

EV3 Programming

Overview for FLL Coaches

A very big high five to Tony Ayad

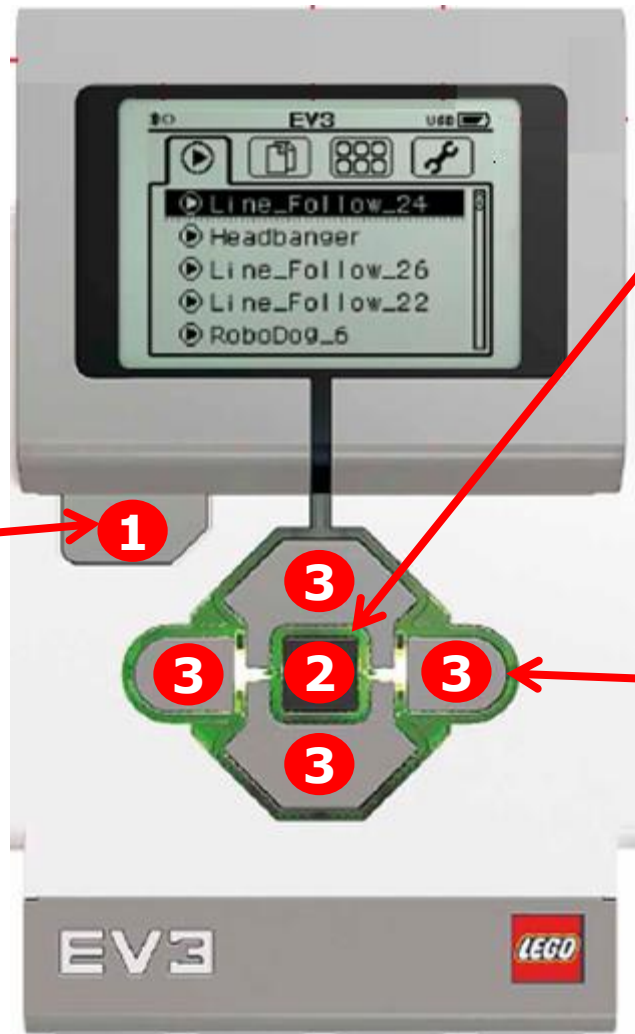
Basic programming of the Mindstorm EV3 Robot

- People Introductions
 - Deborah Kerr & Faridodin Lajvardi - *FIRST*® Senior Mentor
 - Jason Yount - Technical Training Manager - LEGO® Education North America
 - Glenn Swinson , *FIRST*® FLL Logistics Manager
- Main Topic:
 - Jason will walk through a slide set for programming the EV3 robot
 - This call is meant to present Basic programming, not advanced methods, and thus geared toward newcomers to the Mindstorm and FLL experience
- Technical Support – where to get help
- Q&A

Outline

- Purpose:
 - This workshop is intended for FLL coaches who are interested in learning about Mindstorms EV3 programming language.
- Programming
 - EV3 Controller (aka: the “brick”)
 - User Interface
 - Building Blocks
 - Controlling the robot with **MOVE** Blocks
- What is New: EV3 vs. NXT
- Turns - there is more than one way to turn
 - Geometry and Math for the Robot
 - Gyro Sensor
- Advanced Programming
 - Light Sensor
 - Program Control (**WAIT**, **LOOP**, **SWITCH** Blocks)
 - Math (**VARIABLES**, **MATH** and **COMPARE** Blocks)
 - **My Block**
 - Resources

EV3 Brick



1 = Back, this button is used to reverse actions, to stop a running program, and to shut down the EV3.

2 = Center, press the center button to select and accept options, or run a program.

3 = left, right, Up, Down
These four buttons are used to navigate through the various menus.

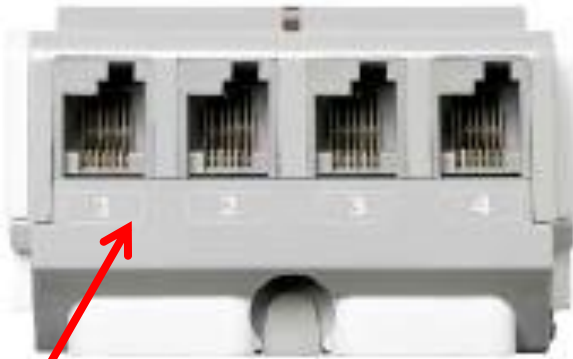
Programmable EV3 Brick

- Four outputs (motors)
- Four inputs (sensors)
- USB, Bluetooth, or Wi-Fi connection
- Improved LCD screen
- 16 MB flash memory
- 64 MB RAM
- SD Card Port: 32 GB
- Multiple onboard utilities
- 1,000 samples per second
- EV3 Brick Button lights
- Sound

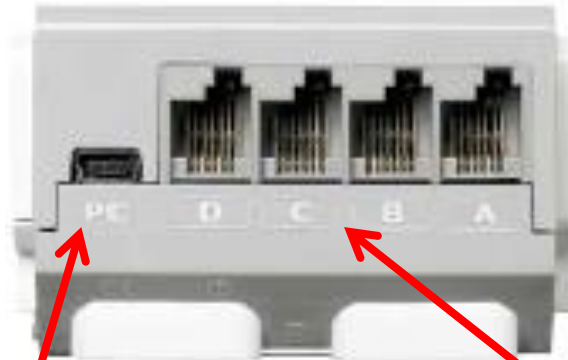


Ports, Sensors and Motors

1, 2, 3, 4 = Input ports used for sensors.



A, B, C, D = Output ports used for motors.



The PC USB port is used to connect to The PC so you can download the Programs into EV3 Controller

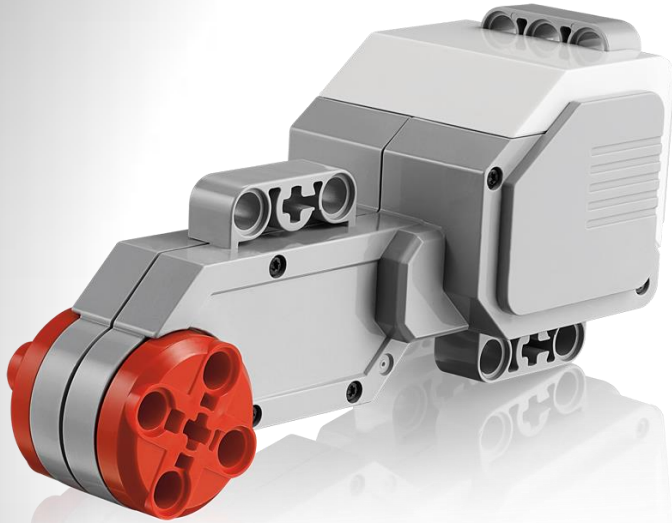


Large Motor



Medium Motor

EV3 Motors



- Two types of motors
- Redesigned to allow easy construction
- The Large Motor is a strong and powerful full motor.
- The Medium Motor is a less powerful motor but runs at a higher revolution rate.
- Both motors have tacho feedback enabling 1 degree resolution.
- Both motors are Auto ID-supported.
- The Medium Motor is smaller and lighter to allow more construction options.

EV3 Ultrasonic Sensor



- Detects distance
- Accurate to 1 cm or 0.3 inches
- Can listen for other ultrasonic sensors
- Improved design for easier build solutions
- Eyes light up to identify which mode the sensor is operating in
- Auto ID

EV3 Color Sensor



- Detects eight different colors
- Detects ambient light, from dark to sunlight
- Detects reflective red light
- Built-in cancelling of backlight makes sensor more reliable
- Improved design for easier build solutions
- Auto ID

Touch Sensor



- Detects pressed
- Detects released
- Detects bumped
- Improved design for easier build solutions
- Auto ID

Gyro Sensor



- Angle mode
- Gyro Sensor mode
- Angle and Gyro Sensor modes
- Can reset accumulated angle value
- Improved design for easier build solutions
- Auto ID

EV3 Navigation

Open a previously saved project

Lobby Button

Open New Project

Open New Project or previously saved ones





- Model Core Set
- Quick Start
- File
- Robot Educator
- Design Engineer...

Basics

- Beyond Basics
- Hardware
- Data Logging
- Tools
- Building Instructions

Configuring Blocks

Straight Move

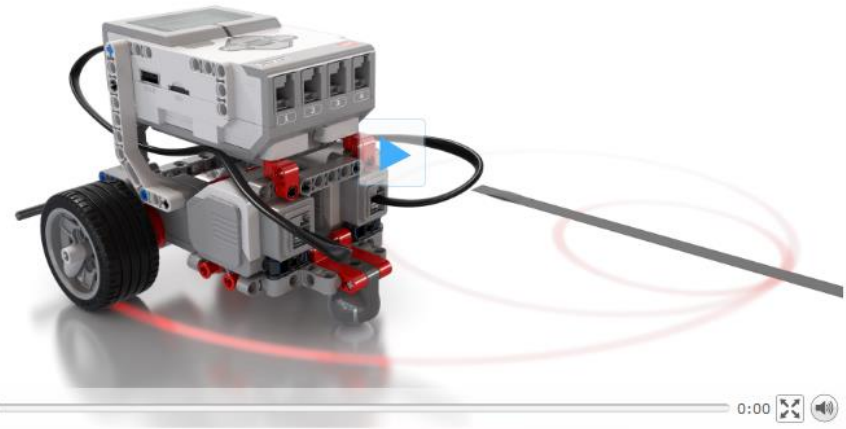
Curved Move

Tank Move 4/9

Move Object

Stop at Line

Stop at Angle



0:00

Tank Move

Use the Move Tank block to steer the Driving Base.

Open



Projects and Programs

The image shows a screenshot of the LEGO MINDSTORMS LabVIEW software interface. The top window, titled "LEGO M", displays a menu bar (File, Edit, Tools, Help) and a taskbar with several open windows: "First Project.ev3* x", "Program x", "Program2 x", "Program3 x", and "MBMove x". A red arrow points from a box labeled "Opened Project" to the "First Project.ev3* x" window. Another red arrow points from a box labeled "Project Properties" to the wrench icon in the taskbar. A third red arrow points from a box labeled "Currently Opened Programs belonging to opened project" to the "Program x", "Program2 x", and "Program3 x" windows. A fourth red arrow points from a box labeled "Click to create a new program within the current project" to the plus sign icon in the taskbar.

Opened Project

Project Properties

Currently Opened Programs belonging to opened project

Click to create a new program within the current project

The bottom window, titled "First Project.ev3* x", shows the project details. It has a "Project Title" field set to "Project". Below this are two tabs: "1 PROJECT PICTURE" and "2 PROJECT DESCRIPTION". The "PROJECT PICTURE" tab shows a photo of a LEGO Mindstorms robot on a table. The "PROJECT DESCRIPTION" tab contains the text "This is my first project". Below the tabs is a "Daisy-Chain Mode" checkbox. At the bottom, there is a "Programs" tab with a table listing the programs in the project.

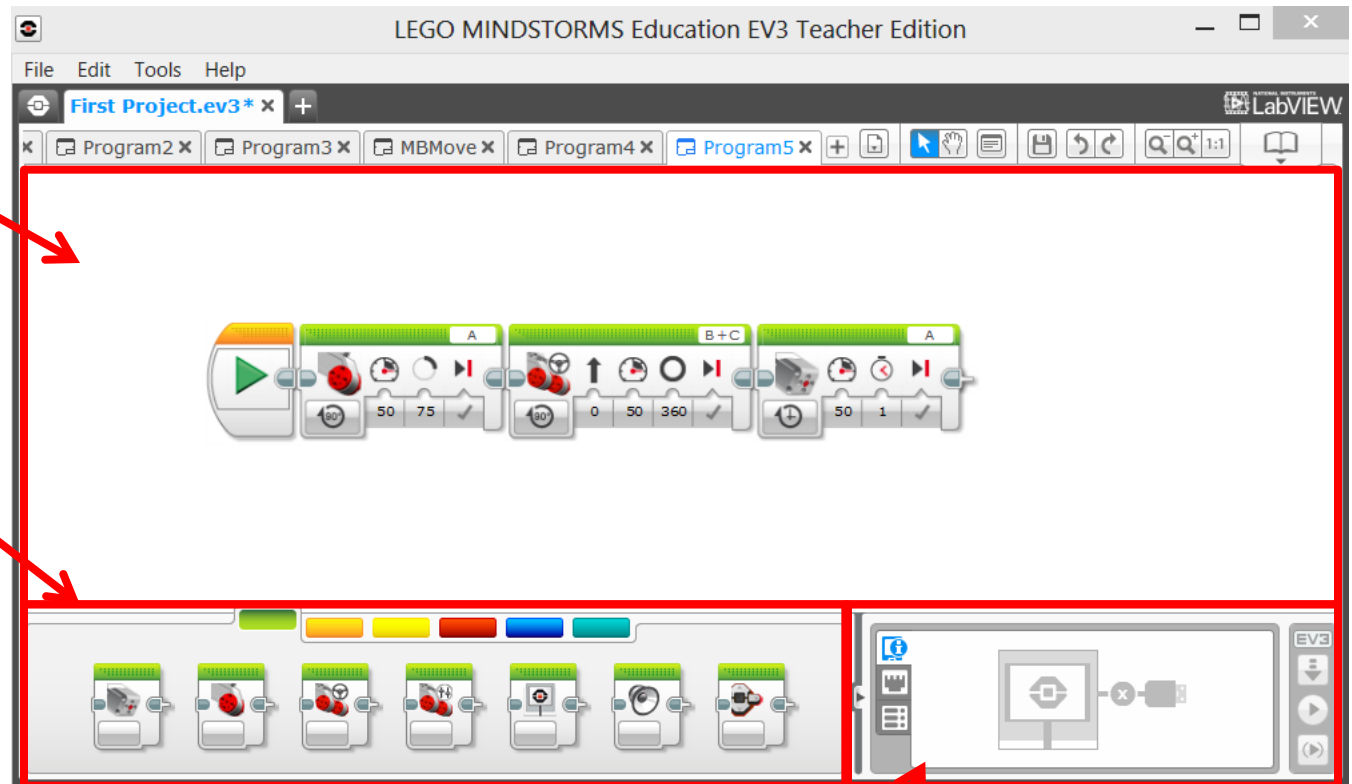
Type	Name	Show	Teacher Only
Program.ev3p	Program.ev3p	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Program2.ev3p	Program2.ev3p	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Program3.ev3p	Program3.ev3p	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Buttons: Copy, Paste, Delete, Import, Export

Programming Environment Workspace

Programming canvas
where you can lay out
the program's blocks /
instructions

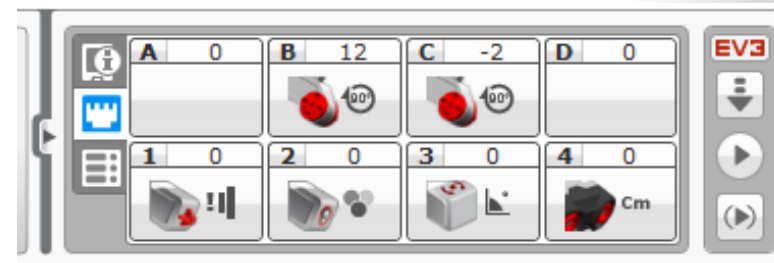
Programming palettes
where you can find the
various building blocks



Hardware page establishes communication with the EV3 brick and where you download programs into the EV3, view memory usages, battery level, and to find out motors or sensors and where they are connected.

The Communication Pane

- Connection status
- Download programs ready to be run
- Download/play programs instantly
- Download a section of a program to run
- Intelligent EV3 Brick status: name and battery level, etc.
- Port status and sensor readings
- Type of connection between the EV3 Brick and the computer (BT, Wi-Fi, or USB)



Multitasking



1/5



Objective

Use multitasking to move the Driving Base and play a sound at the same time.



©2013 the LEGO Group.

Build it

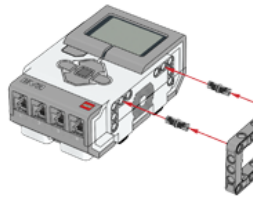
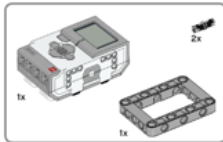


2/5



Click the link(s) below to open the building instructions, then build the model and return to this project to continue. Skip this step if the model is already built.

[Driving Base](#)



©2013 the LEGO Group.

Test it

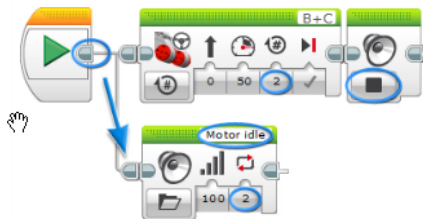


4/5



Recreate the program shown, drag the vertical sequence wire after adding the blocks to the canvas, then download and run to test.

Click Blocks



©2013 the LEGO Group.

Content Editor

- Animations
- Build guides
- Interactive guides
- Sample programs
- Challenges
- Save as a project
- Teacher and student modes
- Edit facility

Palettes

Action Blocks



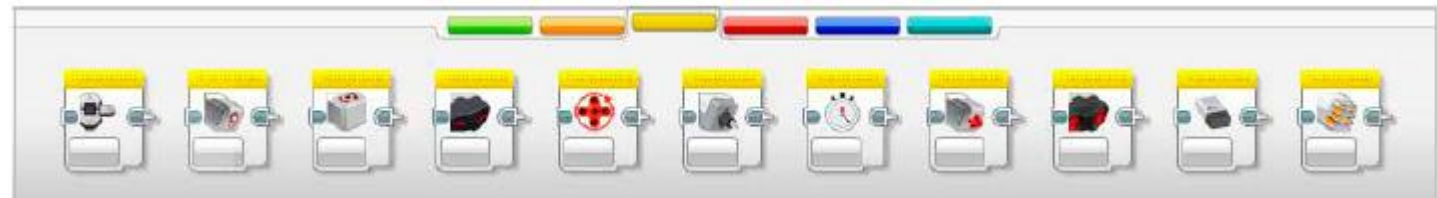
Medium Motor, Large Motor, Move Steering, Move Tank, Display, Sound, Brick Status Light.

Flow Blocks



Start, Wait, Loop, Switch, Loop Interrupt

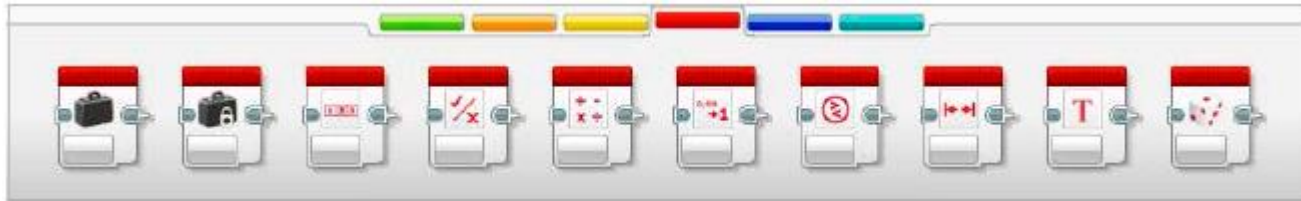
Sensor Blocks



Brick Buttons, Color, Gyro, Infrared, Motor Rotation, Temperature, Timer, Touch, Ultrasonic, Energy Meter, Sound

Palettes

Action Blocks



Variable, Constant, Array, Logic, Math, Round, Compare, Range, Text, Random

Advanced Blocks



File Access, Data Logging, Messaging, BlueTooth, Keep Awake, Raw Sensor Value, Unregulated Motor, Invert Motor, Stop Program

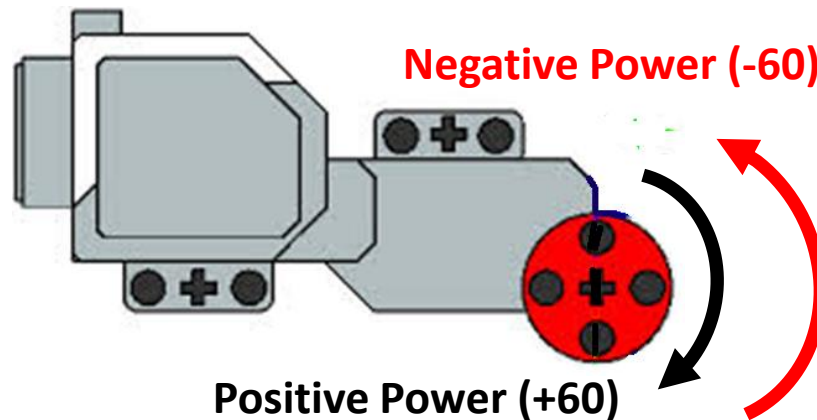
My Blocks



Block you create to repeat same actions in multiple programs. Programmers refer to this as subroutines or functions.

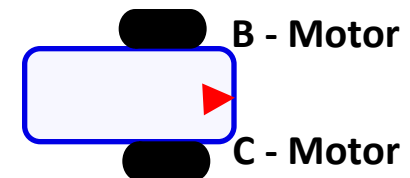
Controlling the EV3 Motors

- Instructing the robot to move and turn is accomplished by the Large Motors which rotate in a predetermined direction where positive amount of power (e.g. 75), will cause a clockwise rotation and negative power (e.g., -45) will cause a counter-clockwise rotation.

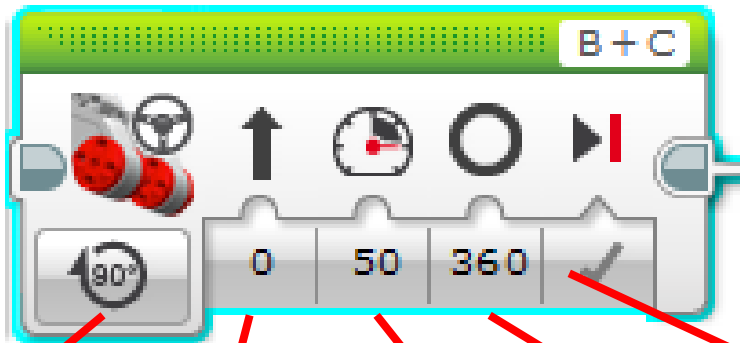


NOTE: the same concept applies the medium motor.

- **All examples used in this document assume the robot configuration and motor is mounted as shown.**

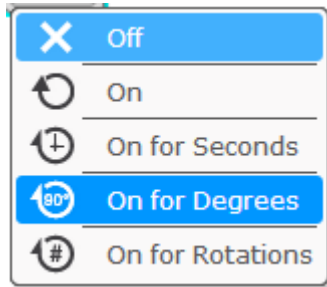


Move Steering Block



Move Steering

- Controls and *regulates* two motors.
- Both motors move either forward (positive power) or backward (negative power)
- Allows steering by applying more power to one of the two motors

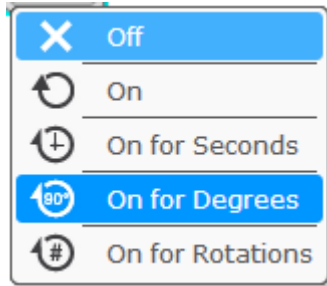
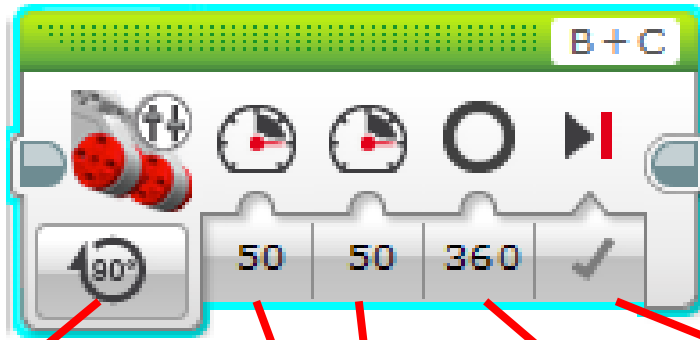


Steering Control from (-100- 100) where
0 = move straight;
Positive# = C more power than B
negative# = B more power than C

Power (100 to -100); positive number is move forward; negative, moves backward.
Note: small amount of power may cause the robot to stall.

Number of motor degrees, rotations or seconds

Move Tank Block



Power (100 to -100); positive number is move forward; negative, moves backward.
Note: small amount of power may cause the robot to stall.

Move Tank

- Control two motors and allows each motor to move with different power level including in different direction for turning or spinning.
- For turning: one motor has zero power; the other has positive (forward) or negative(backward) power.
- When zero power is specified, the motor is locked and will not move to ensure accurate turns
- For spinning, use positive power for one and negative for the other

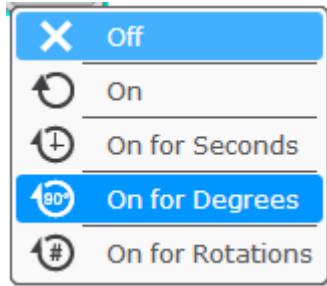
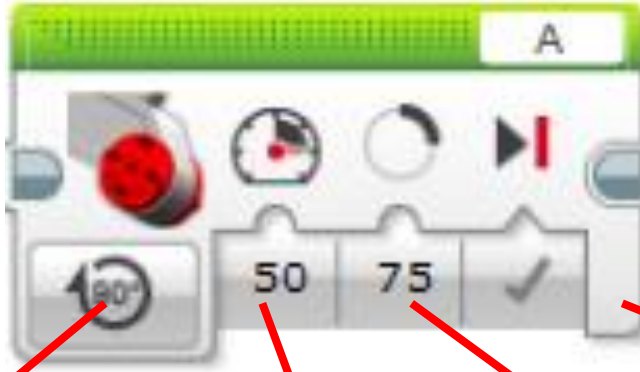


Number of motor degrees, rotations or seconds

Large Motor Block

Large Motor

- Control a single large motor
- When zero power is specified, the motor is locked and will not move to ensure accurate turns

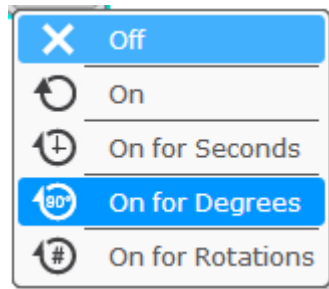
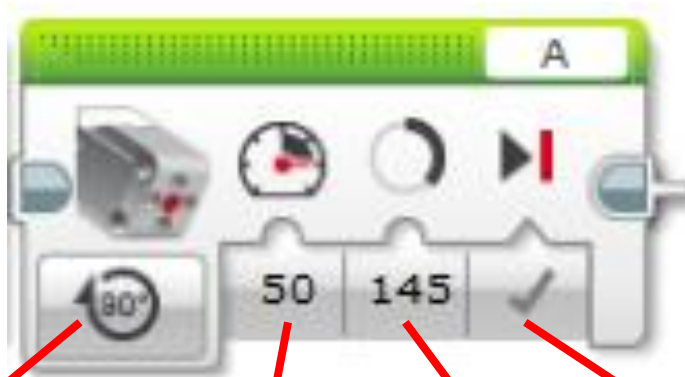


Power (100 to -100); positive number is move forward; negative, moves backward.
Note: small amount of power may cause the robot to stall.



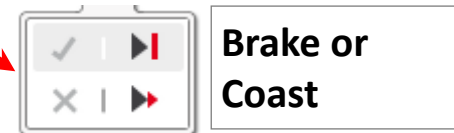
Number of motor degrees, rotations or seconds

Medium Motor Block



Medium Motor

- The Medium Motor block controls the Medium Motor. You can turn the motor on or off, control its power level, or turn the motor on for a specified amount of time or rotations
- When zero power is specified, the motor is locked and will not move
- Use positive or negative power to control direction



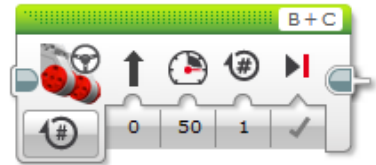
Power (100 to -100); positive number is move forward; negative, moves backward. Note: small amount of power may cause robot to stall.

Number of motor degrees, rotations or seconds

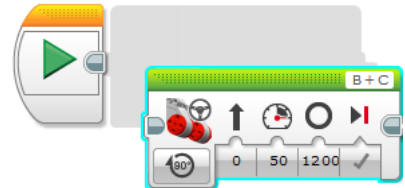
Steps to create a program



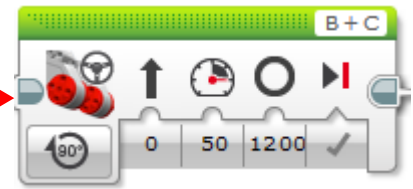
1. Click and hold block with left mouse button to drag it



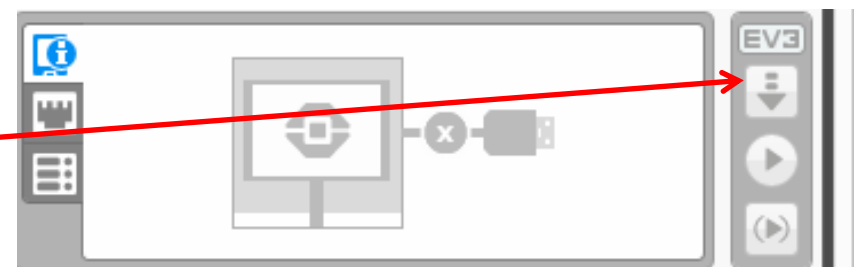
2. Drop the programming block when grey box appears



3. Select / enter options

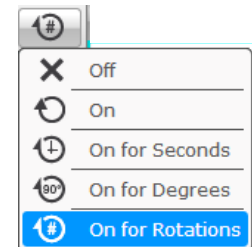


4. Click download to compile and load the program in the EV3 controller



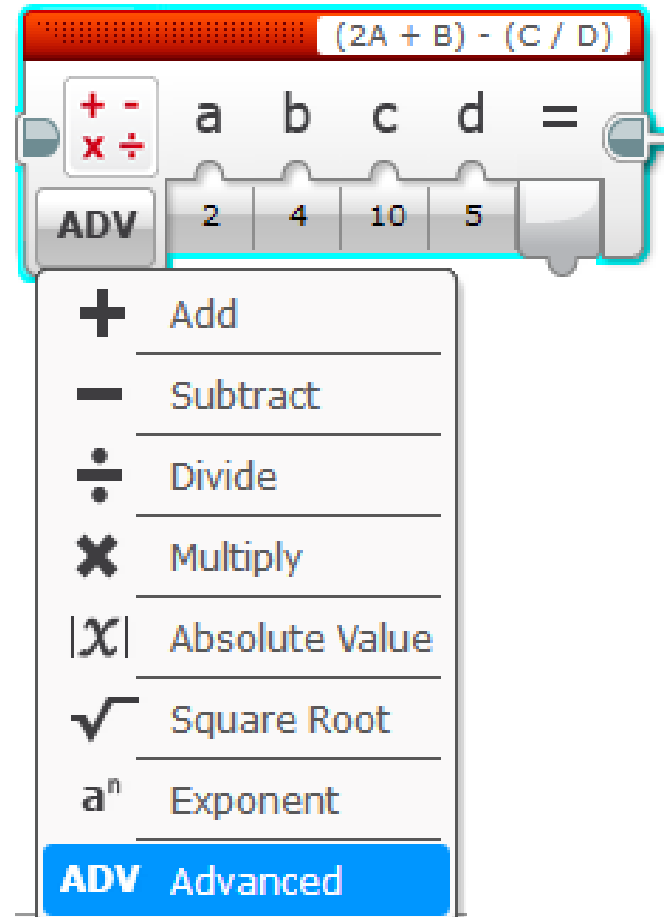
What's new

- All files are stored within the Project file, i.e., programs, my blocks. You can move / copy the project file to other computers and it will work. Now you can backup the entire project or even use a memory stick to store your project!
- Turning on/off the EV3 now takes about 30 seconds
- The **MOVE** block is replaced by the **MOVE STEERING** and **MOVE TANK** blocks
 - **MOVE STEERING** has single power control; motors are regulated, i.e., if one motor moves faster than the other, the faster motor will be slowed down to compensate.
 - **MOVE TANK**: has independent power controls for each motor where one can move faster than the other or even in opposite direction. This too is regulated.
NOTE: with limited testing, it appears that issues using steering in NXT are solved in EV3!
- The **MOTOR** block is replaced by **LARGE MOTOR** and **MEDIUM MOTOR**
- In NXT-G you specified direction, in EV3, you specify either negative or positive power to control the direction of the motors
- The **unlimited** duration option is replaced by **ON**
- The **STOP** option of **MOVE** and **MOTOR** blocks are replaced by **OFF**



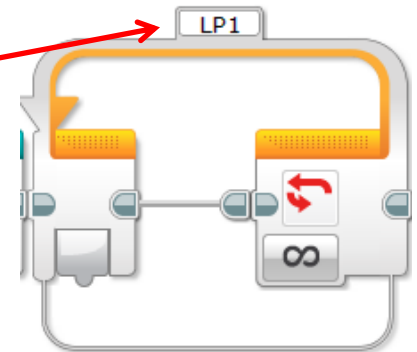
What's new – the much improved **Math Block!!!**

- As with the NXT, you can Add, Subtract, Divide, Multiply, and obtain the absolute value. Now you can calculate an exponent and CREATE YOUR CUSTOM FUNCTION under the **Advanced** option!
- You have up to 4 variable which can be initialized with wires from other math blocks or typed-in.
- Then you can enter your own function and obtain the result
- This reduces the number of math blocks used.
- **QUIZ: what is the answer?**



What's new continued

- LOOP blocks can be given names
- The LOOP block can be stopped with a condition within the loop or with a new **LOOP INTERRUPT**



- Multiple parallel programming sequences can be created using a new **START** block
 - Multiple sequences can run simultaneously
 - Clicking the green arrow of the **START** block, will compile and download the entire program to your EV3, but only the selected sequence will run.



*Walk and
Chew gum
At the same
Time!*



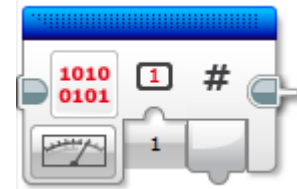
- Use the **Invert Motor** block to have a normal “forward” and “backward” directions swapped. Any programming blocks after the Invert Motor block that would normally make the motor turn clockwise will instead make the motor turn counter-clockwise, and vice-versa.

Programming Bug NOTE: *IT DOESN'T WORK with Move Steering or Move Tank.*

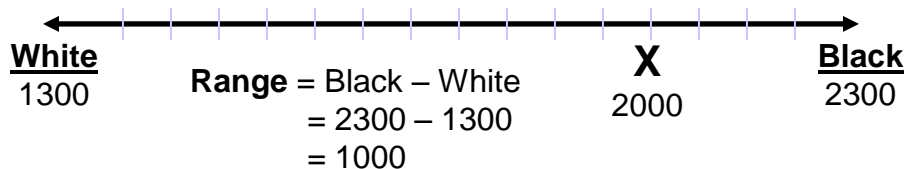


- Built-in light sensor calibration is gone! You have to build your own (see example on the following pages)

What is New - Continued



- In order to use NXT sensors with EV3, a new RAW SENSOR VALUE block is included
- Although you can use the NXT light sensor, you have to write a program to calibrate and interpret the reading using the RAW SENSOR VALUE block
 - REMEMBER: Raw white reading is less than black!!!
- How do we compute light intensity for NXT light Sensor
 - First, we have to know the **white** raw value and the **black** raw value
 - For example, if we measure the white black values as in light calibration, the numbers will be around 1300 for White and 2300 for black
 - To compute the light intensity of "X", we use this formula



Step 1:

$$\begin{aligned} \text{Range} &= \text{Black} - \text{White} \\ &= 2300 - 2000 = 1000 \end{aligned}$$

Step 2:

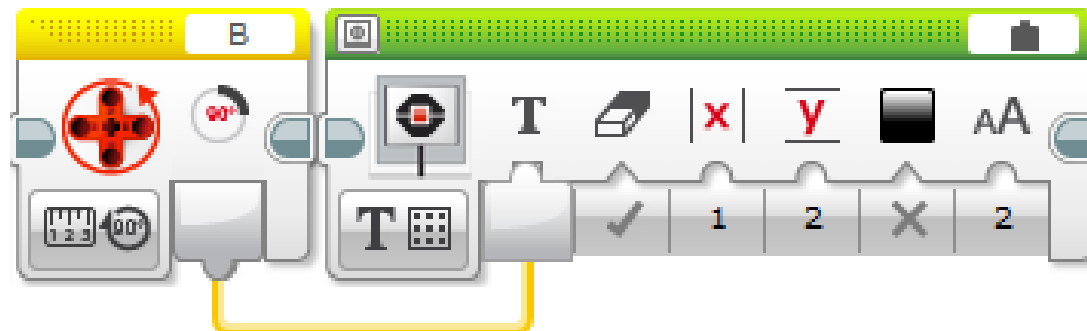
$$\text{Intensity} = \frac{\text{Black} - X}{\text{Range}} = \frac{2300 - 2000}{1000} = .3 \text{ or } 30\%$$

What's New Continued

- The NUMBER to TEXT block, which was used to convert numeric values to text so they can be displayed, was eliminated!
- Logic and numeric Data wires can be automatically converted as described below.

From Type	To Type	Result
Logic	Numeric	False = 0, True = 1
Logic	Text	False = "0", True = "1"
Numeric	Text	Text representation of the number (For example, "3.5")

You take the rotation value (degrees) and display it on the EV3 Controller screen without having to convert the number to text!!!

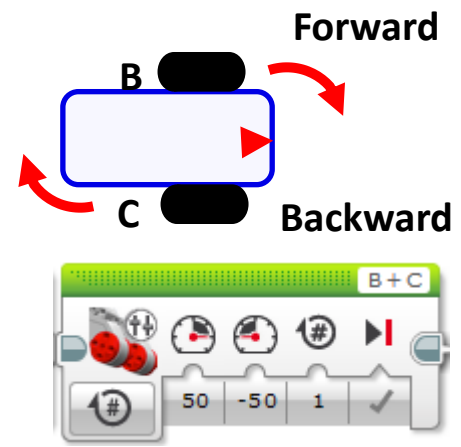
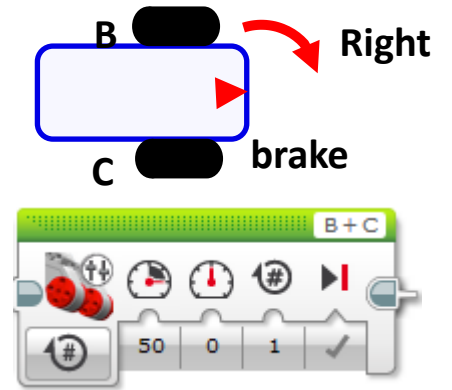


Turn vs. Spin

- There are two ways in which you can make the robot turn
 - Make **ONLY** one motor move, or
 - Using both motors moving in opposite direction, and this is referred to as "**spinning**"
- One Motor move:
 - Right Turn – Use **MOVE TANK** block and select a power level for the "**B**" motor and zero for the "**C**" motor
 - In this case the robot's right wheel will be stationary and the left wheel will move.
- Turning with two motors in opposite direction
 - To turn right, use the **MOVE TANK** block where the "B" motor will turn clockwise (positive power) and the "C" motor will turn counter clockwise (negative power).

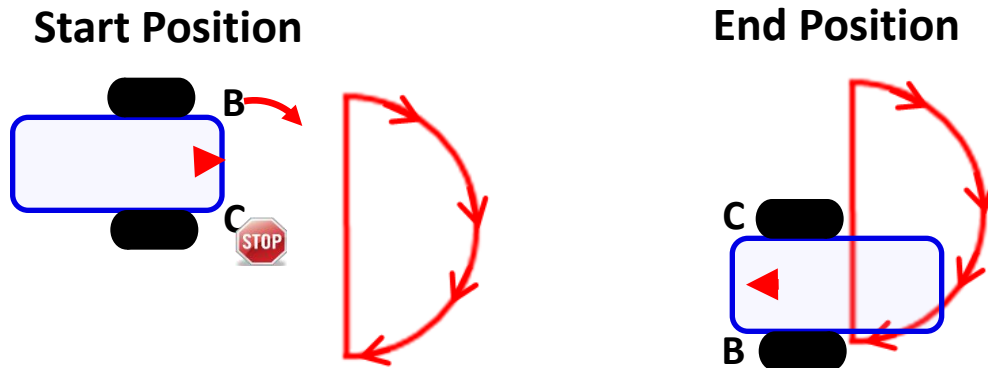


TIP: for turning in a tight spot, use the two motors.

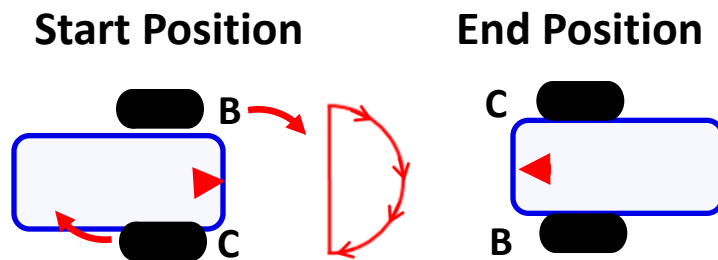


Turns Continued

- To make 180 degree right turn using a tank move or large motor



- To make 180 degree right turn using tank move (spin in place)



- Note the distance travelled is shorter (exactly half) when using both two motors.

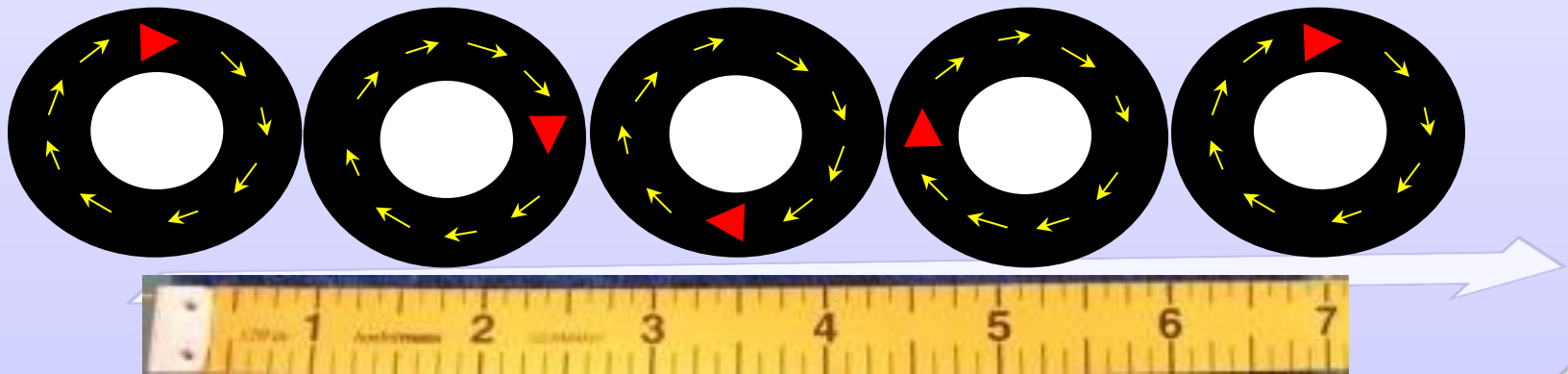


Geometry and Math

Fun way to see that what is learned in school can be applied to the FIRST LEGO League's robots.

Note: it may be a stretch for younger teams that have not covered these concepts in school.

$$C = \pi \times \text{Diameter}$$



Inches to Motor Degrees

- The Circumference of the robot's wheel determines the number of inches it will travel in 1 rotation
- Circumference of wheel = $\pi \times \text{Diameter}$
- Degrees traveled per inch = $360 / \text{circumference of wheel}$

3.2 inches



- Circumference of wheel = $\pi \times \text{Diameter} = 3.14 \times 3.2 = 10.0$ inches
- Degrees traveled per inch = $360 / 10.0 = 36$ motor degrees

1 Rotation = 10 inches

2.2 inches



- Circumference of wheel = $\pi \times \text{Diameter} = 3.14 \times 2.2 = 6.9$ inches
- Degrees traveled per inch = $360 / 6.9 = 52$ motor degrees

1 Rotation = 7 inches

1.6 inches



- Circumference of wheel = $\pi \times \text{Diameter} = 3.14 \times 1.6 = 5$ inches
- Degrees traveled per inch = $360 / 5 = 72$ motor degrees

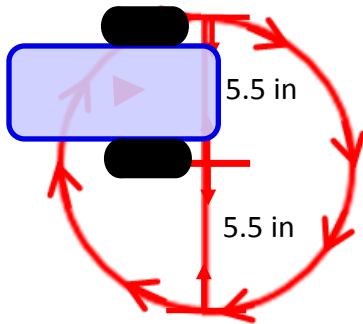
1 Rotation = 5 inches



You can use a ruler or measuring tape to plan mission...

Measuring turn travel distance – More Geometry!

- When the robot turns using one motor, it will make a circle whose Radius is the distance between the wheels



In this case, the radius is 5.5 inches

$$\begin{aligned} \text{Circumference} &= 2 \times \pi \times R \\ &= 2 \times 3.14 \times 5.5 = 34.5 \text{ inches} \end{aligned}$$

- If the robot is using the wheel whose diameter is 2.2 inches and therefore its circumference is 7 inches, how many **wheel rotations** will it take to make a complete robot turn rotation (34.5 inches)?

$$\frac{\text{Circumference of Robot Turn Circle}}{\text{Circumference of Robot's Wheel}} = \frac{\text{Diameter} \times \pi}{\text{Diameter} \times \pi} = \frac{11 \times \cancel{3.14}}{2.2 \times \cancel{3.14}} = 4.9$$

Note: it takes 4.9 wheel rotations to make a complete (360° turn)
(or 4.9 motor degrees to travel 1 degree of turn circle)

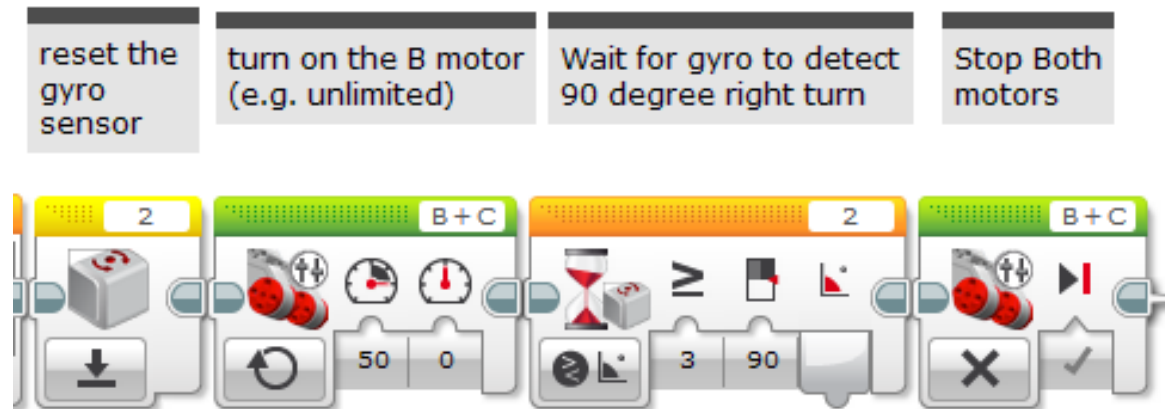
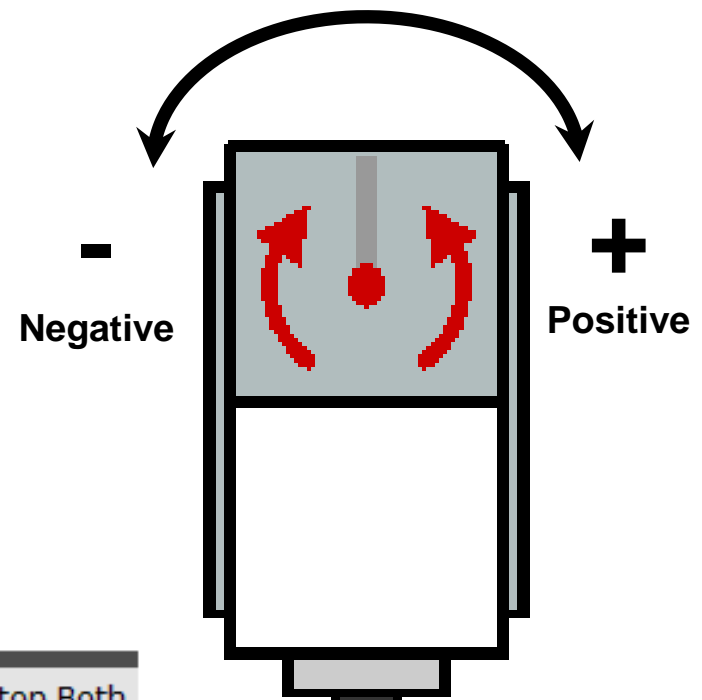
- Example: to make a 90° robot turn
 - Motor Degrees = 90 X 4.9 = 441 Motor Degrees

2.2 inches



Turning the easy way – Gyro Sensor!

- The gyro sensor has Angle Measure Feature to allow you to control turns based upon the turn angle.
- When the Gyro Sensor is attached to the robot, and robot turns to the right, the gyro sensor will report positive number; if the robot turns left, the Gyro Sensor will report a negative number.
- See example below



NOTE: you must reset the gyro sensor as the first step and immediately before turning so that your turn is measured from the robot's current position!

Programming Process

- Analysis and Planning Steps
 - Define the problem
 - Brainstorm solution and select one
 - KEEP IT SIMPLE!
 - Plan and create a flowchart and take measurements
- Programming suggestions
 - Divide the program into small pieces
 - Program one pieces at a time
 - Example: Move the robot to black line
 - Once the step is consistently repeatable, go to the next
 - Whenever possible reuse repeatable combination of blocks using MYBLOCK
 - Ask for help
 - <http://forums.usfirst.org/forumdisplay.php?f=24>
 - The questions should be generic and not specific to strategy

Tips

- For moving straight, the **MOVE STEERING / MOVE TANK** blocks have a built-in PID to regulate the movement of "B" & "C" motors. If one motor falls behind, the **MOVE STEERING** block compensate by applying less power to the faster motor.
- For driving the robot, use the **B & C** motor ports; the **A and D** ports should be used for the robot's arm.
- Using full motor power (100%) may cause erratic robot movement, use 75% or less.
- Conversely, too little power (below 25%) may cause the robot to stall.
- Brake at the end of each **MOVE** block to take advantage of the PID which self corrects to achieve more precise moves.
- Using Degrees is a more accurate way to move motors; using time, will be inconsistent when the batteries become weak
- The **MOVE STEERING / MOVE TRACK** block also keeps track of "errors" that accumulate in multiple blocks and adjusts itself.
- Use the **brake** option and also use the **RESET** block.
- REMEMBER: the tradeoff between speed and accuracy!

Why Color / Light sensors?

One of the ways for the robot to know its location is to take advantage of the markings on the field mat. Every year, the Robot Game's mat has lines or dark markings that can be detected by the color Sensor.

In this section, we will cover the following

- How do color sensors work?*
- How to calibrate the color sensor?*

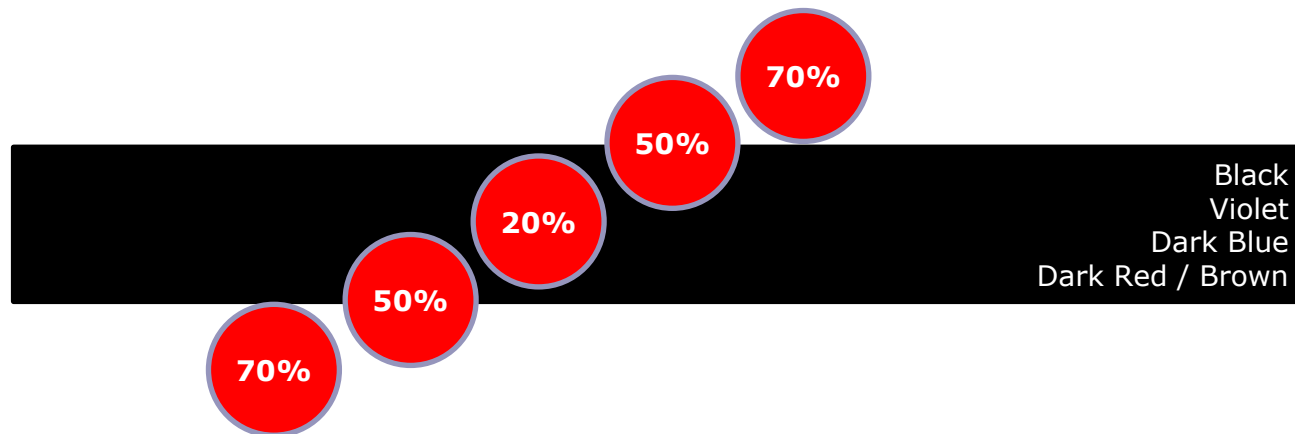
Common use Programming Examples:

- Move until a dark line is encountered by the robot,*
- Align the robot with a black line*
- Follow a line*

Color Sensors – reflected light mode...

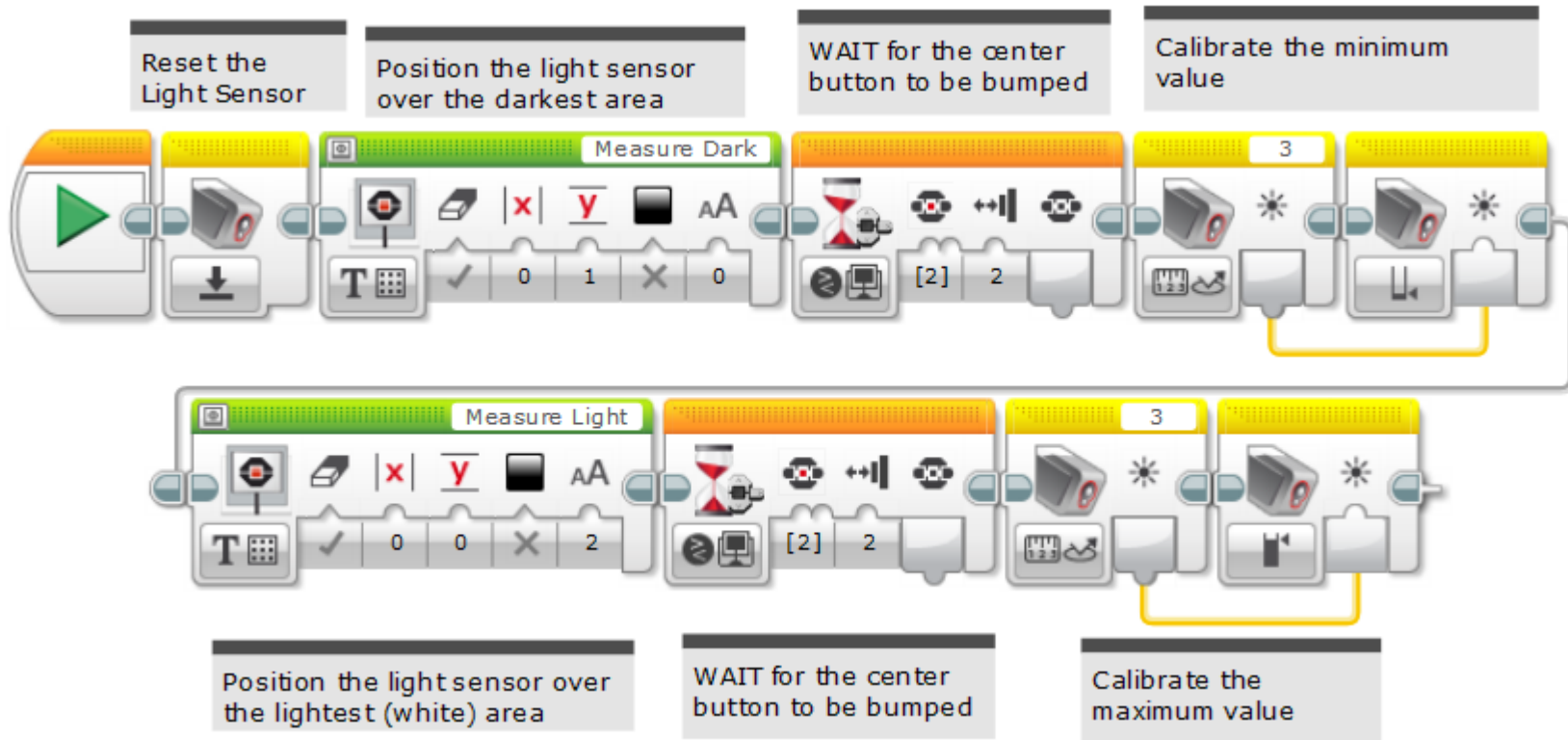


- The **COLOR SENSOR** shines a light on the mat and reads the reflected brightness level (**intensity**) level, i.e. dark or bright, to help the robot recognize its position and through programming take action.
- Light **intensity** ranges from 0-100%
- On a bright (white) area of the mat the light **intensity** value will be above 50%
- On darker area (blue, black, green,...), the light **intensity** value will be below 50%



- To obtain more accurate reading, make sure the light sensor is close to the mat (less than ½ inch)
- Calibrate the sensor whenever light conditions change (see calibrate slide)

How to write a program to calibrate the light sensor



WAIT for light value block



- The **WAIT** block keeps checking for the specified light intensity value and when the condition is met, the next step in the program is executed

The screenshot shows the LEGO Mindstorms software interface. A 'WAIT' block is selected, and its configuration menu is open. The 'Compare' option is chosen, and the 'Reflected Light Intensity' sensor type is selected. The value '50' is entered in the input field. A dropdown menu for comparison operators is also visible, showing options like '=', '≠', '>', '<', '≥', and '≤'. Red arrows point from text boxes to specific parts of the interface: the port number '3', the value '50', the 'Compare' menu, and the comparison operator dropdown.

Port: identify the NXT port where the light sensor is connected

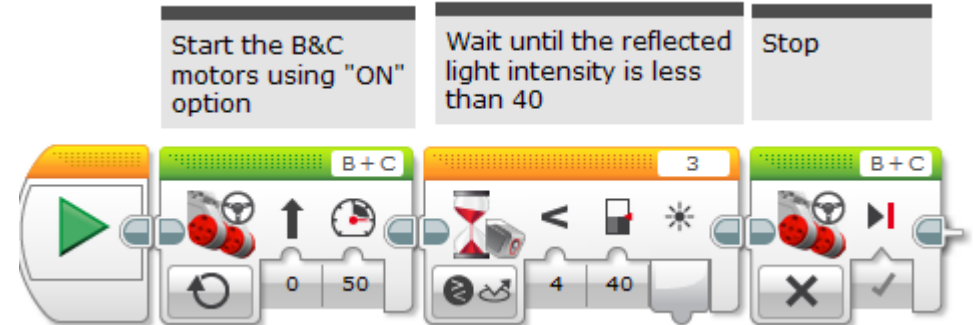
Enter value you would like to test.

Select ">" greater than or "<" less than.
Example: if you select "<" and entered 50, and the light sensor encounters a black block, the result will be the test is "True"

Wait Block – Examples

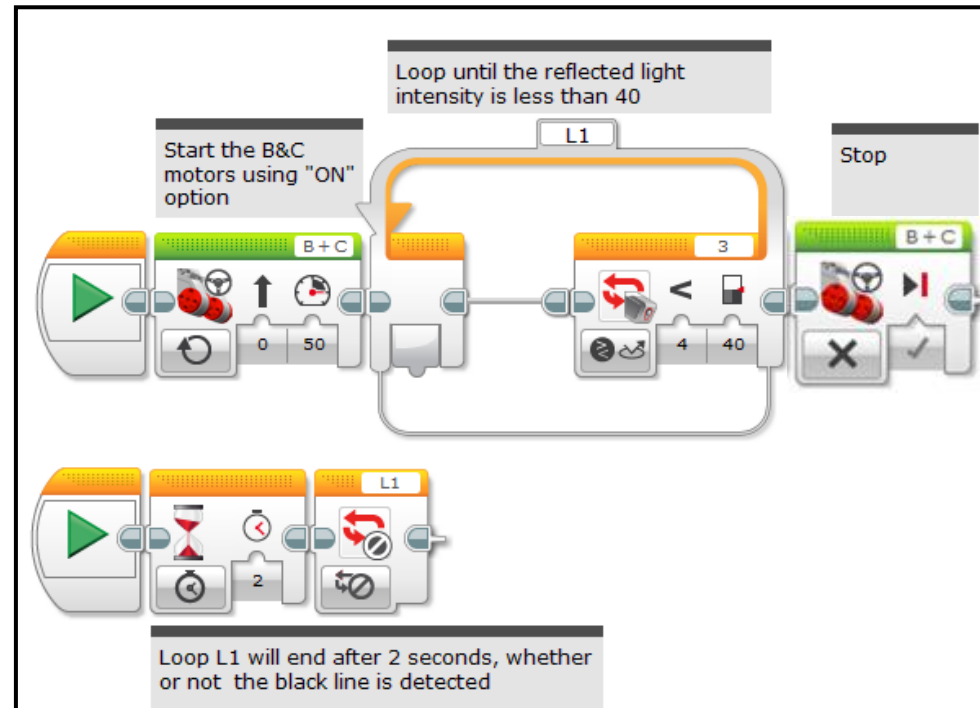
Example 1:

Move until the robot detects a (black) dark line and stop



Example 2:

Move to detect a black line, using a **LOOP** block instead of **WAIT**. Simultaneously use the **WAIT** for time block and if two seconds elapsed, end the loop using **LOOP INTERRUPT** and stop the motors.



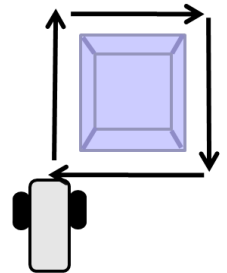
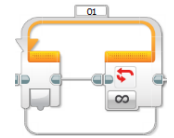


Advanced Blocks and Example

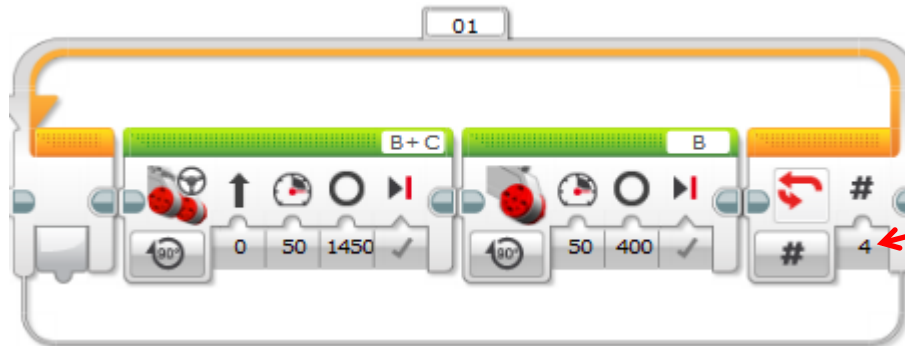
- *To create a line follower program you will need to use the light sensor and repeatedly check the light value and adjust the robot's position*
- *LOOP Block*
- *COMPARE Block – Line follower examples*
- *Additional Blocks*
 - *VARIABLE*
 - *MATH*
 - *COMPARE*
- *Data Hubs*

The LOOP Block


- Sometimes, there are actions that you want to repeat. The **LOOP** block allows you to repeat those actions until an end condition is met (or becomes TRUE).
- Example: make the robot move around a box and return to its starting position
- To move along the box sides, it takes 8 blocks as follows:



Using the **LOOP** block, only



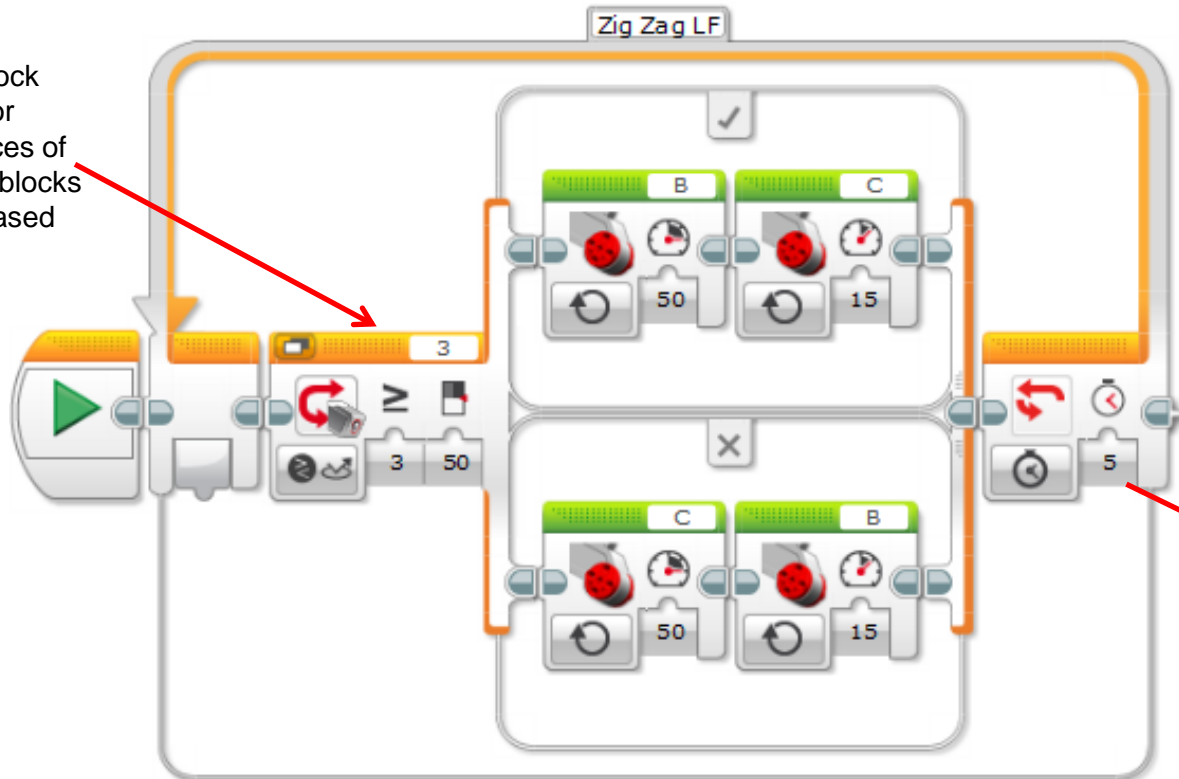
Repeat the loop 4 times

 Warning: Deleting the LOOP block will also delete all the blocks within the loop. You can move the blocks out of the loop, then delete it.

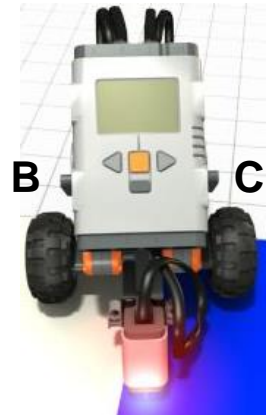
SWITCH block

- The **SWITCH** block will check for a condition and will take one action if the condition is true and another action if the condition is false

If the light intensity is greater than or equal to 50, The B motor will move faster than C



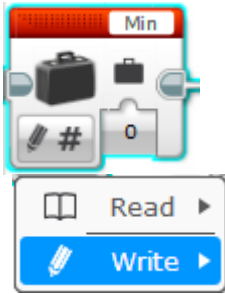
The Switch block contains two or more sequences of programming blocks that are run based on condition.



Repeat the loop for 5 seconds

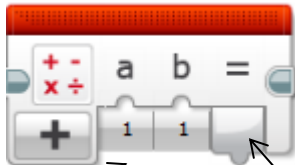
If the light intensity is less than 50, The C motor will move faster than B

Advanced Blocks



VARIABLE block

- The variable block is a “bucket” where you can store information and retrieve it at a later time.
- There are three types: Number, Text, and Logic



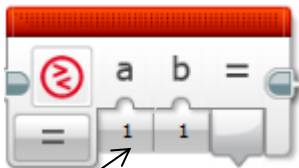
MATH block

- The math block allows the addition, subtraction, division or multiplication of two numbers.

Input

Result

+	Add
-	Subtract
÷	Divide
x	Multiply
x	Absolute Value
√	Square Root
a ⁿ	Exponent
ADV	Advanced



COMPARE block

- The compare allows you to determine if one number is greater than, less than or equal to another number.

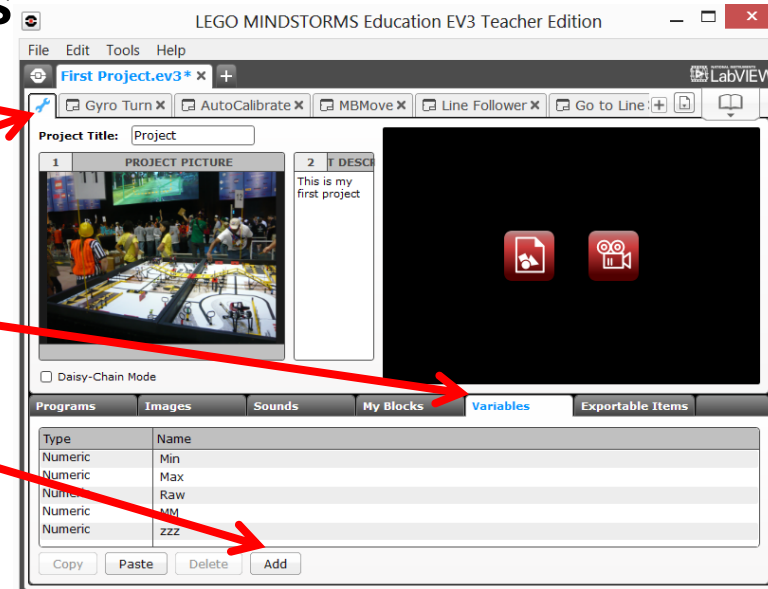
Input

Result (True or False)

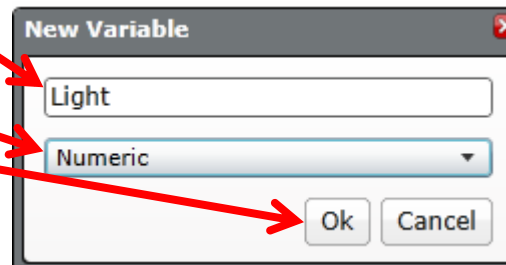
=	Equal To
≠	Not Equal To
>	Greater Than
≥	Greater Than or Equal To
<	Less Than
≤	Less Than or Equal To

VARIABLES –create your own variables

- To create your own variables
 1. Click "Project Property" icon
 2. Click the "Variables" Tab,
 3. Then click on "Add"



4. Enter the variable name
5. Select the type
6. Press OK



My Block

- **My Block** is a combination of one or more blocks that are grouped into a single "Block". Once created, it can be used in many programs. **My Block** can be used in the following ways:
 1. Minimize the coding, if certain actions are repeated in multiple programs. Often, you need to repeat certain steps, for example, different programs require that the robot would back into the wall to re-orient itself. This process can be placed into a **My Block**.
 2. Divide a program into smaller manageable pieces
 3. Reduces the amount of memory used.
 4. Clarify the action taken in programs by creating self explanatory **My Blocks**
- Example 1: your program already includes 22 blocks and you're not finished. It is time to consider breaking up the program into "chunks", i.e., **My Blocks**.
- Example 2: For turning left or right you use the **LARGE MOTOR** block. When someone is reviewing the program, they can't tell which way the robot is turning if you are only using **MOTOR** block. Solution: create a "Turn Left" **My Block** to make it easy to understand the program.
- Example 3: One you've fine tuned a perfect 90 degree turn, you can create your own **My Block** for the perfect 90 degree left and right turn.

My Block Example – Moving using inches instead of degrees

- Objective:

- Build a move **My Block** that takes one input called duration which represent the number of inches

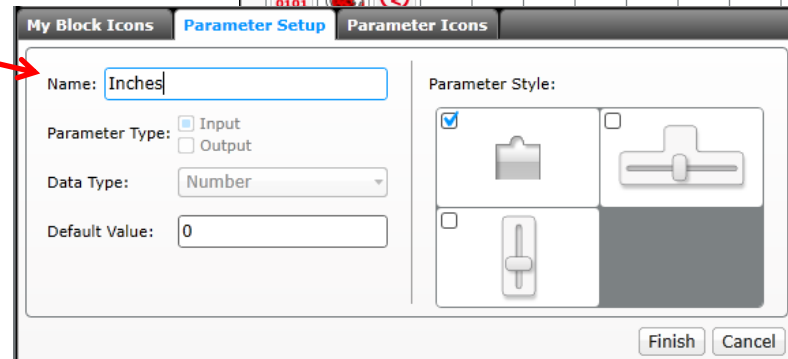
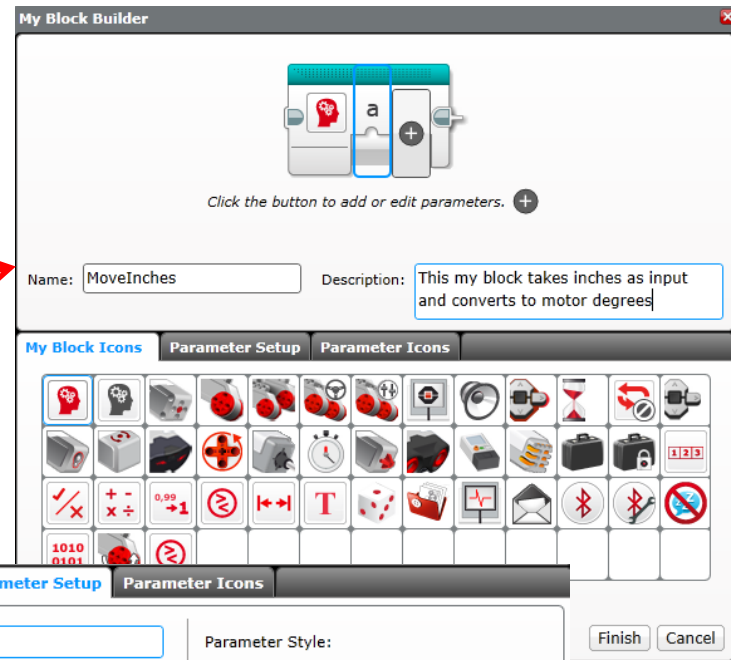
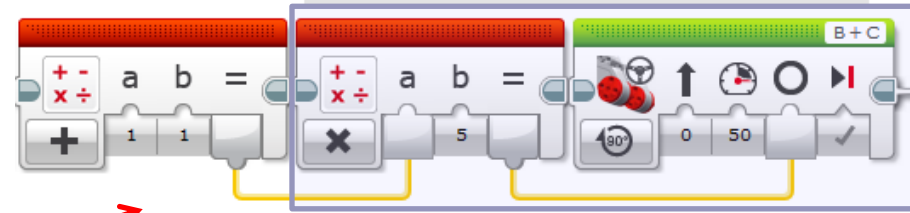
1. Select the Math & Move blocks

2. Under tools, select "**My Block Builder**"

3. Type **My Block** Name "MoveInches", and enter an optional description and select a my block icon

4. Click on the Parameter Icon, then enter parameter name and click on Finish.

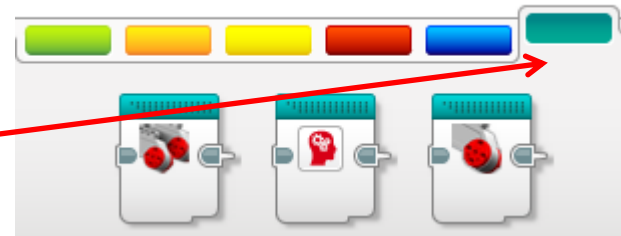
The math block multiplies "A" times "B" which is set to 5 to convert inches to motor degrees. The result is fed into the MOVE STEERING block's Degree hub



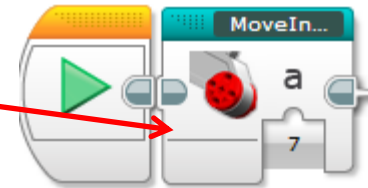
Using My Block

- Once you've created **My Block**, it will become available on the My Block palette

1. Select the My Block palette



2. Drag and drop the **My Block** named "MoveInches" into the program



- Type the number of inches into the "a" input value...That's it.

My Block Tips

- Use **My Block** to break down a large program into two or more **My Blocks**
- If you create a **My Block** with one input and decided later to add another input, you'll have to start over
- If you create a **My Block** with two inputs and decided later to remove one, you cannot delete the input; either start over, or ignore it.



Finally, the resources

- FLL Forum where you can find answers to your questions from other coaches <http://forums.usfirst.org/forumdisplay.php?f=24>
- *FIRST* <http://www.usfirst.org/roboticsprograms/fll/>
- Video tutorials for both EV3 and NXT-G www.stemcentric.com
- A great tutorial on how to program in NXT-G www.ortop.org/NXT_Tutorial
- Instructions to build a variety of robots and sample programs www.nxtprograms.com
- Wish you had extra LEGO pieces? www.bricklink.com
- LA Region FLL website <http://fll.larobotics.org/>
- LA Region FLL Google Group <http://groups.google.com/group/LARFLL>
- If all else fails tony.ayad@gmail.com or fll@larobotics.org