FLL® Programming

2017 Workshop Series

2017

Prerequisites & Equipment Required

- Basic computer skills
- Assembled EV3 Educational robot or equivalent
- Computer or Laptop with LEGO® Mindstorms software installed
- Computer login ID and password
- EV3 USB cable
- Centimeter tape measure

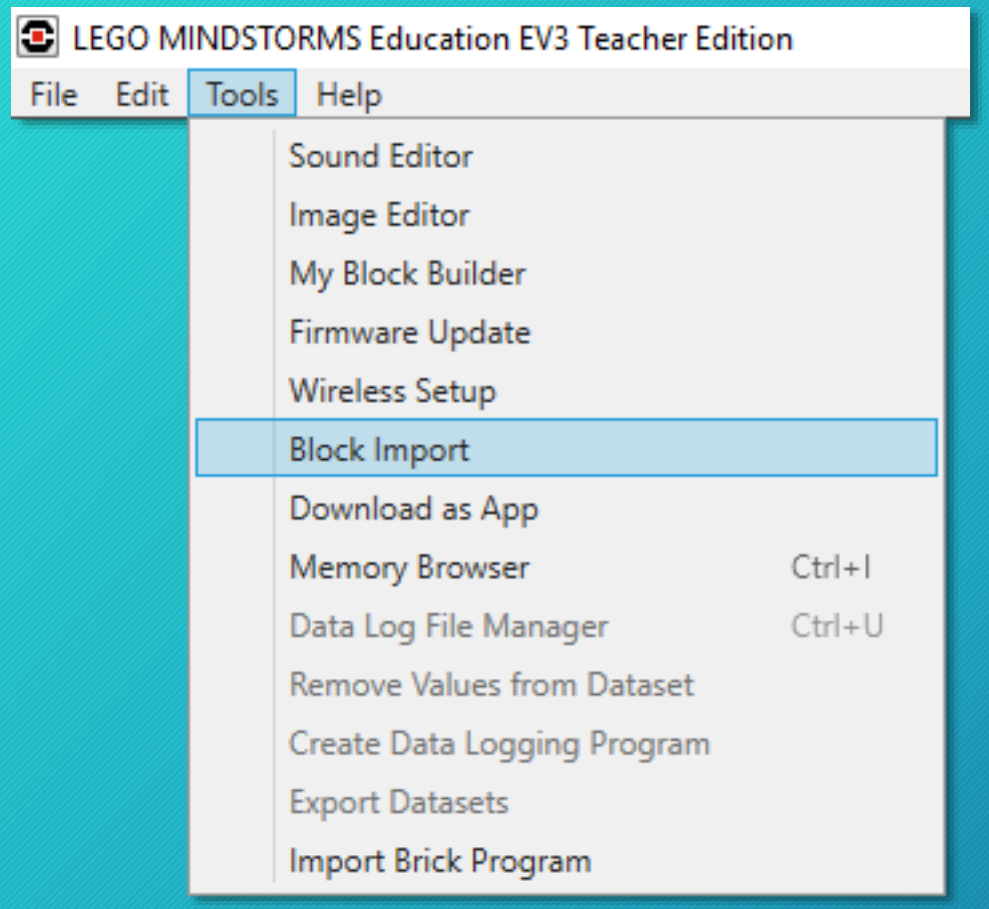
Icons defined

Informational/Instructional slide

Hands-on slide

Bluetooth connection information

Import additional program blocks

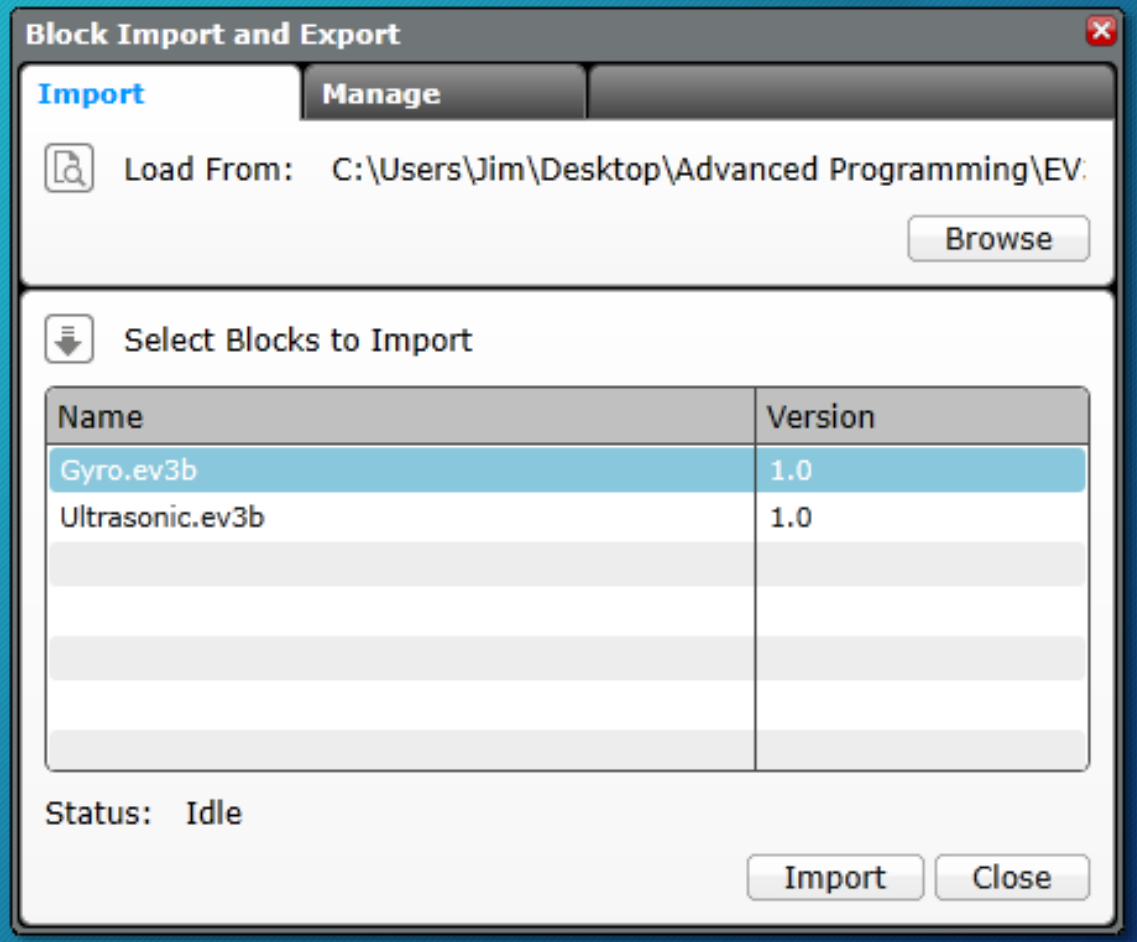


- If you are using the Home Version of EV3 software, you will need to import the Gyro and Ultrasonic program blocks

1. Download the blocks from LEGO® MINDSTORMS® download website
   https://www.lego.com/en-us/mindstorms/downloads
2. In Mindstorms® application, from the Tools drop down menu, select Block Import, the Block Import and Export dialog displays

Import additional program blocks

3. Select the Browse button, and navigate to where you downloaded the blocks
4. Select Open
5. Select the block to import
6. Select the Import button



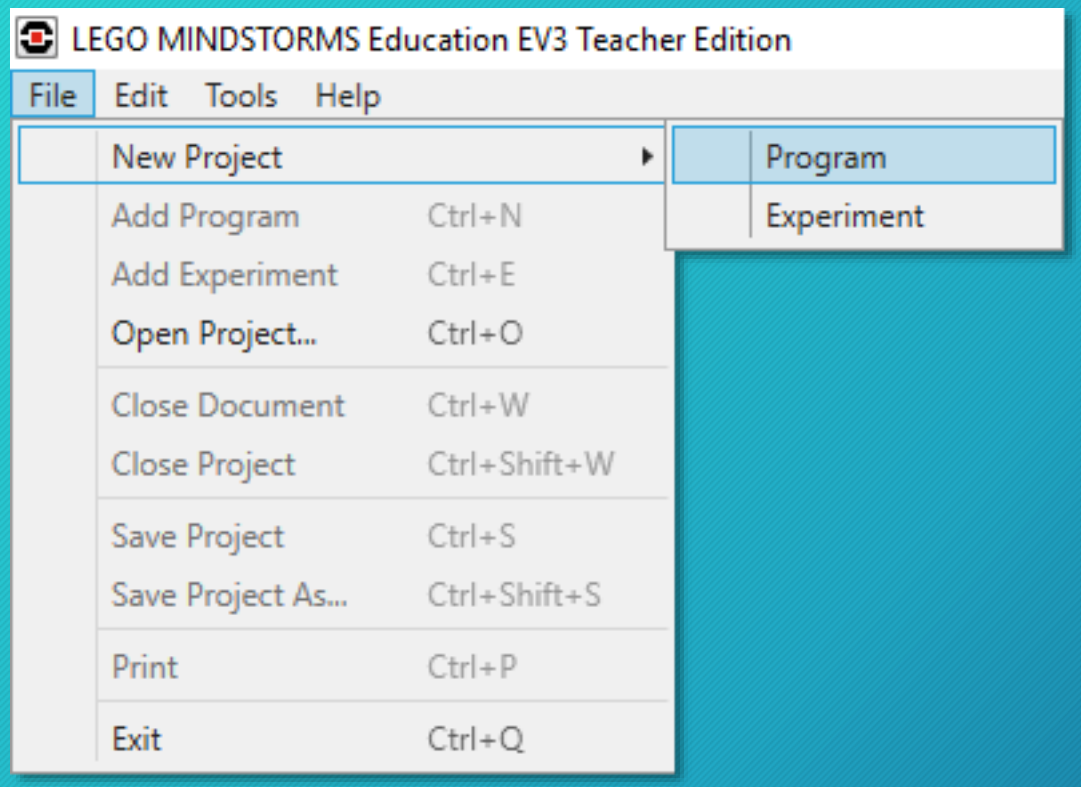
What is a Program?

- A program is a sequence (a list in order) of instructions that tells the robot how to perform a task.
- The robot does exactly what you tell it to do, and only what you tell it to do.
- You, the programmer determines what actions the robot will perform by the programming.

Session 1

1. EV3 software introduction
2. Programming the robot to move a set distance
3. Connecting with Bluetooth
4. Building a My Block
5. Fundamentals of turns
6. Strategy

Open LEGO® Mindstorms® EV3



Double click the LEGO® Mindstorms® EV3 icon, application launches

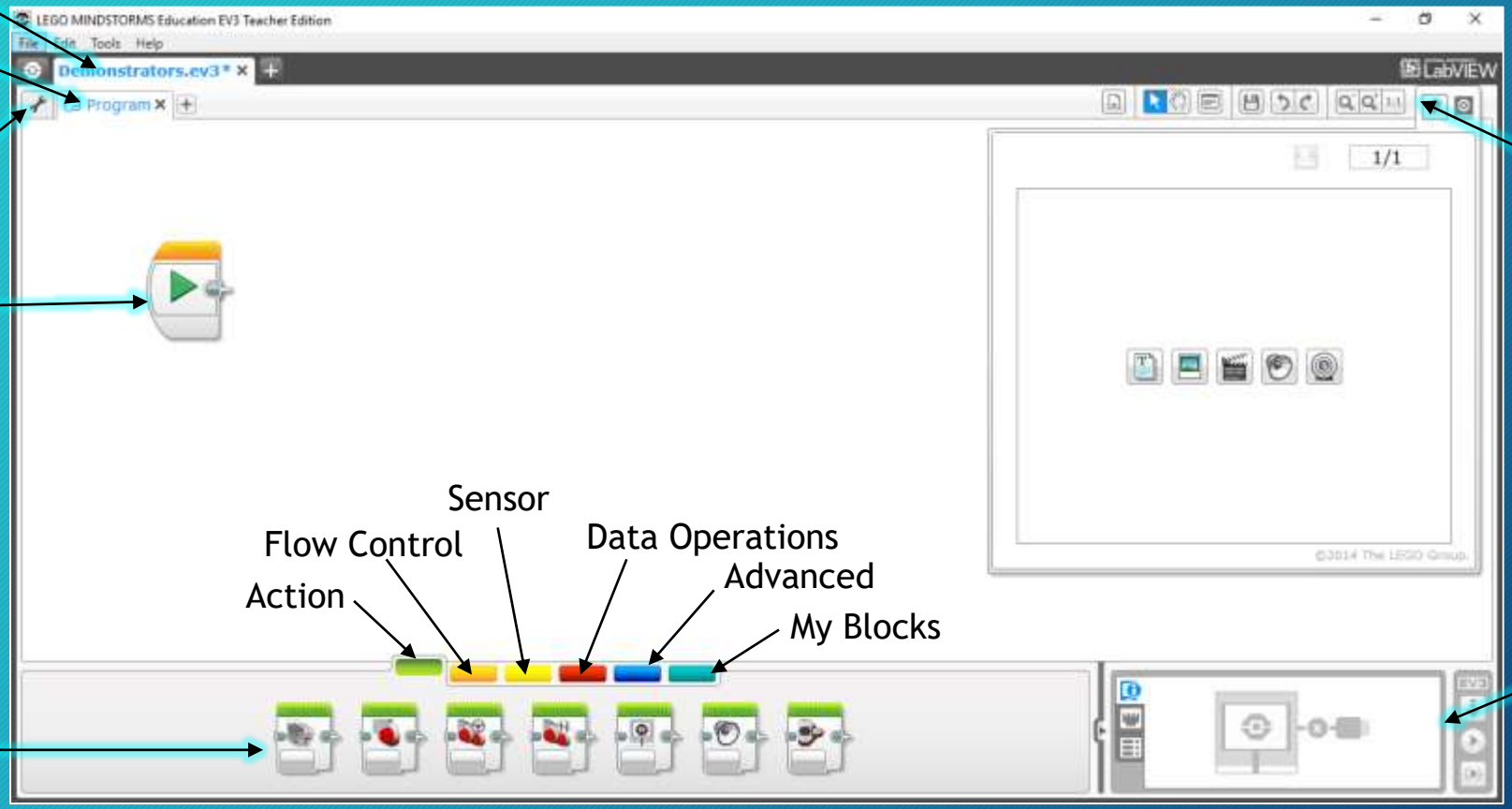
Select File ➔ New Project ➔ Program

Select File ➔ Save Project as

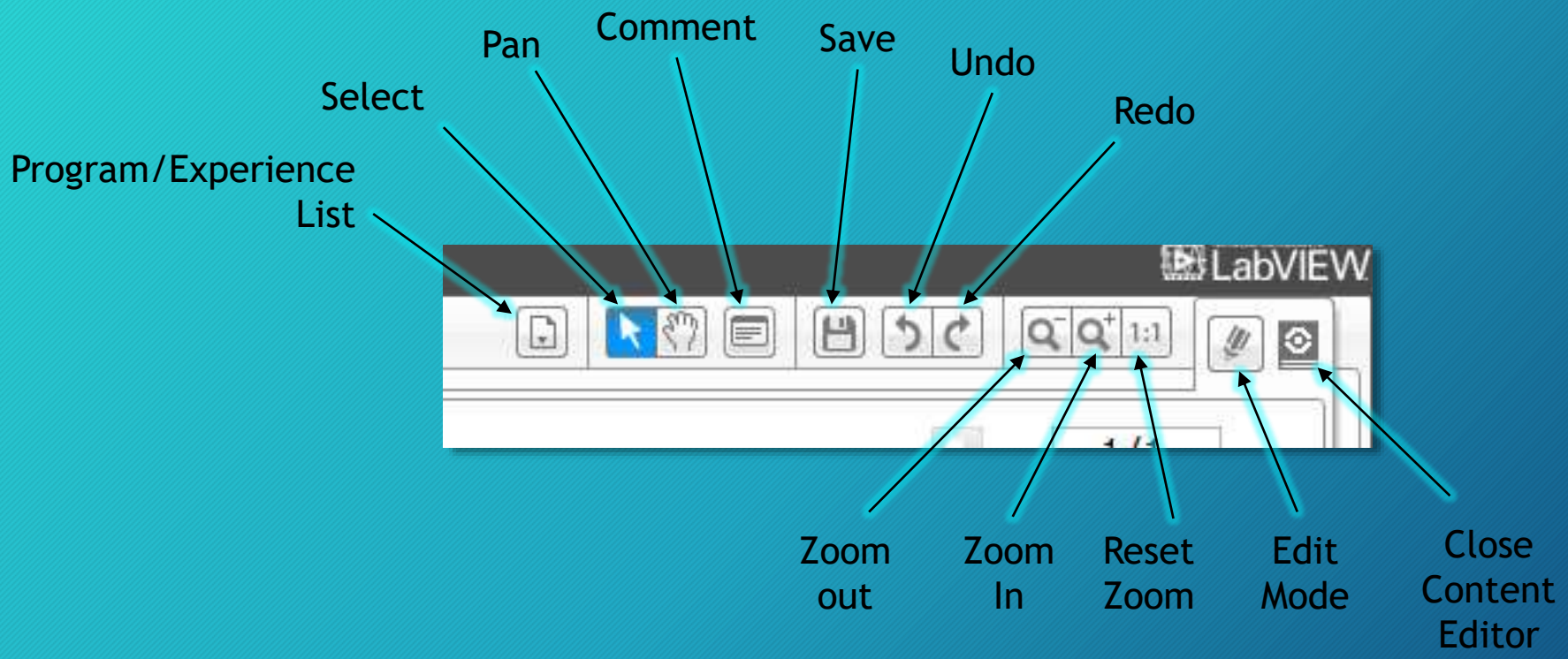
Create a folder to save your work to (Use FLL-[your first name] for folder name) Example: FLL-Jim

- Name Project: [your first name]
- Select Save button

Mindstorms® EV3 Software Overview



Toolbar – Upper right



EV3 Action and Flow Control Blocks

Action (green) blocks

Medium Motor | Large Motor | Move Steering | Move Tank | Display | Sound | Brick Status Light

Flow Control (orange) blocks

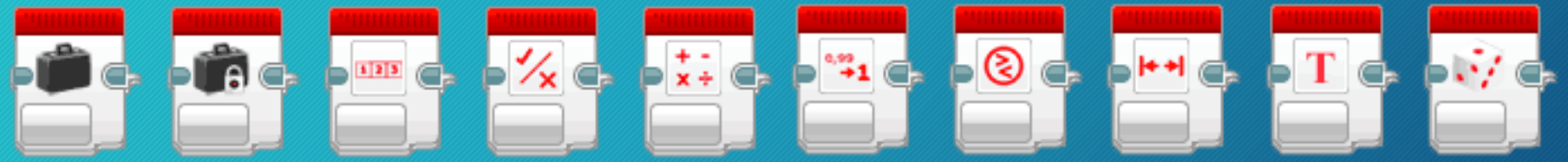
Start | Wait | Loop | Switch | Loop Interrupt

EV3 Sensor and Data Operations Blocks

Sensor (yellow) blocks

Brick Buttons | Color Sensor | Infrared Sensor | Motor Rotation | Timer | Touch Sensor

Data Operations (red) blocks



Variable | Constant | Array Operations | Logic Operations | Math | Round | Compare | Range | Text | Random

EV3 Advanced blocks

Advanced (dark blue) blocks

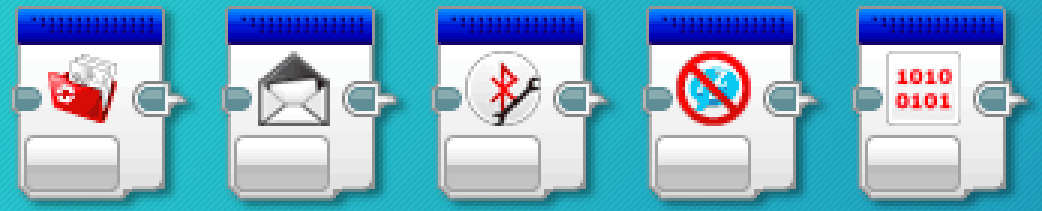
File Access

Messaging

Blue Tooth Connection

Keep Awake

Raw Sensor Value



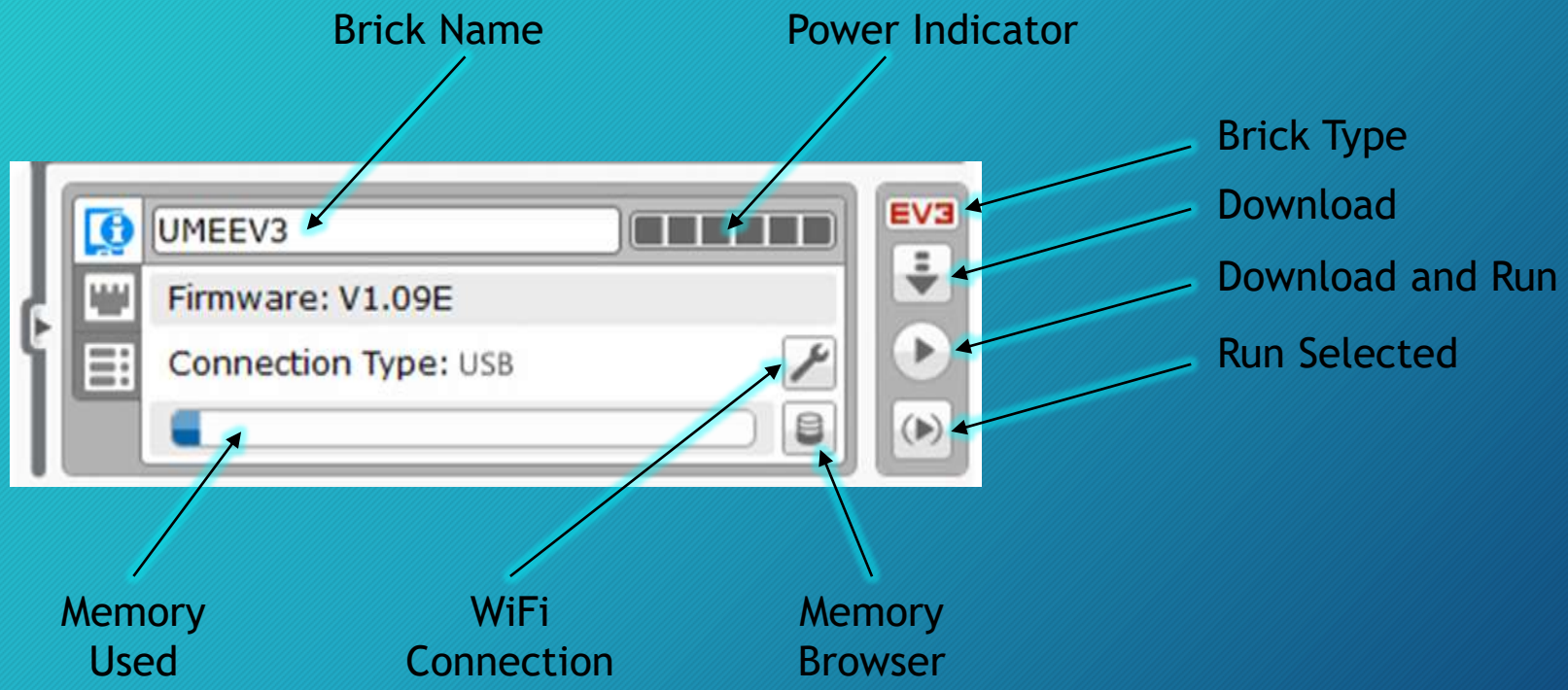
Unregulated Motor

Invert Motor

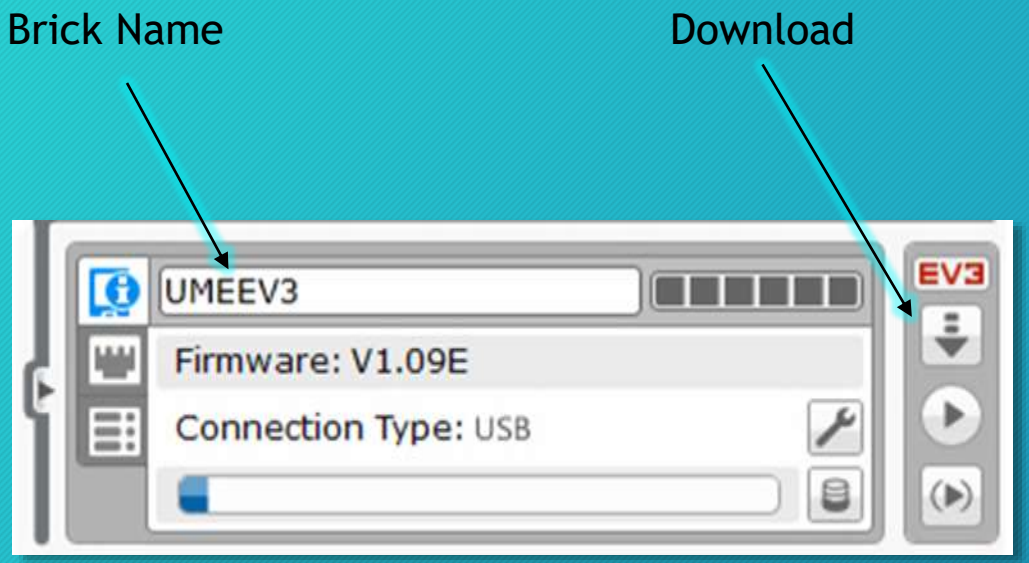
Stop Program

Comment

EV3 Brick Information – Lower right

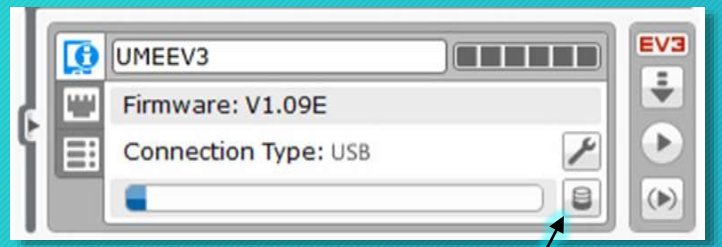


Name your robot



1. Connect robot with USB cable to the computer
2. Select Brick Name entry box
3. Using keyboard, enter the name of your robot
4. Select the Download button
5. View name on the EV3 screen

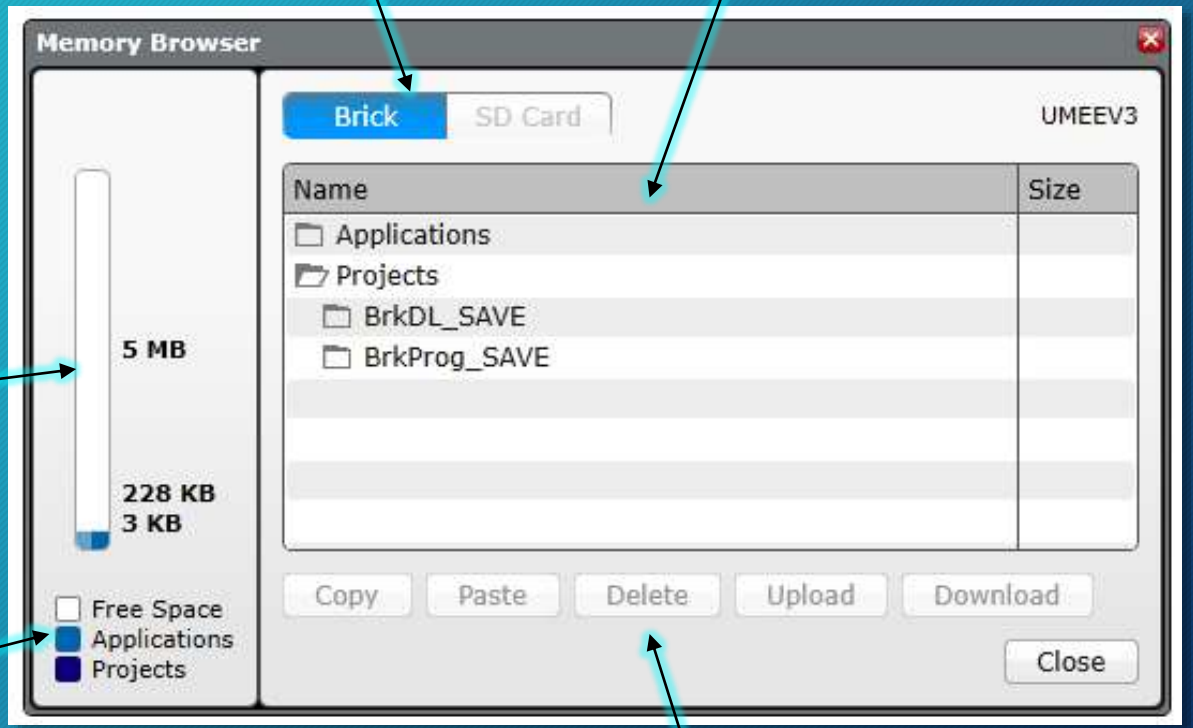
Memory Browser Dialog



Memory Browser

Brick/SD Card tabs

Application and Project files

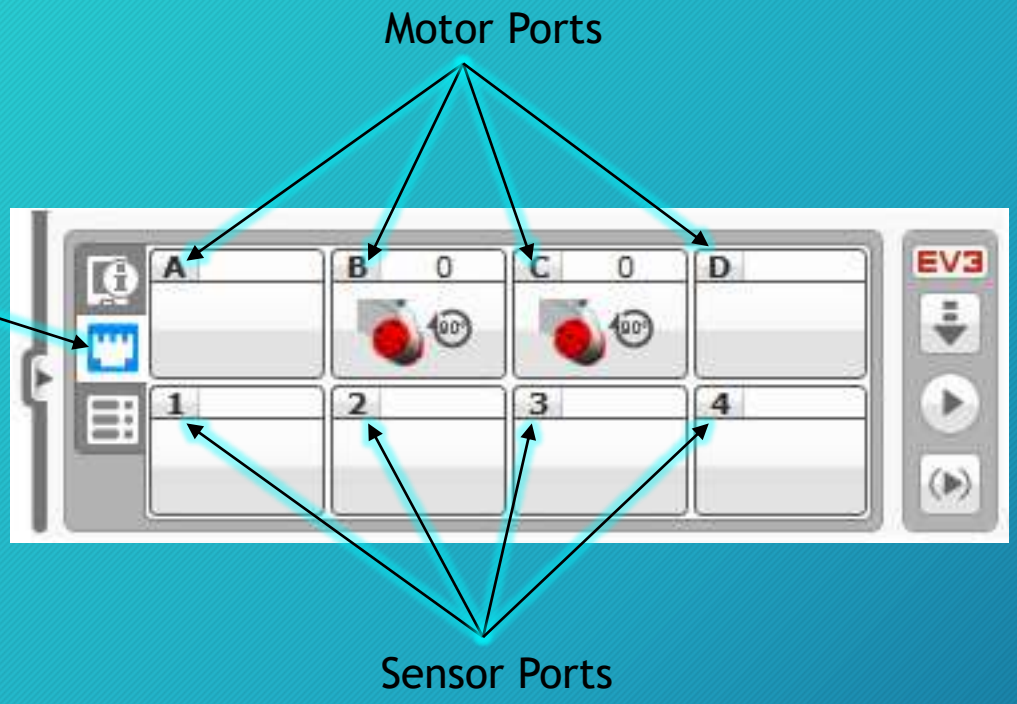


Memory Bar

Key

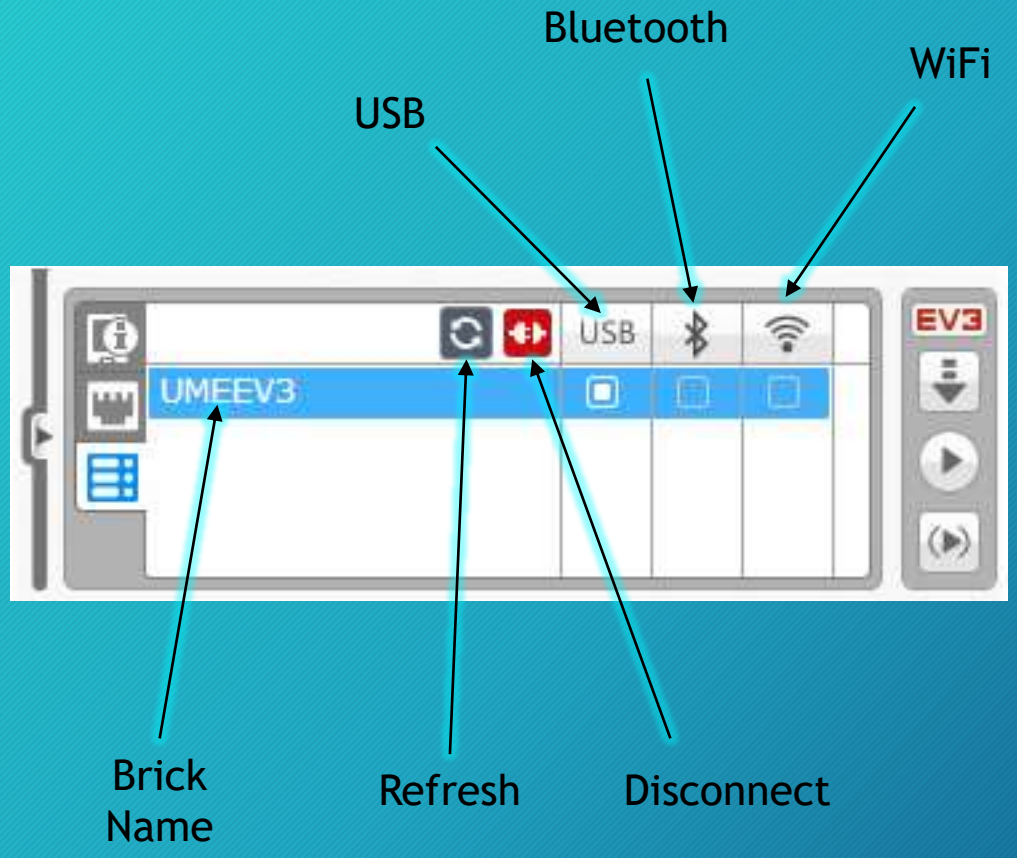
Action Buttons

Port View



- Displays motors and sensor current connected to the EV3 Brick
- Displays current reading for motors and sensors

Available Bricks



Project Properties

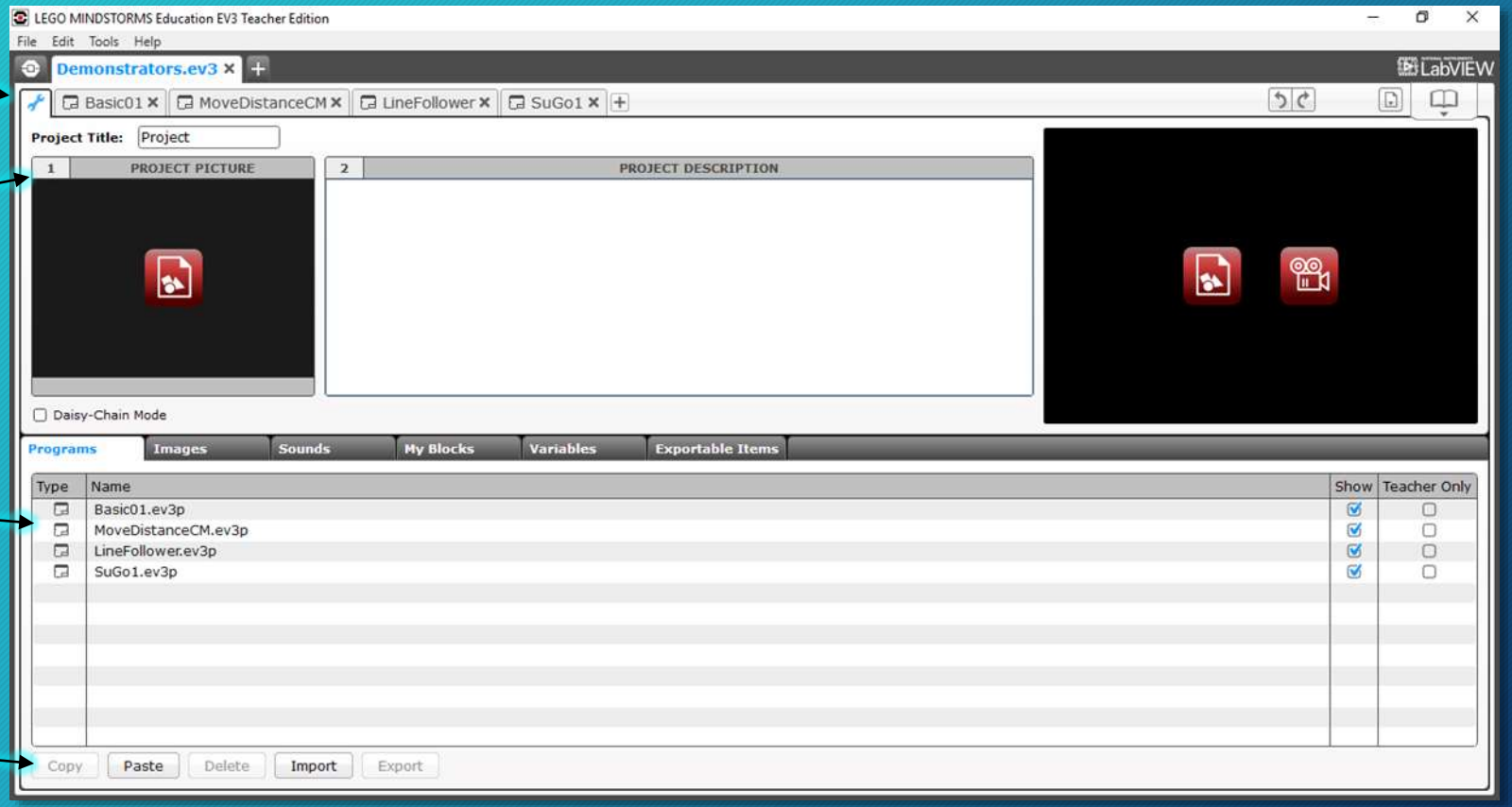
Program Tabs

Project Picture

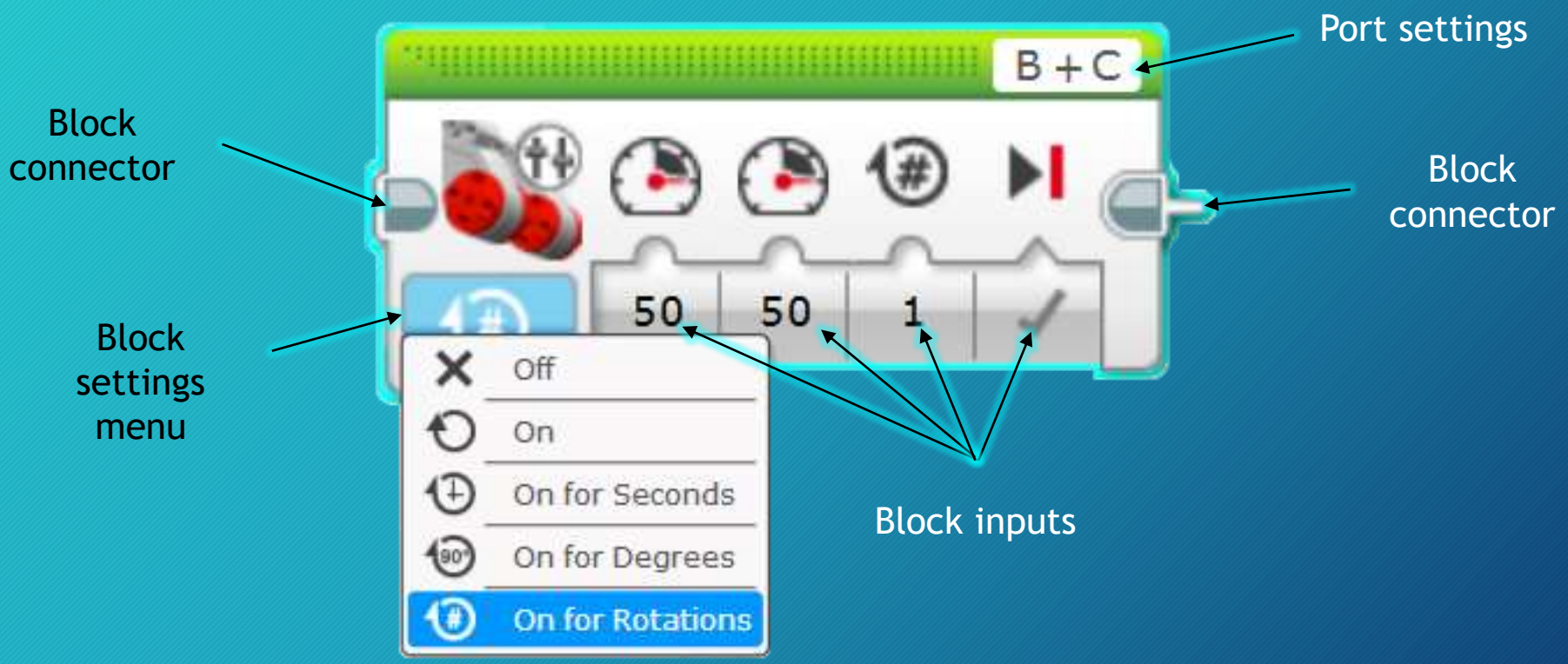
Tabs

Program Type and Name

Action Buttons



Block settings



Moving forward a set distance

To program the robot to move a set distance in inches or centimeters:

- You need to know circumference of the wheels that are powered by the motors to move the robot.

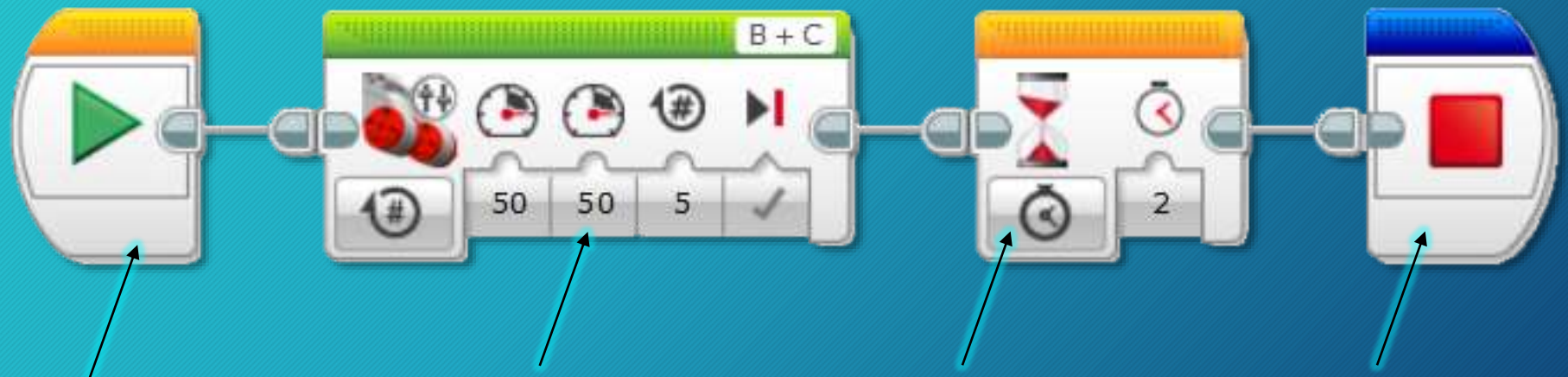
- There are several ways to determine the circumference:
  - Calculate from what is printed on the wheel
  - Measure wheel diameter
  - Use robot to determine circumference (best!)

Naming the program

- Select Program tab
- Using the keyboard, enter program name: Move5
- Select Save

Calculate wheel circumference with the robot - Program

- Create a program that moves the wheels five (5) rotations forward, and waits for 2 seconds.



Calculate the circumference using the robot - Setup and test

1. Use a reference on the robot to mark starting point
2. Run the Move5 program
3. Using a tape measure, measure distance traveled by robot in centimeters
4. Record distance
5. Repeat steps 1-4 five times

TIP: Use centimeters for measuring units.

Calculating wheel circumference

Add the five distances traveled together:

87.3 + 87.6 + 87.9 + 87.0 +88.2 = 438

Average distance traveled:

438 ÷ 5 = 87.6

(Total of the test runs ÷ Number of test runs = Average of test runs)

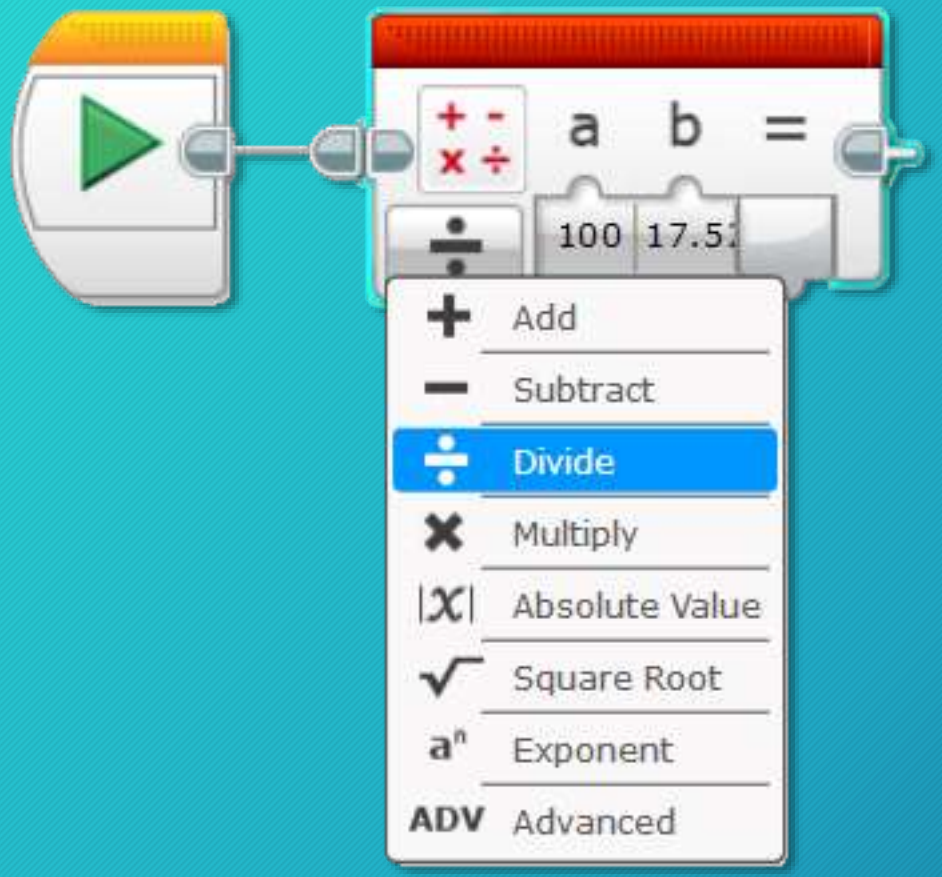
Calculate the circumference using the robot

Distance traveled ÷ Motor rotations = Wheel circumference

OR

87.6 cm ÷ 5 = 17.52 cm

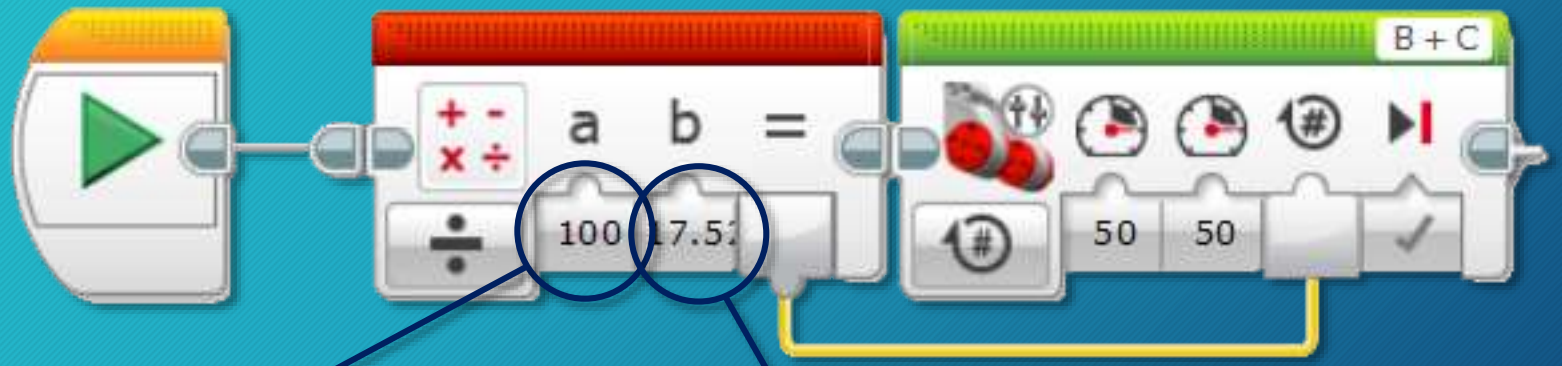
Move forward a specified distance



1. Add a new program and name it “MoveStraight”
2. Add a Math block to calculate rotations
3. Change the math function to division ÷
4. Enter the value for the distance to move in input **a**
5. Enter the calculated circumference value in input **b**

Move forward a specified distance

6. Add a Move Tank block (check motors)
7. Wire output (=) of Math block to Rotations input on Move Tank Block
8. Save Project



The math: distance to move (cm) ÷ wheel circumference (cm) = number of rotations

Move forward a specified distance

9. Test program to verify it works
10. Adjust circumference value if distances are off

What would happen if we put larger wheels on the robot?

a) It would move less than 100 cm
b) It would move further than 100 cm
c) No change, it would move 100 cm

Moving consistency is goal more so than exact distance.

Connecting with Bluetooth

- Bluetooth allows you to download programs without the USB cable. This is a great time saver in practices.
- Bluetooth connection also allows you to see sensor data real-time in the Port view, and see what block is executing in your program. These can be great troubleshooting tools

Connecting with Bluetooth

Naming your brick will help prevent others nearby from selecting your brick and accidently deleting or downloading programs to yours.

1. Enable Bluetooth on the computer.
2. Power on the EV3 brick.
3. In the EV3 interface, under settings, select Bluetooth.

Connecting with Bluetooth

4. Enable Bluetooth and Visibility. Then, select the large check mark at bottom.

Connecting with Bluetooth

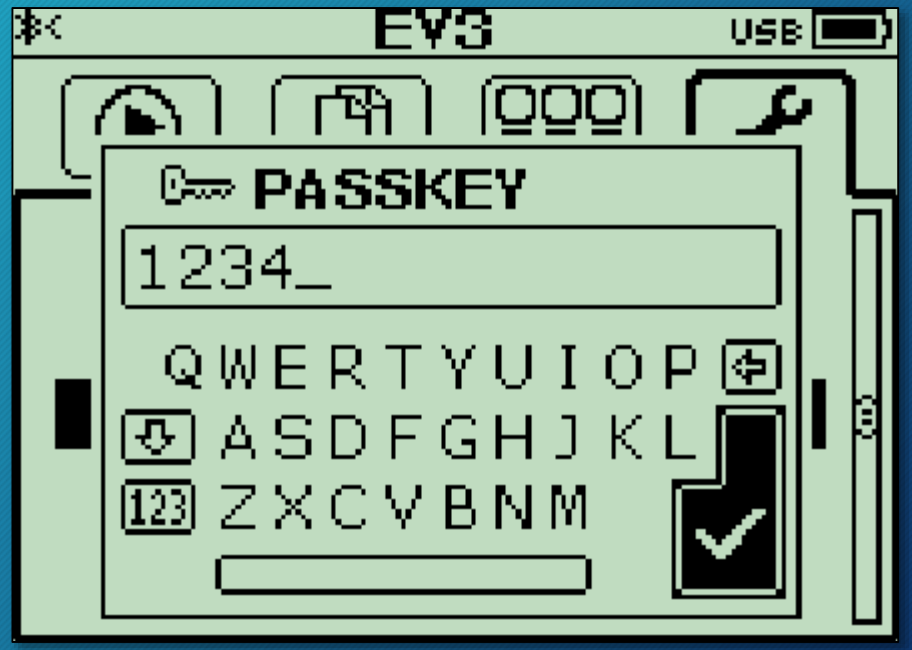
4. Confirm that Bluetooth is enabled.
5. On computer, pair the EV3 brick and computer.

Connecting with Bluetooth

7. On the EV3 interface, when you see Connect? dialog, select the check mark.

Connecting with Bluetooth

8. On the EV3 interface, read the PASSKEY value (default: 1234) and select the check mark. Ignore the underscore character at the end of the passkey value.
9. On computer enter passcode.

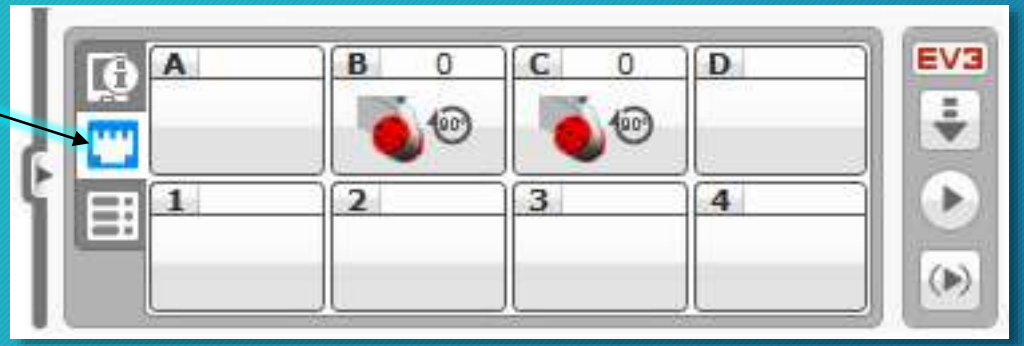


Bluetooth must be off at FLL® tournaments.

Using Port View

- Open Port View
- Run the move distance program while observing the motors in Port View happens?
- Run the program a second time observing the program blocks. What happens?

Port View



You can observe a lot just by watching - Yogi Berra

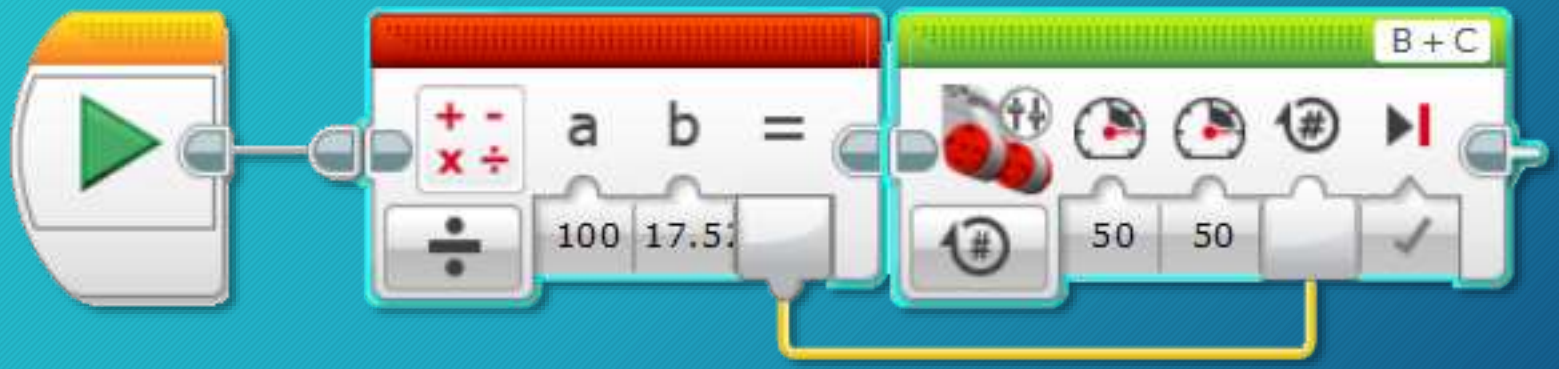
My Blocks

What are My Blocks?

My Blocks allow programmers to group multiple programming blocks in a specific order into a single block for easy reuse.

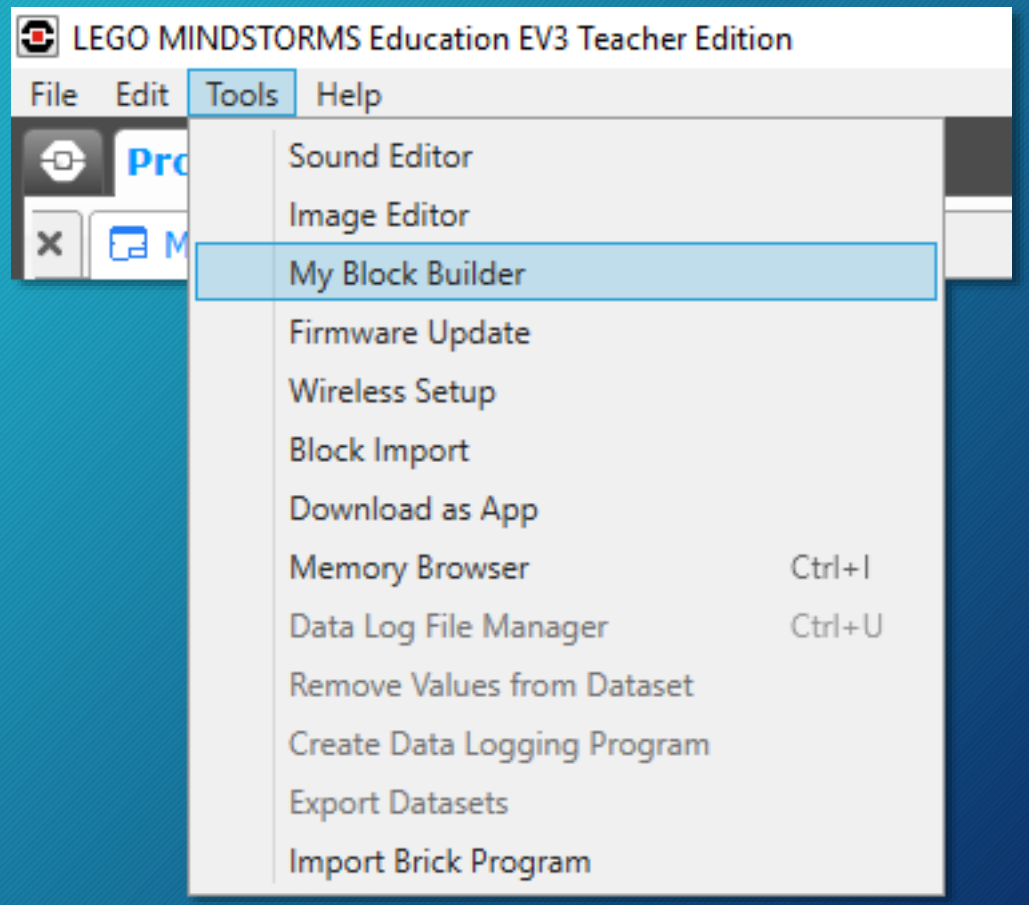
How to make a My Block

- Select the blocks that will make up the my block, blocks are highlighted with blue border. Do not highlight the Start block.



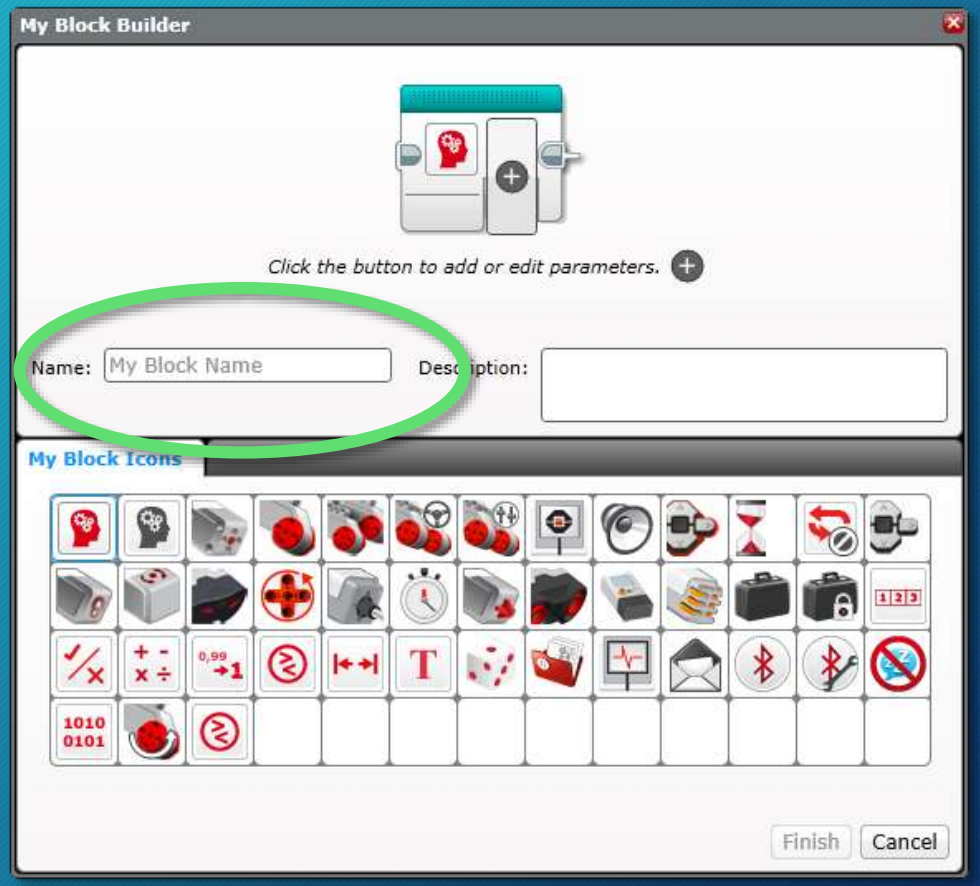
Make a My Block – My Block Builder

1. From dropdown menus select Tools ▶ My Block Builder
2. My Block dialog opens



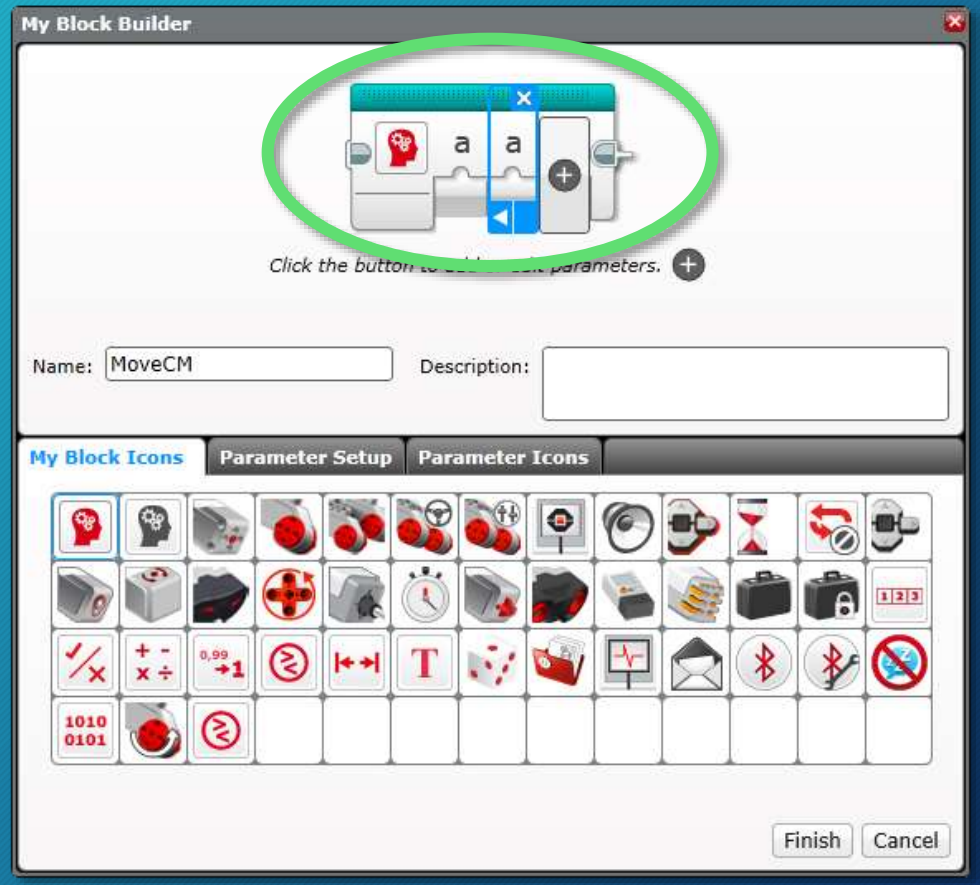
Make a My Block – Name My Block

3. Using the keyboard, enter **MoveCM** for the My Block Name



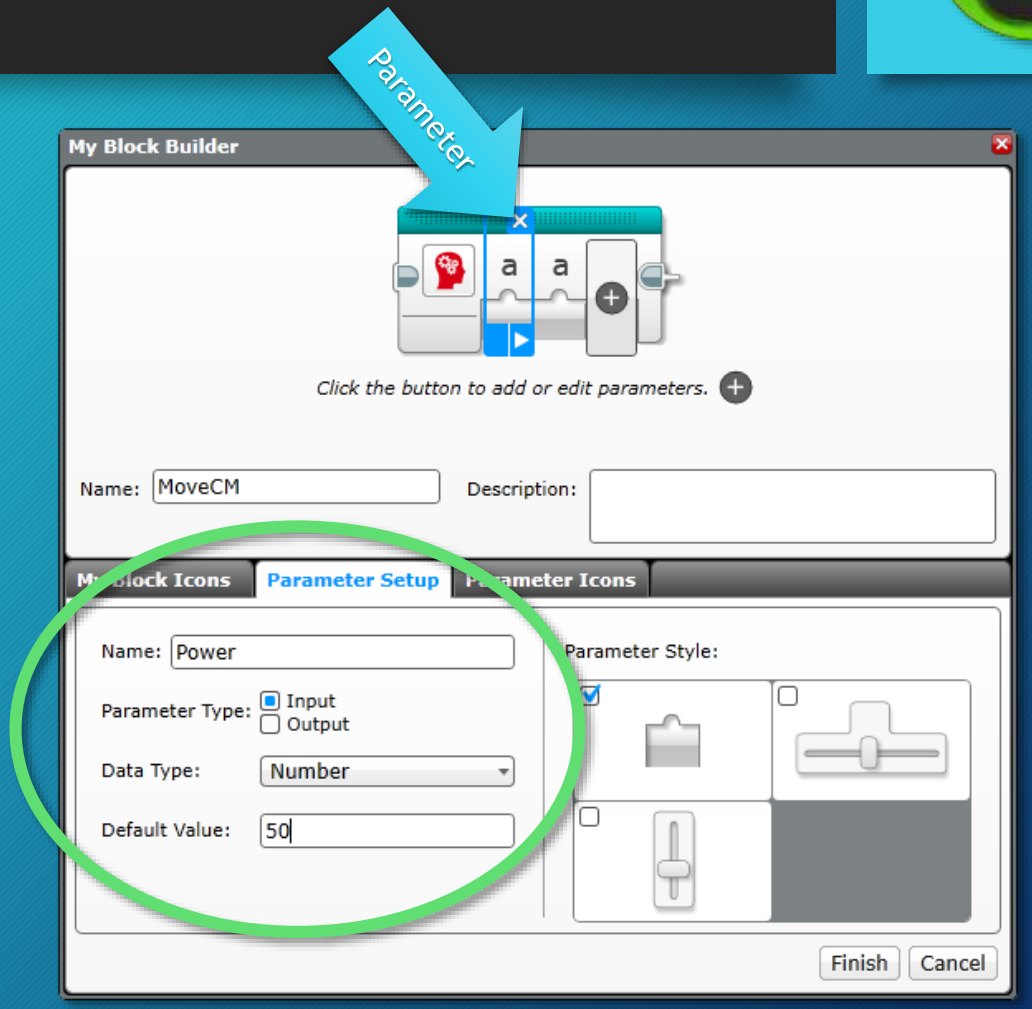
Make a My Block – Add Parameters

4. Select the + twice to add two Parameters



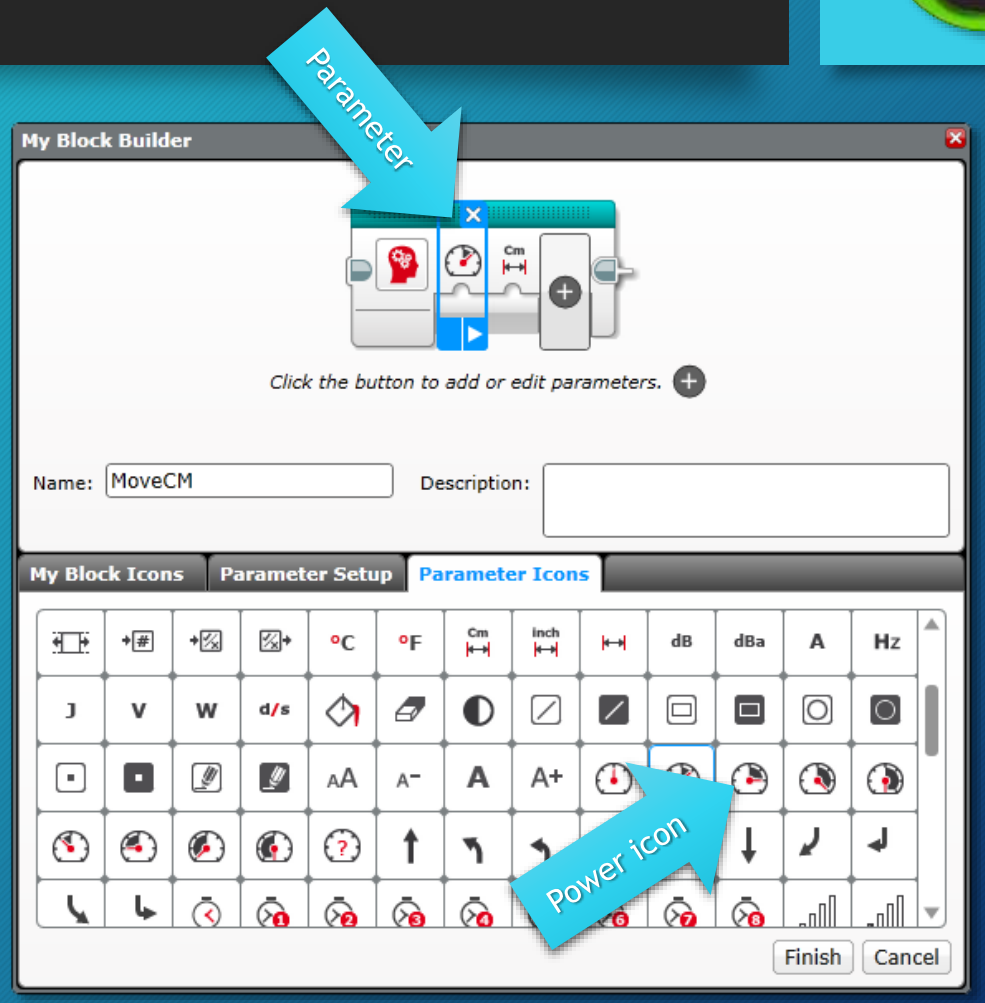
Make a My Block – First Parameter setup

5. Select the first parameter
6. Select the Parameters Setup tab
7. Using the keyboard, enter a parameter Name (Power)
8. Leave the Parameter Type as Input
9. Leave the Data Type as Number
10. Enter Default Value (50)



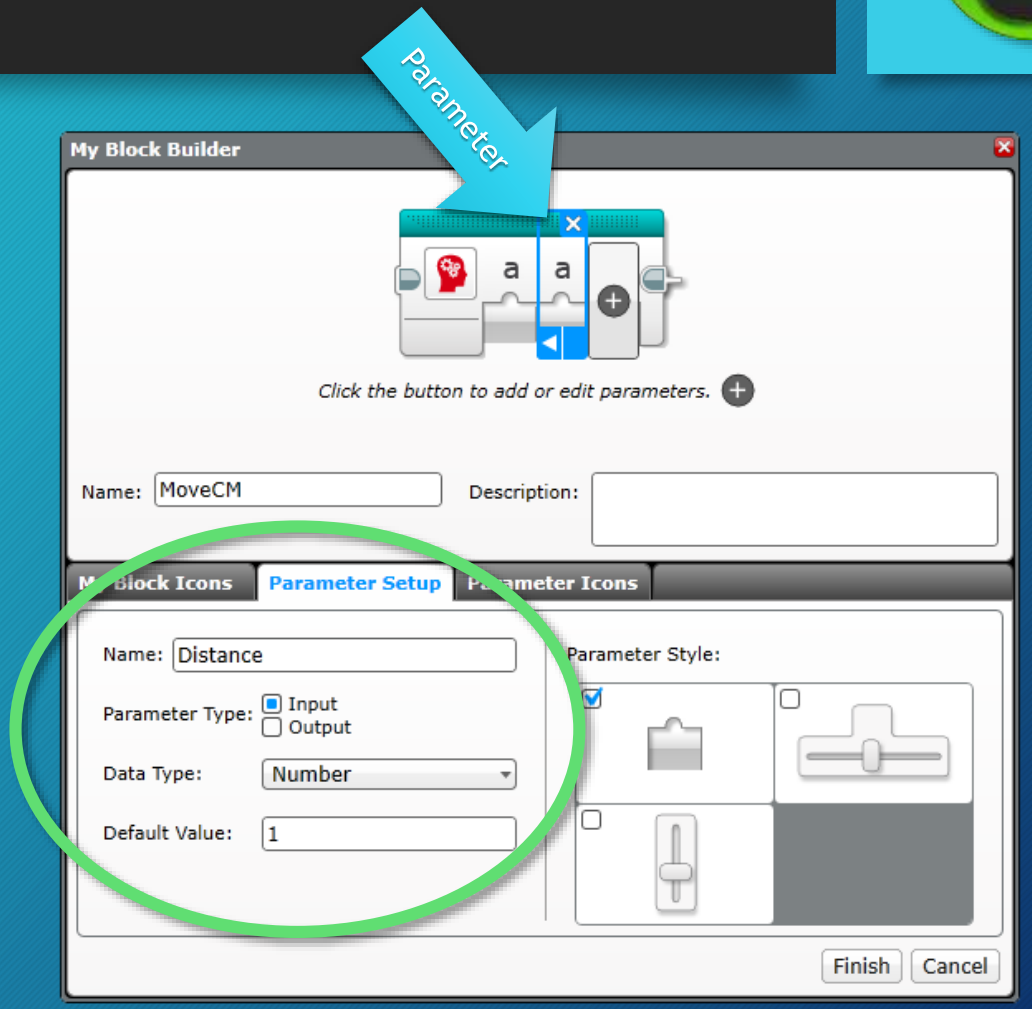
Make a My Block – Select icon

11. Select the Parameter Icons tab
12. Scroll down to the power icons
13. Select power icon



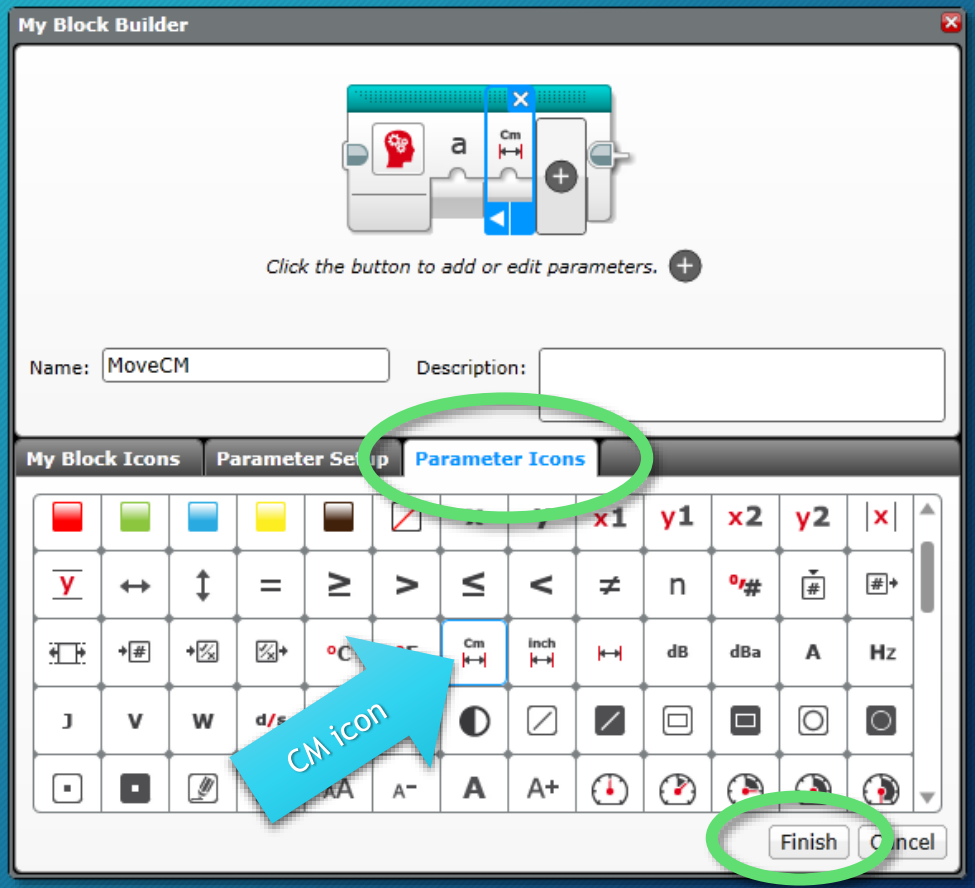
Make a My Block – Second Parameter setup

14. Select the second parameter
15. Select the Parameters Setup tab
16. Using the keyboard, enter a parameter Name (Distance)
17. Leave the Parameter Type as Input
18. Leave the Data Type as Number
19. Enter Default Value (1)



Make a My Block – Select icons

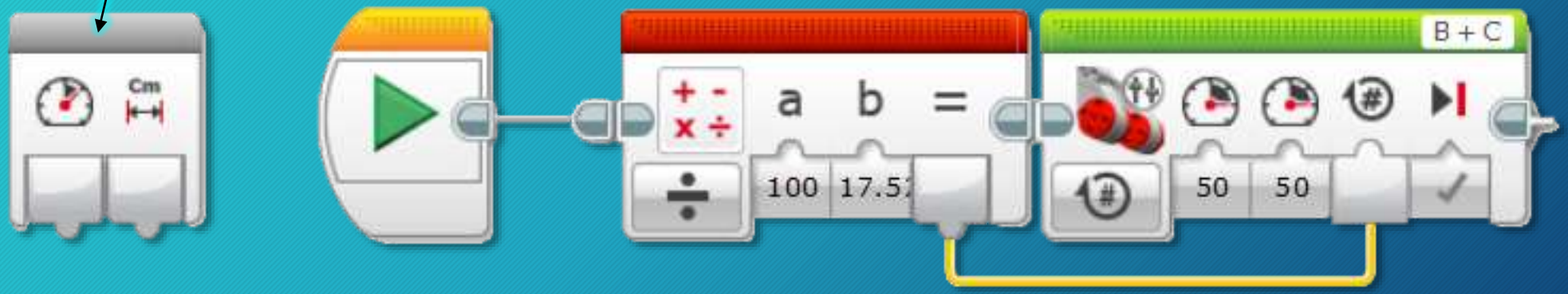
20. Select the Parameter Icons tab
21. Select CM icon 
22. Select Finish button
23. Programming screen displays with an additional block



Make a My Block – Programming screen

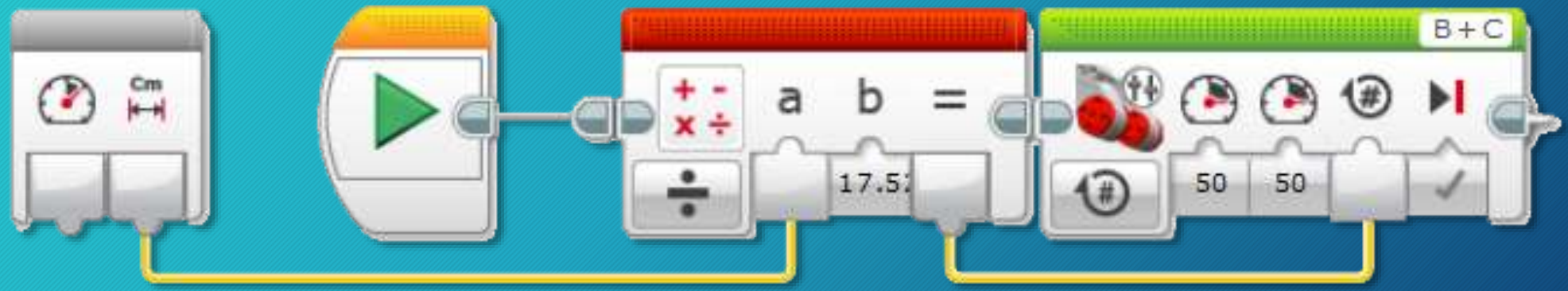
Parameter

Block inputs



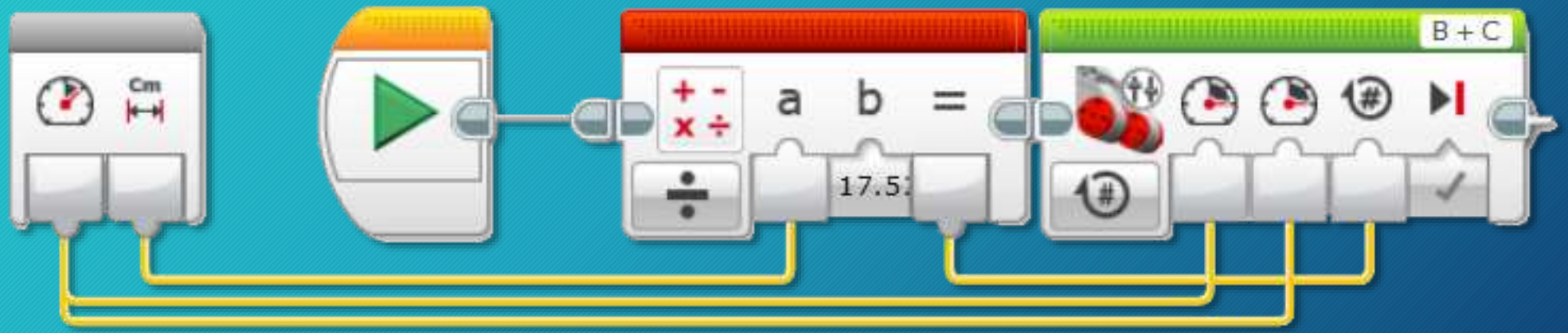
Make a My Block – Connecting parameters

- Drag a wire from the distance parameter Cm to the a input on the Math block



Make a My Block –

- Drag a wire from the power parameter to the B motor input on the Move Tank block
- Drag a second wire from the power parameter to the C motor input on the Move Tank block.

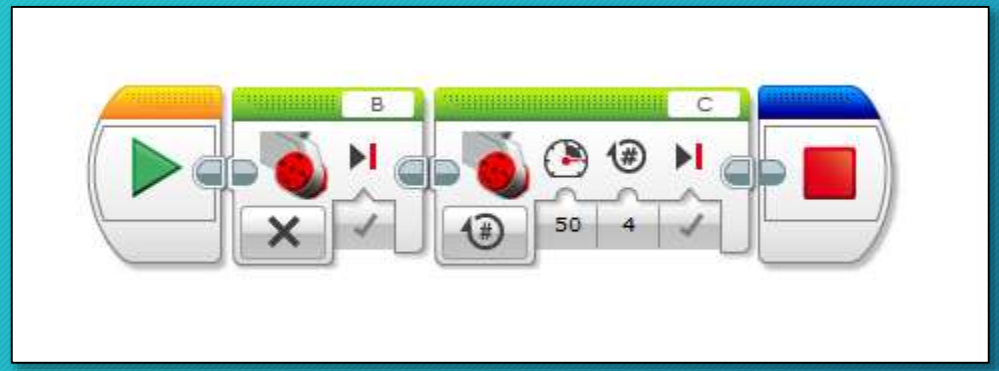


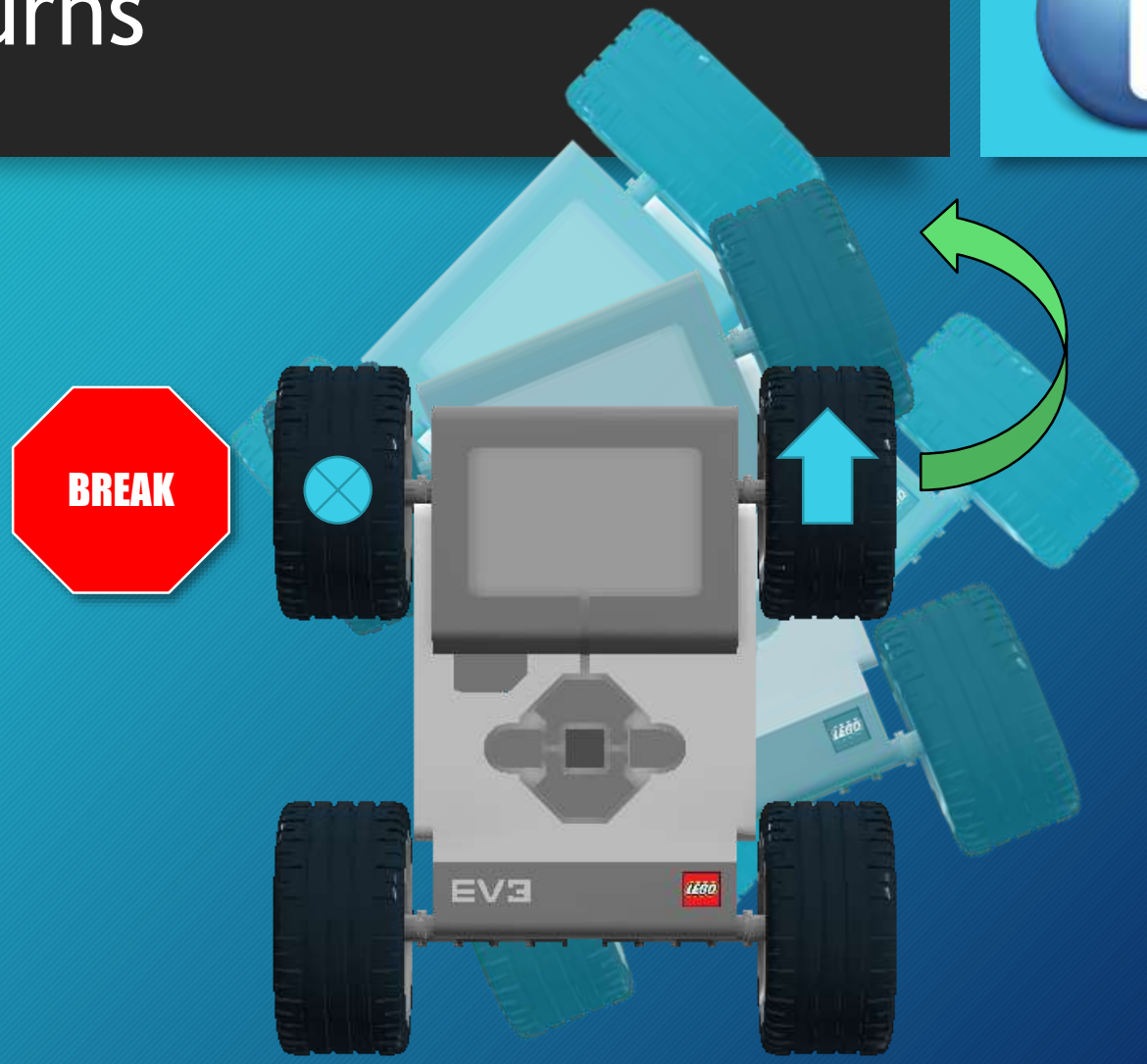
Make a My Block – Test

1. Insert MoveCM My Block with power at 50 and distance at 50
2. Insert a Wait block set to 3 seconds
3. Insert MoveCM My Block with power at 25 and distance at 25
4. Insert a Wait block set to 3 seconds
5. Insert MoveCM My Block with power at 100 and distance at 100
6. Insert a Wait block set to 1 seconds, end with a Stop block

Fundamentals of pivot turns

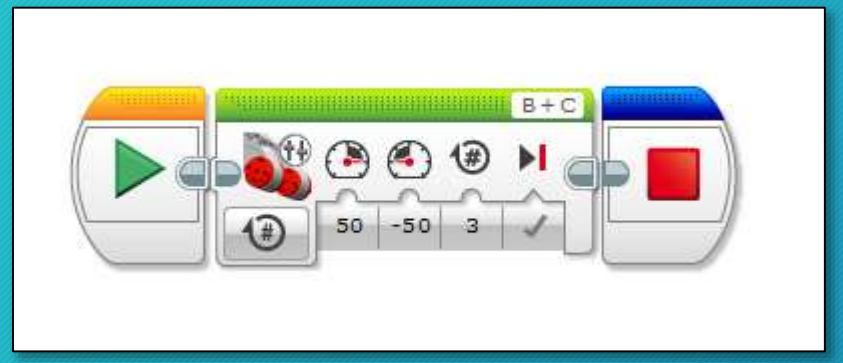
- Powering one wheel and breaking the other will rotate the robot around the mid-point of the breaked wheel.





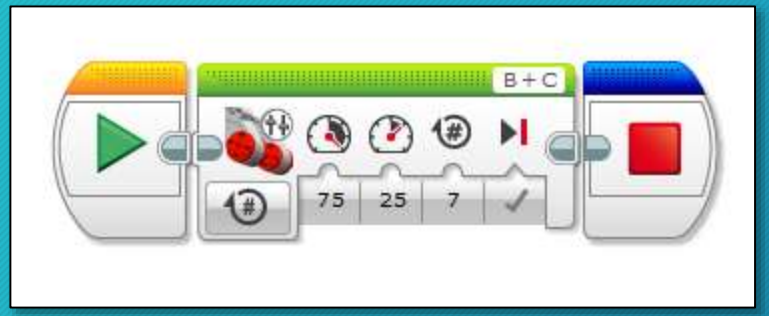
Fundamentals of pinpoint turns

- Powering the wheels in opposite directions will rotate the robot around the mid-point of the axle.



Fundamentals of arc turns

- The robot turns when driving wheels move at different speeds
- The robot turns towards the slower wheel
- The greater the difference in speeds, the tighter the turn



Challenge

Build and test a program that has your robot navigate in a 60 cm square ending on the spot it started.

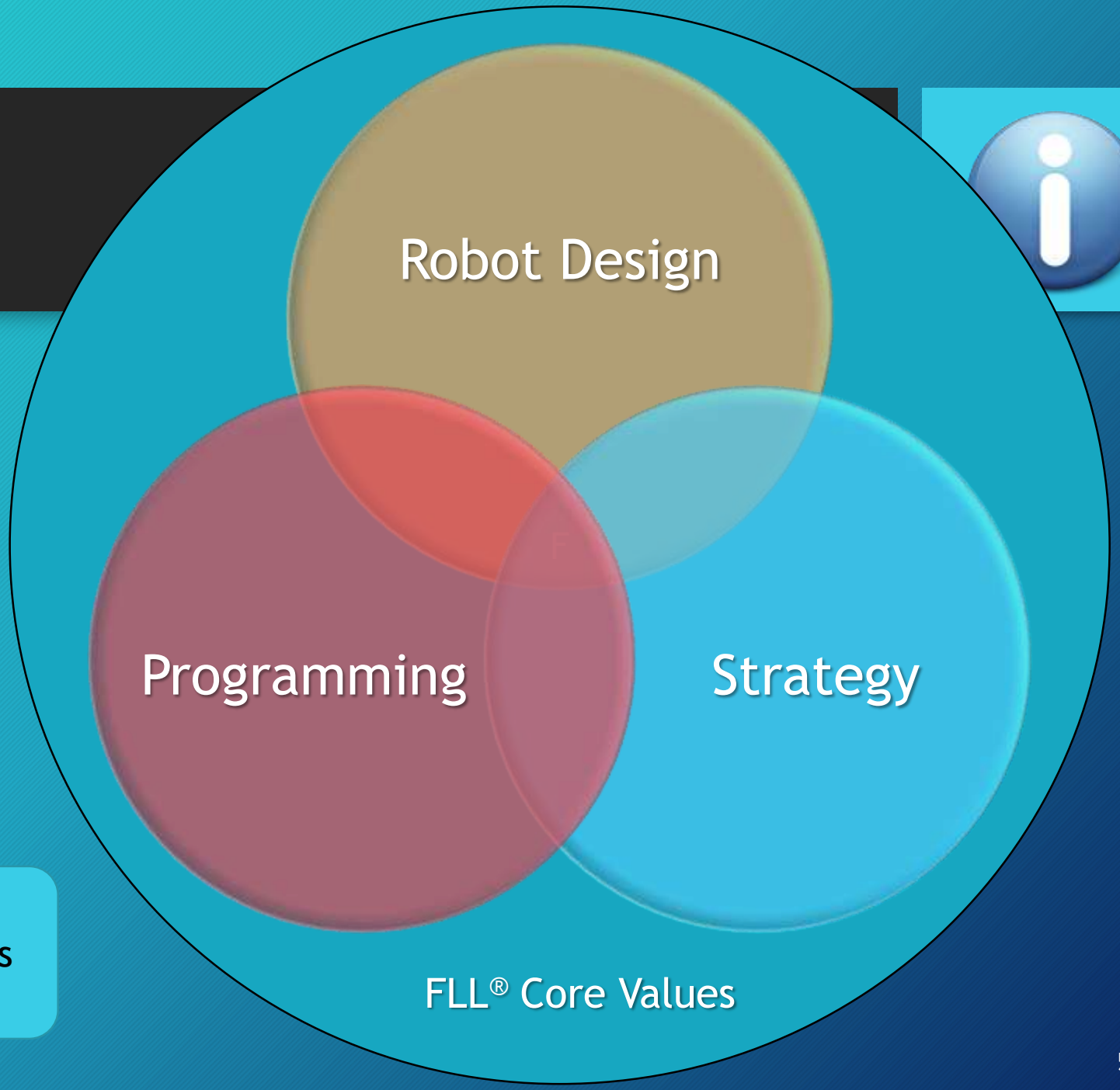
Use the programming workflow chart to document what the robot needs to do, then program.

Achieving Top Robot Performance

Teams that are consistently top performers are always balanced in robot design, programming, and strategy.

Teams will naturally be stronger in one area, but to neglect the other areas will result in inconstant results.

Robot Design, Robot Challenge and Project are what we do, Core Values is how we do them.



Strategy

- Good strategy with good programming are essential to consistently good robot performance
- Programming helps overcome the limitations of the hardware

Great robot + poor strategy + poor programming = inconsistent scores

Fair robot + good strategy + good programming = consistent scores

Robot Game Strategy - Base

- The robot must always start from base
- Base is the only place where technicians (drivers) can handle and make changes to the robot

Robot Game Strategy - Time

- Matches are 2:30 minutes or 150 seconds
- When the Robot is in base, it's not scoring
  - Minimize time spent in base
- Travel on the field takes time
  - Minimize time spent moving from place to place
  - Solve multiple missions in the same region on the same robot run

Robot Game Strategy - Reliability

- Distance: Errors increases with distance
- Missions that are close become easier
- Missions that are far become harder
  - Use field elements (lines, walls, models) to guide the robot to make things seem “close”

Robot game strategy - Humans

- The Robot does exactly what physics and programming tell it to do
- Humans (drivers) make mistakes and are inconsistent
- Design the robot and strategy to avoid human mistakes and reduce time in base

Republic of Pi's design mantra

- Whenever the robot or humans make a mistake in scoring, redesign the robot/programming so that mistake cannot happen again.

Tip: Start every mission from same spot

- Put solid edges/guides on robot so that you can align robot with solid edges, not by sight-aiming
- Always start robot at same location and facing the same direction
  - Makes for faster setup in base between mission runs
  - Less chance for errors

Strategy - Navigation

A key to scoring is to consistently navigate (move) the robot into position to accomplish the mission

- What a program(mer) needs to know to navigate:
  - Where the robot currently is
  - How precisely you know where it is
  - Where the robot needs to go
  - Is there anything in the way
  - What can help guide you there

The robot must navigate (move) consistently to score consistently.

Navigation & Odometry

What is Navigation? Examples:

- Follow heading for rotations
- Follow heading until black

What is odometry?

- Using distances and turn angles for navigation is called “odometry”
- It's useful, but consistency depends on the quality of robot components
- Mindstorms® robots can have a lot of odometry error

Sources of odometry error

- Friction / Lack of Friction (Wheel slippage)
- Gear slack
- LEGO motors have 5°-15° degrees of gear play
- Battery charge
- Timing issues
- Gyro drift (LEGO gyro can have +/- 3° of error)

Small angles lead to large offsets

Suppose a robot travels 100 centimeters, but its heading is "off" by 1 degree

Q: How far off will it be after 100 cm?
A: 1.74cm

The FLL® field is 8 feet, or 243.84 cm, so at 8 feet you would be 4.23 cm off or 1.66 inches off

Overcoming error strategy

- Use field elements for navigation such as lines, walls, and mission models
- If your robot can find a line, wall, model, or something on the other side of the field, you accurately know its location.
- One rule of thumb is to never make more than two turns without re-orienting the robot using something on the field.

Session 2

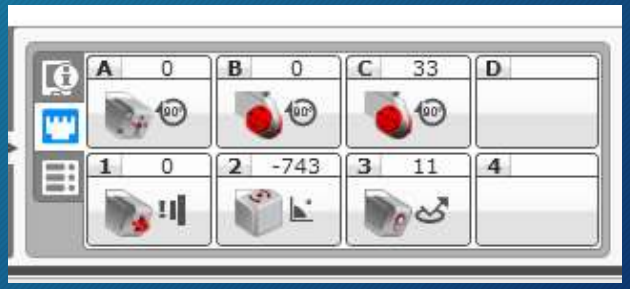
1. Using sensors to help navigate
2. Working with loops
3. Build a gyro following loop
4. Build a line follower
5. Sequence Programming

Using sensors to improve navigation

- Gyro Sensor
  - Can help the robot move in a straight line
  - Can help control how far the robot rotates (turns)
- Color sensor
  - Can detect colors on the mat
  - Can detect and follow lines on the mat

Gyro sensor

- Gyro sensor detects rotation around an axis
- The axis for the LEGO® gyro is indicated by the point (dot) and arrows on the sensor.
- The gyro sometimes shows movement even when it is still, this is called bias and drift, and must be corrected before using the gyro.

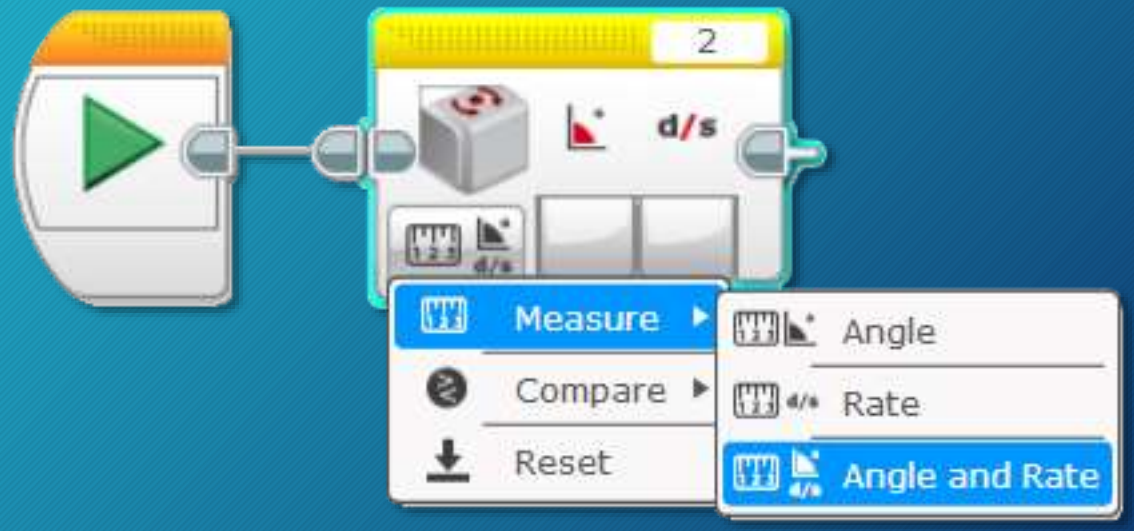


Port View

Reducing gyro drift program

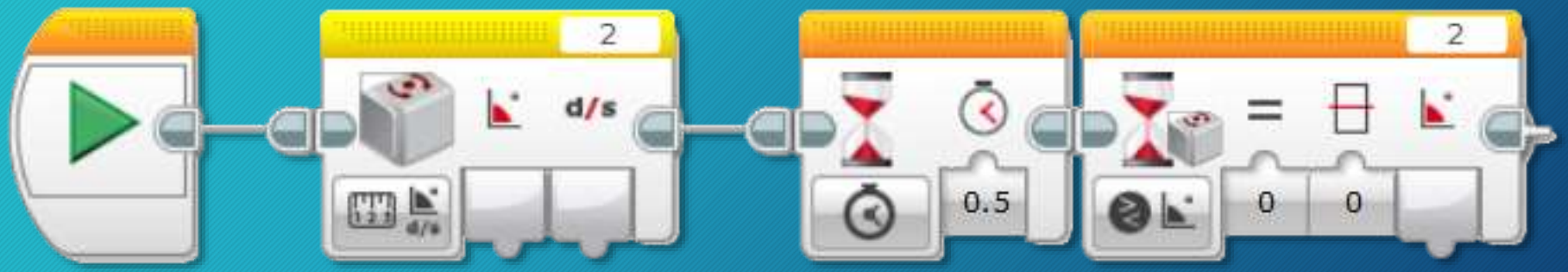
The following block sequence recalibrates the gyro sensor to stop drift:

1. Open new program. Name it “GyroCal”
2. Add Gyro Sensor block
3. Set Gyro Senor block to Measure ▶ Angle and Rate



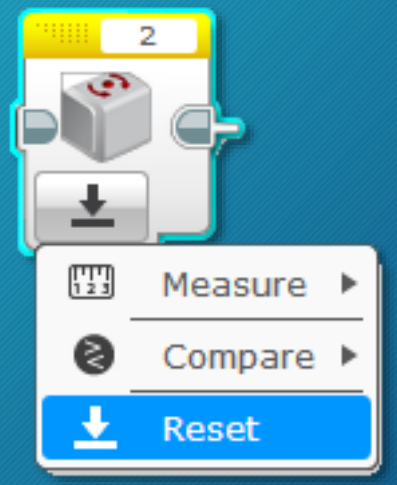
Reducing gyro drift program

4. Add Wait block, set Wait block for .5 seconds
5. Add a second Wait block
6. Set Wait block to Gyro Sensor ▶ Compare ▶ Angle
7. Set Compare Type to 0 (=) and Threshold Value to 0



Reducing gyro drift program

- Only need to perform this once at beginning of program/robot run
- Requires 2-3 seconds to complete (EV3 flashes)
- Gyro must be stationary while calibrating
- Can be ran prior to beginning of match during setup at the table



≠ Recalibrate gyro

Trip: Gyro block, Reset option doesn't recalibrate the gyro!

Turning using the gyro to turn

1. Open New Program
2. Add Gyro block set to Reset
3. Add a Wait block set to 0.1 seconds
4. Add Large Motor block and set port to B, with input to On, and 25% power
5. Add a second Large Motor block and set port to C with input Off and Brake to True

Turning using the gyro to turn

6. Add a Wait block, set to Gyro ▶ Change ▶ Angle
7. Set the Direction to 2 (Any) and the Amount: 90
8. Add a Large Motor block and set port to B with input Off and Brake to True
9. Add Gyro block set to Reset

Challenge

- Make the gyro turn into a My Block.
- Build and test a program that has your robot navigate in a 75 cm square ending on the spot it started using the gyro turn My Block to make the turns and the MoveStraight My Block.

Loops

What does a Loop block do?

A “loop” block repeats what is contained in it until a set criteria is met.

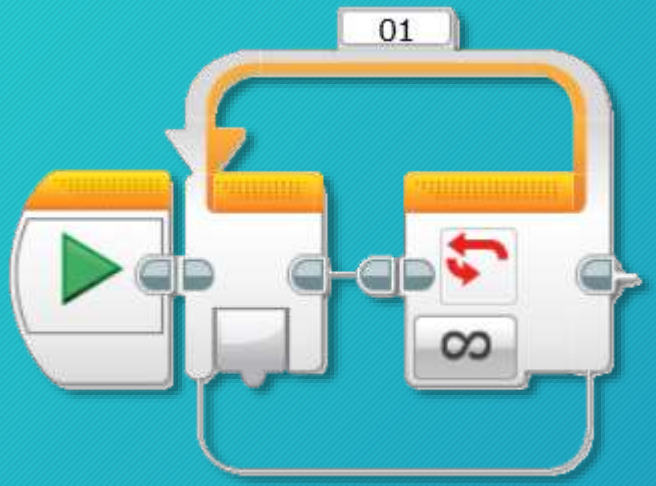
Loops – Basic

A basic Loop block:

What does this Loop Block do?

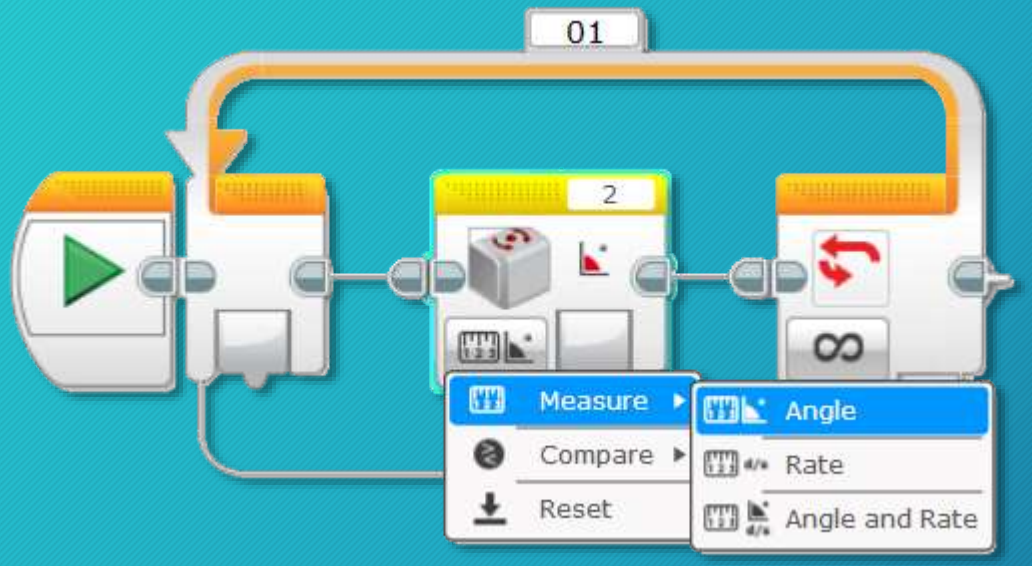
A gyro-following loop - programming

1. Open new program, name it “GyroLoop”
2. Add a Loop block, set to infinite



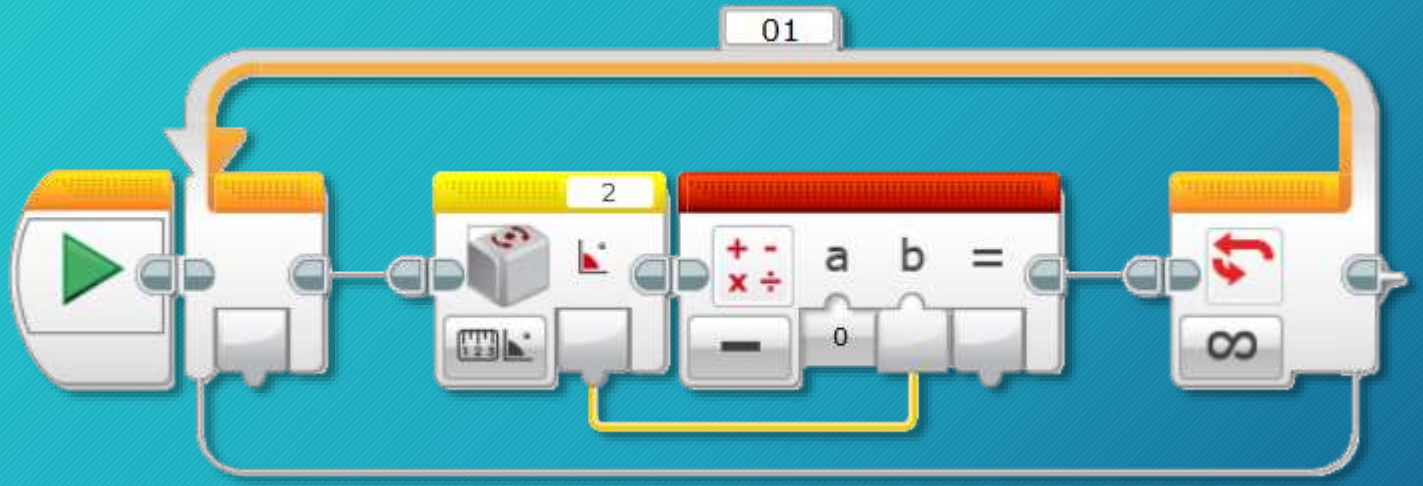
A gyro-following loop - programming

3. Add Gyro Block
4. Set Gyro block to Measure ▶ Angle



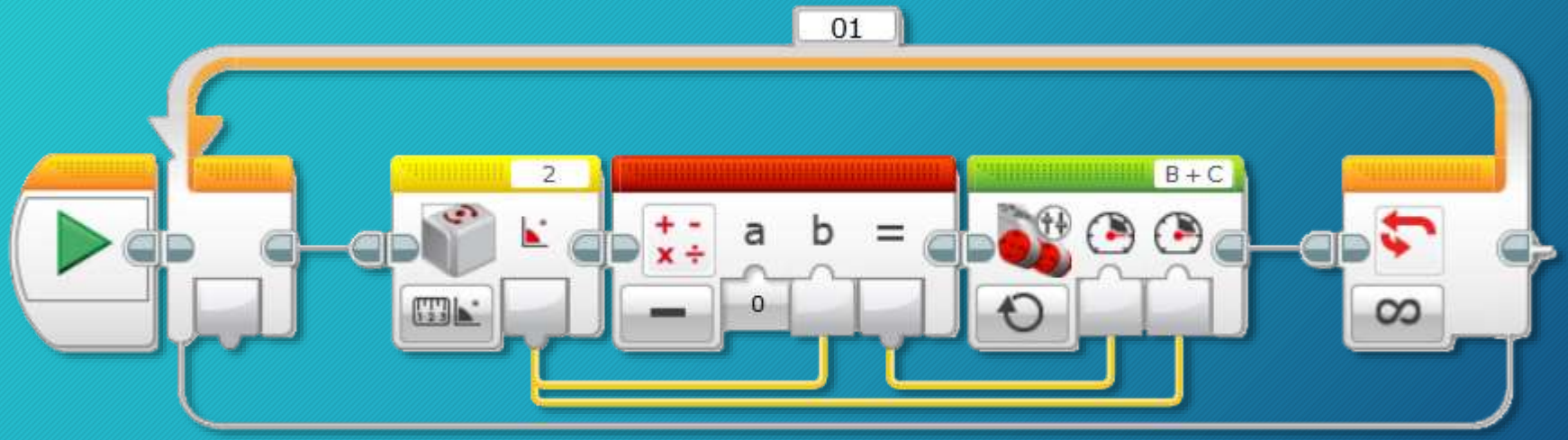
A gyro-following loop - programming

5. Add Math block, set to Subtract and a to 0
6. Drag thread from Gyro to b



A gyro-following loop - programming

7. Add Tank Steering block, set to On, motor power at 0
8. Drag Thread from Math block = to B motor Power
9. Drag Thread from Gyro block to C motor Power



Gyro-following loop test

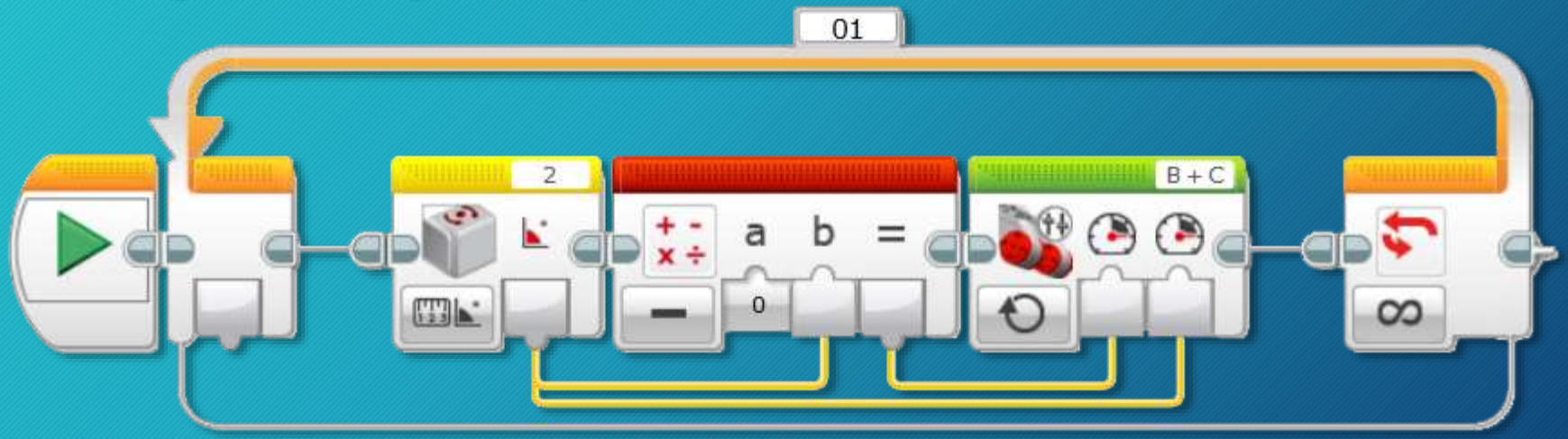
- Run the gyro-following loop, program. What is the robot doing?
- Pick the robot up and turn it slight and set it down. What happened this time?

Troubleshooting

- If the robot is turning slowly, you to run may need the gyro calibration program.
- If the robot spins out of control, try swapping the motors cables and check the ports to make sure they are in B and C.

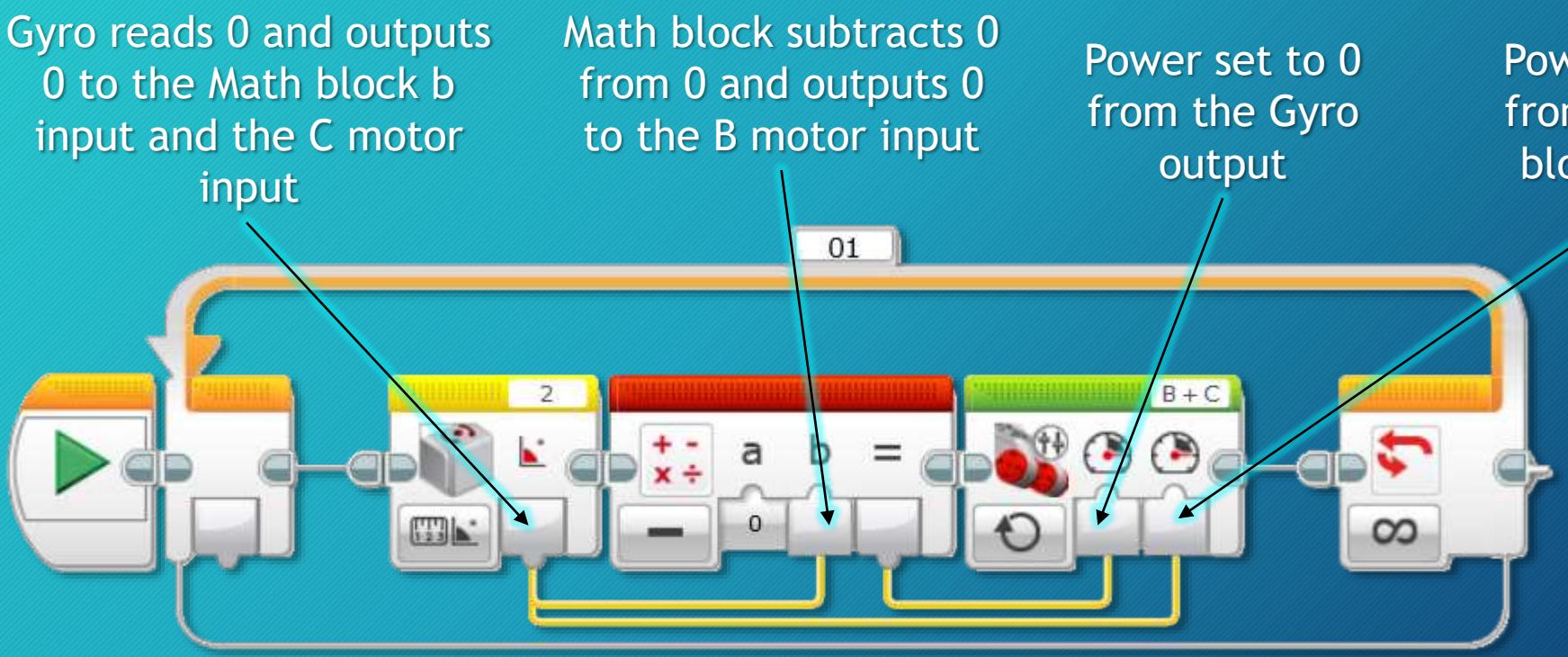
Gyro-following loop explanation

- The Gyro Sensor block reads current angle and outputs the value
- The Math block changes the (+/-) sign of the gyro angle value and provides that to the B motor
- The C motor gets a original value, so one motor get positive power the other gets a negative power



Gyro-following loop explanation

- What happens here when the gyro angle reading **is** zero?



Gyro-following loop explanation

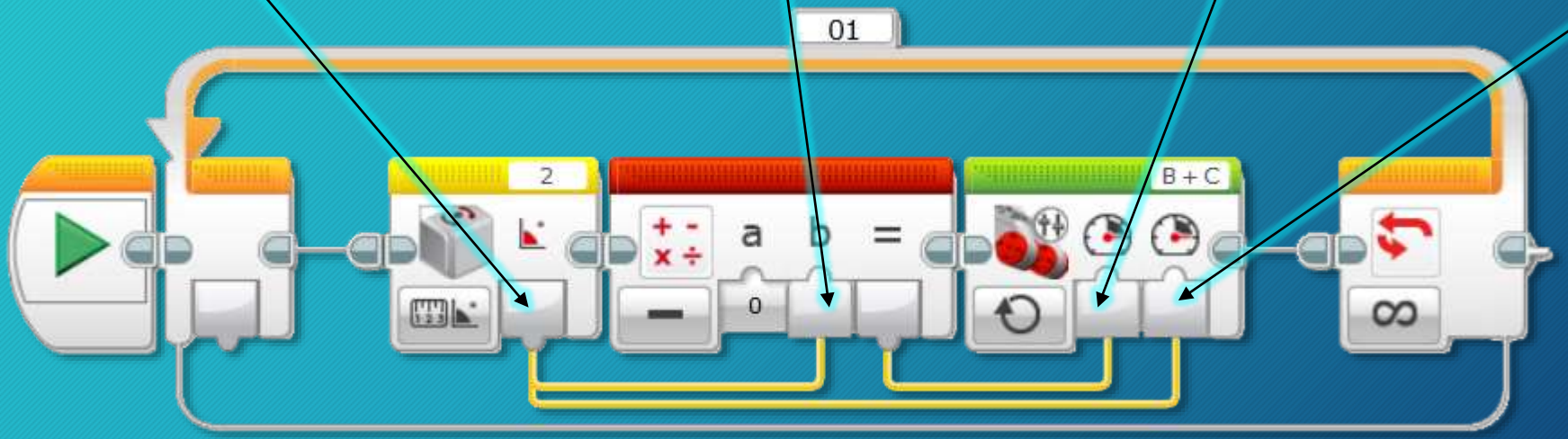
- What happens here when the gyro angle reading **is not** zero?

Gyro reads 2 and output 2 to the Math block b input and the C motor input

Math block subtracts 2 from 0 and outputs -2 to the B motor input

Power set to -2 from the Gyro output

Power set to 2 from the Math block output



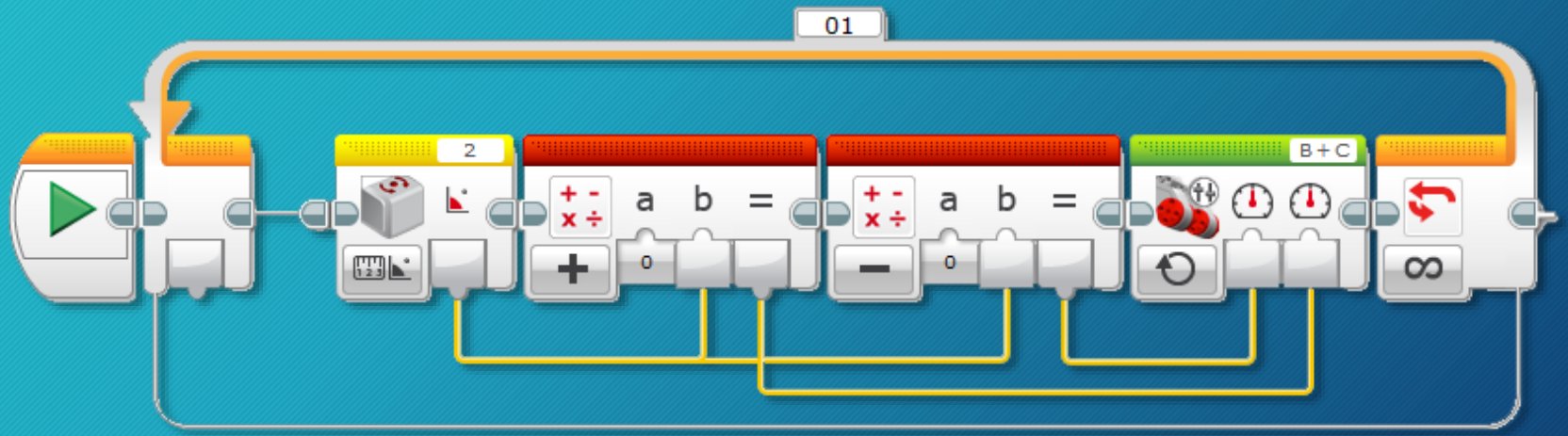
A gyro-following loop is proportional control

- The power to the motors is proportional to how far the gyro sensor is away from zero (the “error”).
- For example:
  - If the gyro detects it is at 2 degrees, it powers the motors at 2% power.
  - If the gyro detects it is at 15 degrees, it powers the motors at 15%

This is cool, but how does it help my robot in ***FLL***®?

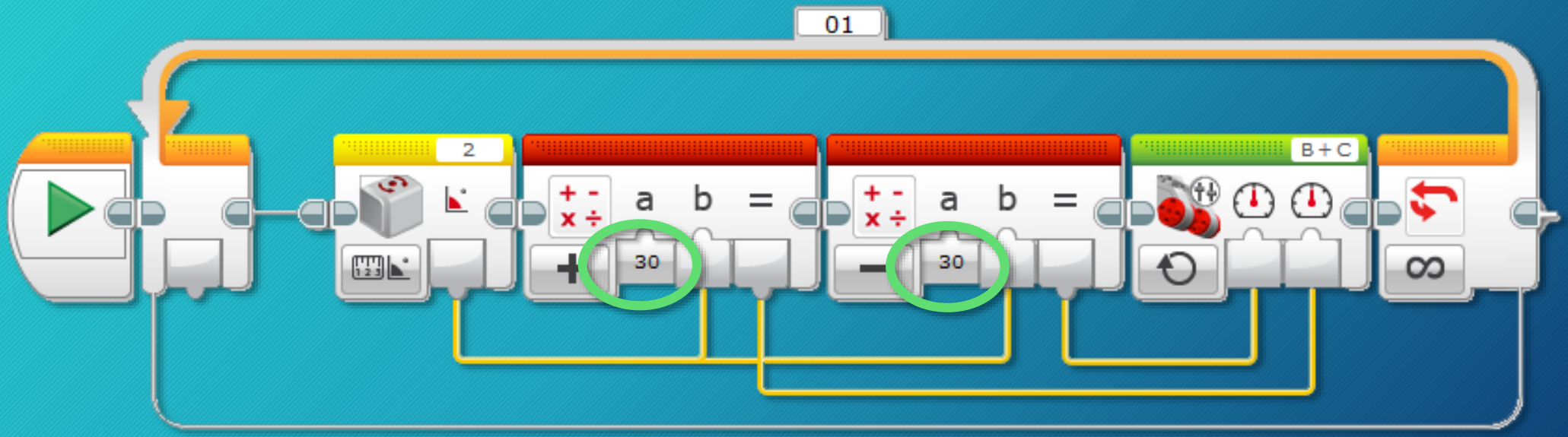
A gyro-following loop - continued

11. Add a math block to the loop set to plus (+)
12. Add a thread from the Gyro sensor block to the b inputs of both Math blocks
13. Add a thread from the first Math block = to the Move Tank block power input
14. Add a thread from the second Math block = to the Move Tank block power input



A gyro-following loop - continued

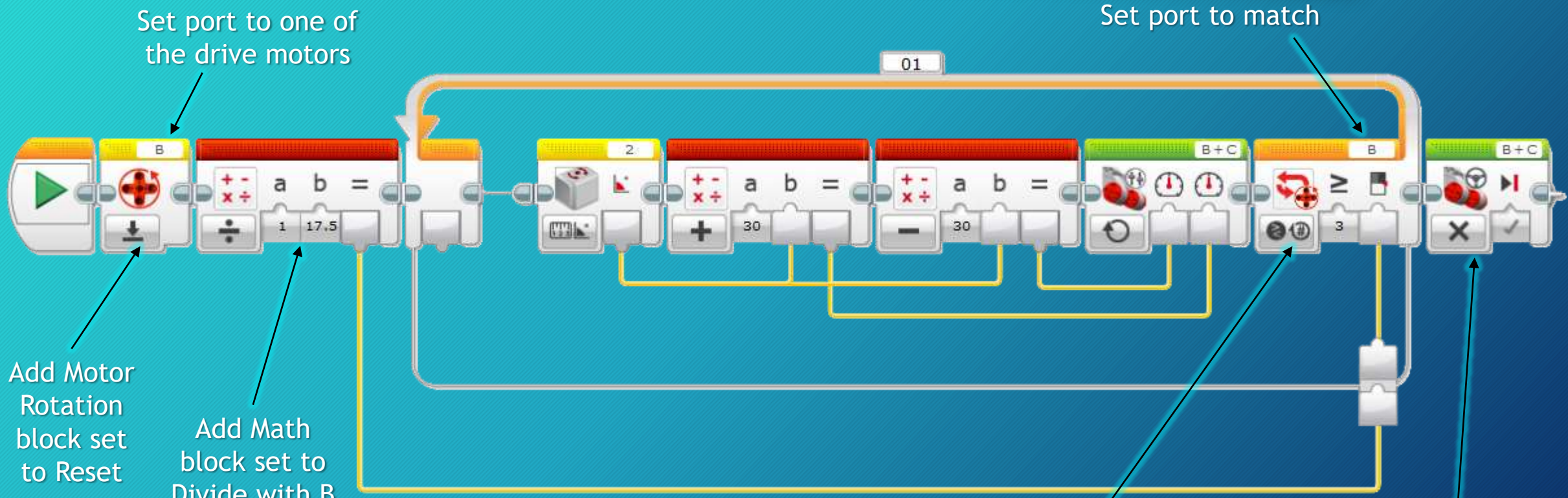
15. Change both zeroes in the Math blocks a input to 30.



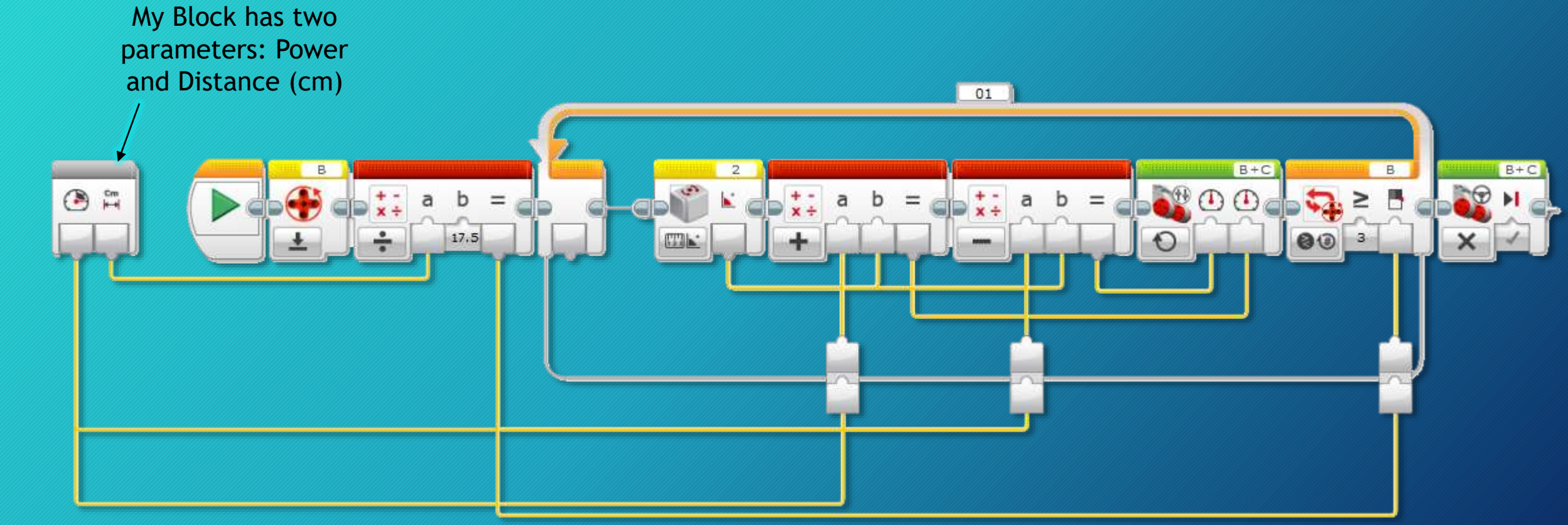
A gyro-following loop - Summary

- When gyro angle is zero: both motors have a speed of 30 and the robot moves straight ahead at 30.
- When gyro angle is not zero: one motor moves faster than 30 and other moves slower than 30 and the robot moves forward but turns toward zero angle.

Add distance control to gyro-following loop



Distance control gyro-following loop My Block



Understanding LEGO® color sensors

- Color sensors have several “modes”
- Color – used to detect specific colors: black, blue, green, yellow, red, white, brown
- Ambient light – amount of light reaching the sensor
- Reflected light – same as ambient, but sensor's LED is turned on
- In all of these modes, external lighting can affect readings
- Shielding helps external light from causing false readings

Sensor should be mounted 0.5 cm to 2.0 cm above the surface

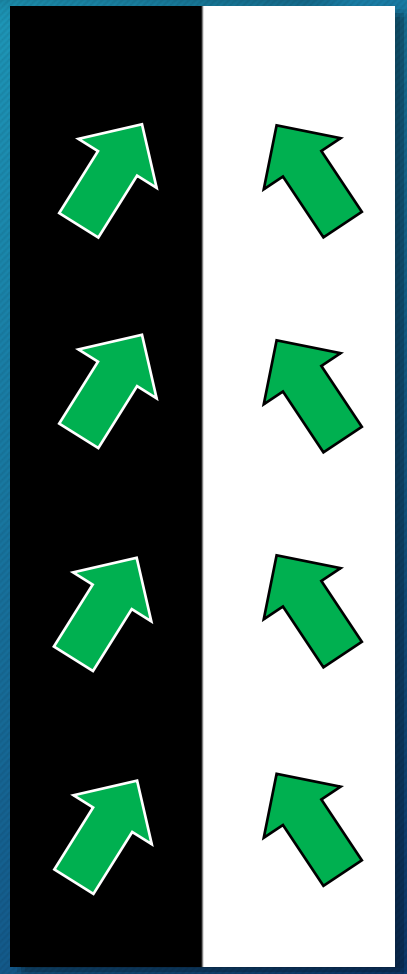
Reflected light mode

- The sensor returns a value from 0 to 100
- 0 = Color sensor receiving very little light
- 100 = Color sensor receiving a lot of light

- Use port view to see what the robot is sensing

Line / edge following

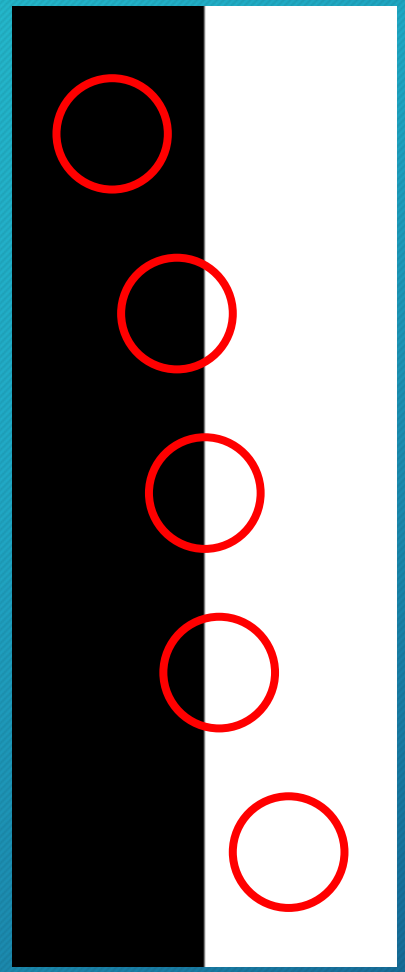
- Use the color sensor to follow lines (actually edges) on the field
- Basic idea:
  - When the robot sees black, turn right
  - When the robot sees white, turn left
- This causes the robot to alternate along the “edge” where white and black meet



Reflected light mode

What sorts of values would the color sensor see at each red circle position?

What should the robot do to stay on the edge where black meets white?



5 = turn right a lot

20 = turn right a little

35 = go straight

50 = turn left a little

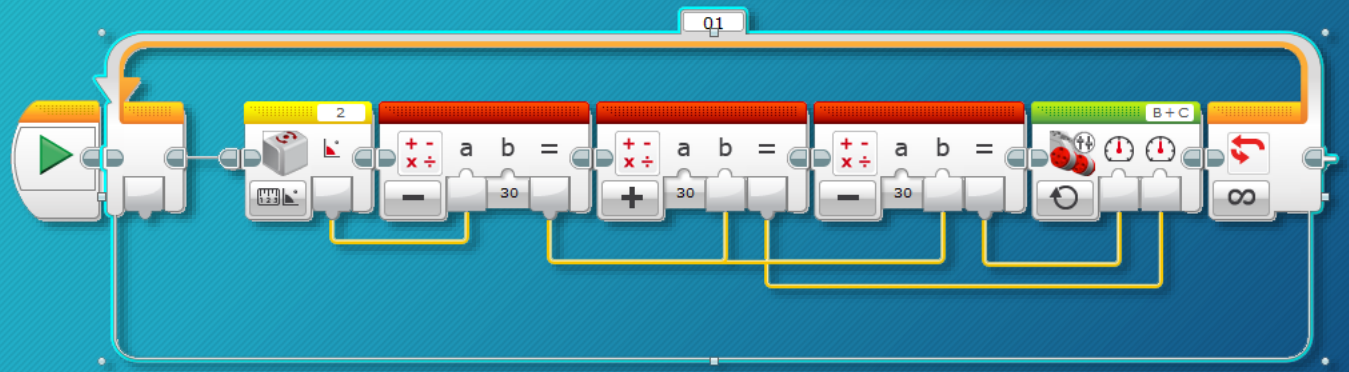
70 = turn left a lot

Proportional edge following

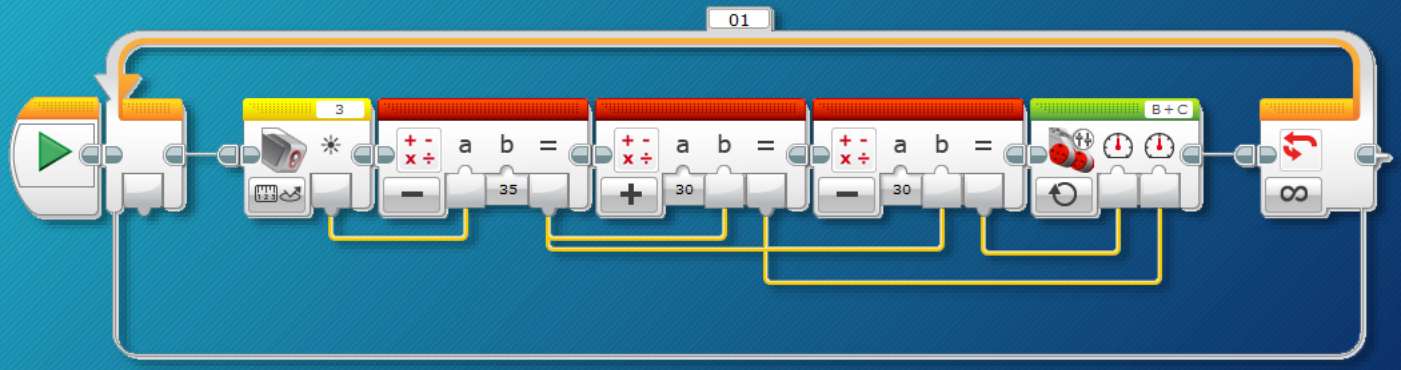
- The light sensor must be in front of the driving wheels for line edge following to work
- With a little tuning, a robot can very precisely follow a line (not waggling)

Proportional edge following

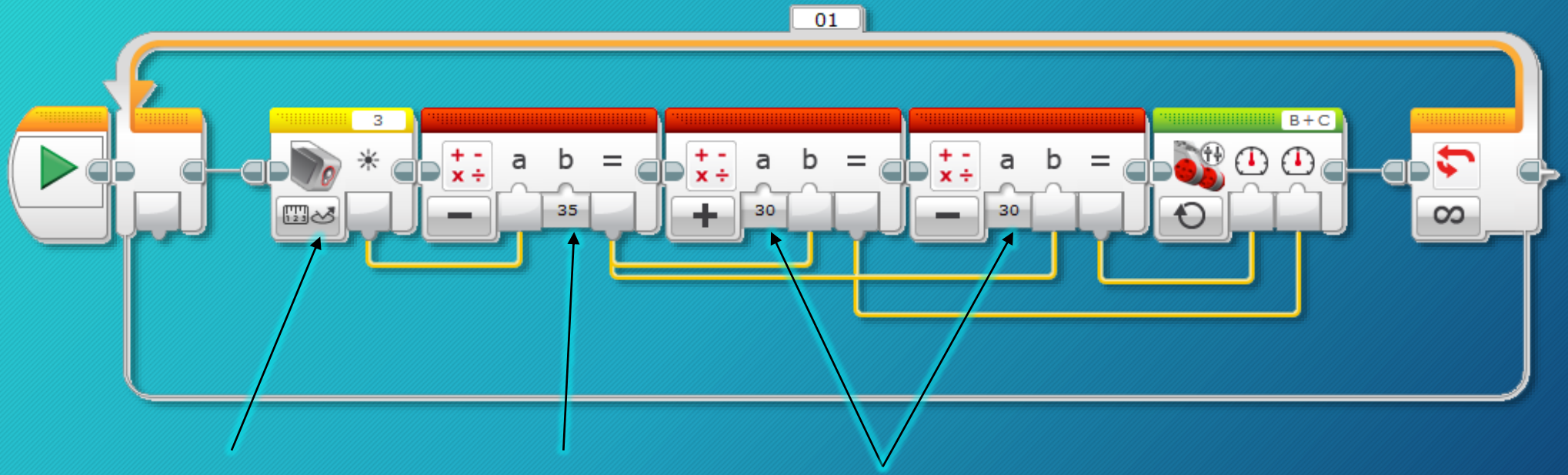
Make a copy of your gyro following loop program



Change the gyro sensor to reflected light sensor



Proportional edge following



Color sensor:
Measure ▶ Reflective
Light Intensity

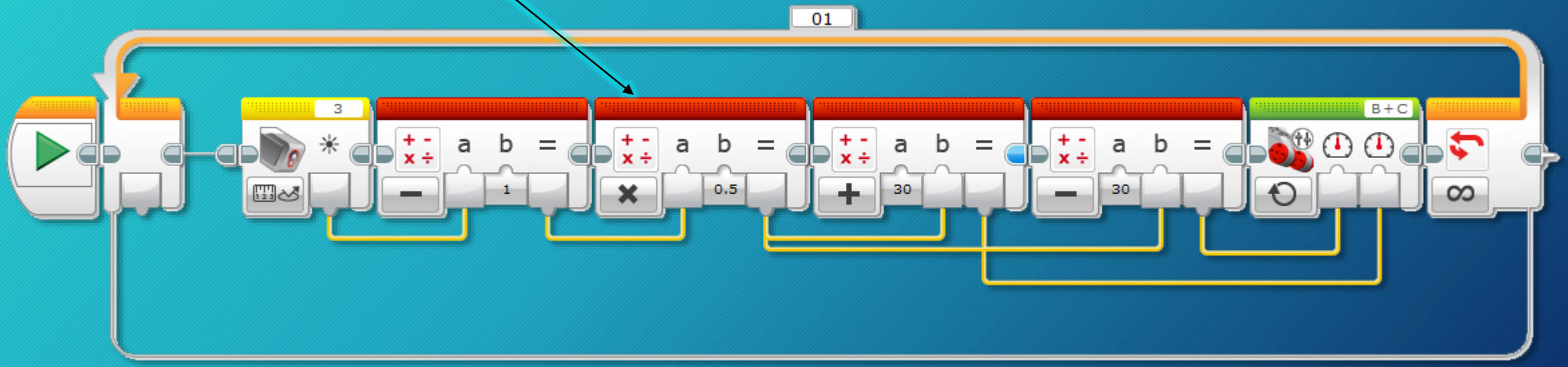
Value at “edge”
Midpoint

Forward
base speed

Proportional edge following w/gain

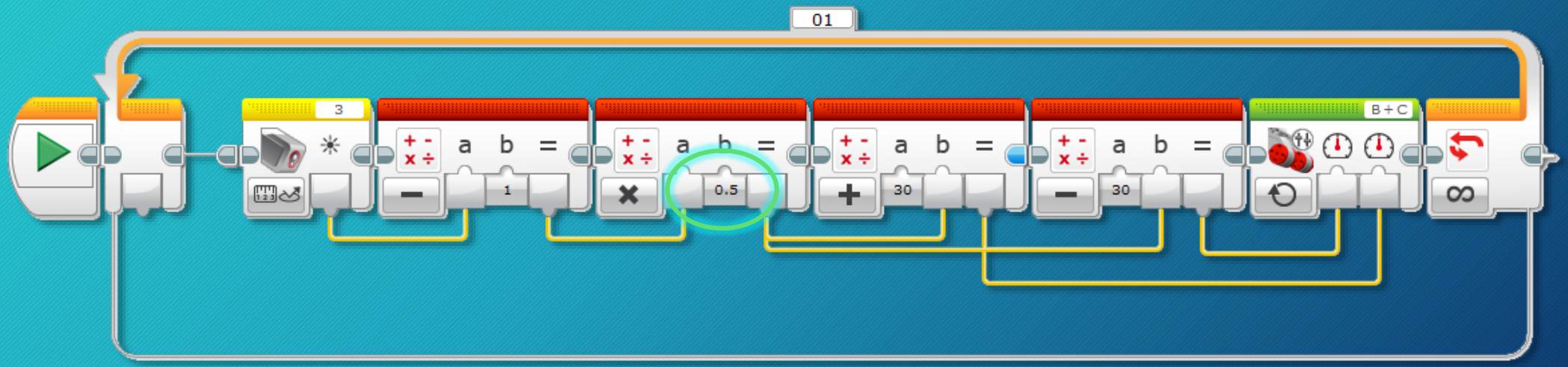
Add Math block set to:
Multiply and **b** input at 0.5

- Adding a “gain” factor gives you additional control over how the robot follows the edge.



Proportional edge following w/gain

- Higher gain = robot makes sharper turns
- Lower gain = robot makes shallower turns
- If robot is “waggling”, decrease gain
- If robot isn't finding the line, increase gain



Sequencing Program

- A “master sequence program” combines all missions into a single program in the desired run order.
- This reduces time spent in Base by not requiring technicians to navigate to, and select the next program to run.

Most ***FLL***® teams create separate programs for missions (or trips) out of Base

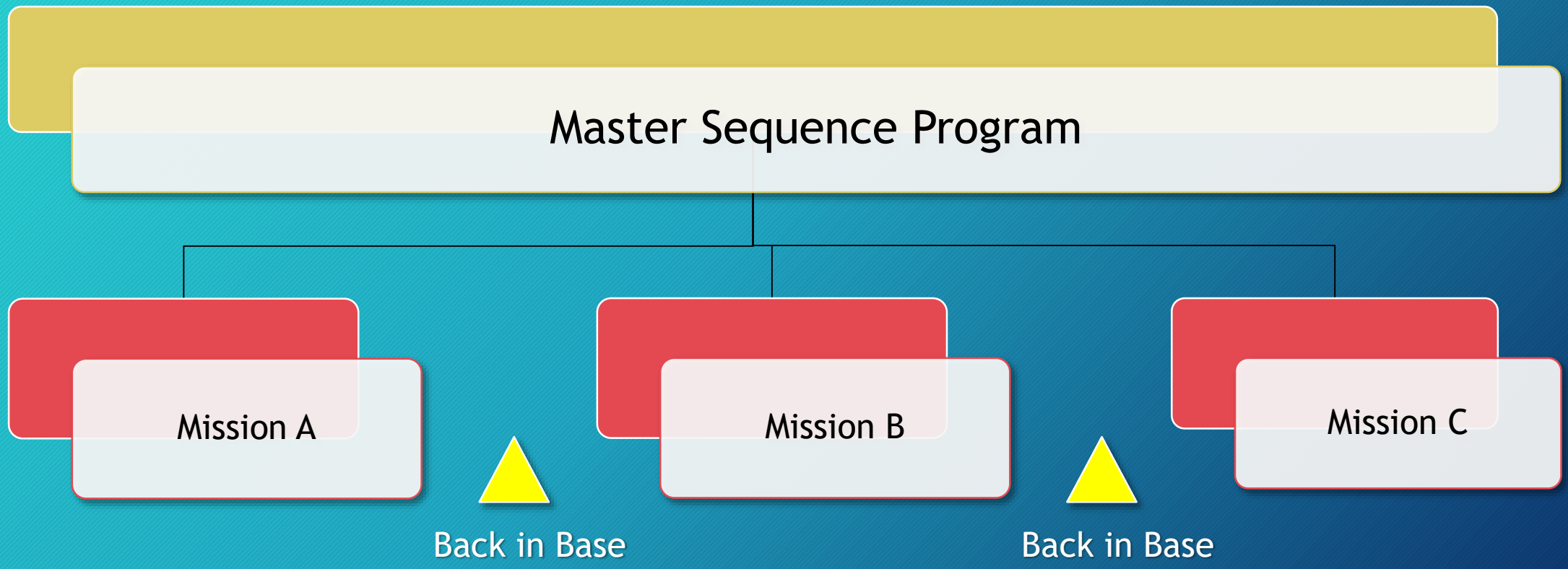
Master sequence programs

- There are many different types of “master sequencer” programs. We are reviewing two, a basic, and an intermediate level.
- Master sequence programs typically have each mission program saved as a separate My Block, with the sequence program calling each of these My Blocks in the desired order.

Master sequence program components

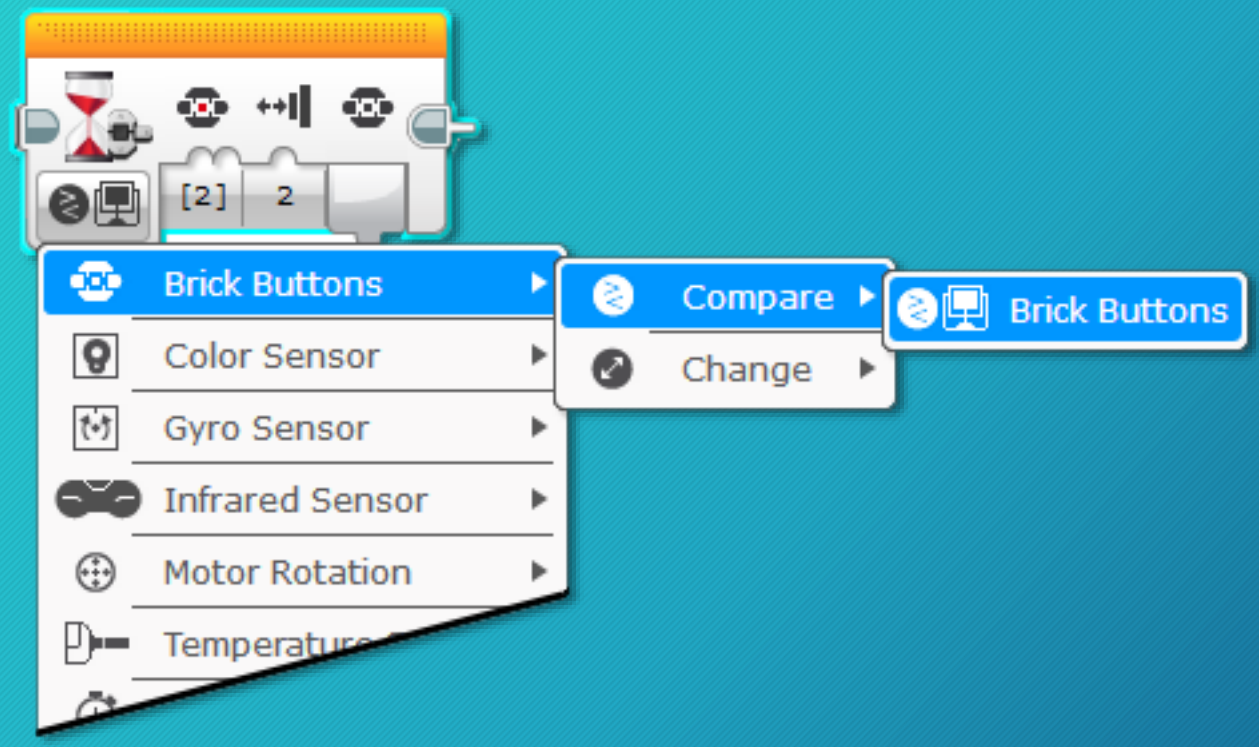
- A mission My Block is the programming needed to solve a single or a series of missions made into a My Block.
- A trip is a series of mission My Blocks used from leaves Base until the robot returns to Base.
- The master sequence program runs the trips from a single program in order, and automatically advances from one trip to the next. More advanced sequence programs allows the technician to navigate back and forth between trips allowing the technician to change the order on the fly.

Basic master sequence diagram

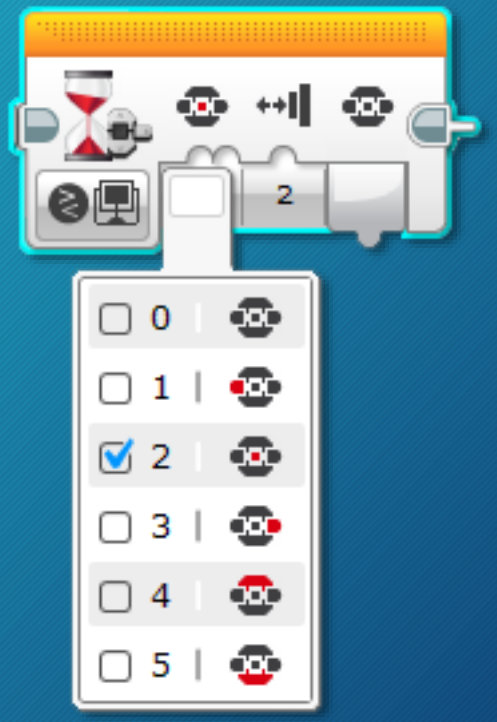


Wait block – Brink Button

1) Set the Wait block to Brick Buttons
▶ Compare ▶ Brick Buttons

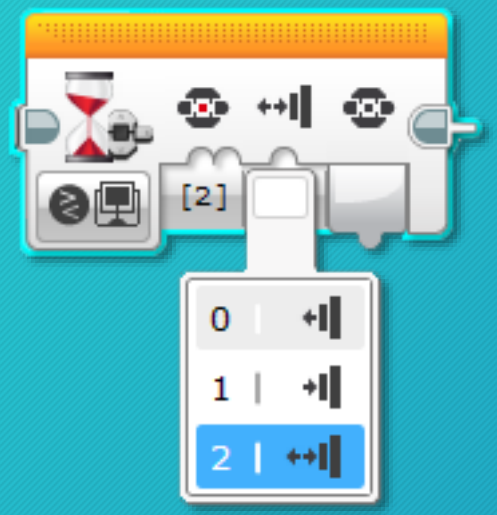


2) Set Button to 2
Center button



Wait block – Button push

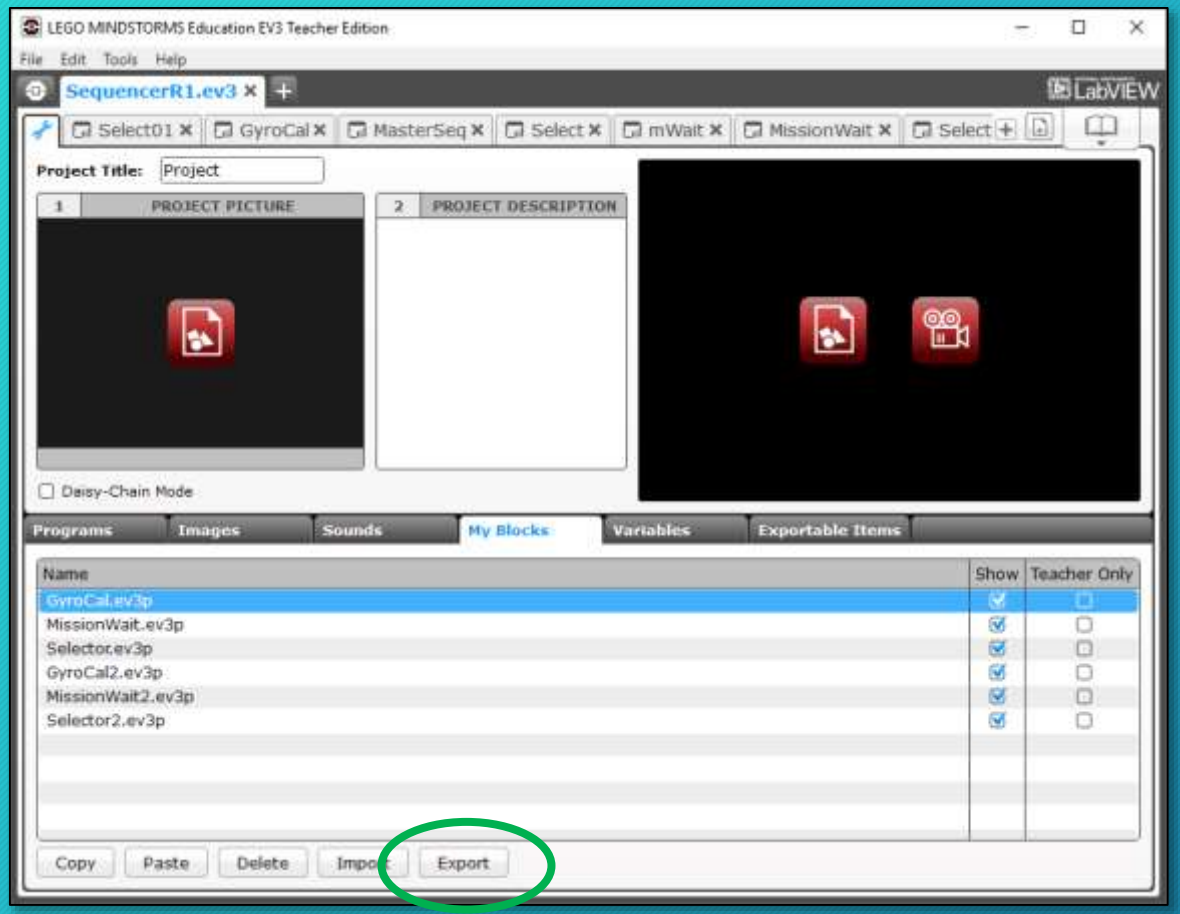
3) Set State to 2
Pressed and Released



4) Wait block
final

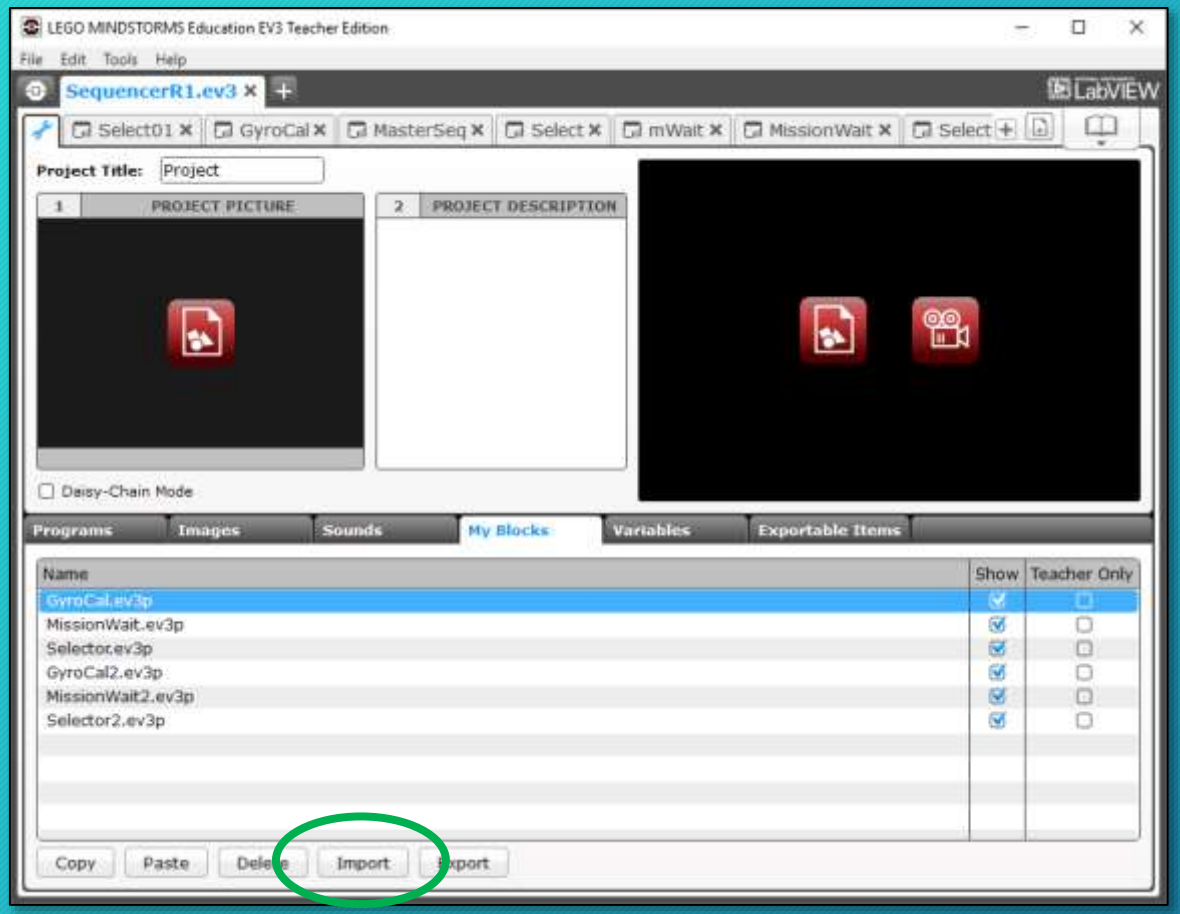
- This Wait block pauses the program until the center button on the EV3 Brick is pressed and released.

Export My Blocks from program



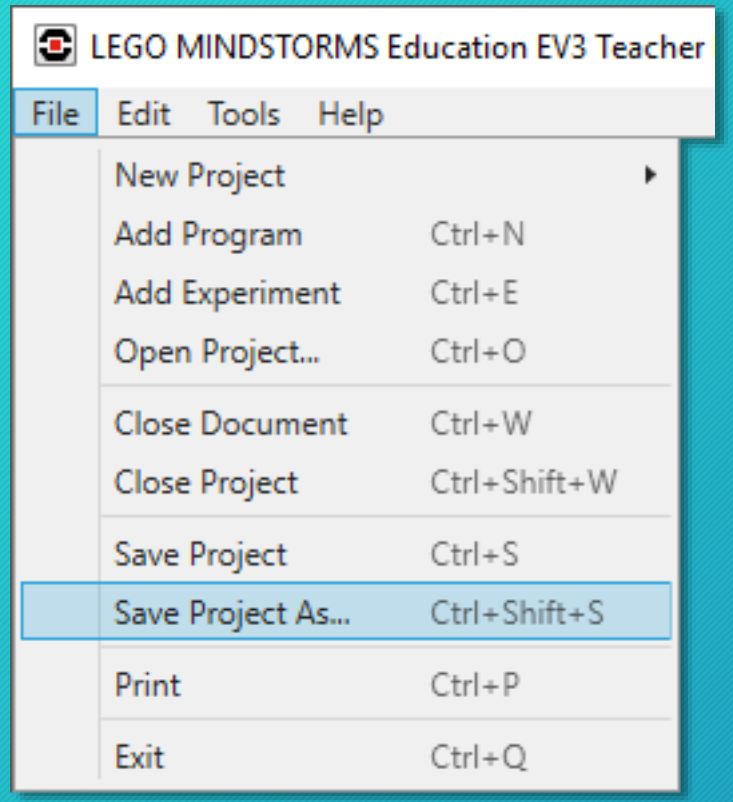
1. Open the Project with the My Block to export
2. Select Project Properties
3. Select the My Block tab
4. Select the My Block to export
5. Select the Export button
6. Name and save the My Block
7. Repeat for each My Block you want to export

Import My Blocks into program



1. Open the Project where you want to import the My Blocks
2. Select Project Properties
3. Select the My Block tab
4. Navigate to where the My Blocks was saved (exported)
5. Select the My Blocks to import
6. Select the Import button

Moving Mission My Blocks



- Exporting/Importing mission My Blocks will duplicate each My Block used in the mission separately.
  - Example: If you used three “PivotTurn” My Blocks in the mission, when you export/import it, you will have PivotTurn1, PivotTurn2, and PivotTurn3 My Blocks each with the unique values set in it.
- Suggest using the Save Project As option, and then deleting the programs not wanted in the new project.

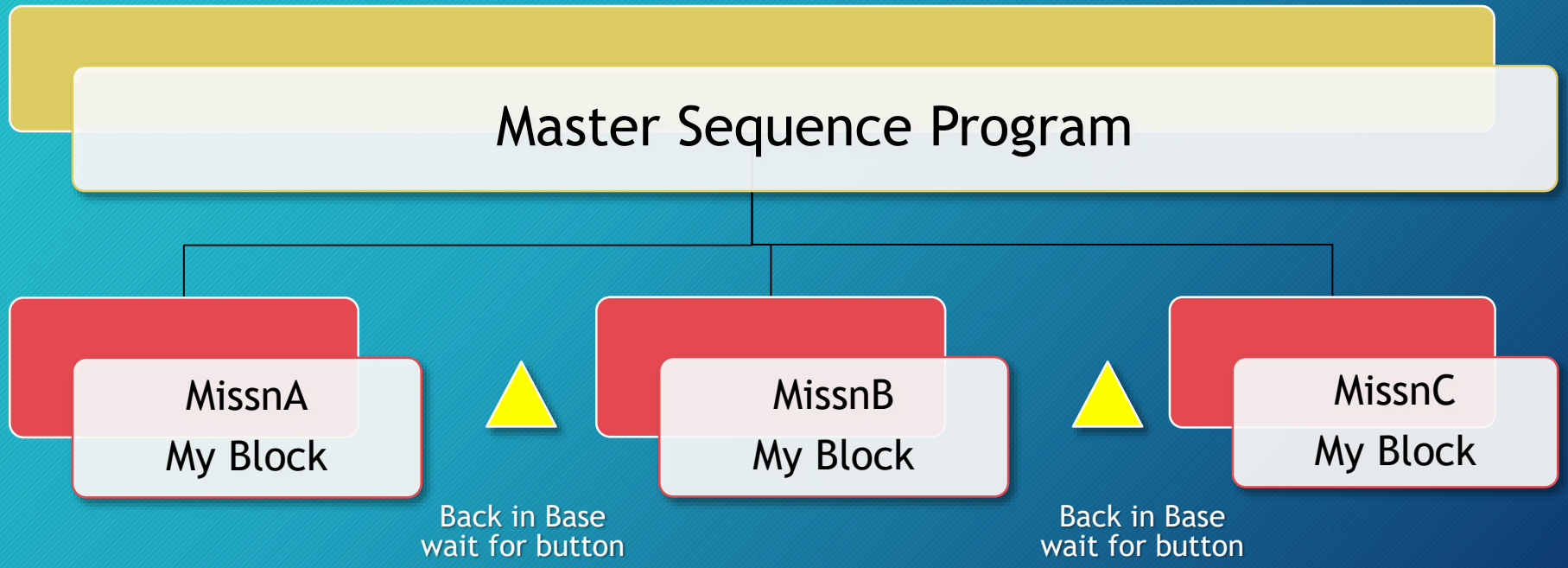
Sequencer challenge setup

Open new Project, and build three programs:

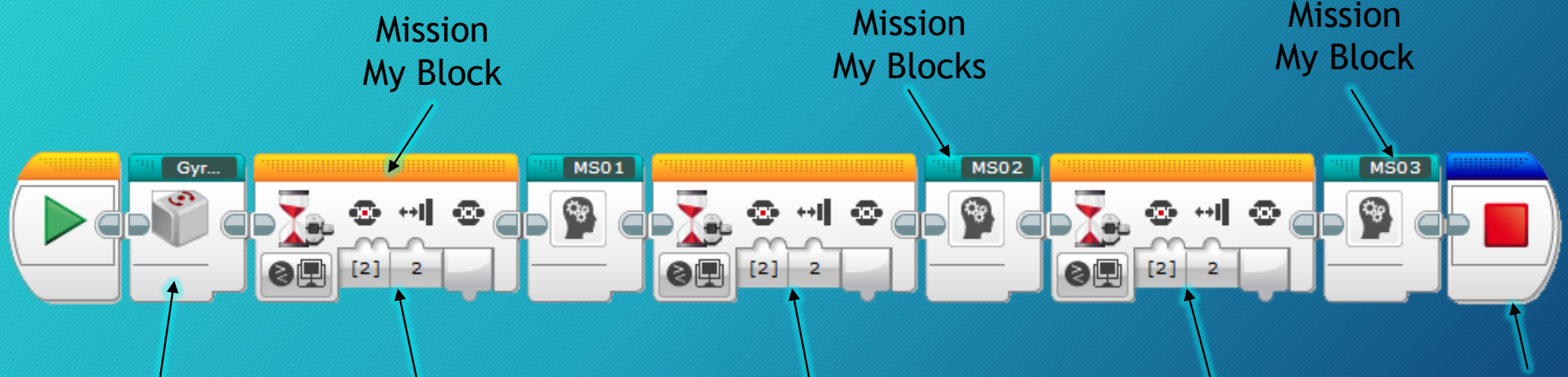
1. MissnA: Moves forward 100 cm, turns 180º and moves 100 cm, and plays “one” sound
2. MissnB: Moves forward 75 cm, turns right 90º moves forward 25 cm, turns 180º moves forward 25 cm, turns left 90º, moves forward 75 cm, and spins 2 rotation and plays “two”
3. MissnC: Moves forward 10 cm, right turns 45º, moves 120 cm, turns 180º moves 120 cm, and spins 3 rotation and plays “three”

Basic master sequence diagram

1. Make each of the six programs into a My Block
2. Build the basic sequence program based on the chart below



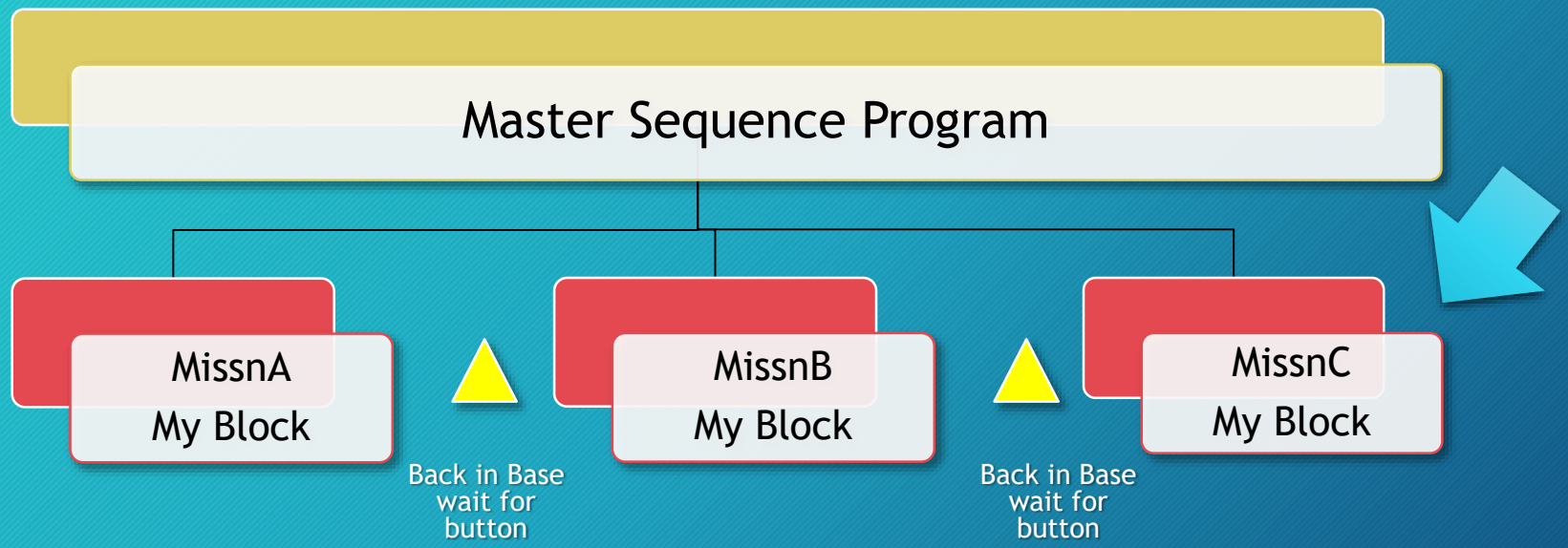
Basic sequence program



Improving the master sequence program?

Using our simple master sequence program, how would you rerun a program if needed?

For example, if Mission C failed, and you had time to try it again, how would you?



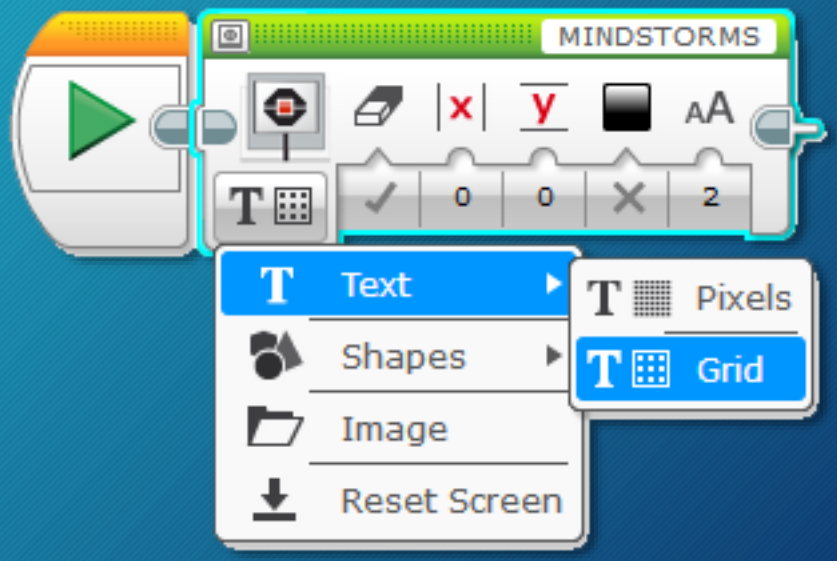
Ways to improve the sequence program?

- Identify which program is next to run for the Technician (Driver).
- Make it easy to select, repeat, or skip missions as needed.

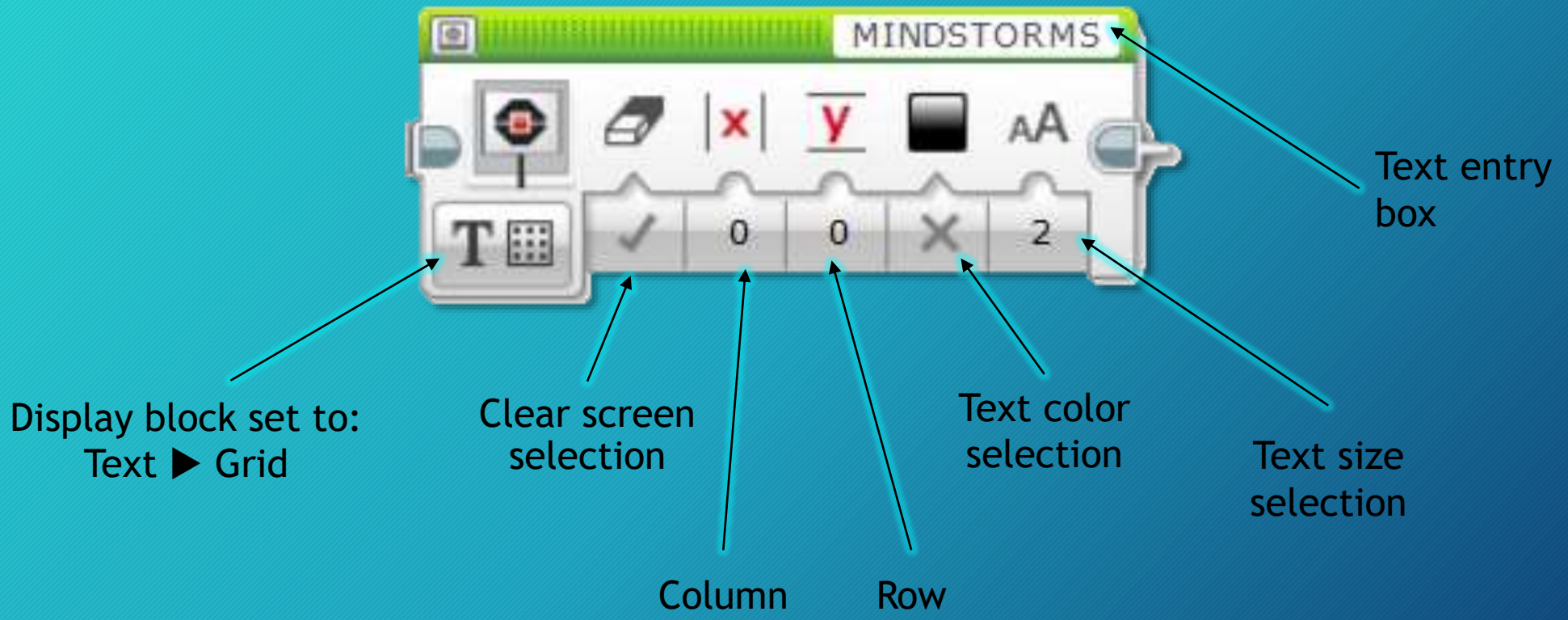
Master sequence program with mission trip displayed on EV3 screen

To tell the driver what mission will be run next create a “MissionWait” My Block

1. In an new program, add a Display block set to Text ▶ Grid



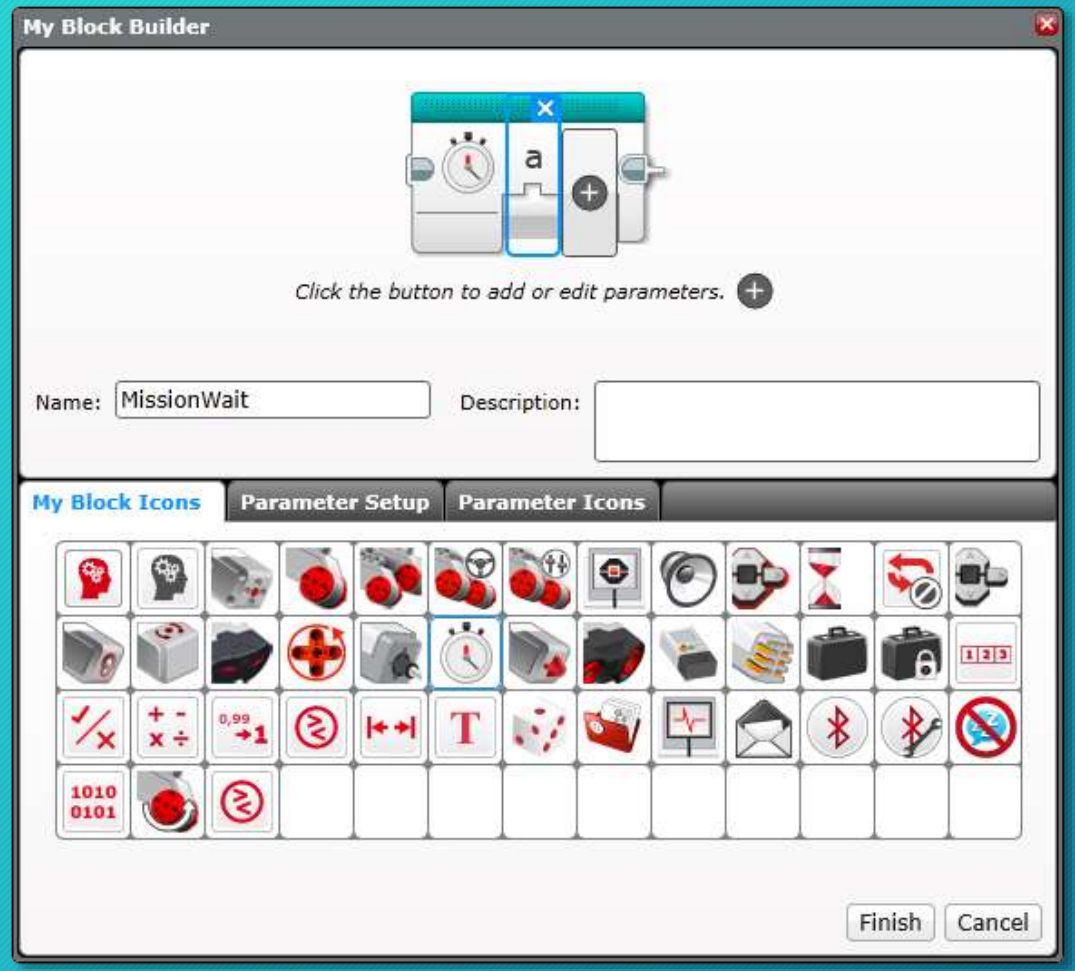
Master sequence program with mission trip displayed on EV3 screen



Master sequence program with mission trip displayed on EV3 screen

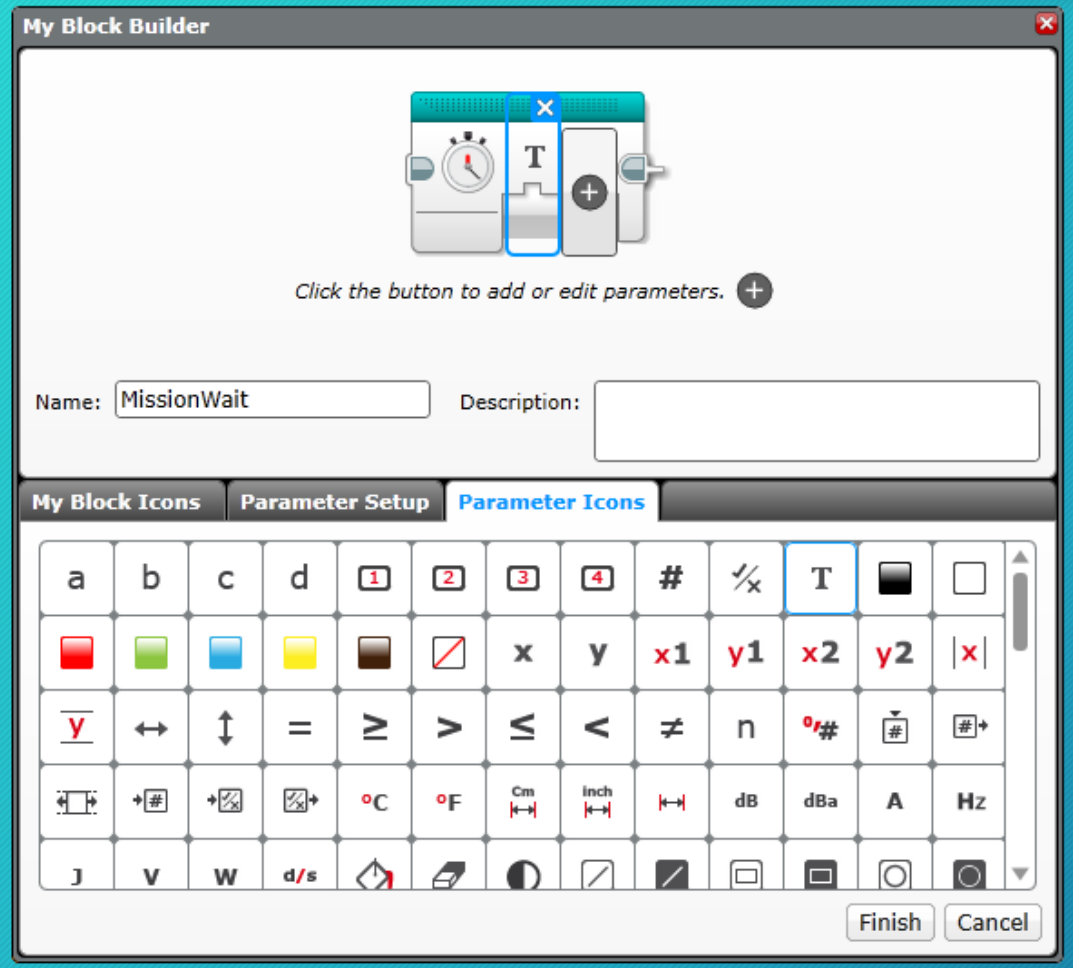
2. Add a Wait block set to Brick Buttons ▶ Compare
3. Set button to [2] (Center) and State to 2 (Bumped)
4. Select the Text entry box and set to Wired
5. Select the two blocks

Master sequence program with mission trip displayed on EV3 screen



6. From the drop down menu, select Tools ▶ My Block Builder
7. Enter My Block Name (MissionWait)
8. Select My Block icon
9. Click + to add parameter

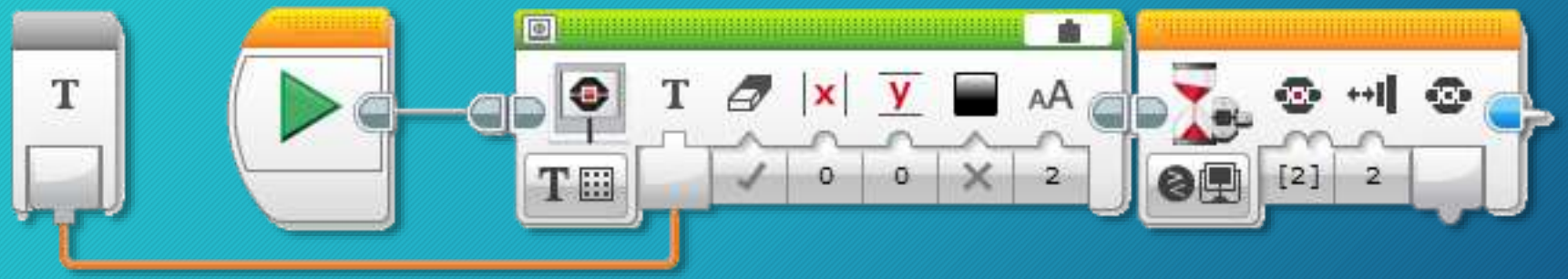
Master sequence program with mission trip displayed on EV3 screen



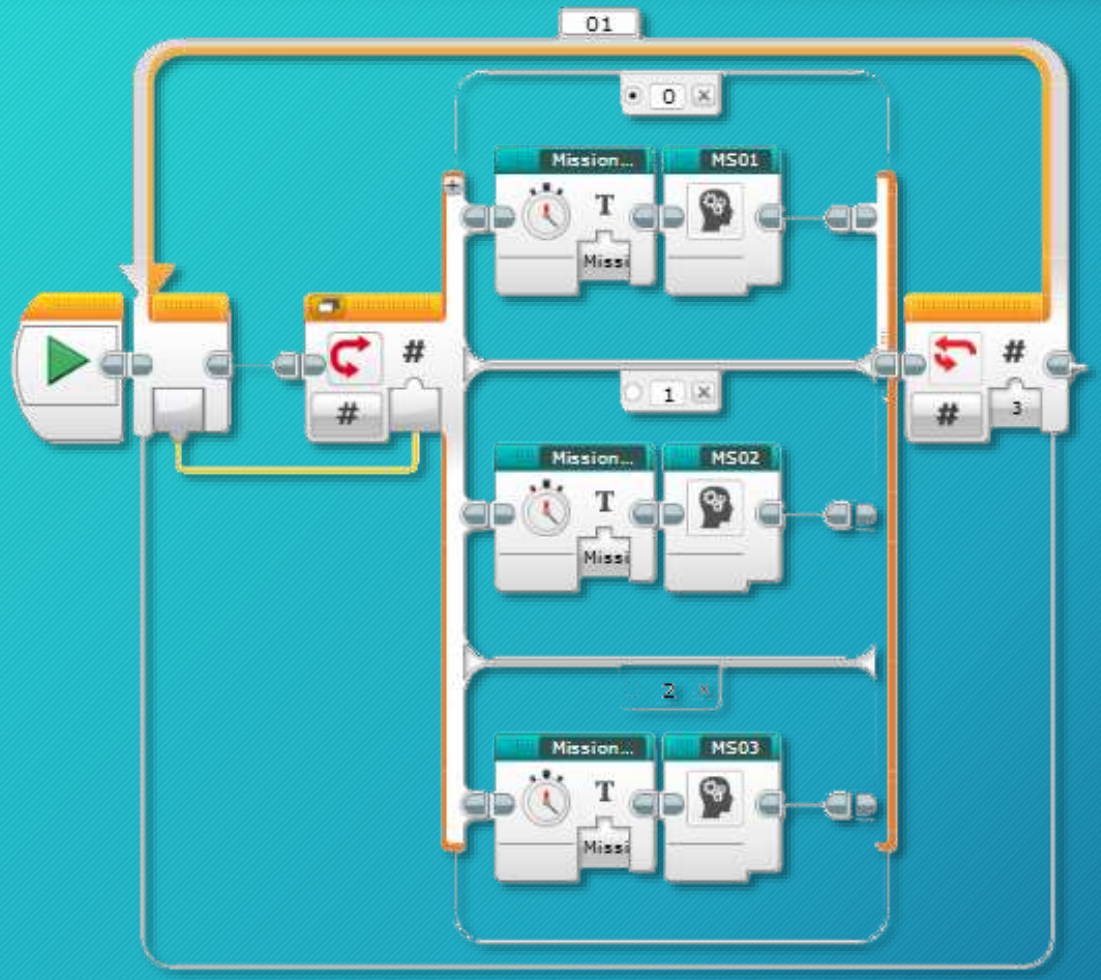
10. Select the Parameter Setup
11. Enter parameter name (Text)
12. Change the Data Type to Text
13. Select the Parameter Icons tab
14. Select the icon for the parameter

Master sequence program with mission trip displayed on EV3 screen

15. Wire the parameter to the text input
16. Save Project

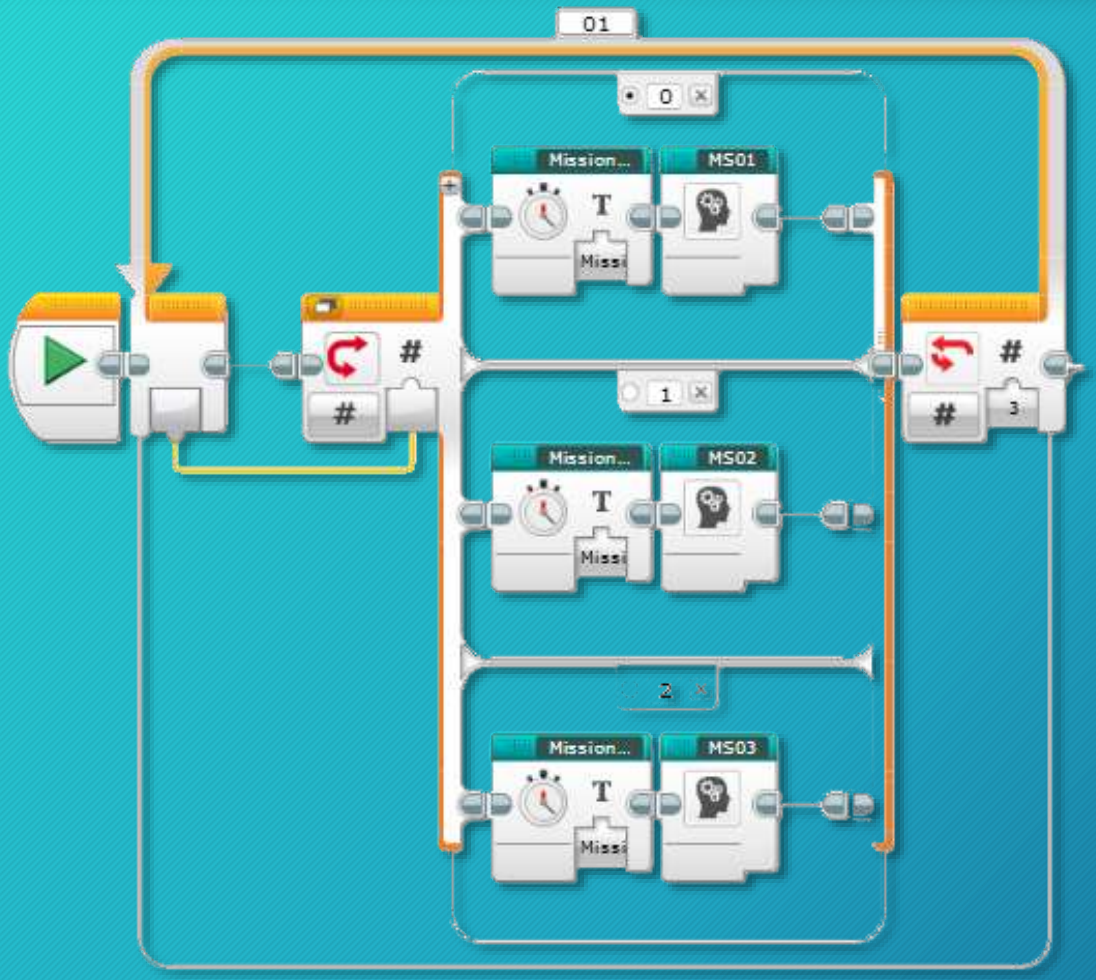


Master sequence program with mission trip displayed on EV3 screen



1. Open a New Program named MasterSeq
2. Add a Loop block set to Count (right side of Loop block)
3. In the Loop block add a Switch block set to Numeric
4. Select the **+** to add an option

Master sequence program with mission trip displayed on EV3 screen

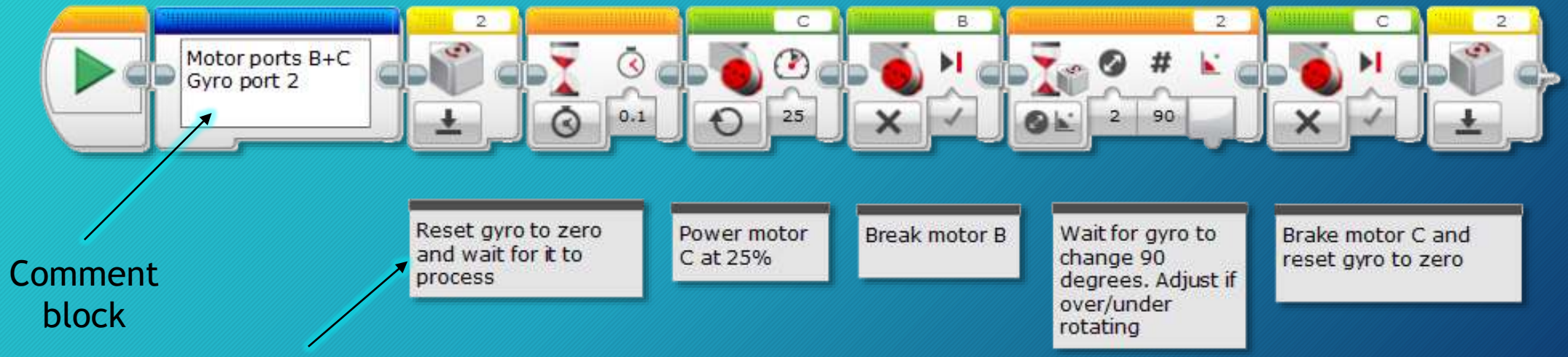


5. Add a MissionWait My Block to each option and enter text (Mission1, Mission2, or Mission3)
6. Add missions My Blocks in the same order as the first sequence program
7. Save, download, and test your program

Tip: Selecting the icon on the Switch block color bar toggles between Switch views

Adding comments to program

- Properly commenting your programs not only helps others to understand what you did, but you to remember what you did and to locate sections of the program quickly.



Basic My Blocks list for FLL®

- Gyro calibrate
- Move distance in cm with gyro
- Pivot right
- Pivot left
- Pin point right
- Pin point left
- Line follower
- Master sequence program

These My Blocks can be programmed before the robot chassis is done.

Everything is awesome!