Advanced Mindstorms Programming for FLL

Patrick R. Michaud
Republic of Pi FLL #3034
pmichaud@pobox.com

October 26, 2013

Goals for this clinic

Help teams get better robot performance

Identify better programming techniques

Provide tips that have worked for our team

Point out traps that have caused us frustration

Consistency

Good programming and strategy are essential to consistently good performance

Needed to overcome the limitations of hardware

Great robot + poor strategy == inconsistent scores

Fair robot + good strategy == consistent scores

My Blocks

- Organize a set of blocks into a sequence
- Fundamental programming concept
- Use for:
  - Any block sequence that is reused frequently
    - Move a distance
    - Turn an angle
    - Follow a line
  - To organize programs into more readable units

TIP: Create a My Block for each mission

TIP: Combine multiple mission My Blocks into “trip” My Blocks

Moving forward a distance
Introduction to My Blocks

Move forward a distance

Specify distances in linear units (in, cm)

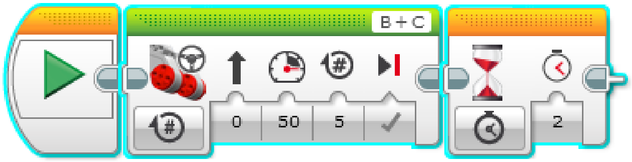
Need to know circumference of driving wheels

Several options:

- Calculate from printed wheel diameter
- Measure wheel diameter
- Use robot to determine circumference (best!)

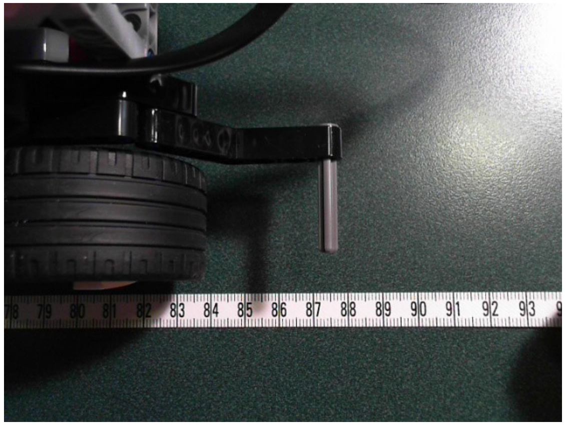
Calculating circumference

Create a program that moves forward 5 rotations, then waits for 2 sec



Run program and measure distance traveled by robot

wheel_circumference =
distance / motor_rotations



87.6 cm / 5 == 17.52 cm

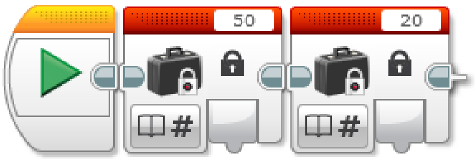
TIP: Always have a measuring tape handy

TIP: Use centimeters for measuring units

Move forward a distance

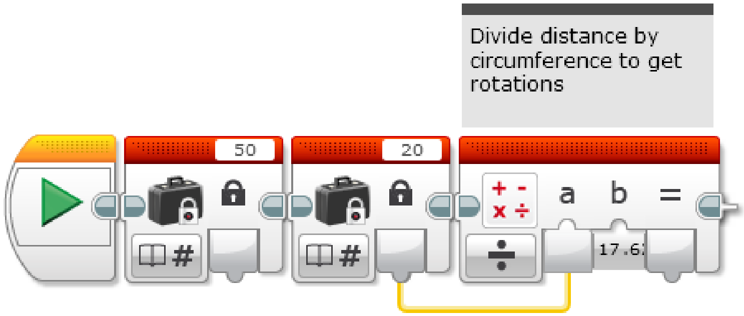
Start with an empty program

Add constant blocks for power and distance



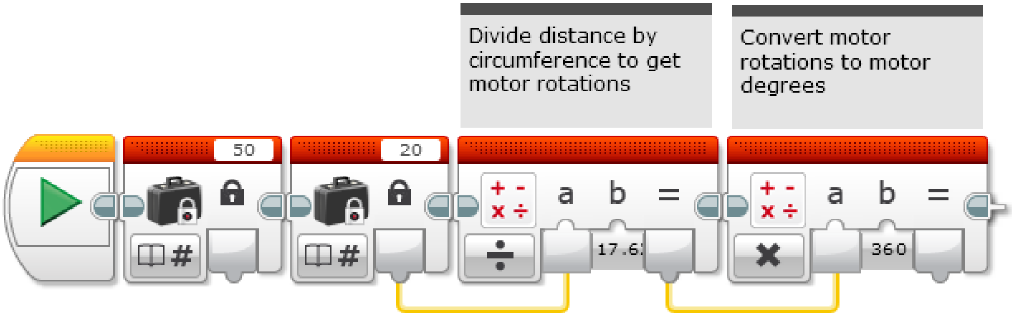
Add a division block to calculate rotations

- Wire A input to distance
- Set B value to wheel circumference


Divide distance by circumference to get rotations

Move forward a distance

Add a multiplication block to convert rotations to degrees

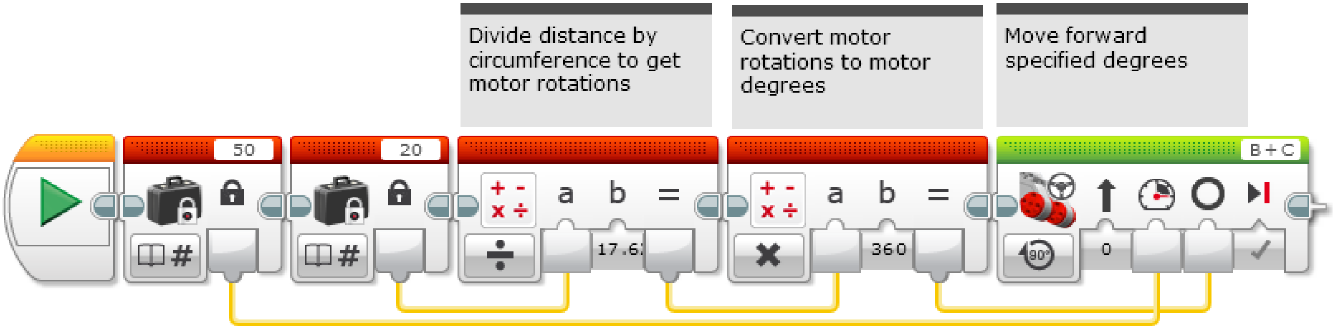


Divide distance by circumference to get motor rotations

Convert motor rotations to motor degrees

Move forward a distance

- Add a Move block
  - Wire power input to power constant block
  - Wire degrees input to output of multiplication block



Divide distance by circumference to get motor rotations

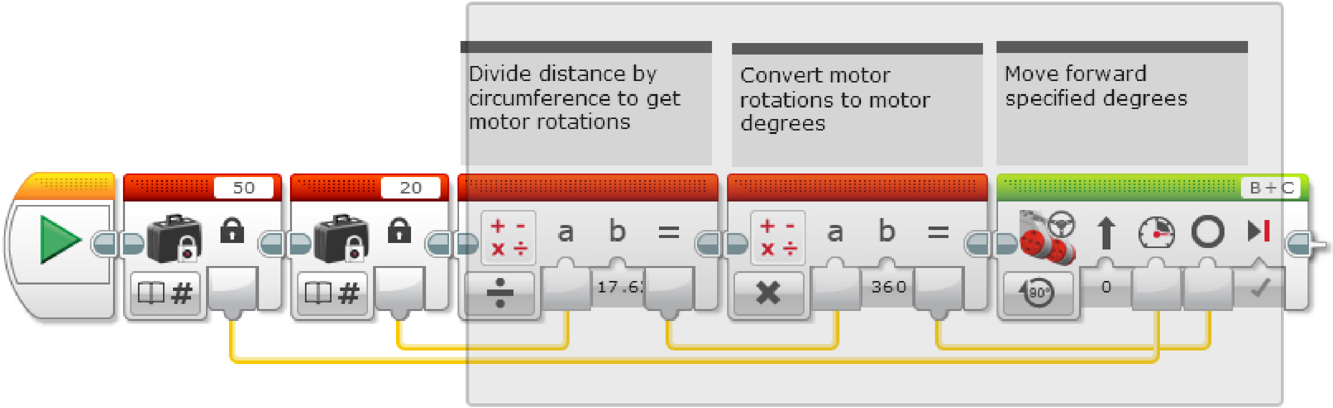
Convert motor rotations to motor degrees

Move forward specified degrees

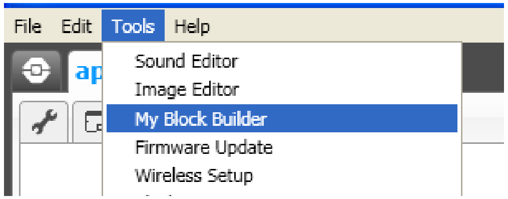
- Test the program to verify it works
- If distance is off, adjust circumference value

Create a “movedist” My Block

Drag to select everything but constant blocks

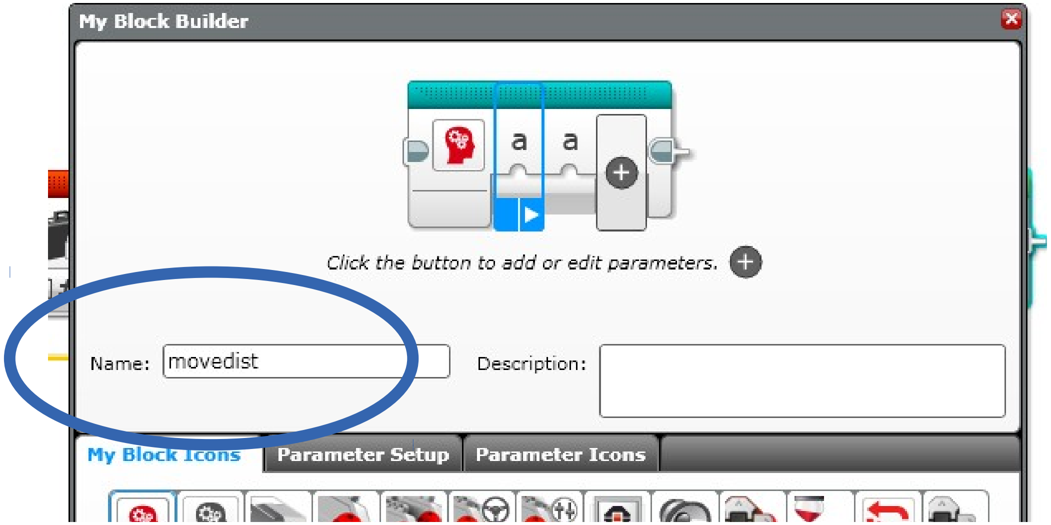

Divide distance by circumference to get motor rotations
Convert motor rotations to motor degrees
Move forward specified degrees
50
20
17.6
360
B+C
0
a
b
=

Select “My Block Builder” from Tools menu



Create a "movedist" My Block

Give the My Block a name



My Block Builder

Click the button to add or edit parameters.

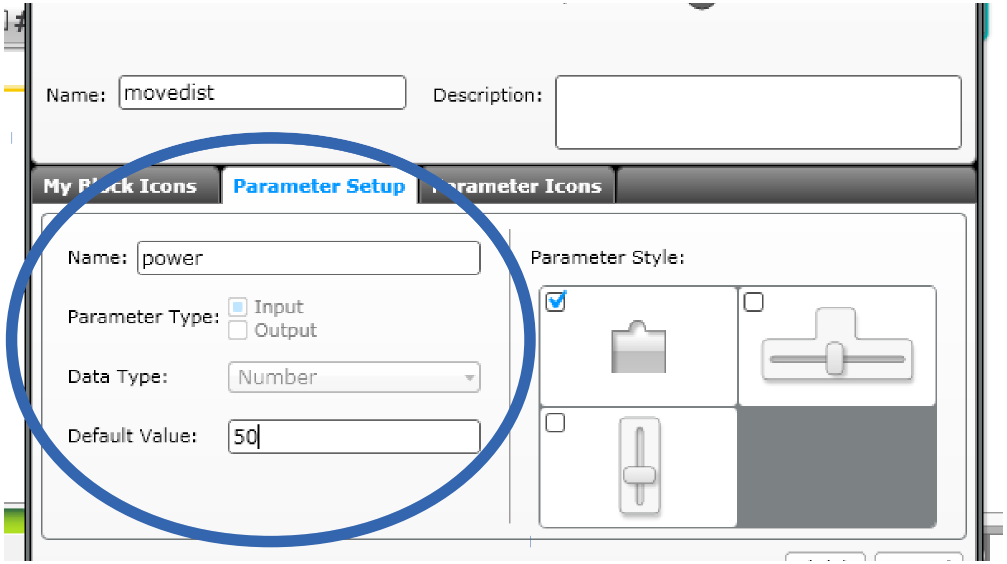
Name: movedist

Description:

Click "Parameter Setup"

Name the parameters "power" and "cm"

You can also provide default values



Name: movedist

Description:

Name: power

Parameter Type: ☑ Input ☐ Output

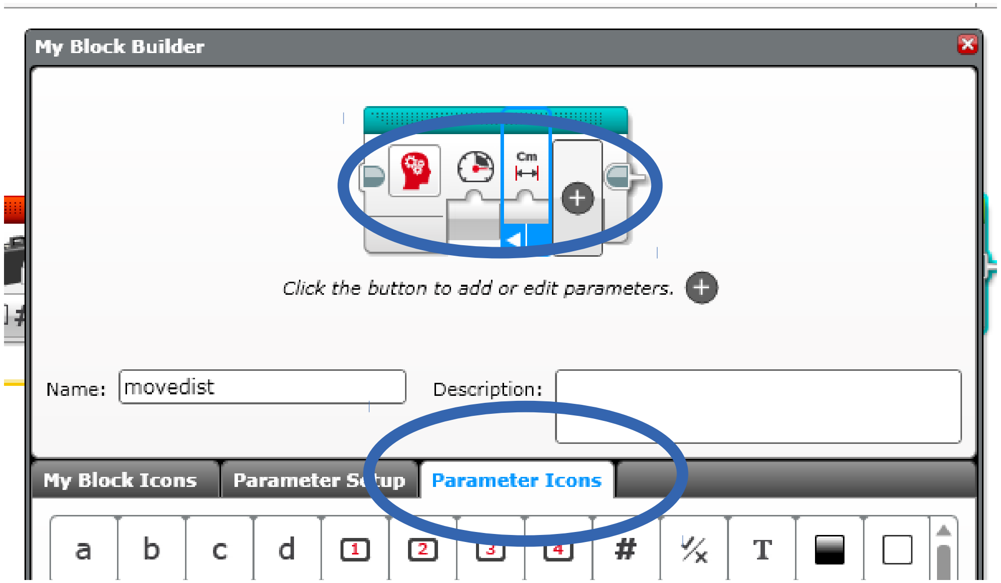
Data Type: Number

Default Value: 50

Parameter Style:

Create a “movedist” My Block

Click “Parameter Icons” to change input icons



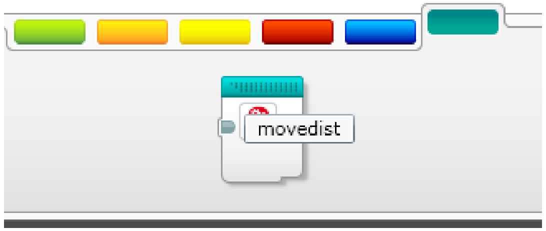
Click the button to add or edit parameters.

Name: movedist

Description:

Click “Finish”

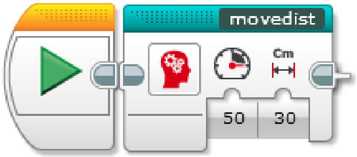
You now have a “movedist” block on the My Blocks palette



TIP: Be consistent with parameter names and icons in your My Blocks

Create a “movedist” My Block

Create a new program to test the “movedist” block.



Experiment: What happens if negative power or distance is given?

TIP: In the EV3 software, negative power and distance values cause the motors to reverse

Turning the robot

Robot turns

Many types of turns

- Point turn – robot spins in place
- Pivot turn – robot turns about a fixed wheel
- Wide turn – robot turns about an arc

Fundamental concept

The robot will turn when one wheel moves at a different speed from the other

The greater the difference in speeds, the tighter the turn

Pivot turns

- One wheel turns while other is stationary
- Our team has primarily used pivot turns
  - Most reliable and repeatable

Pivot turn formula

$$motor\_degrees =$$
$$turn\_angle * wheel\_track / wheel\_radius$$

“Motor degrees” is how far to move the turning motor

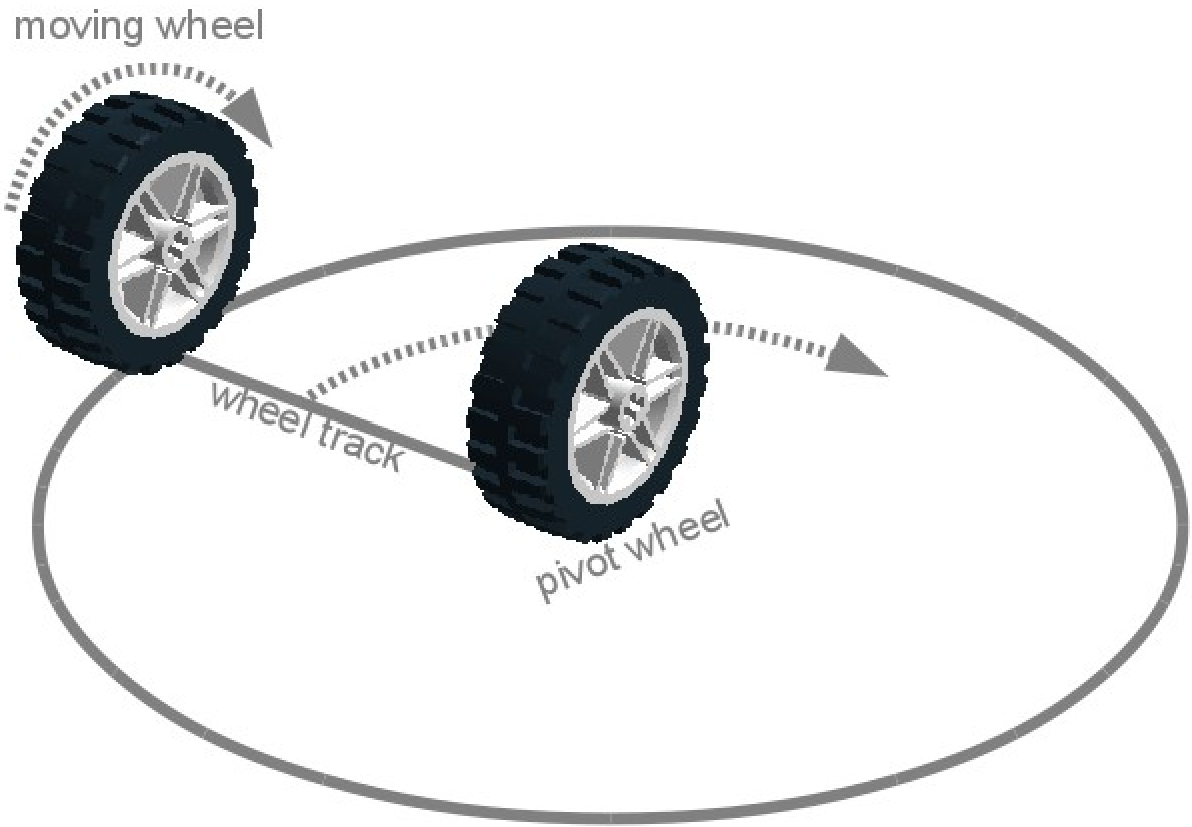
“Turn angle” is degrees robot is to turn

“Wheel track” is distance between two wheels

Wheel radius can be calculated from circumference

Pivot turn formula

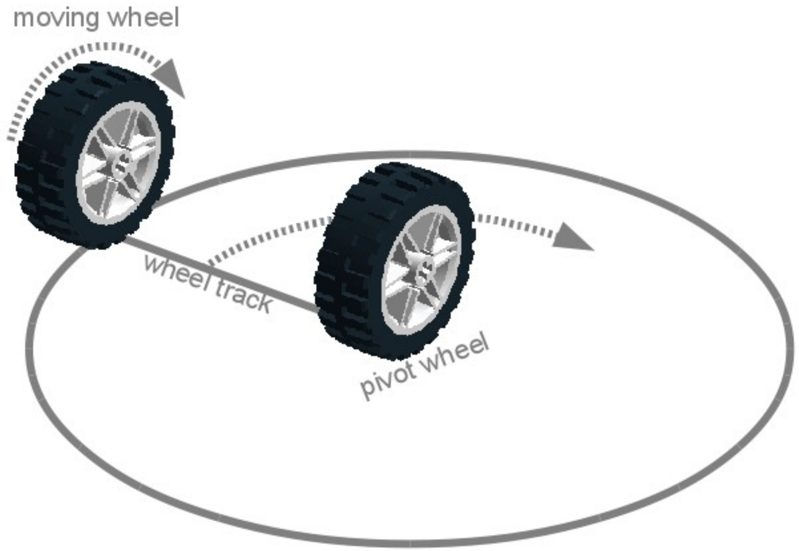
motor_degrees =
*turn_angle * wheel_track / wheel_radius*



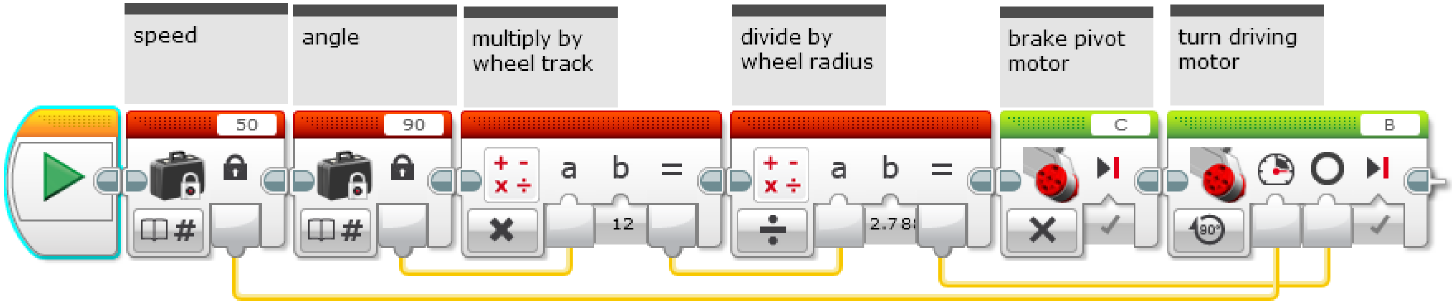
Pivot turn formula

motor_degrees =
turn_angle * *wheel_track* / *wheel_radius*

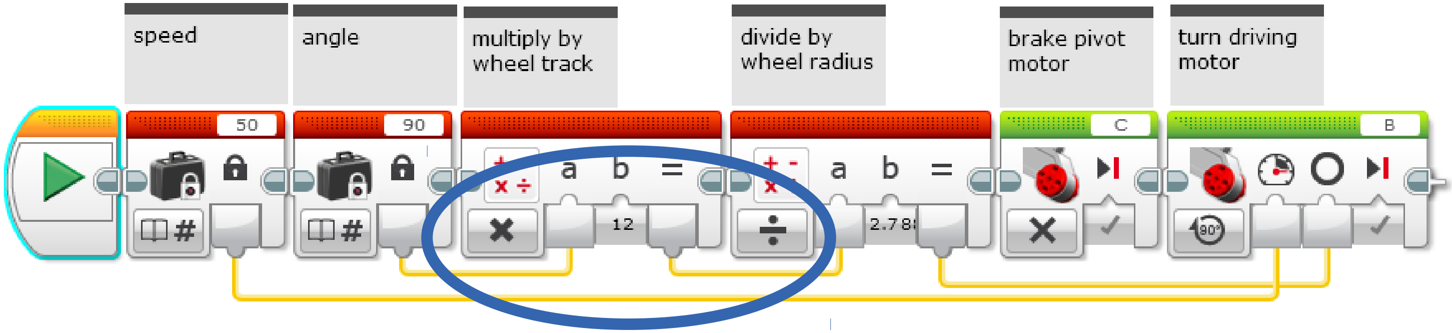
If wheel track is 3x wheel radius, the robot will turn 360 degrees when the moving wheel makes 3 rotations.



Pivot turn My Block



Pivot turn My Block



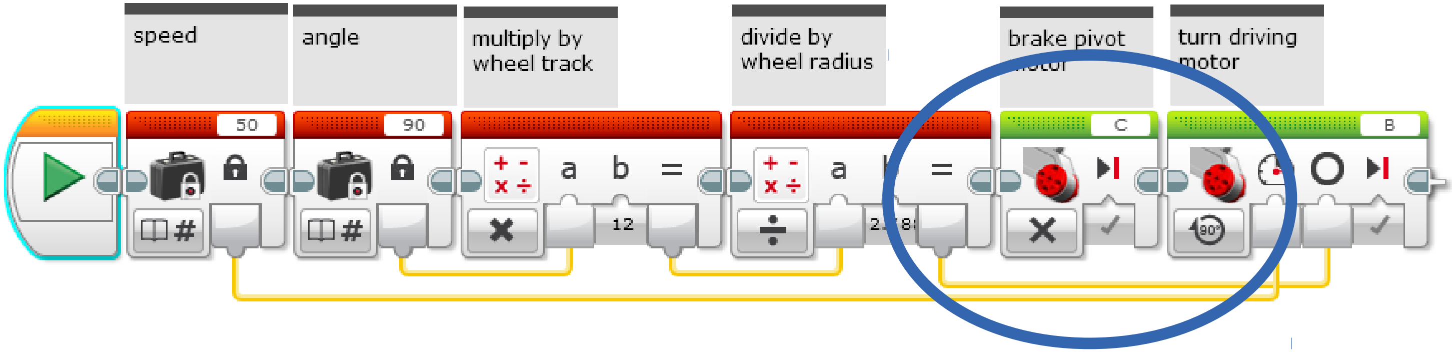
Calculate wheel track value experimentally:

Start with an estimate of wheel track

Adjust up or down until robot turns proper angle

TIP: LEGO stud centers are exactly 8mm apart

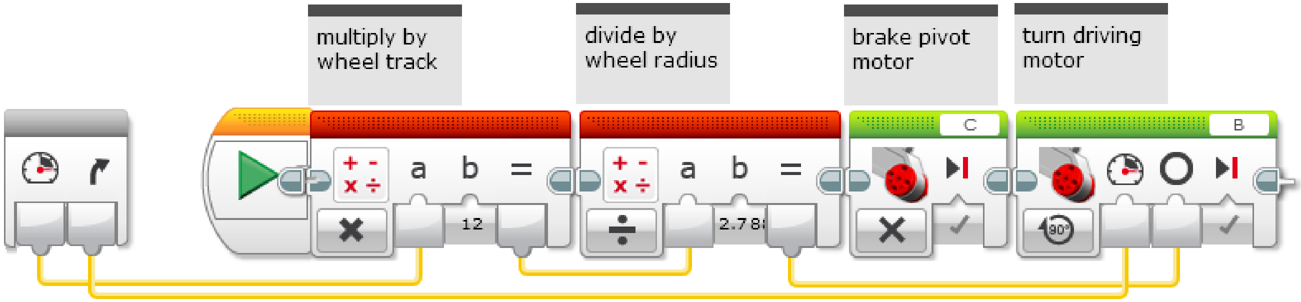
Pivot turn My Block



TRAP: Be sure to positively brake the pivot wheel

Otherwise, wheel can “coast” and affect turn

Pivot turn My Block



Once everything is working, turn it into a My Block

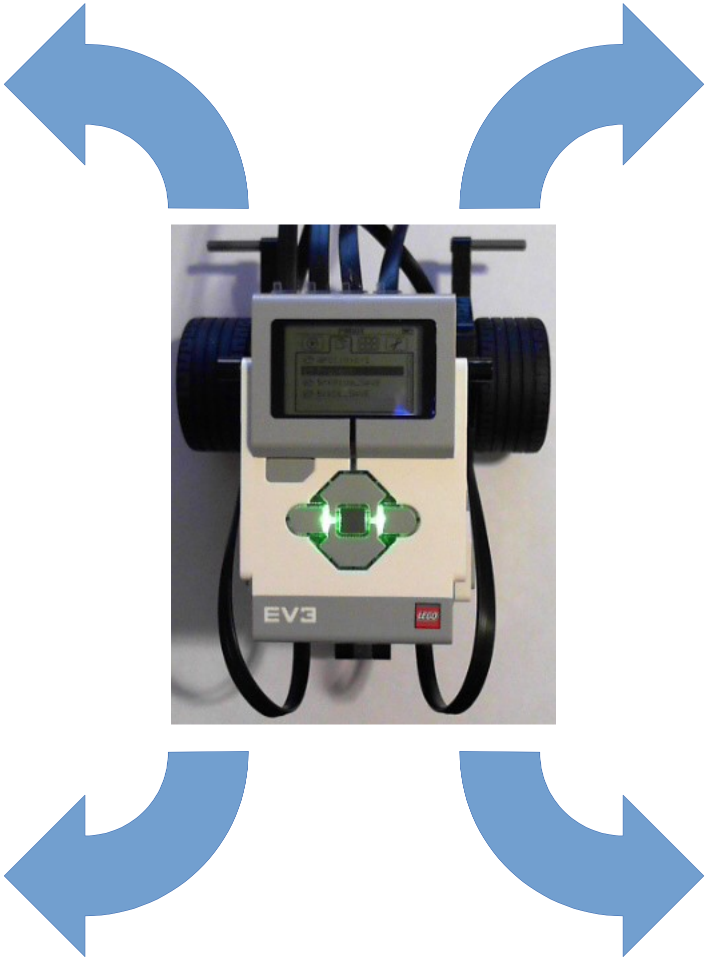
Can have separate blocks for turning left and turning right

Or combine using a switch block and logic input

Four pivot turn directions

Strategy: The robot has *four* pivot turn directions available

Keep all of them in mind when planning navigation



Stops

Stops

TRAP: Be sure the robot comes to a full stop between moves

When Move blocks complete, they brake the motors

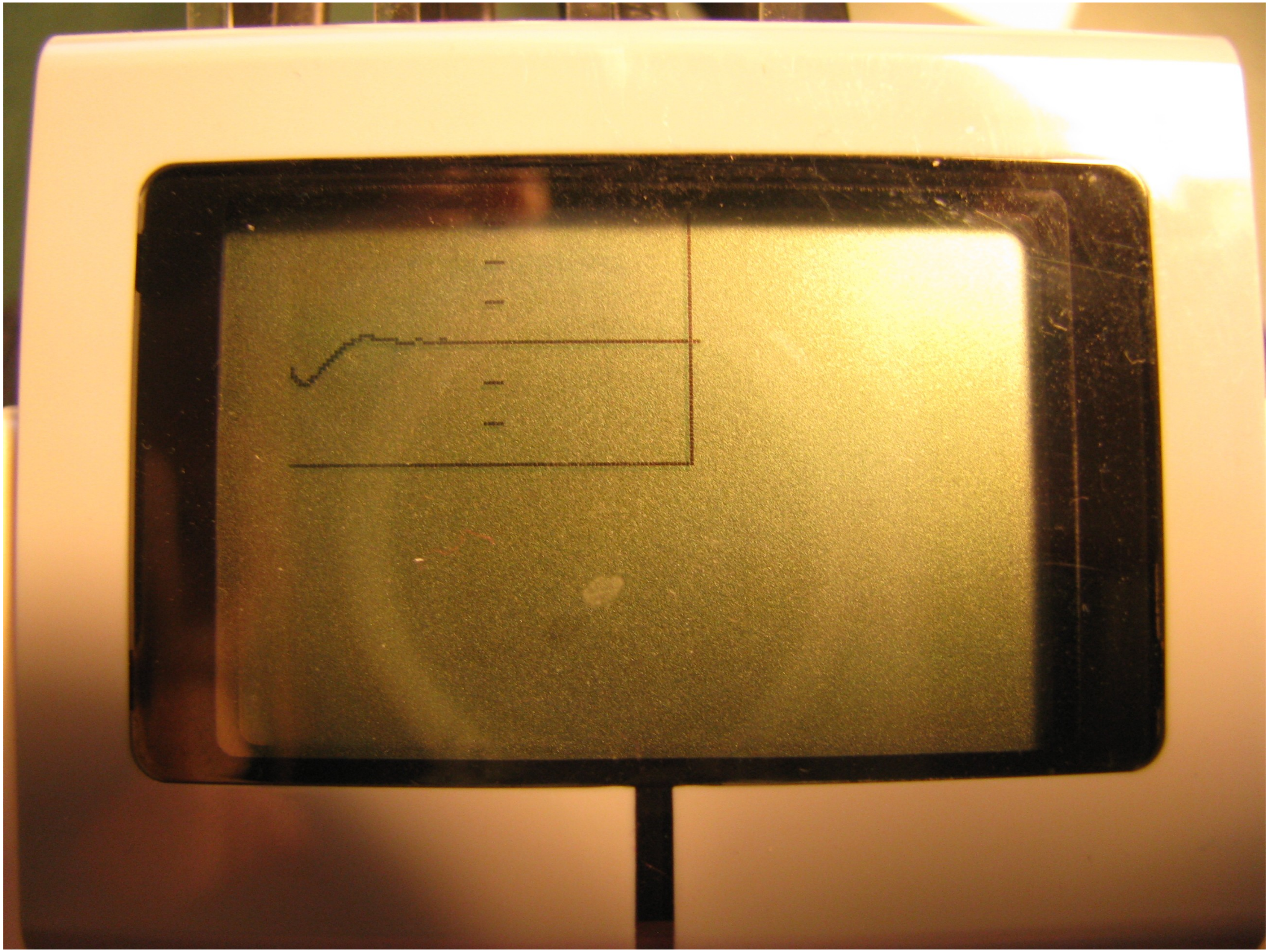
Inertia carries the robot further though, and the motors have to back up a little bit

This takes a little time

Your programs need to account for this

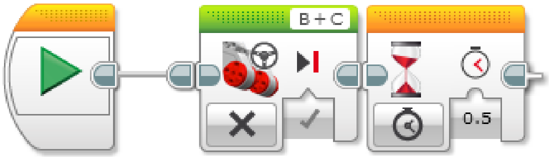
Video

Rotation sensor after brake

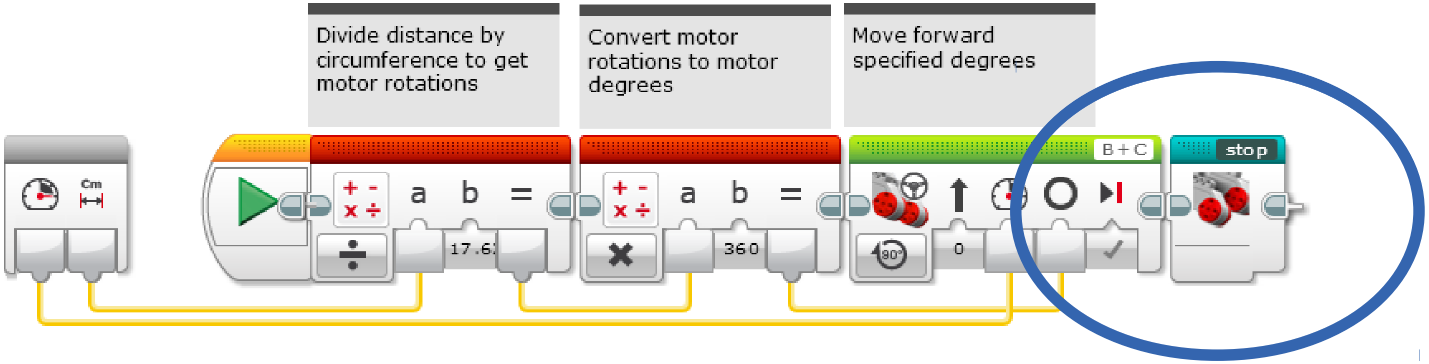


"Stop" My Block

A simple My Block to use for stopping



Place it at the end of any movement My Blocks where you want to be sure the robot has stopped


Divide distance by circumference to get motor rotations

Convert motor rotations to motor degrees

Move forward specified degrees

Odometry error

Odometry

Using distances and turn angles to navigate a robot is called “odometry”

It's useful, but depends on the quality of robot components

Mindstorms robots can have a lot of odometry error

Sources of odometry error

Friction

Gear slack

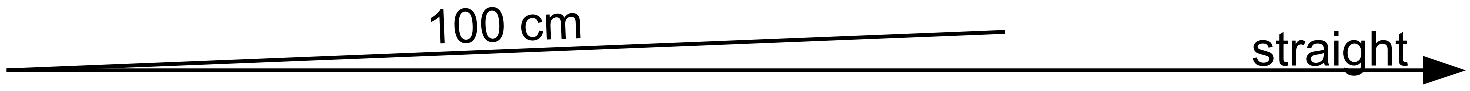
Wheel slippage

Battery charge

Timing issues

How significant?

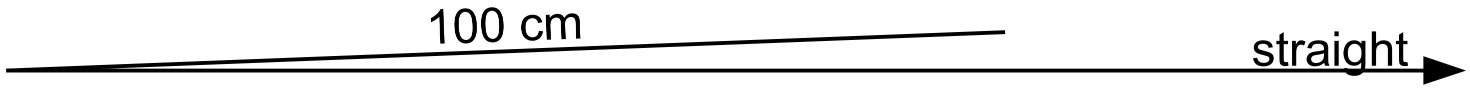
Suppose a robot travels 100 centimeters, but its heading is "off" by 1 degree:



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

How significant?

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

If you're trying to reach something small on the far side of table, you need more accuracy.

How significant?

LEGO NXT motors regularly have 5-10 degrees of “slack” in the internal gearing

A robot built with Mindstorms parts can easily have 5 degrees of “error” per turn

TIP: Run the same program multiple times, use Post-It flags to mark the results

Overcoming odometry error

Strategy: Use field elements for navigation

- Lines
- Mission models
- Walls
- Other

Strategy: Never make more than two turns without re-orienting with something on the field

Stopping at a field line

Light and color sensors
can be used to stop when
reaching certain places on the field

TIP: For Nature's Fury, the colored scoring area lines may not be thick enough to use reliably. Be sure to test carefully before relying on them.

Understanding light sensors

Light sensors have several different “modes”

- Color – used to detect specific colors
  - black, blue, green, yellow, red, white
- Ambient light – the amount of light reaching the sensor
- Reflected light – same as ambient light, but the sensor's LED is turned on

In all of these modes, external lighting can affect the readings

Reflected light mode

The light sensor returns a value from 0 to 100

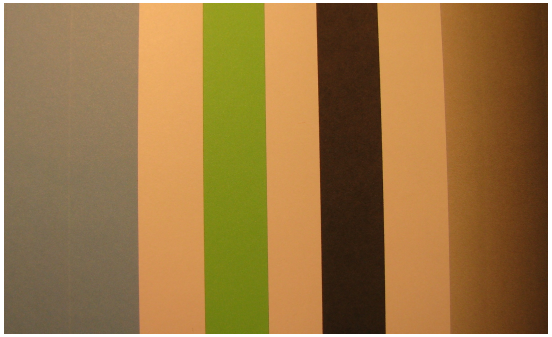
0 == sensor is receiving almost no light

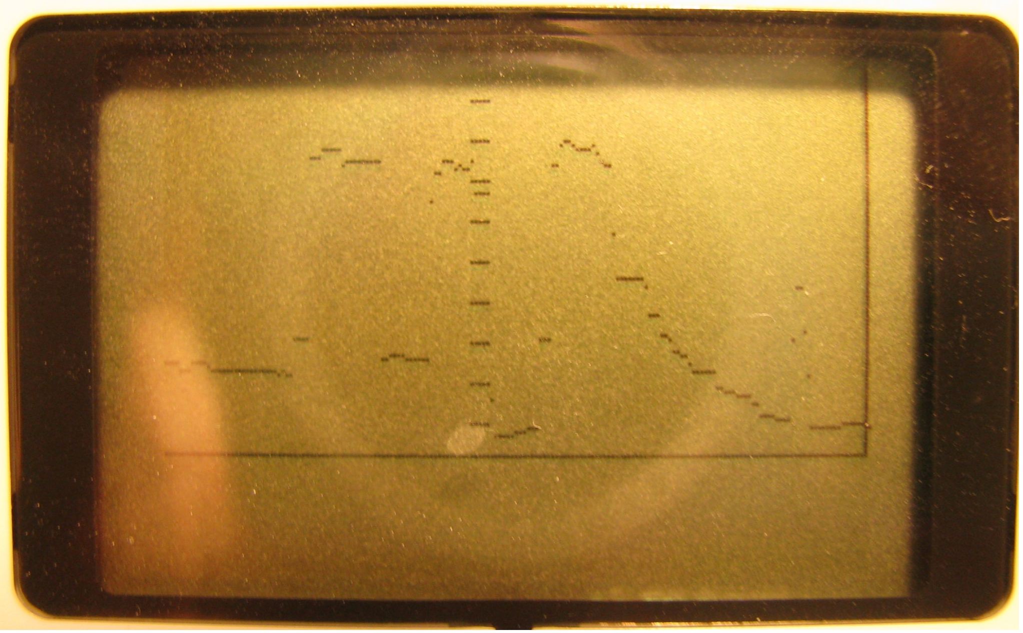
100 == sensor is receiving a lot of light

Use the robot to determine what the sensor is detecting

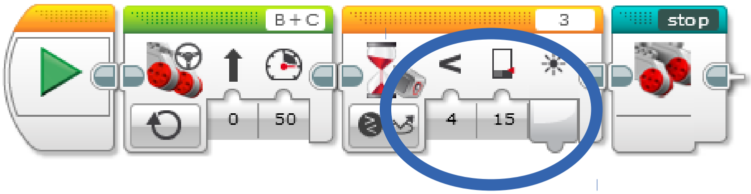
Light graph

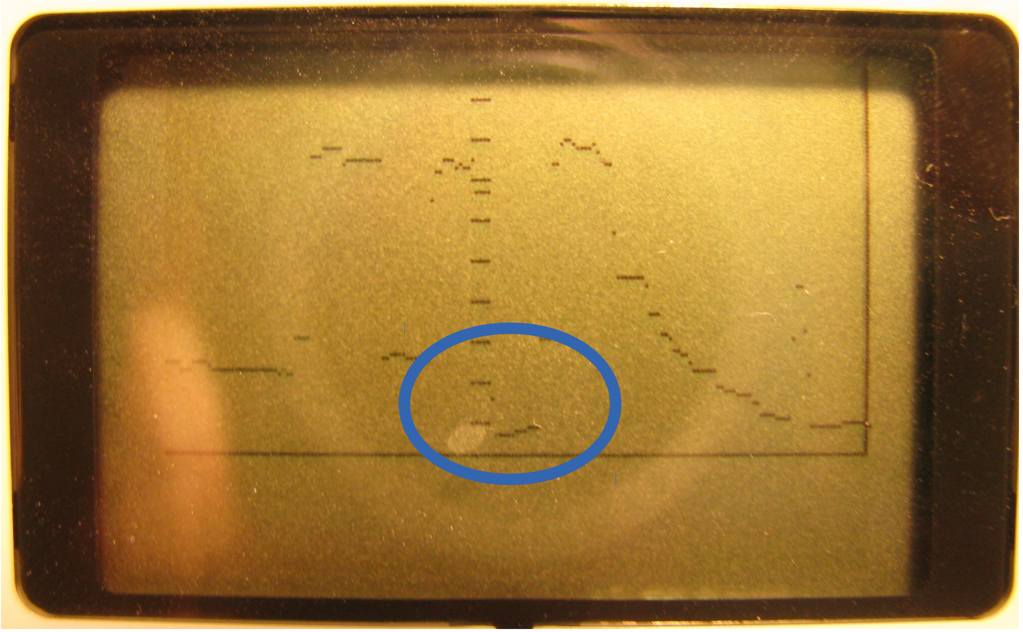
Tip: Write a program to graph light values as the robot moves





Stop at a black line





TIP: Our team has always used raw light values w/o any light sensor calibration

Following a line
(actually following an edge)

Line (edge) following

There are many ways to follow lines

Our team uses a simple proportional line follower to follow a boundary between light and dark areas

2013-10-28

More slides coming soon