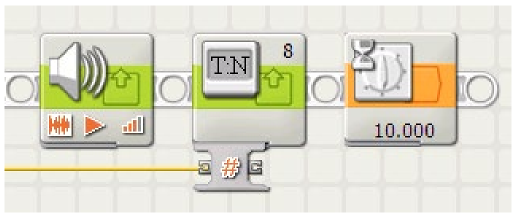
6

6 Probability Games

6.1 Even or Odd Game (Mission)

- A system is in either even or odd state
- A player wins, if a touch sensor is bumped when it is in even state
- A player loses, if a touch sensor is bumped when it is in odd state
- A player is given 10 chances to play
- After the game, total winning count is displayed
- What will be the expected winning count?



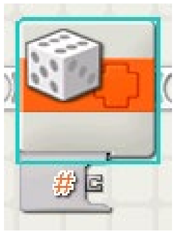


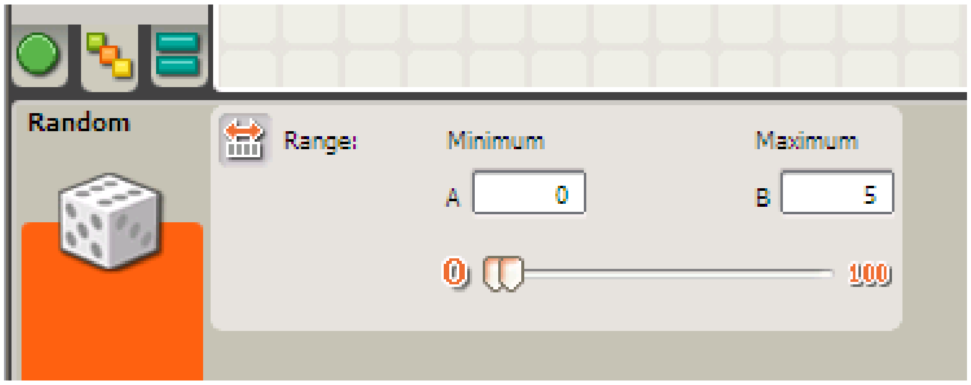
6.2 Dice Roll Game (Mission)

- A die has six sides, numbered one through six (internally 0~5)

- One touch bump simulates the roll
- A player wins, if the number is greater than 3
- Otherwise the player loses.
- A player is given 10 chances to play
- After the game, total winning count is displayed.
- What will be the expected winning count?

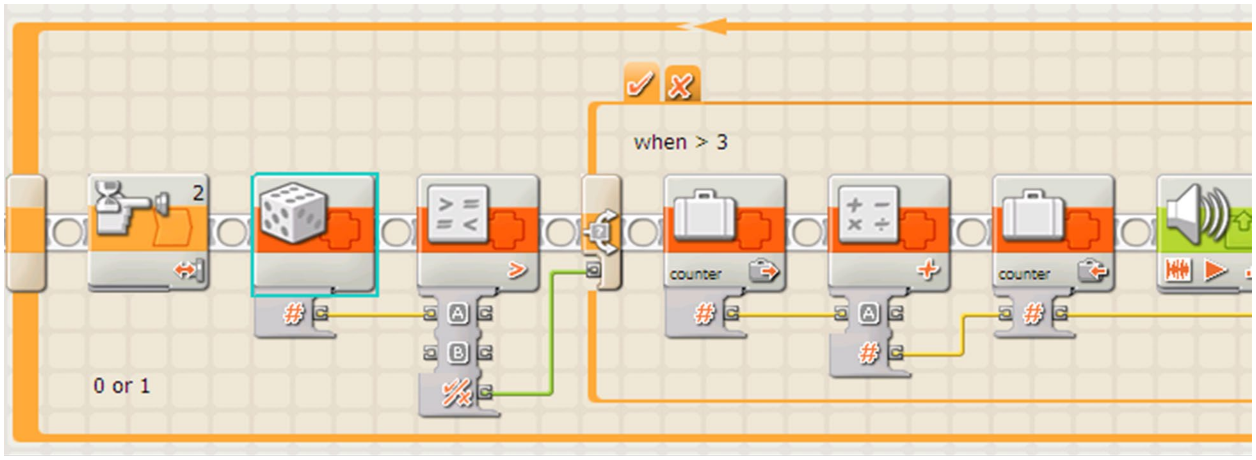
Hint:





Random

Range: Minimum A 0 Maximum B 5



> 2

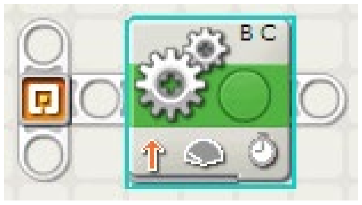
(DiceRoll.rbt)

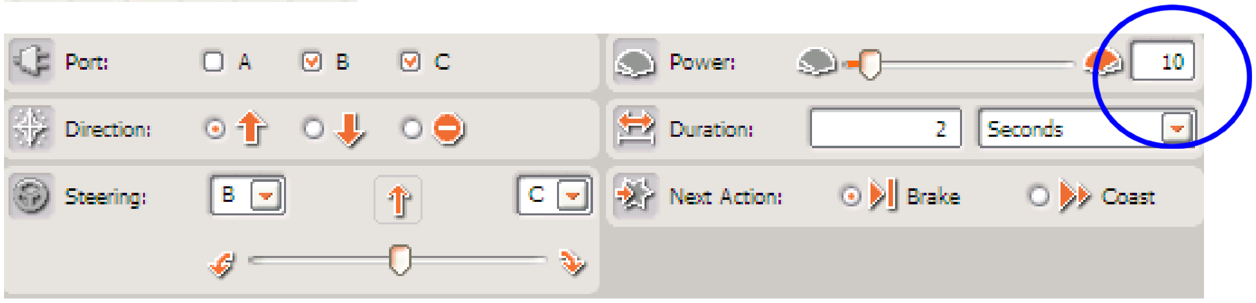
7

7 Data Analysis

7.1 Mission: Go straight for 2 seconds using different power levels

Let the robot go forward for 2 seconds with power level 10 % at 8 volt:




Port: A B [✓] C [✓]

Direction: [✓ forward] backward stop

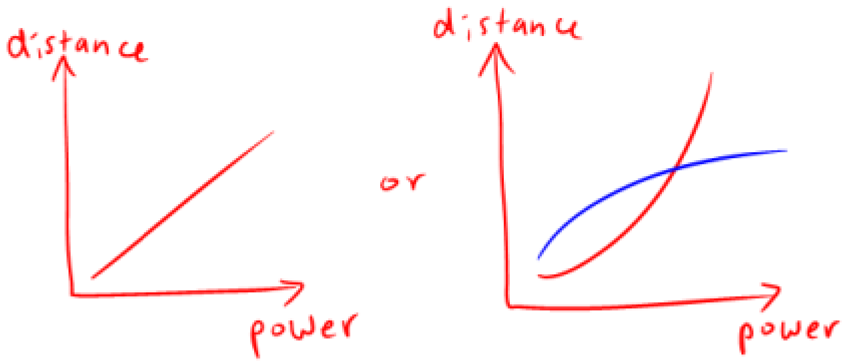
Steering: B C

Power: 10

Duration: 2 Seconds

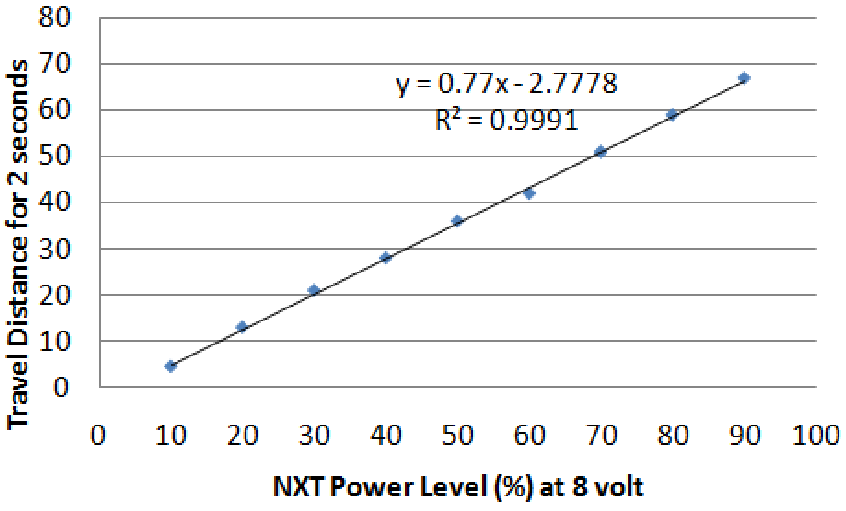
Next Action: [✓ Brake] Coast

Then measure the travel distance (cm) by increasing the power level. Will it be linear of non-liner??



Let's do the "Linear regression" which is to find best-fit line using Excel.

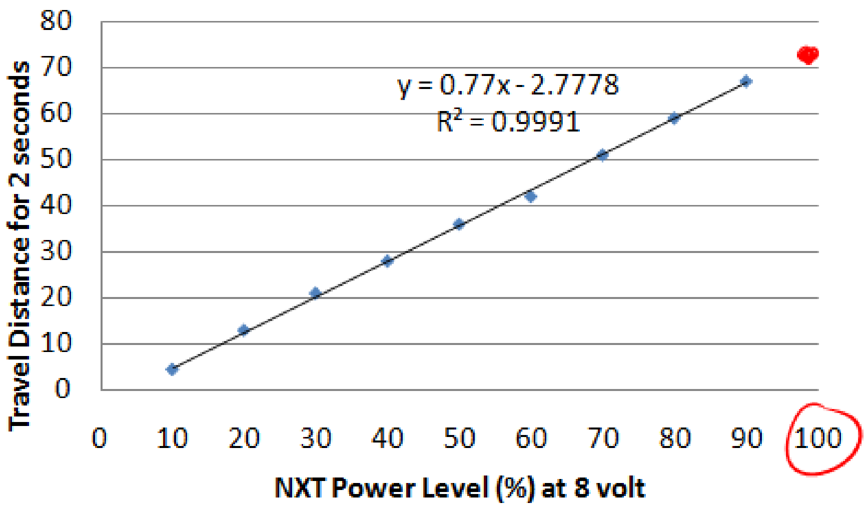
power	distance
10	4.5
20	13
30	21
40	28
50	36
60	42
70	51
80	59
90	67



What will be the travel distance when power level is 100%? The travel distance when power level is 100% can be estimated without running the robot. Use the equation found by the linear regression!

$0.77 \cdot 100 - 2.7778 = 74.2222$

power	distance
10	4.5
20	13
30	21
40	28
50	36
60	42
70	51
80	59
90	67
100	?

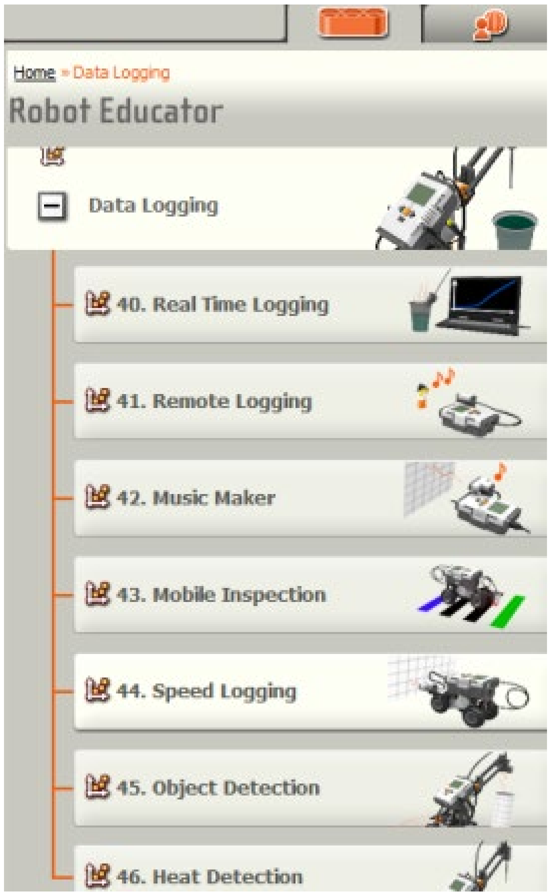


Actual running gave me 74cm. See an Excel 2007 tutorial for the linear regression on the web at:
http://www2.selu.edu/Academics/Faculty/jtemple/485/excel%202007%20tutorial.pdf

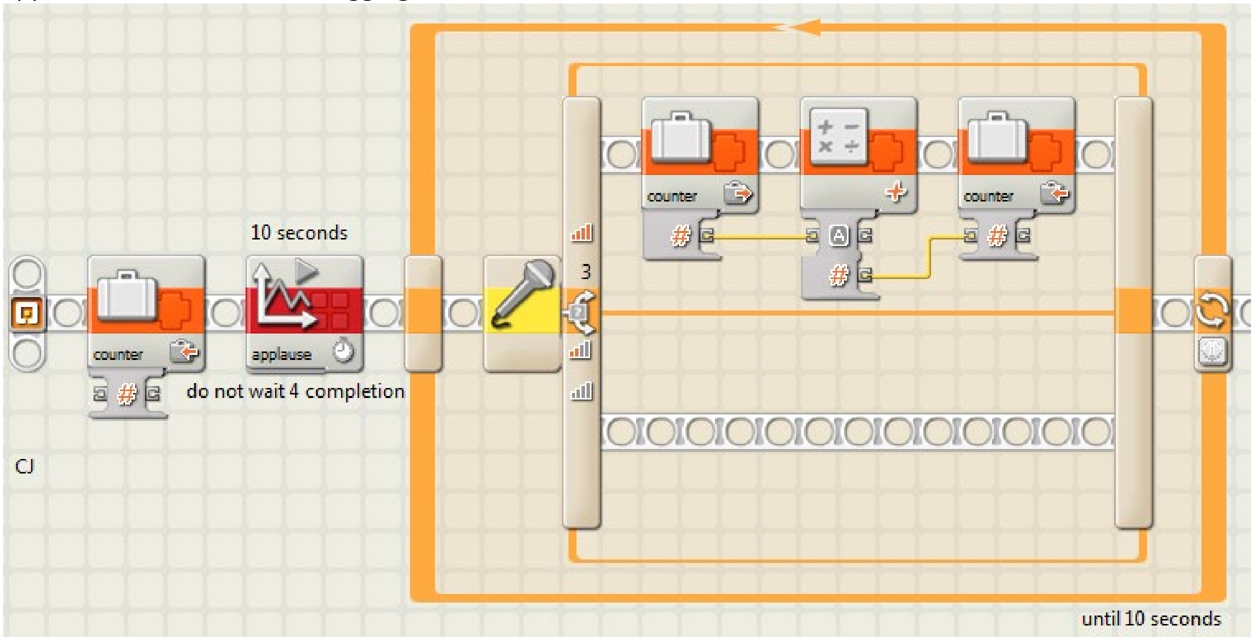
Exercises: Do the similar experiment and data analysis on the following:

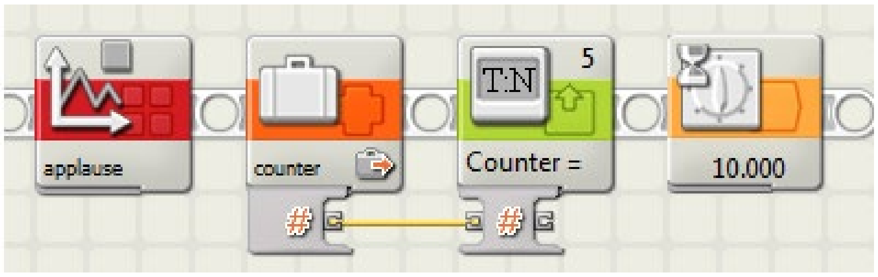
- Distance vs. Time (linear?)
- (Distance with fixed 2 rotations) vs. (power level) should be constant?

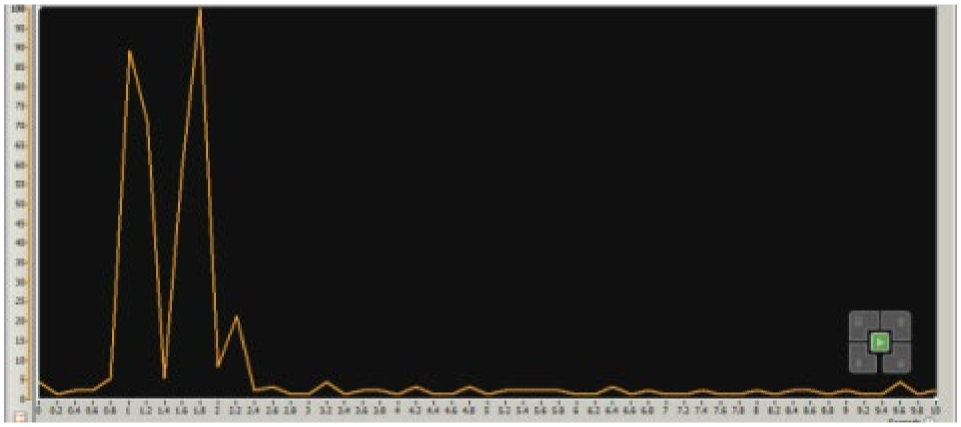
7.2 Mission: NXT-G Data Logging

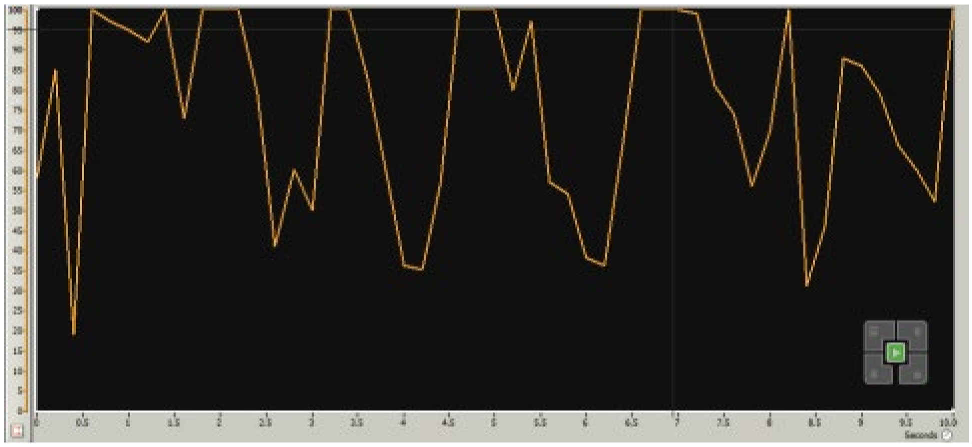


Applause Meter with Data Logging:






Counter = 147

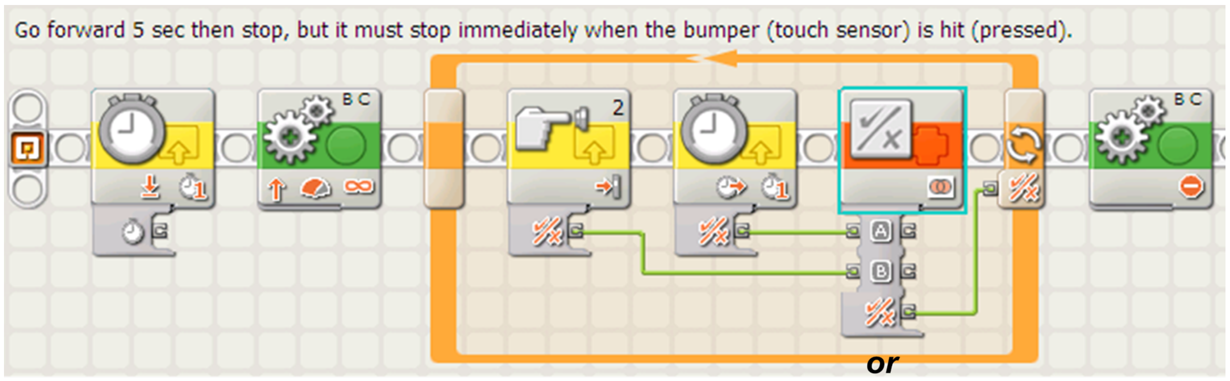

Counter = 2540

8

8 Logic / Boolean Logic

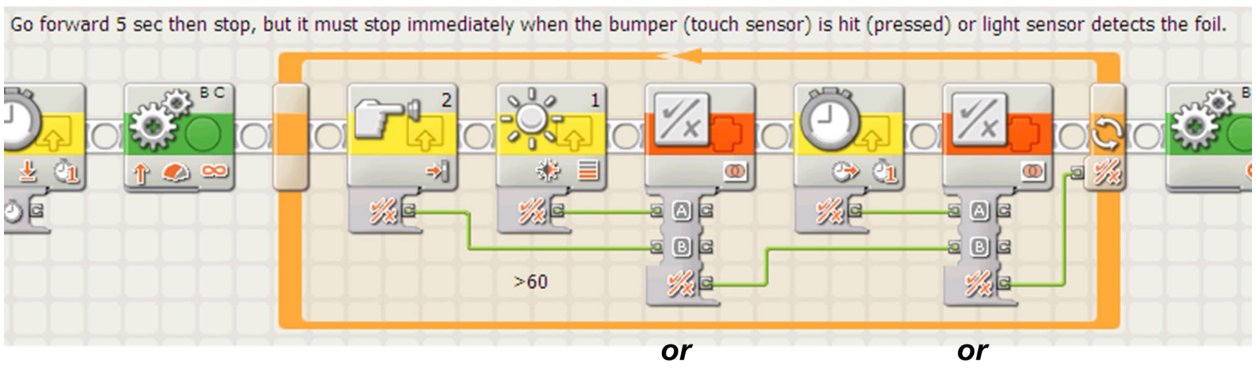
8.1 Watching two sensors

Watching timeout or touch (watching2sensors.rbt)


Go forward 5 sec then stop, but it must stop immediately when the bumper (touch sensor) is hit (pressed).

8.2 Watching three sensors

Extend watching2sensors.rbt. Stop when Timeout, touch, or (bright light > 60)


Go forward 5 sec then stop, but it must stop immediately when the bumper (touch sensor) is hit (pressed) or light sensor detects the foil.

8.3 Negation of Conjunction $\neg(p \wedge q) \Leftrightarrow \neg(p) \vee \neg(q)$

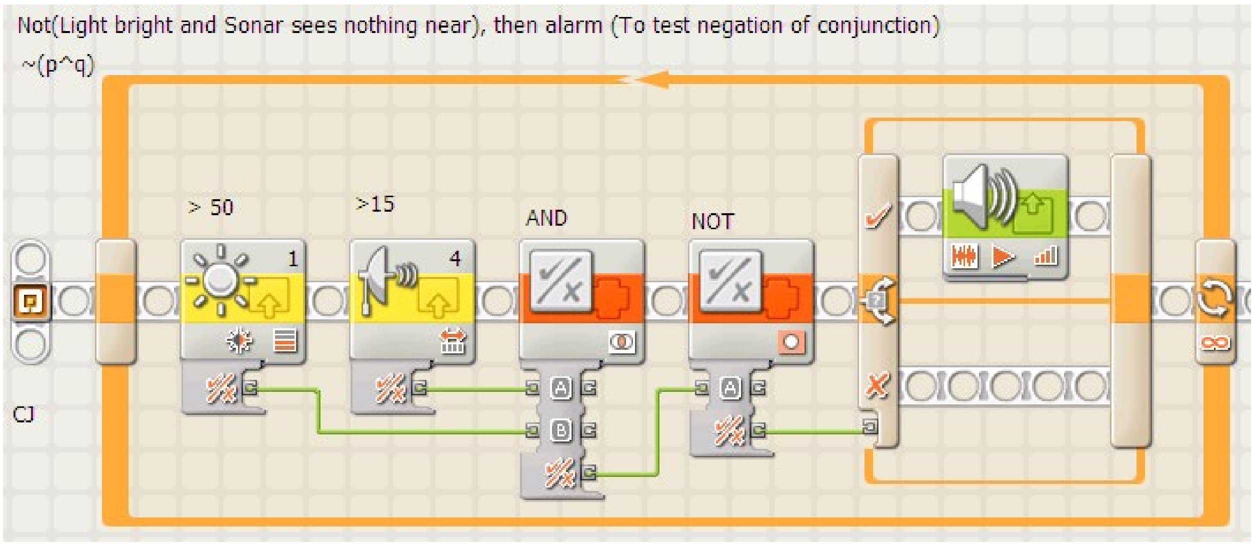
Since it is false that two things are both true, at least one of them must be false.

p	q	$p \wedge q$	$\neg(p \wedge q)$	$\neg p$	$\neg q$	$(\neg p) \vee (\neg q)$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

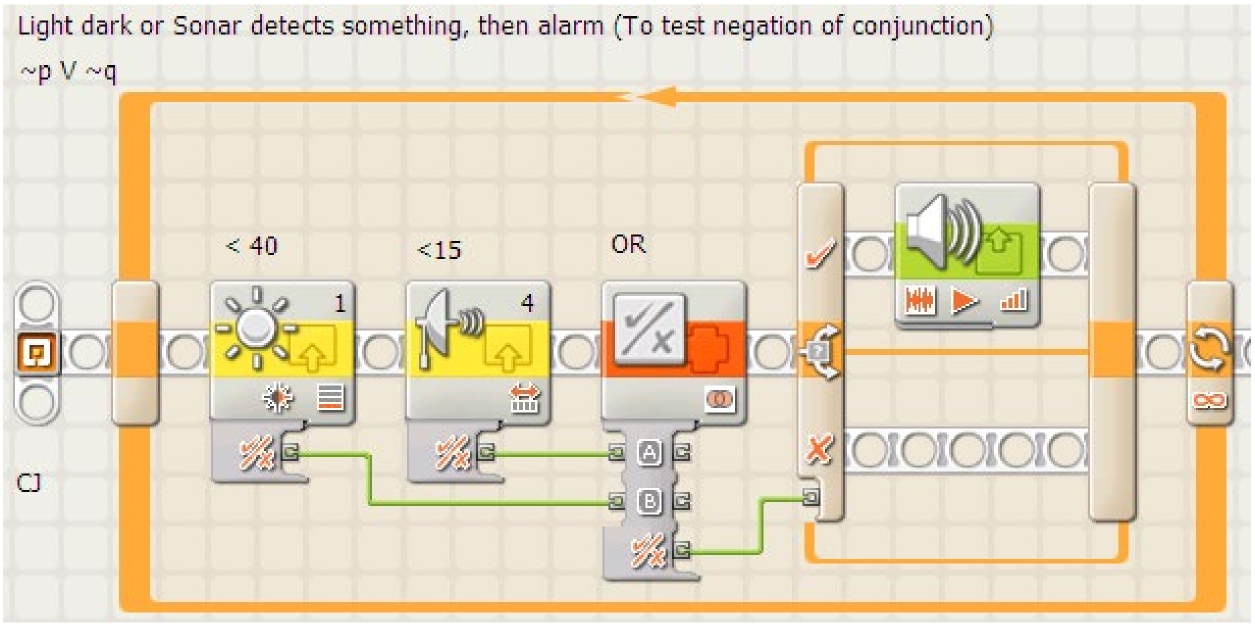
Make an alarm sound when the following *is false:*
light sensor sees bright *and* sonar sensor sees no object

$\Updownarrow$

Make an alarm sound when
light sensor sees dark *or* sonar sensor sees an object


Not(Light bright and Sonar sees nothing near), then alarm (To test negation of conjunction)
~(p^q)

(negConj1.rbt)



Light dark or Sonar detects something, then alarm (To test negation of conjunction)

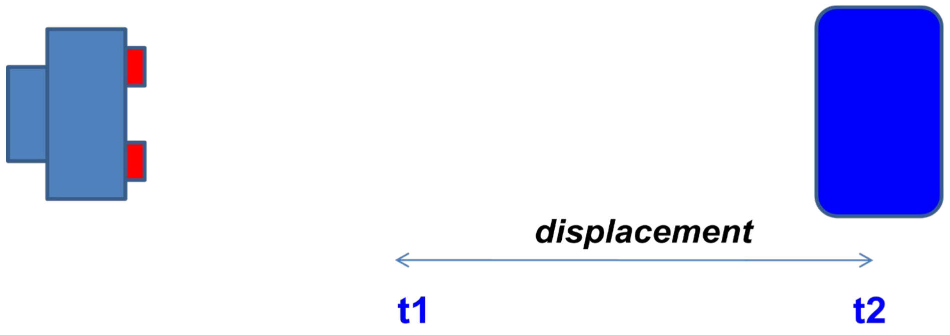
~p V ~q

(negConj2.rbt)

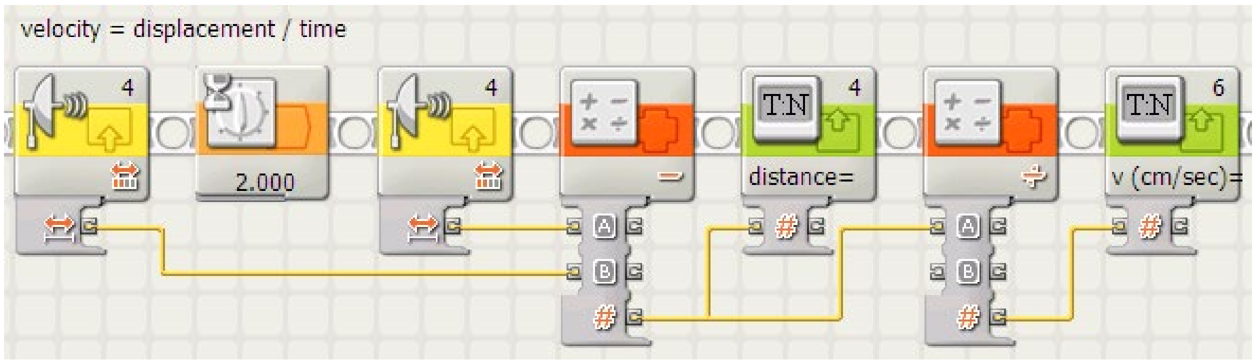
9

9 Science

9.1 Calculate velocity of an object using a sonar sensor



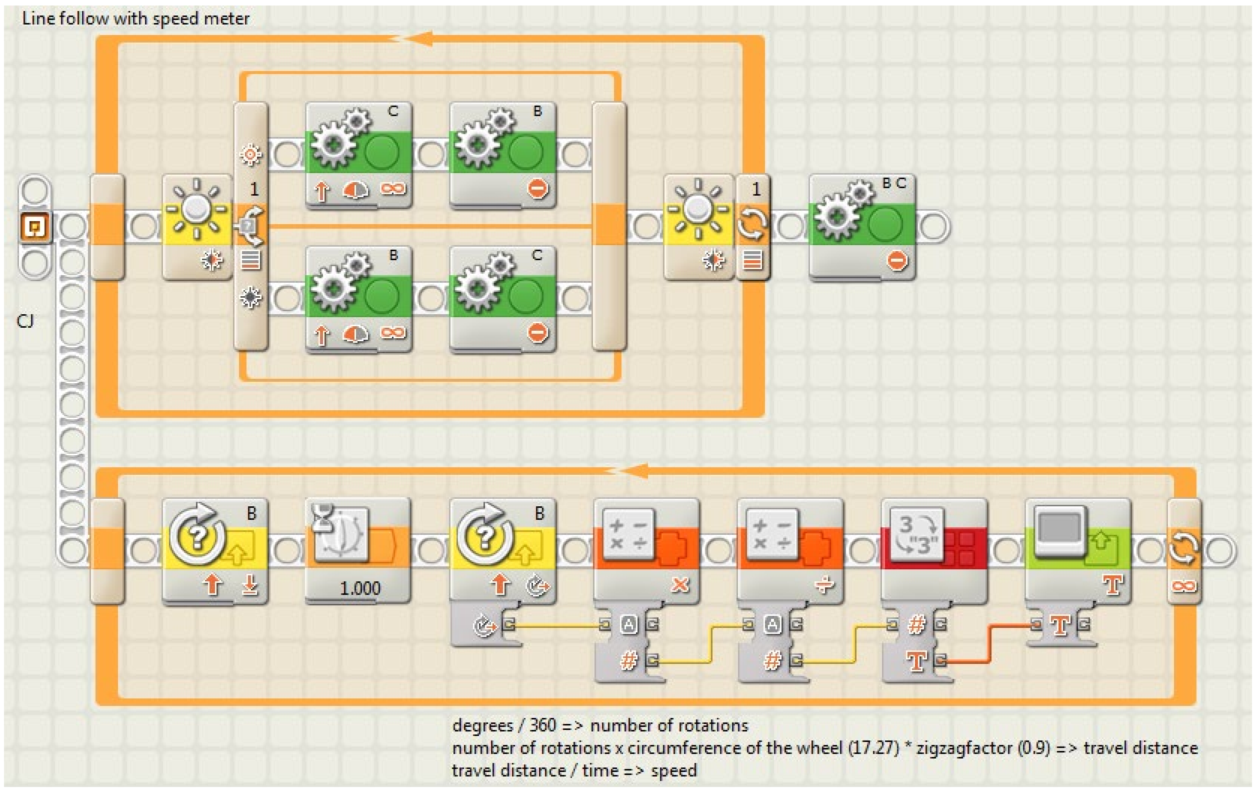
v = Displacement / Time (cm/second)



(Velocity.rbt)

9.2 Speed Meter

s = Distance / Time (cm/second)



degrees / 360 => number of rotations
number of rotations x circumference of the wheel (17.27) * zigzagfactor (0.9) => travel distance
travel distance / time => speed

9.3 Triathlon Challenges

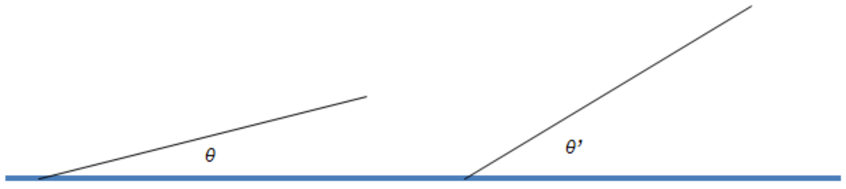
Challenge 1: Shot Throwing/Put
Throw a Lego ball – the shot - as far as possible. The robot can be stationary or moving. In case of moving, it must not pass a line.

Challenge 2: Push
Place a heavy object on a table. Find the object and push it off the table. Increase the weight of the object. Maximum weight of the robot must be specified before the competition.

- Wheeled division: Non sticky wheels are allowed to use
- Free division: any material can be used for this division

Challenge 3: Hill Climbing

- Wheeled division: construct a robot using wheels and program it to follow a line on a ramp. The ramp will be steeper and steeper. Using sticky wheels are not allowed.
- Free division: any material can be used for this division



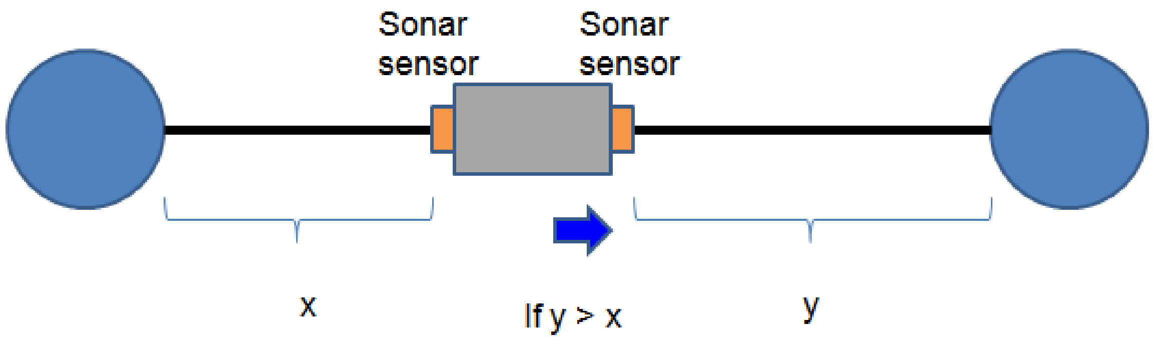
$\theta < \theta'$

10

10 More Challenges and Projects

10.1 Centering

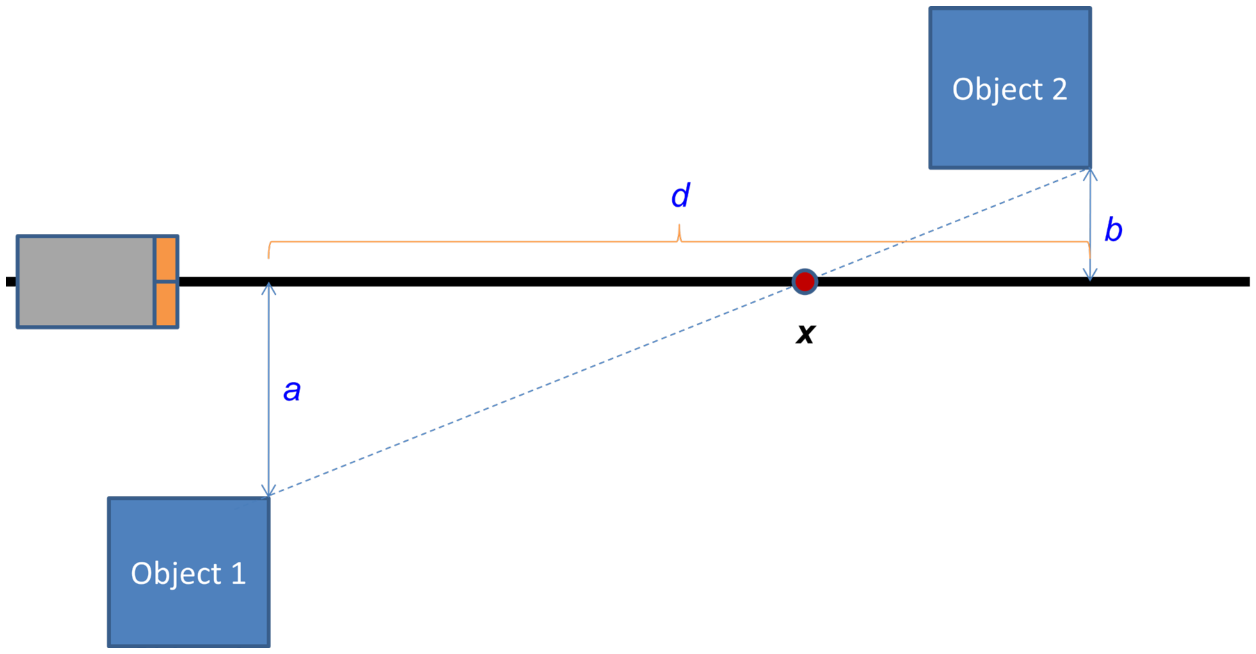
Locate the robot in the middle of two objects.



Hint: If y > x then follow right until x and y values are (almost) same.

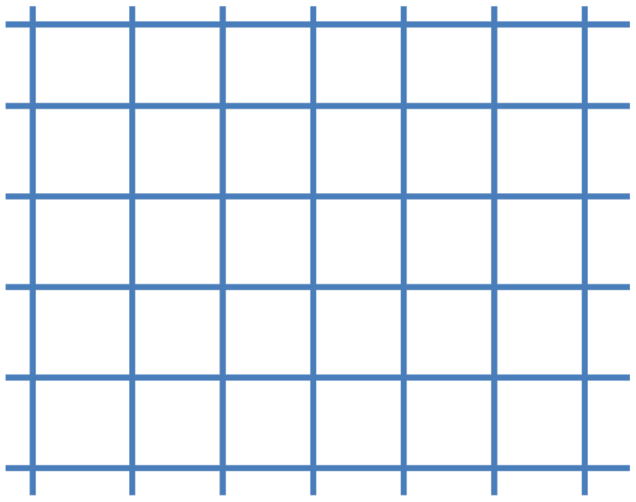
10.2 Locate at z

Hint: measure a, d, and b while following the line. Solve the system of equations to find x. Go backward (d – x) distance.

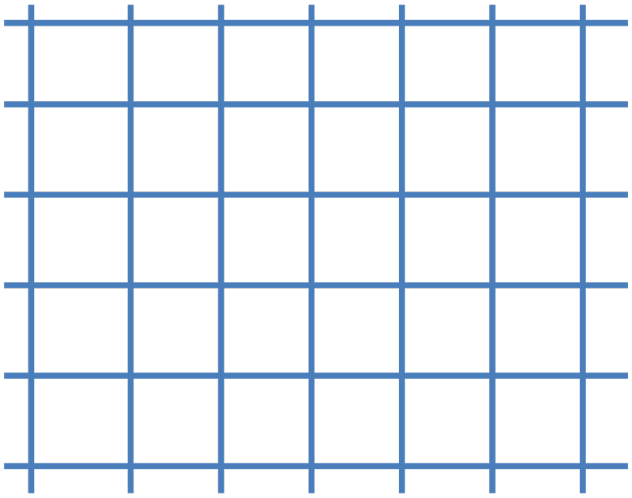


10.3 Go to (x, y) point

A robot is placed at (0, 0) point facing north on the 1st quadrant of the Cartesian plane. Let it go to (2, 3) point and stop. You need at least two light sensors for this challenge.



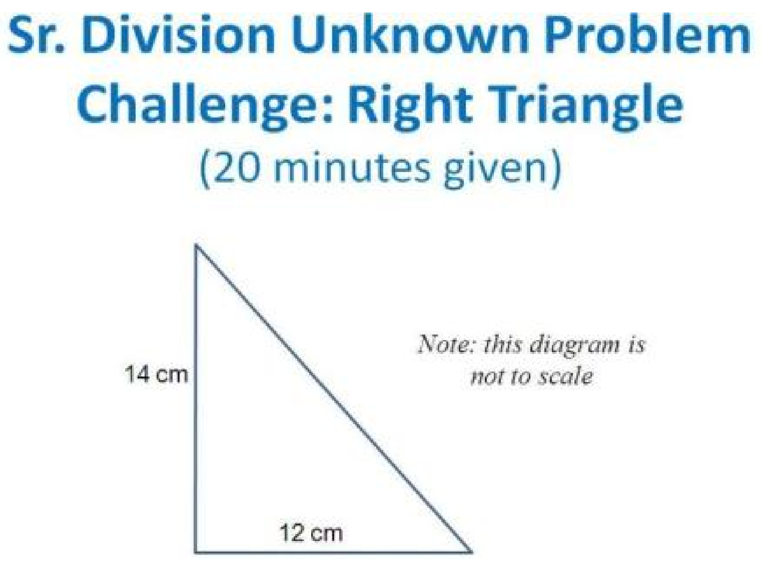
10.4 Draw y=*ax*+*b* *approximately* on the 1st quadrant of the Cartesian plane



Find y intercept by hand calculation. Go north to the y intercept. Calculate the angle to turn for the slope of the line. Go straight for some distance.

10.5 Drawing a right triangle – Robofest 2009 Game

The 2009 RoboZone Unknown Problem Challenge required the robot to draw a polygon. The number of sides and side lengths were revealed at the competition.



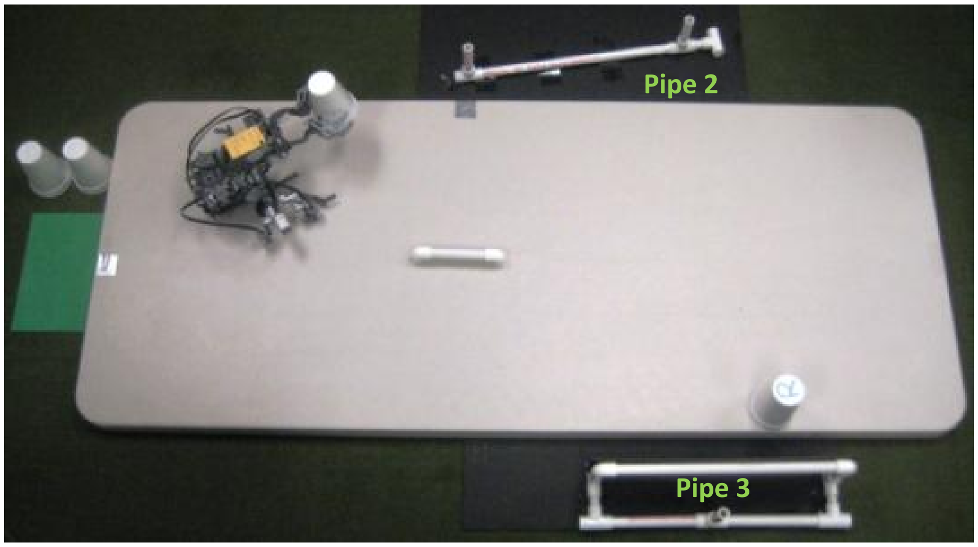
Sr. Division Unknown Problem Challenge: Right Triangle

(20 minutes given)

Required the use of Trigonometry:

$$\arctan\theta = b/a$$

10.6BTOS (Block the Oil Spill) – Robofest 2011 Game

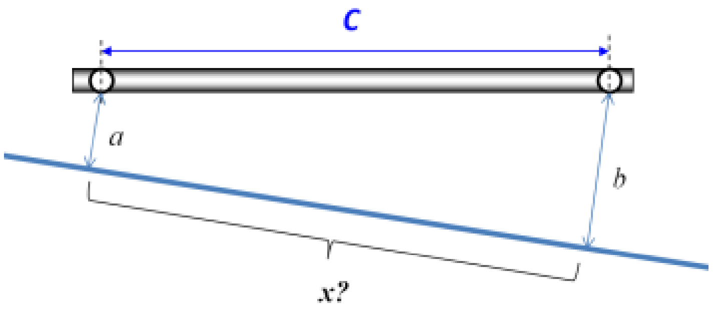


In order to stop the oil spill in deepwater, an autonomous robot is being sent to cap the leaking pipes.

Learning objectives: logic, ratio, proportion, circles, angles, math operations, measuring, and Pythagoras' theorem

Cap the pipe 2

Constants a, b, c are given. Find x by solving a system of two equations.

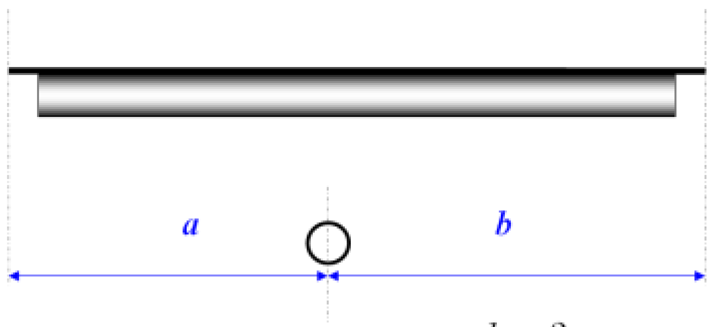


$$\begin{cases} a+b = d \\ a/b = c \end{cases}$$

Cap the pipe 3

Constants a, b, c are given. Calculate b and a+b, by solving a system of two equations.

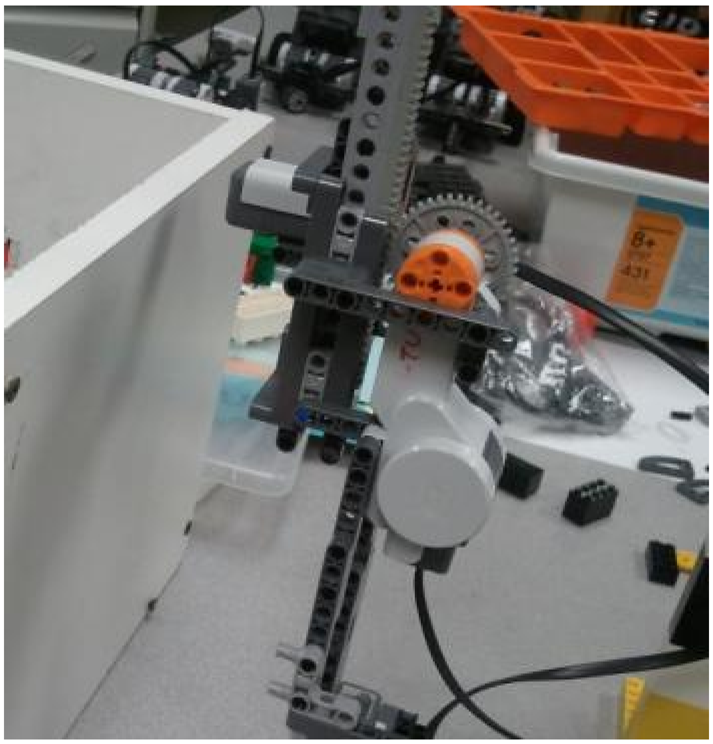
$$a/b = c$$

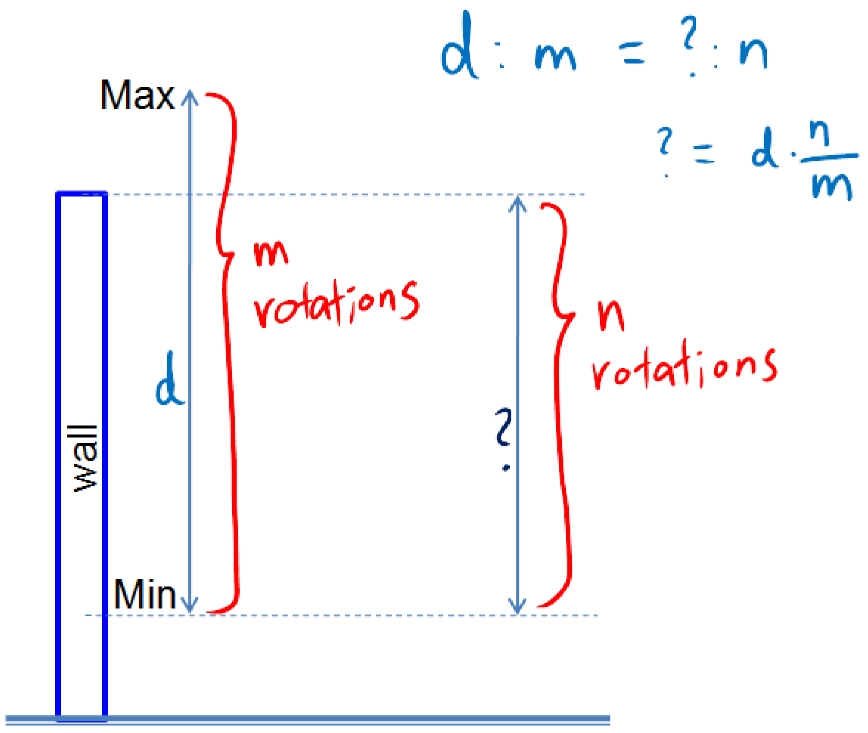


$$b = ?$$
$$a + b = ?$$

$$\begin{cases} x^2 + y^2 = c^2 \\ y = b - a \end{cases}$$

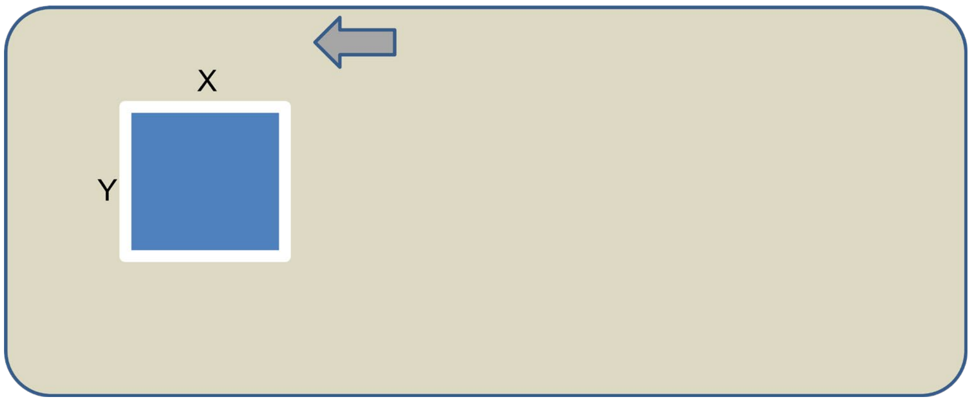
10.7 Measure the height of a wall





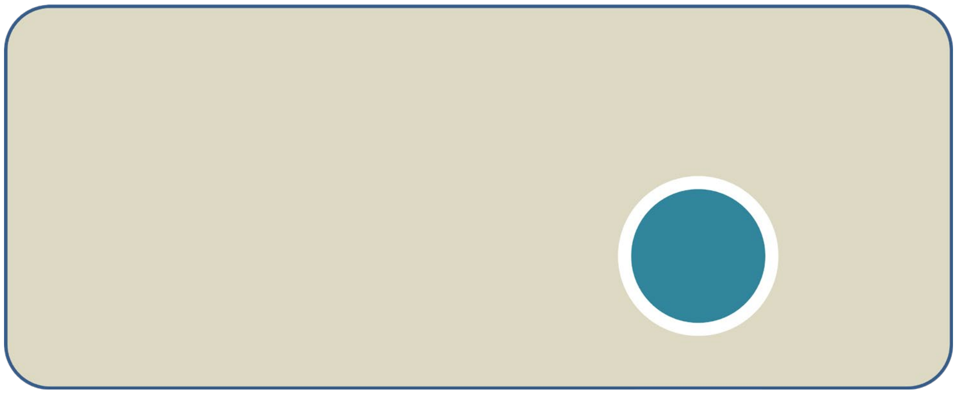
$$? = \frac{n}{m}(\text{max} - \text{min}) + \text{min}$$

10.8 Measure the volume of a box



10.9Measure the volume of a cylinder

The location of a cylinder is decided after the robot starts. Find it and measure the volume.



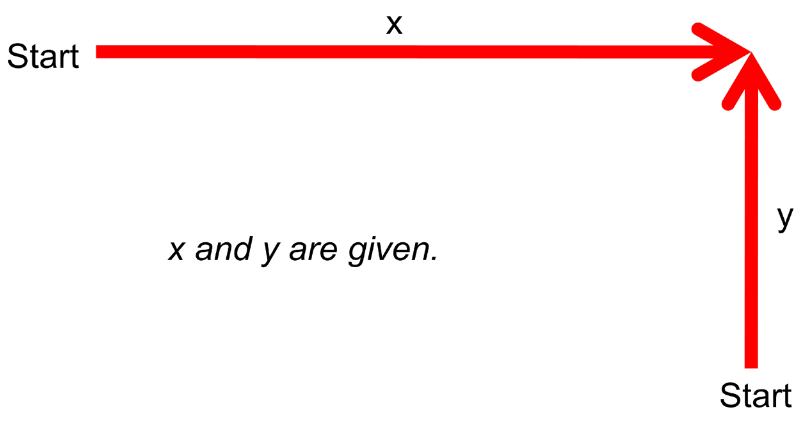
10.10 Follow right side of a black line. Stop at the end of the line

Idea 1: If turn left counter becomes greater than 2, stop

Idea 2: If it does not find black line for a specified amount of time, stop

10.11 Make a Collision

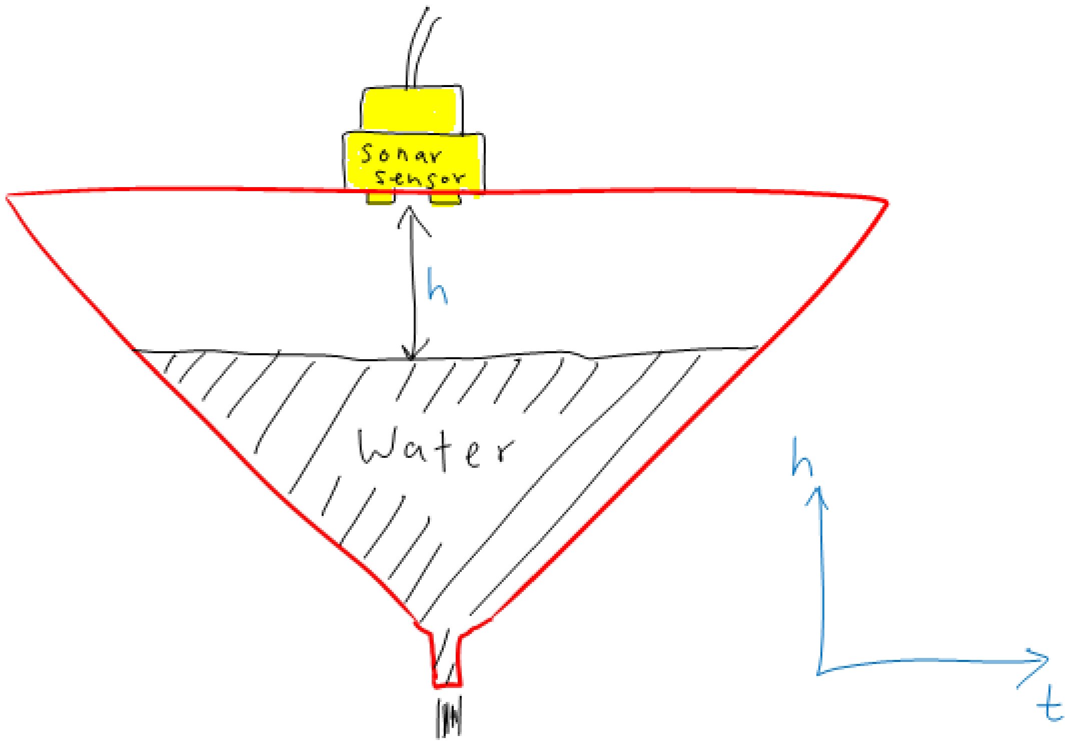
- Build two robots to go as straight as possible. X and Y distances are given. Calculate proper power for each robot to collide in the following course.
- To build a robot to go straight is not easy since two motors are different. Do the line following version of the above challenge.



10.12 Funnel Math

- Collect data to draw the graph.
- Develop math model for the funnel.

– Compare the two.



References and Credits

Chung, C. and Engalan, J. (2000). Lego Robot Programming and Construction Guide – Learning with Lego Robot Games and Introduction to RCX Code Programming, LTU

Chung, C. (2006). Robotics Programming Workbook for Robofest, LTU

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books, Inc.

Weiss, S. (2004, December). Teachers' Knowledge and Skills are Key to Improving Student Achievement in Science, Math. *The Progress of Education Reform*, 6 (1), Education Commission of the States.

Wagner, S.P. (1998). Robotics and children: Science achievement and problem solving. *Journal of Computing in Childhood Education*, 9(2), 149-192.

Wang, E (2004). Engineering with Lego Bricks and RoboLab