North Texas FLL
Coaches' Clinics

Advanced FLL Programming

Patrick R. Michaud
pmichaud@pobox.com

October 2015

Goals

- Get more consistence performance
- Learn advanced programming techniques
- Share tips that have helped our team
- Point out traps that cause frustration

Topics

Specifying distances in centimeters

Programming and robot game strategy

Loops and sensor blocks

Moving along a heading with gyros

Line following / edge following

Understanding navigation error

Background

Hopefully you already know about...

- Compiling and downloading programs to EV3
- Motor / move blocks
- Wait blocks
- Touch sensors

Moving forward a distance
Review of My Blocks

Move forward a distance

Specify distances in linear units (inches or cm)

Need to know circumference of driving wheels

Several options:

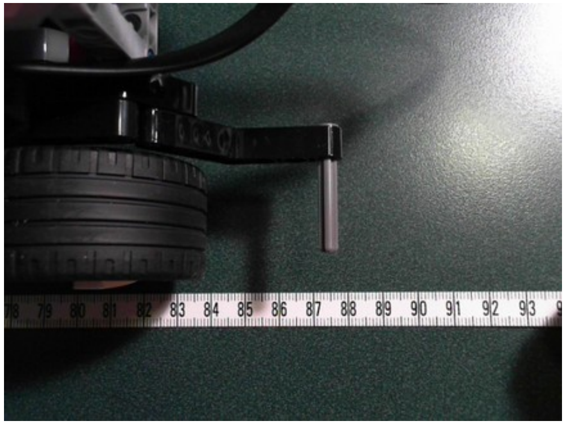
- Calculate from printed wheel diameter
- Measure wheel diameter
- Use robot to determine circumference (best!)

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

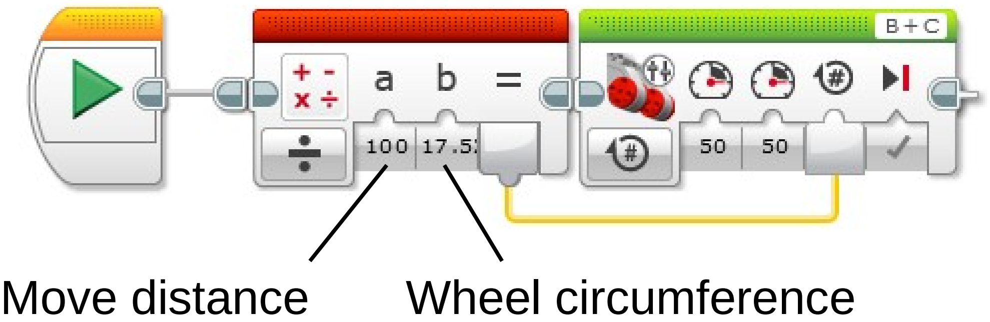
Move forward a distance

Start with empty program

Add a Math division block to calculate rotations

Add a Move block

Wire output of division to rotation input

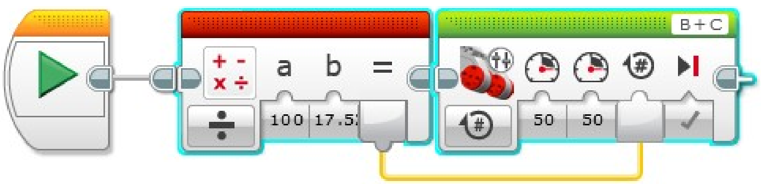


Test program to verify it works

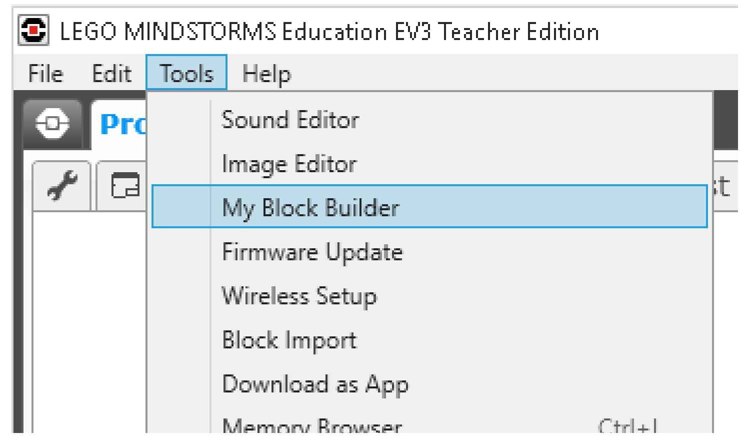
Adjust circumference value if distances are off

Make a My Block (review)

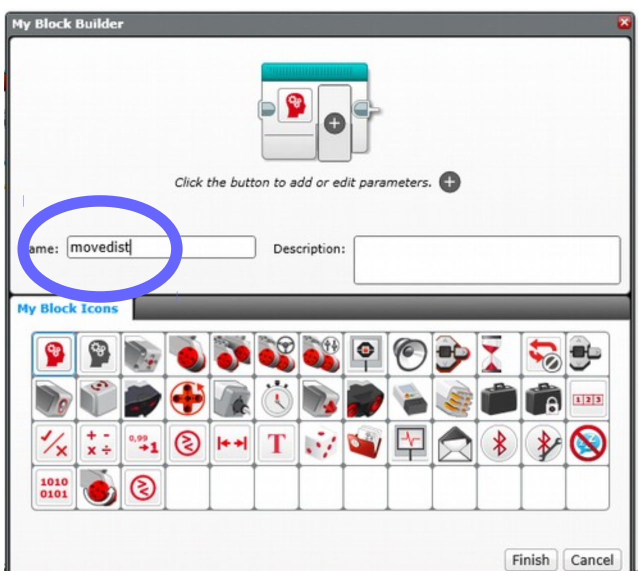
1. Select blocks



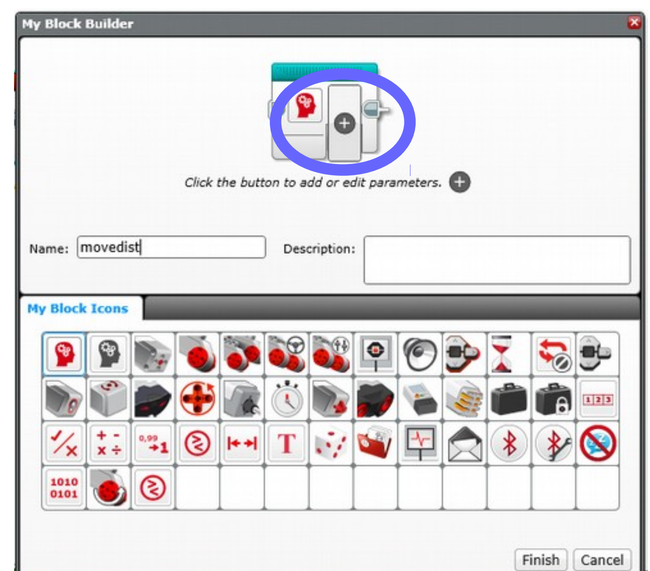
2. Tools → My Block Builder



3. Name the My Block

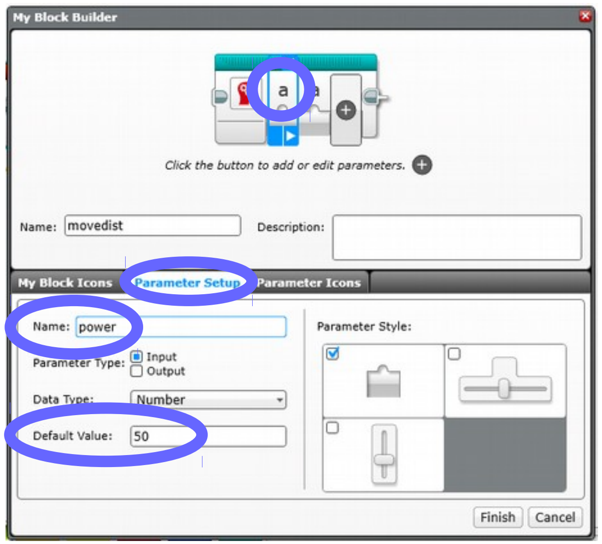


4. Add two parameters

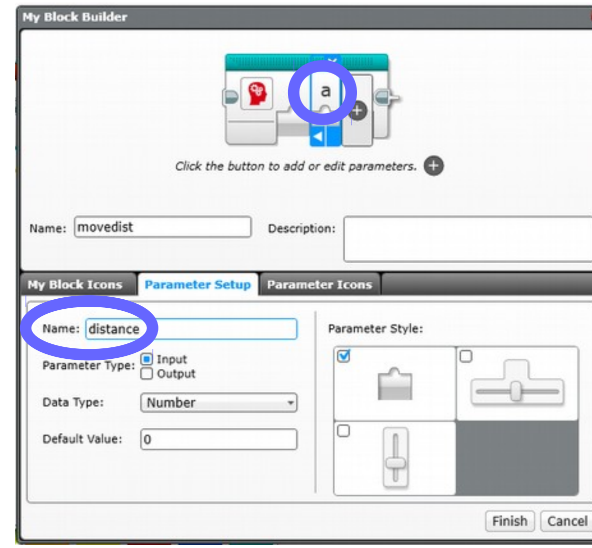


Make a My Block (review)

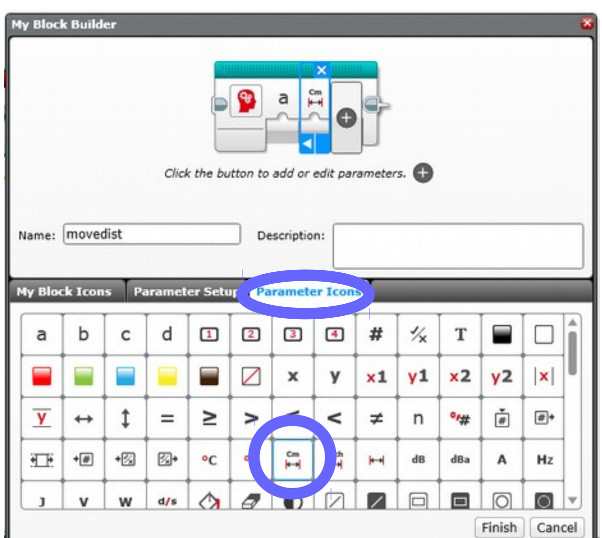
5. Set up power parameter



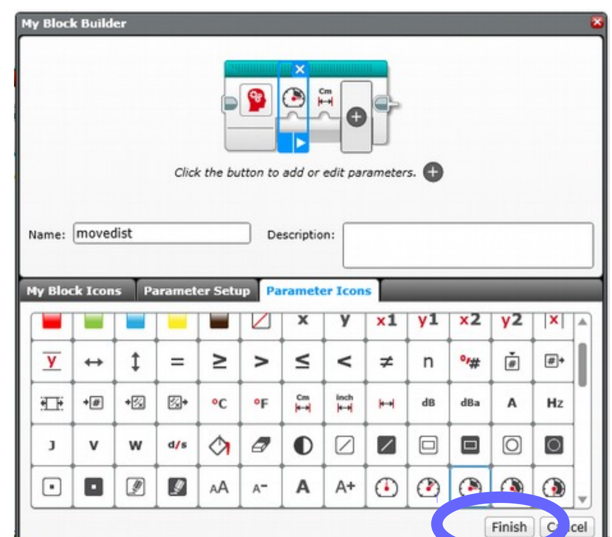
6. Set up distance parameter



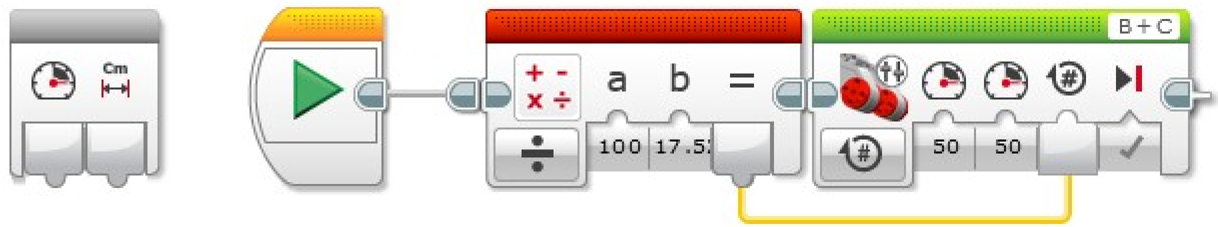
7. Select parameter icons



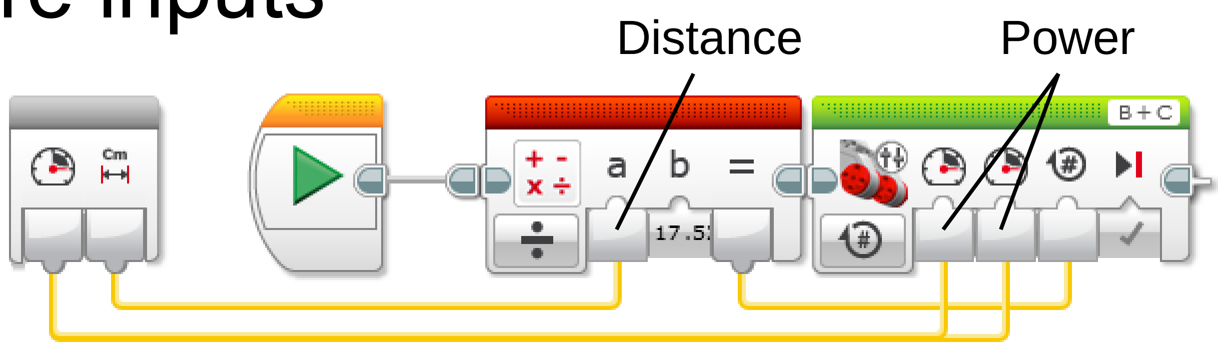
8. Click Finish



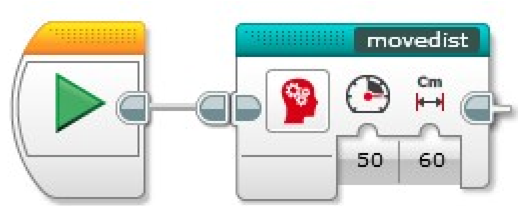
Make a My Block (review)



9. Wire inputs



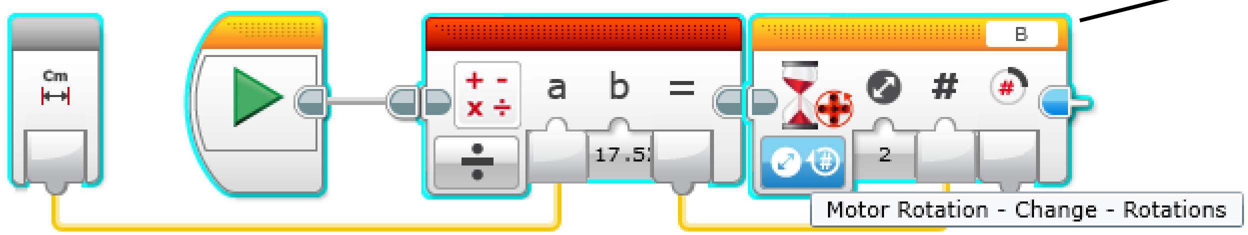
10. Test



Change power and distance settings here to test

Wait for distance My Block

Also useful: Wait for a distance (in cm)



Be sure to set correct motor port

Example usage:

Motors on

Wait 50cm

Wait for black

Stop

Robot game strategy

Consistency wins

Good programming and strategy are essential to consistently good performance

Programming overcomes the limitations of the hardware

Great robot + poor strategy → inconsistent scores

Fair robot + good strategy → consistent scores

Robot Game Strategy - Base

The robot must always start from Base

Base is the only place where changes can be made

Robot Game Strategy - Time

Matches are 2:30

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

Robot Game Strategy - reliability

Distance:

- Error 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 say 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* so that mistake
cannot happen again.

Tip: Start every mission from same spot

Put solid edges on robot

- Align robot with solid edges, not by sight-aiming
- Robot can always start with known location and heading
- Faster setup in Base between mission runs

Place flat edge against wall

Pick a marking to align robot

To save match time, always start from same spot

Navigation

A key to scoring is to move robot consistently

Things a program(mer) needs to know to navigate:

- Where the robot currently is
- How precisely you know where it is
- Where the robot is going
- What's in the way, or what can guide you there

Robot needs to be able to move in a straight line

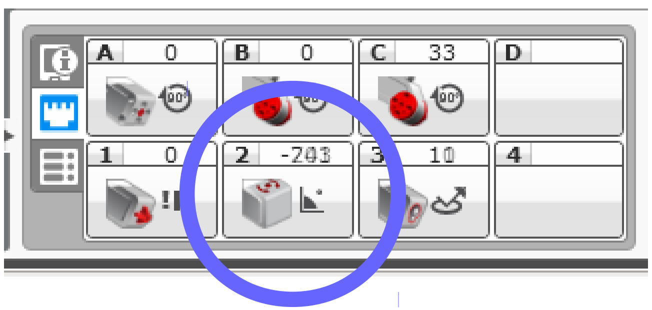
Moving in a straight line with gyro sensor

Gyro sensor detects rotation about an axis

It can help robot follow a straighter line (cf. driving a car)

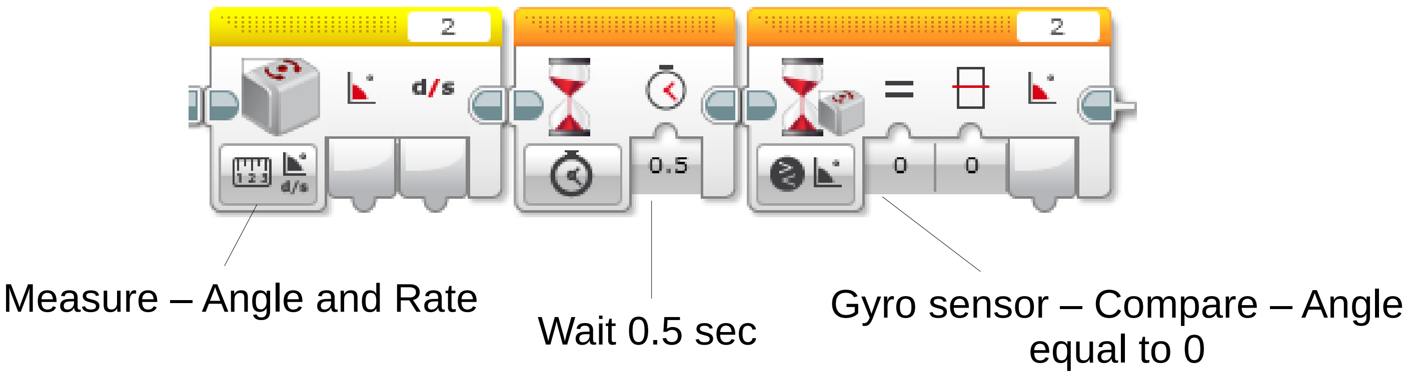
First must correct for sensor *bias* and *drift*

gyro sometimes shows movement even when still



Reducing gyro drift

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

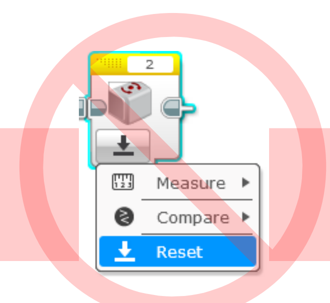


Perform this once at beginning of program

Requires 2-3 seconds to complete

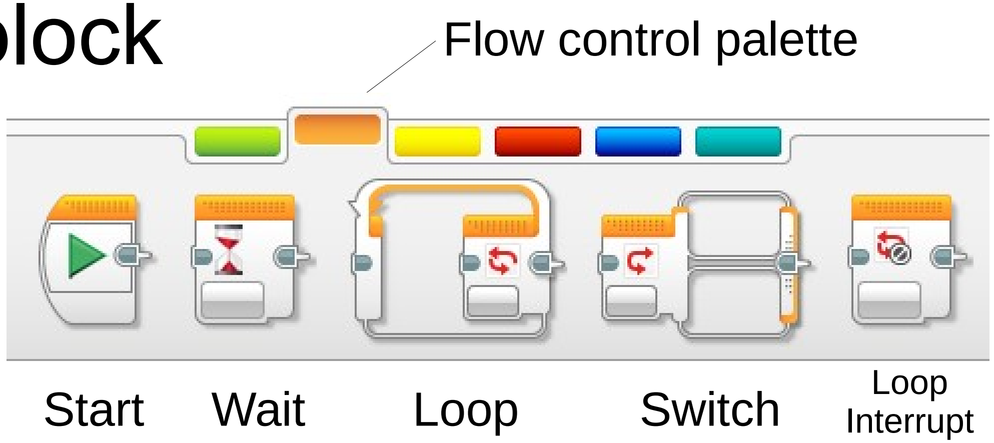
Gyro must be stationary while calibrating

Trap: “Gyro reset” block doesn't recalibrate gyro!

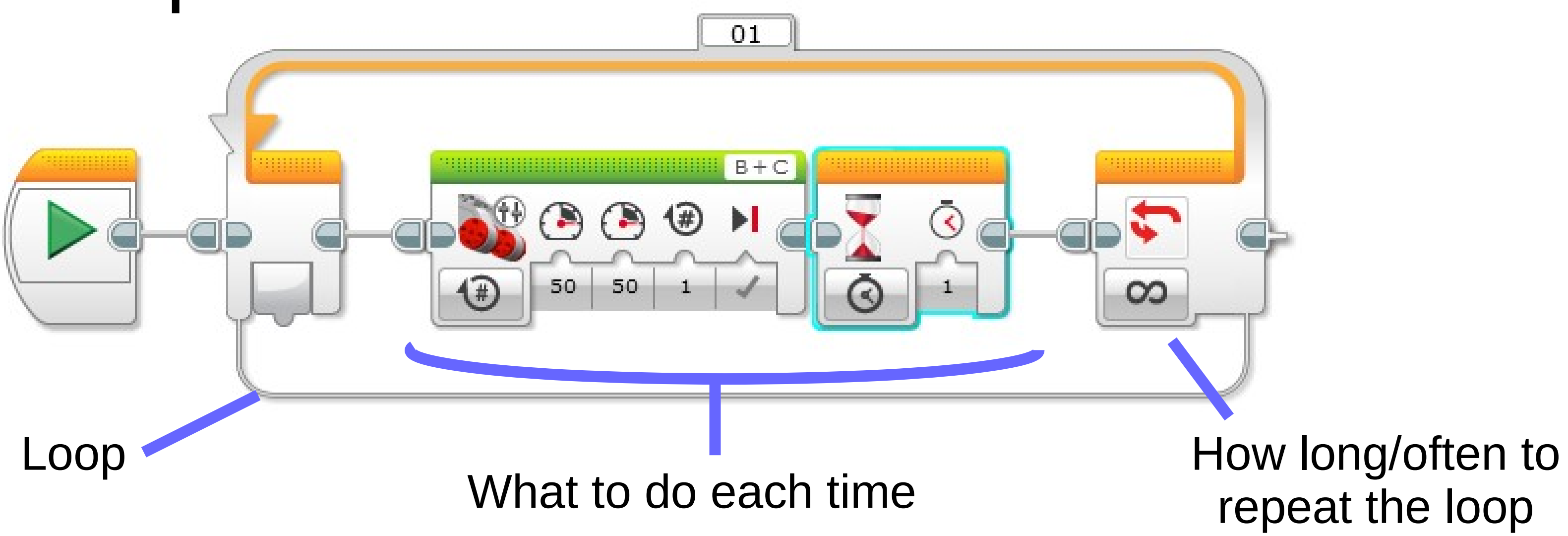


Loops

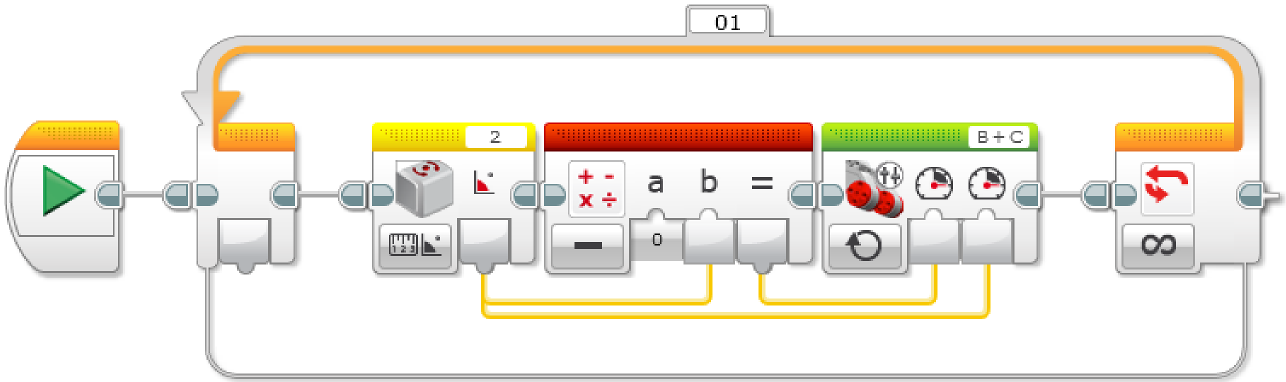
To do something repeatedly (like steering), use a “loop” block



A basic loop block



A gyro-following loop



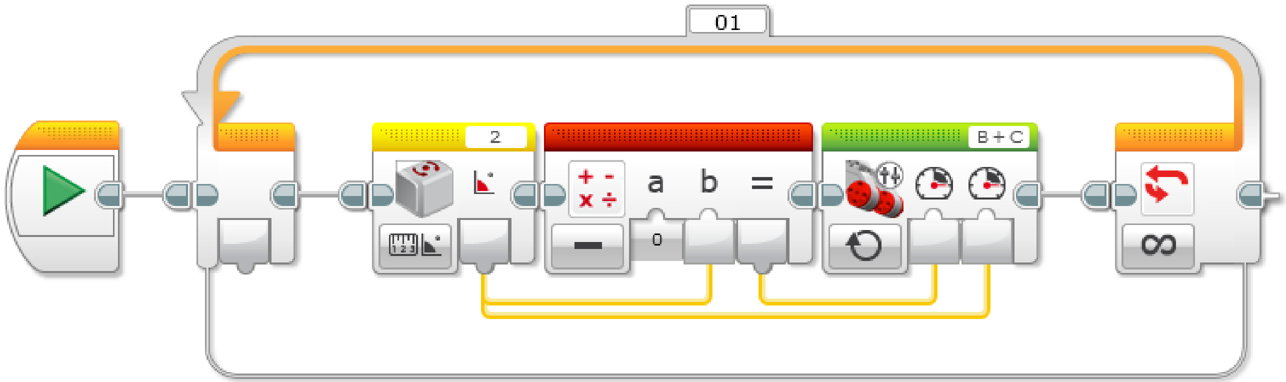
The gyro sensor block reads an angle

The math block flips the (+/-) sign of the gyro angle

One motor gets a negative value
the other gets a positive value

What happens?!?

A gyro-following loop



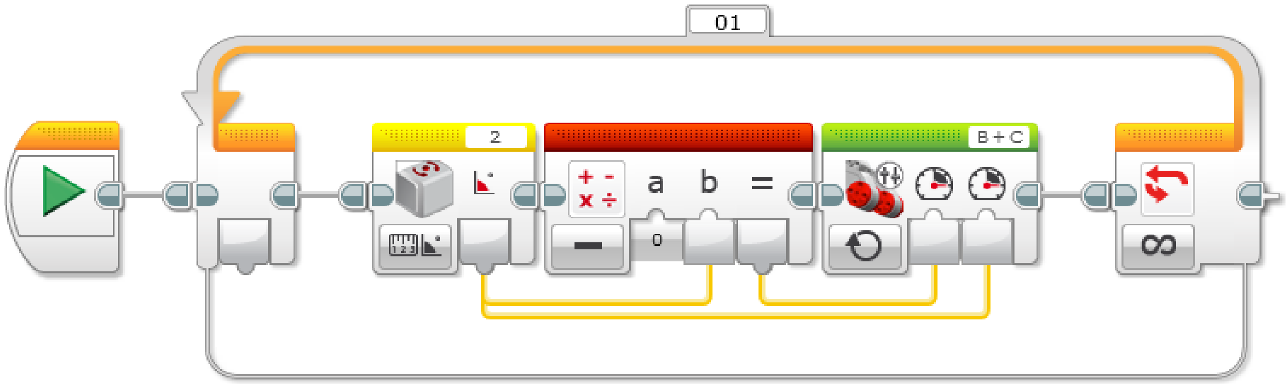
What happens when driving motors move in opposite directions?

What happens here when the gyro angle is zero?

What happens here when the gyro angle is not zero?

→ The robot always turns towards zero!

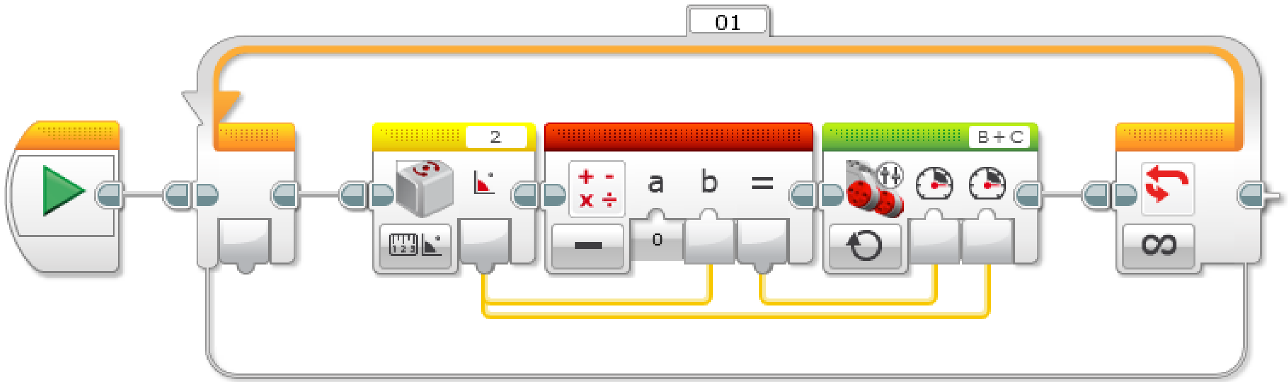
A gyro-following loop



Proportional control loop:

The power to the motors is proportional to how far the gyro sensor is away from zero (the “error”).

A gyro-following loop

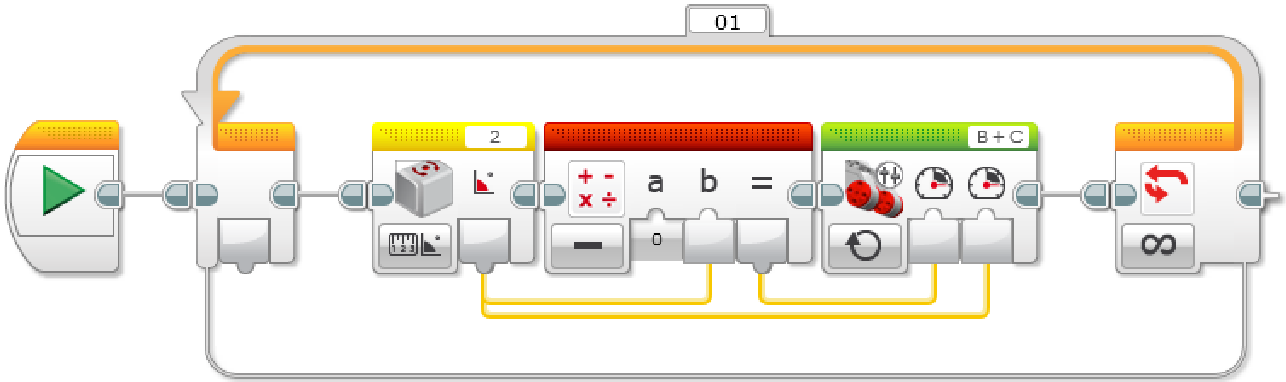


Try it yourself!

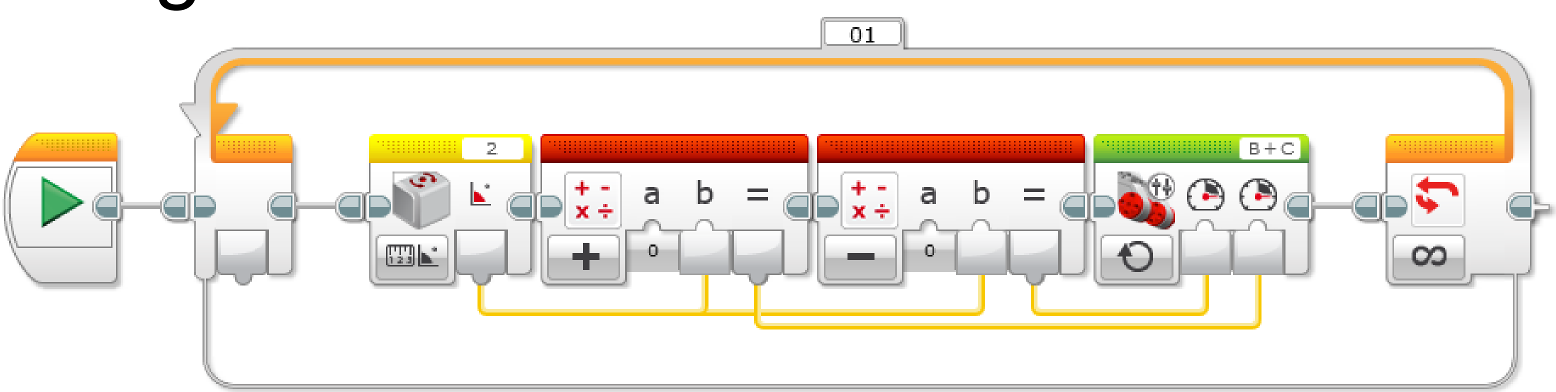
If the robot spins wildly out of control, try swapping the B+C inputs of Move Tank

You may need the gyro calibration code

A gyro-following loop

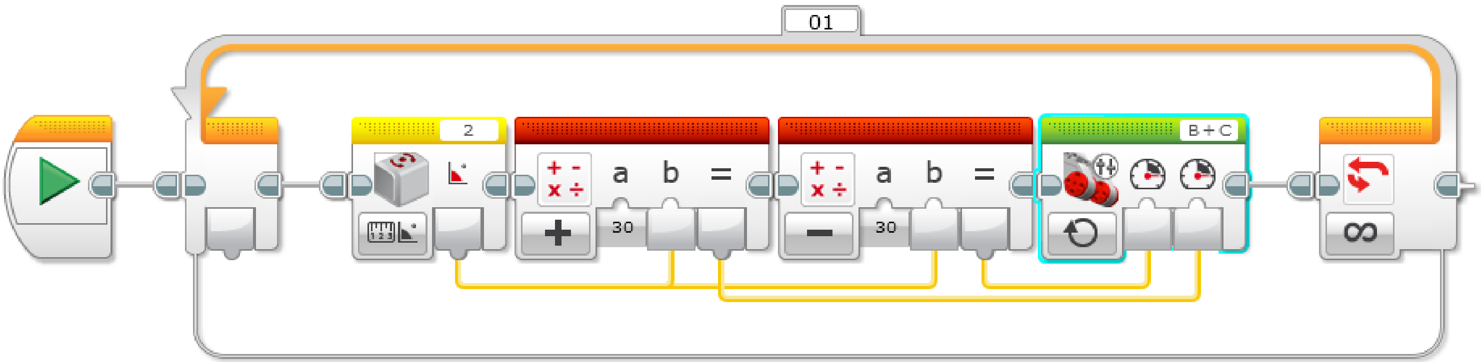


Let's add a math block to the loop that adds the gyro angle to zero:



Does this change anything?

A gyro-following loop



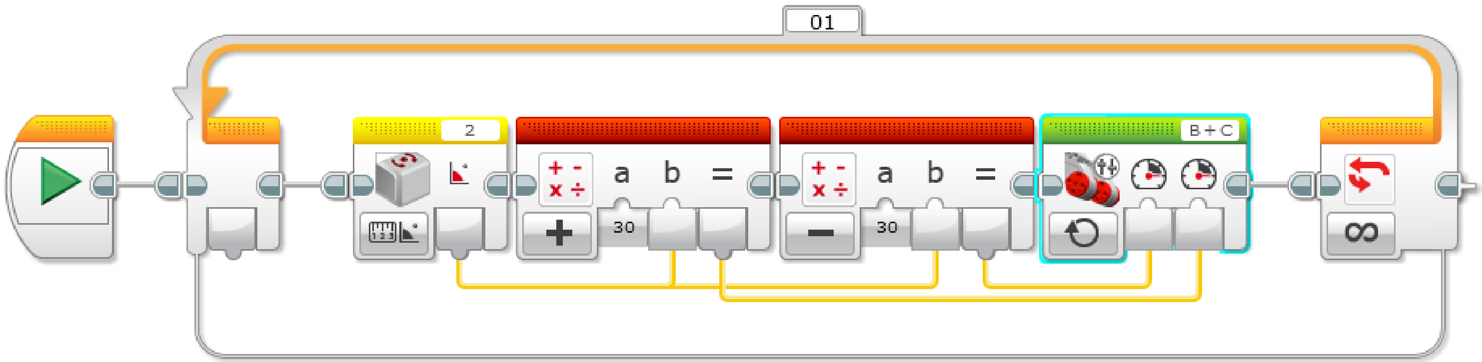
Now change the zeroes in the math block to 30.

What will this do...

...when the angle is zero?

...when the angle is not zero?

A gyro-following loop



When gyro angle is zero:

- both motors have a speed of 30
- robot moves straight ahead

When gyro angle is not zero:

- one motor moves faster than 30 and other moves slower than 30
- robot moves forward but turns toward zero angle

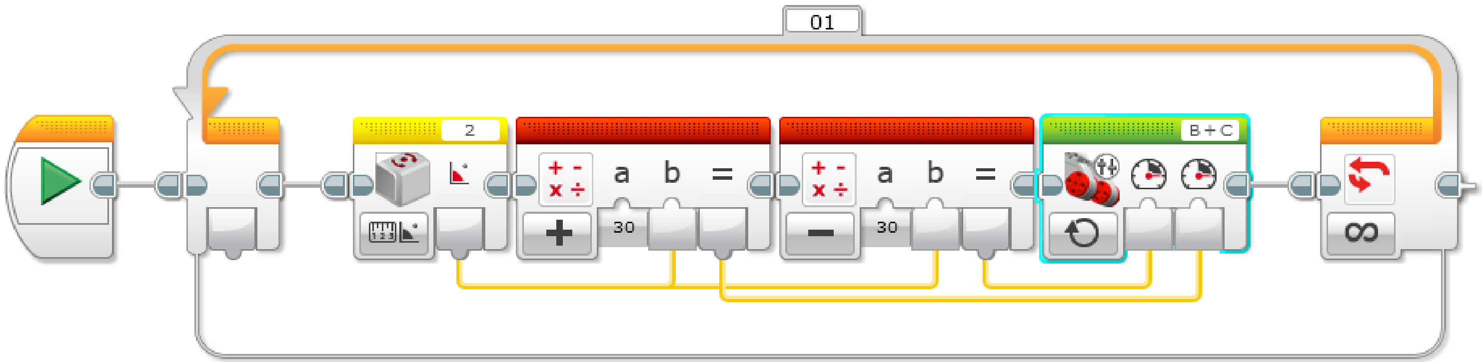
Fundamentals of turns

The robot turns when driving wheels move at different speeds

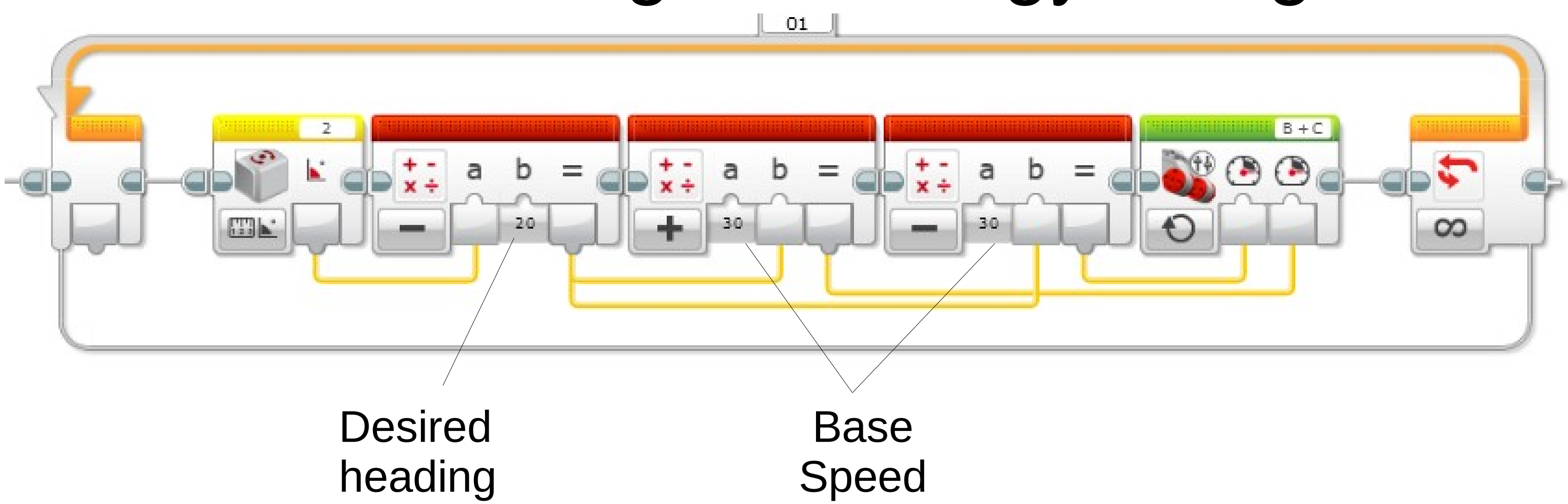
The robot turns towards the *slower* wheel

The greater the difference in speeds, the tighter the turn

A gyro-following loop



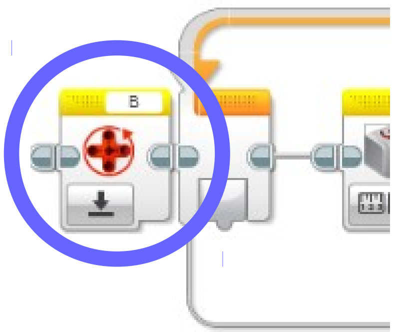
To follow a gyro angle other than zero, subtract the desired heading from the gyro angle:

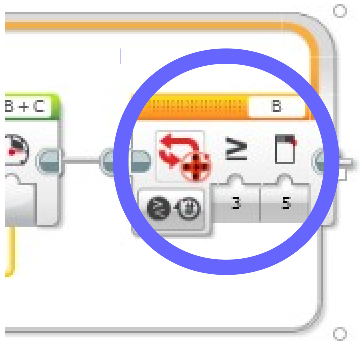


Exiting the loop - version 1

One way to exit the loop

- Add a “reset motor” block before the loop
- Tell the loop to exit based on motor rotations

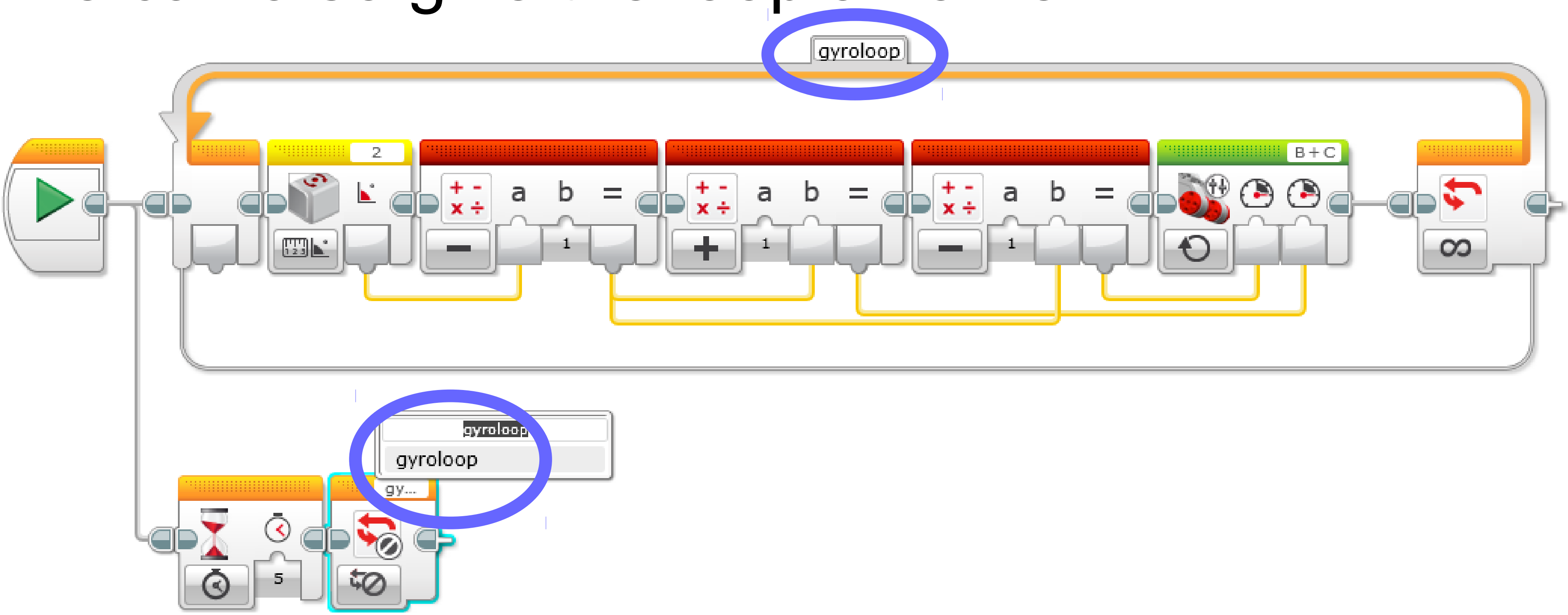




- Be sure to set the ports to a driving motor!

Exiting the loop - version 2

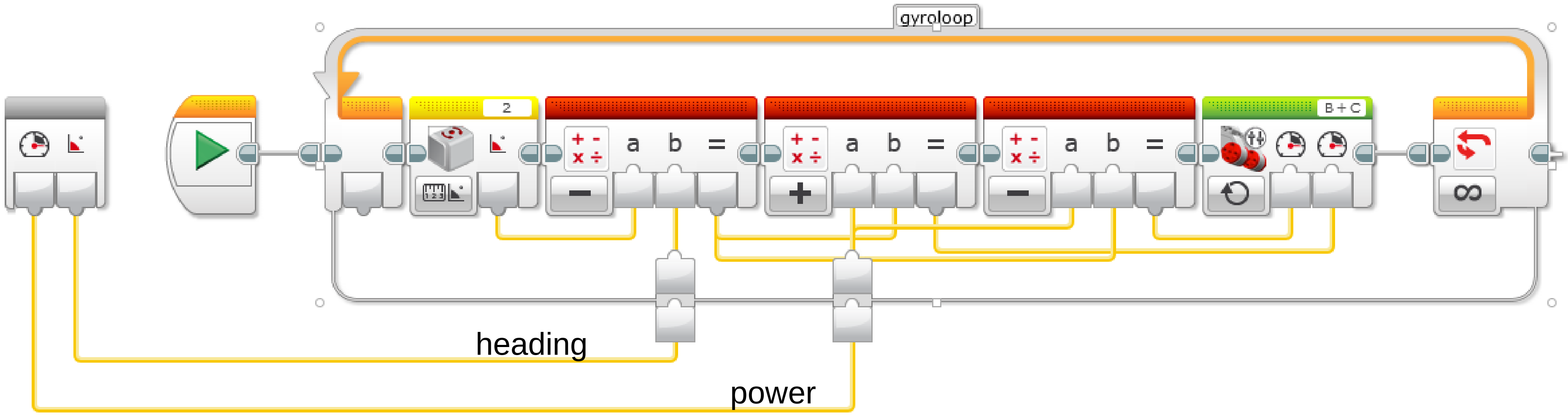
We can also give the loop a name

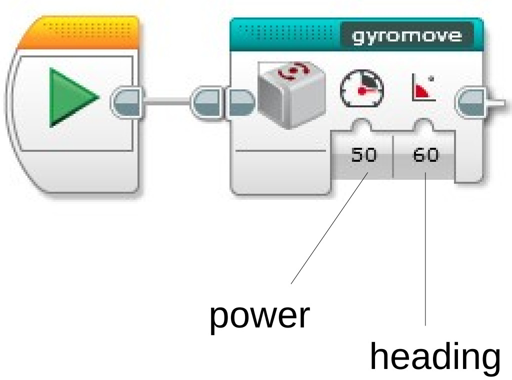


and use a "Loop interrupt" block in parallel to cause it to exit

A gyro-following My Block

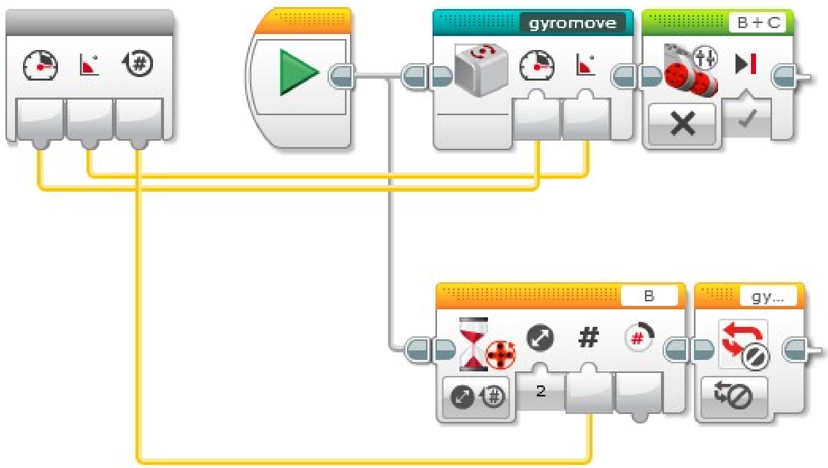
My Block to follow a heading until interrupted:



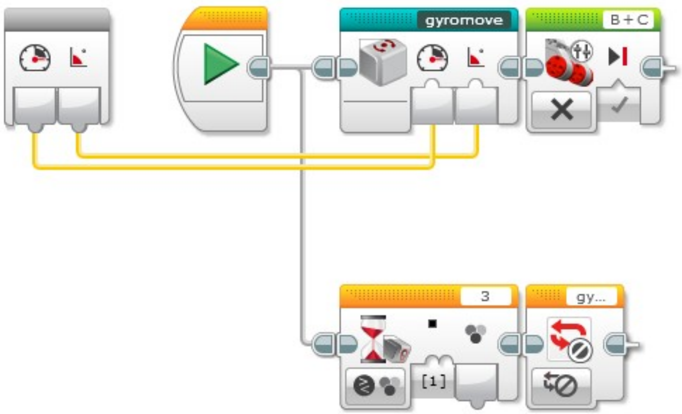


Example movement

Follow heading for rotations



Follow heading until black



Intermission

Navigation error

Using distances and turn angles for navigation is called “odometry”

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

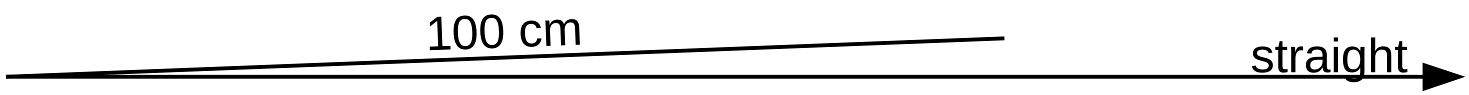
Mindstorms robots can have a lot of odometry error

Sources of odometry error

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

Small angles lead to large offsets

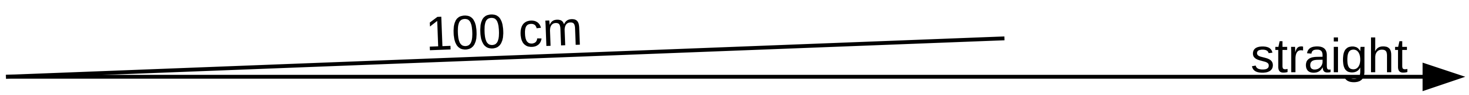
Suppose a robot travels 100 centimeters, but its heading is "off" by 1 degree



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

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 100cm?

A: 1.74cm

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

Overcoming error

Strategy: Use field elements for navigation

Lines	Walls
Mission Models	Other

If your robot can find a line, wall, model, or something on the other side of the field, you accurately know its location.

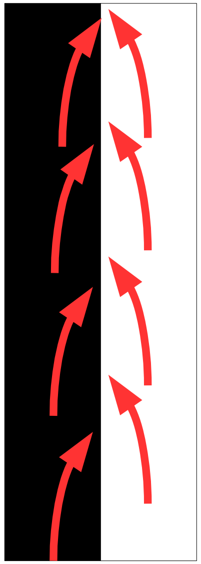
Our guideline: Never make more than two turns without re-orienting the robot using something on the field.

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

Understanding LEGO light sensors

- Light 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
- Sensor should 0.5cm to 2.0cm from surface
- Shielding helps a lot

Reflected light mode

The sensor returns a value from 0 to 100

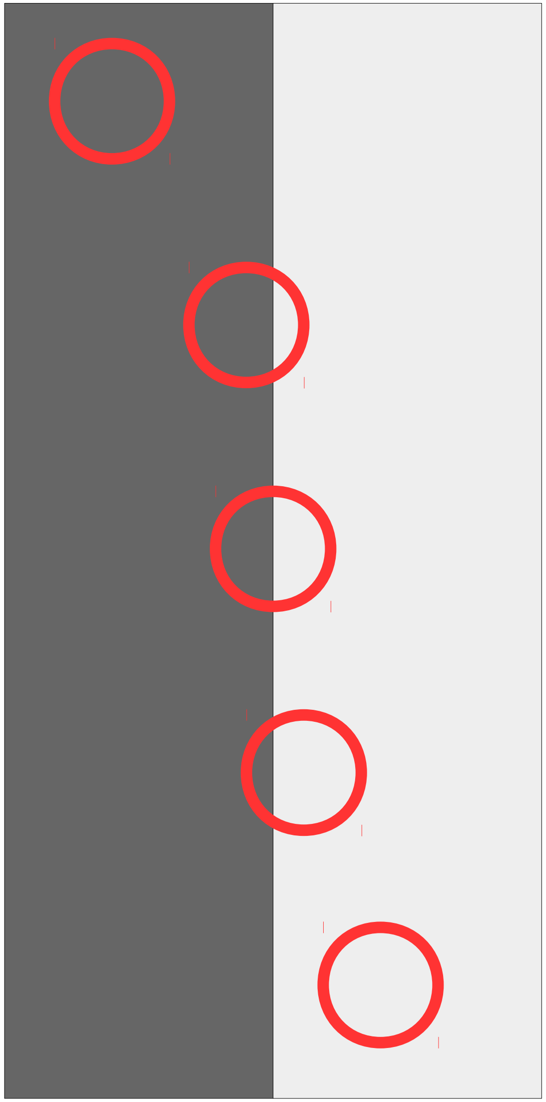
- 0 == sensor receiving almost no light
- 100 == sensor receiving a lot of light

Use port view to see what the robot is sensing

Reflected light mode

What sorts of values would the sensor see?

Proportional Control!



5 – turn right a lot

20 – turn right a little

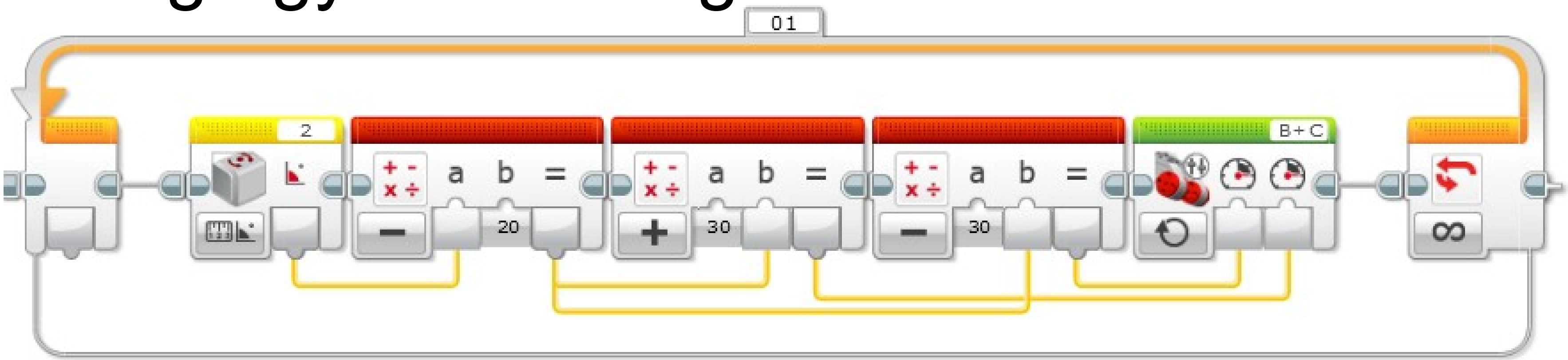
35 – go straight

45 – turn left a little

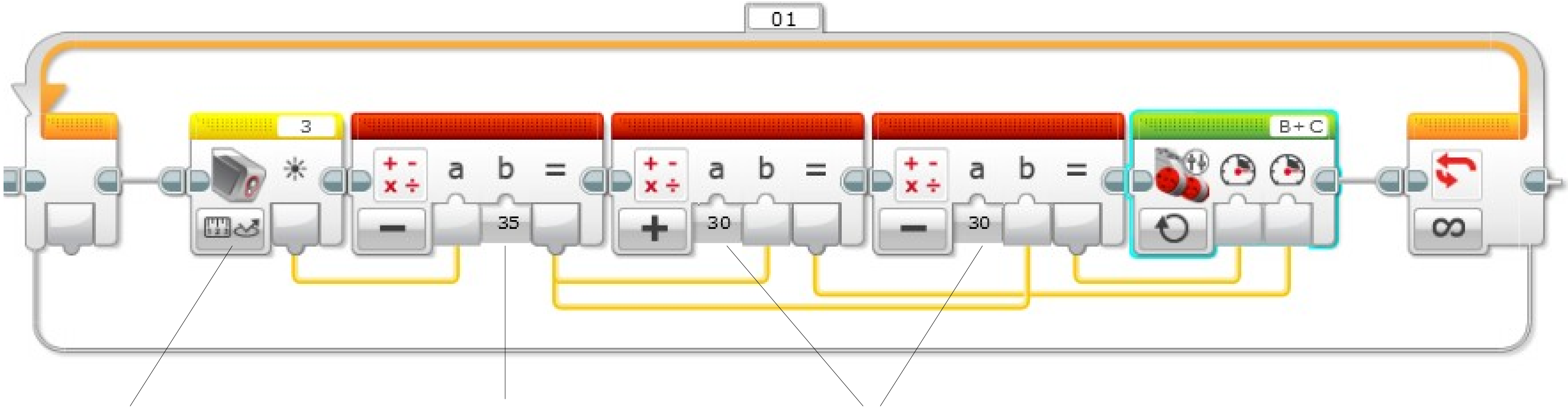
58 – turn left a lot

Proportional edge following

Change gyro-following sensor



to reflected light sensor



Measure
Reflected light

Value of “edge”
midpoint

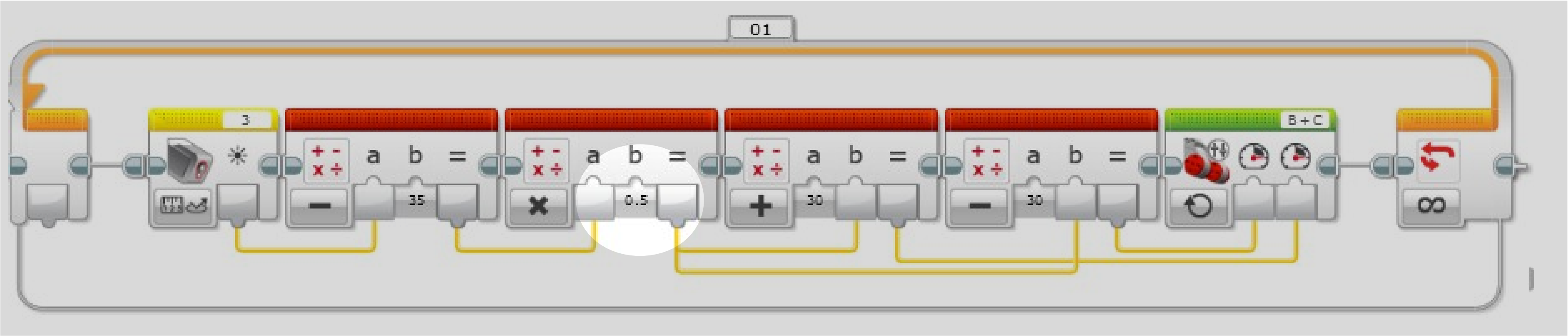
Base forward
speed

Proportional edge following w/gain

Sometimes you also want a “gain” factor

Higher gain → sharper turns

Lower gain → shallower turns



If robot is “waggling”, decrease gain

If robot isn't finding the line, increase gain

Proportional edge following

The light sensor must be in front of the driving wheels for edge for edge following to work

With a little tuning, a robot can very precisely follow a line

More stuff goes here

Thank you!

Questions?

Patrick R. Michaud
pmichaud@pobox.com
republicofpi.org

Join the NorthTexasFLL group!