FLL Workshop – Day 3
Intermediate FLL Programming

Patrick R. Michaud
pmichaud@pobox.com

University of Texas at Dallas
June 2017

Goals

Get more consistence performance

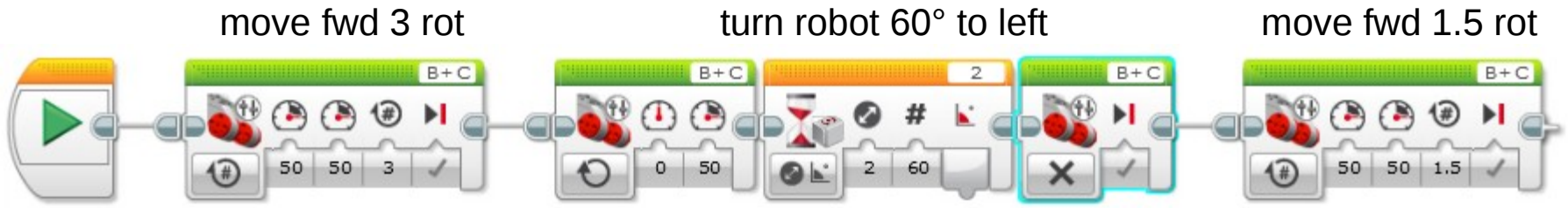
Learn advanced programming techniques

Share tips that have helped our team

Point out traps that cause frustration

Simple My Blocks - turning

A program to move, turn, then move:

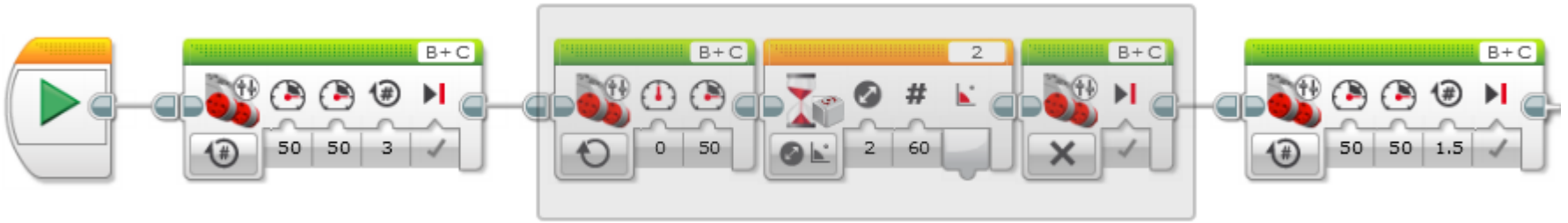


Adding three blocks for every turn will get tedious

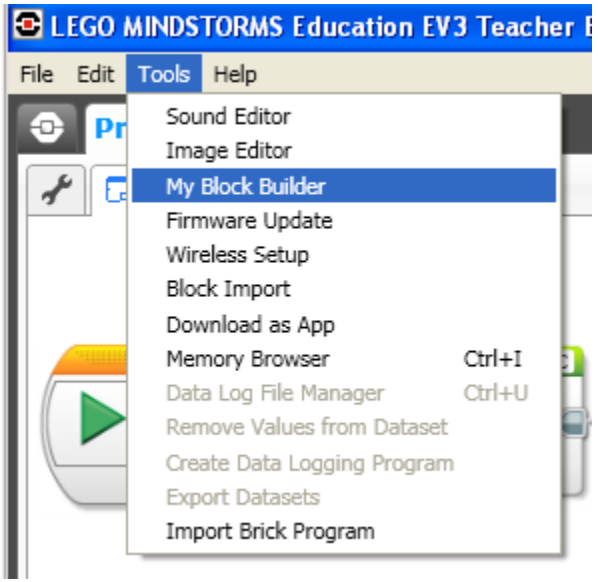
We can create custom blocks called “My Blocks”

Creating a My Block

Start by drag-selecting the blocks to be used



Select “My Block Builder” from the Tools menu



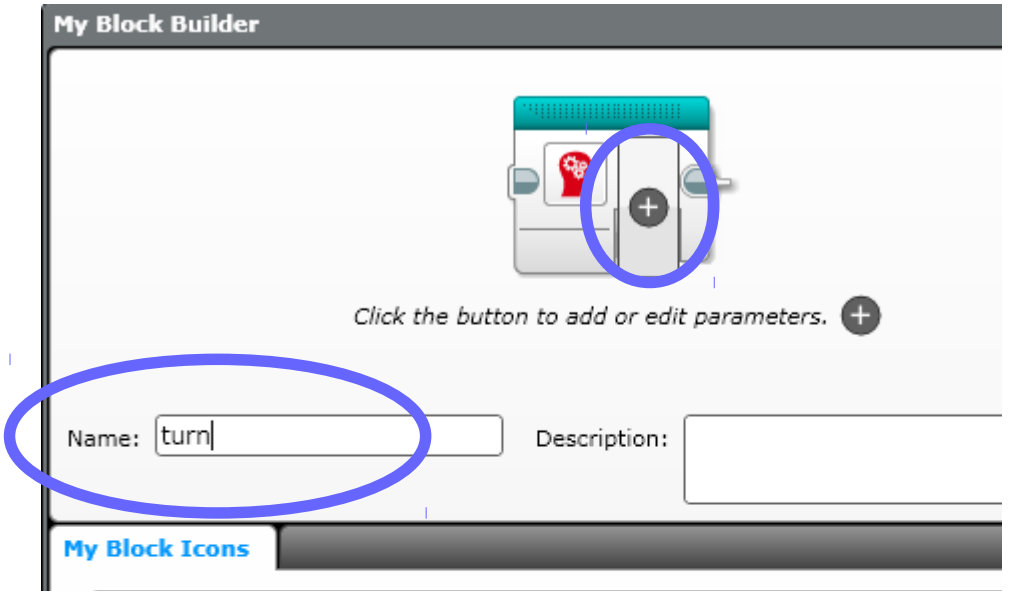
Creating a My Block

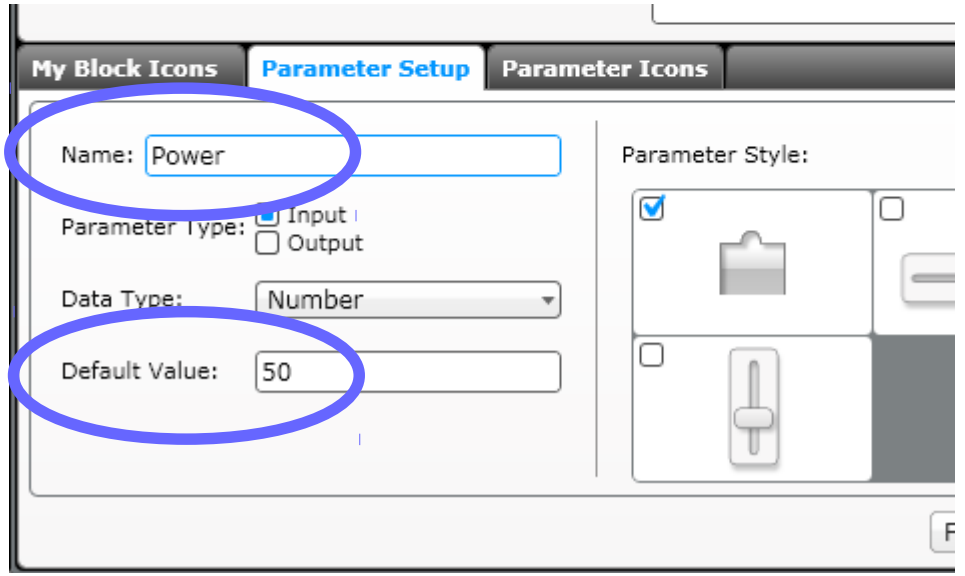
Give the My Block a name

Click the “+” to add a parameter

Click “Parameter Setup”

- Name parameter “Power”
- Give it a default of 50



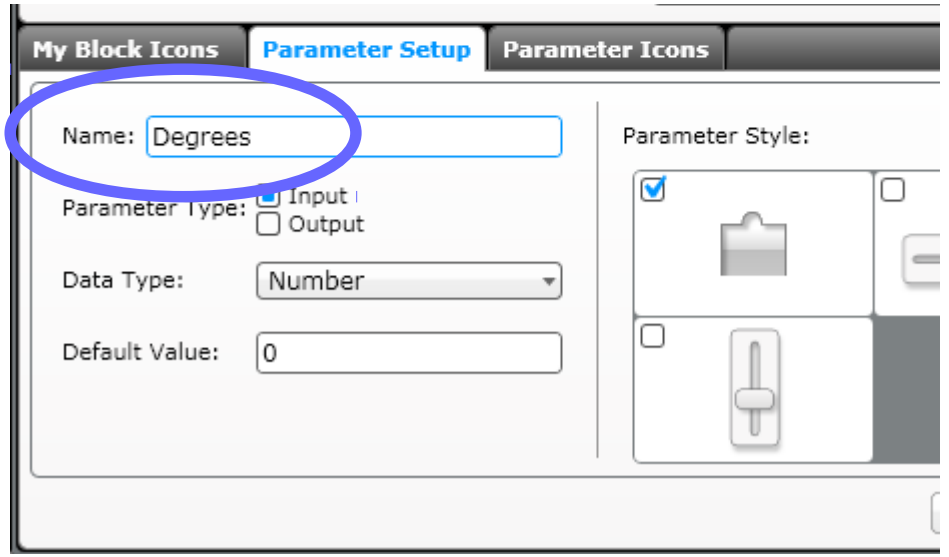


Creating a My Block

Click “+” again to create another Parameter

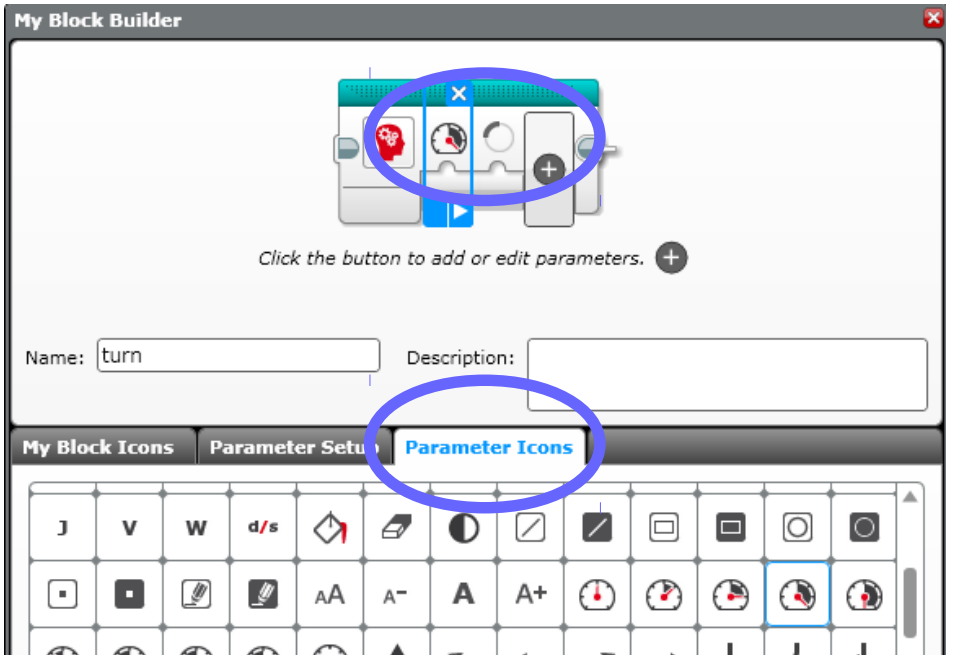
Name it “degrees”

Default value 0



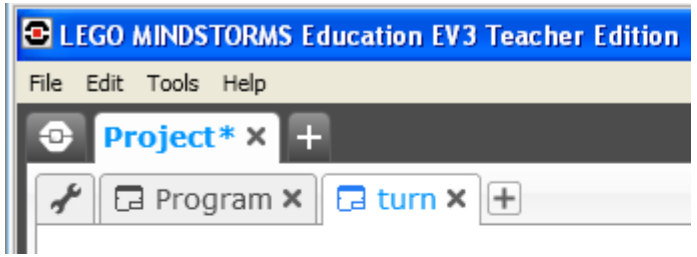
You can change the parameter icons to be more meaningful

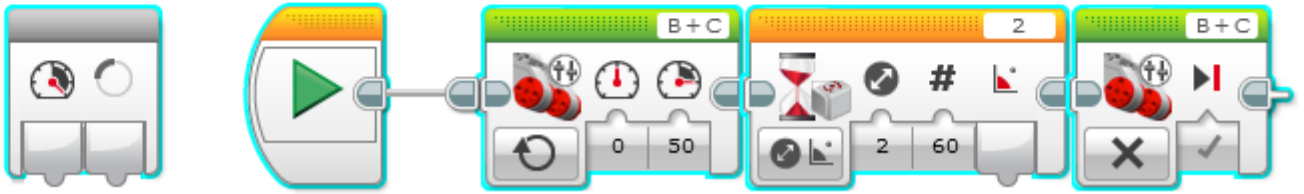
Click “Finish”



Creating a My Block

A new program is created for the My Block



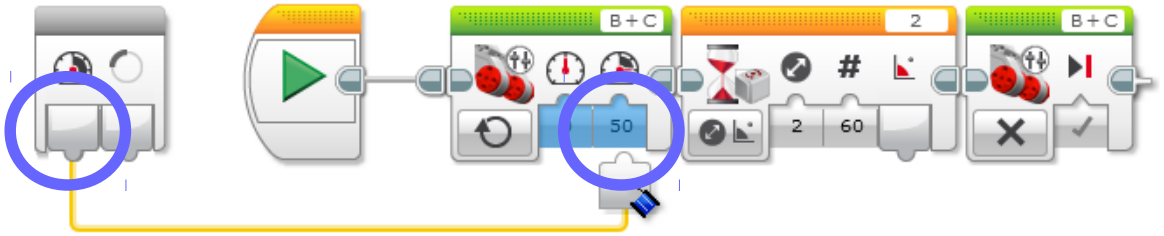


The grey block on the left has our “parameter inputs” for power and degrees

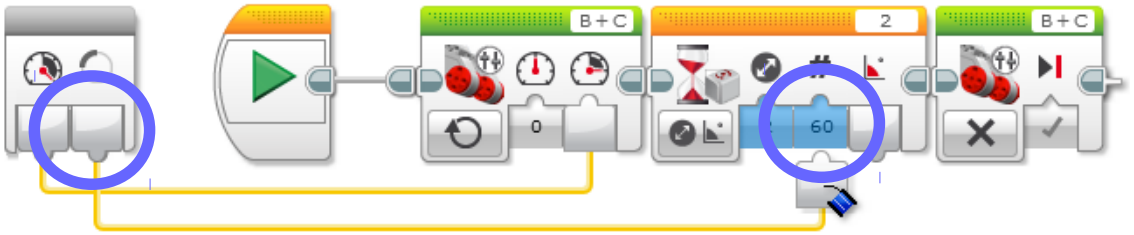
We have to wire them to the appropriate blocks

Creating a My Block

"Drag" a wire from the "Power" parameter to the C motor power input

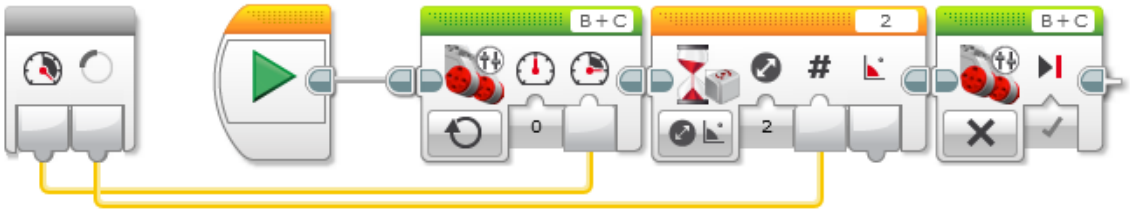


Drag a wire from the "Degrees" parameter to the degrees input of the wait block

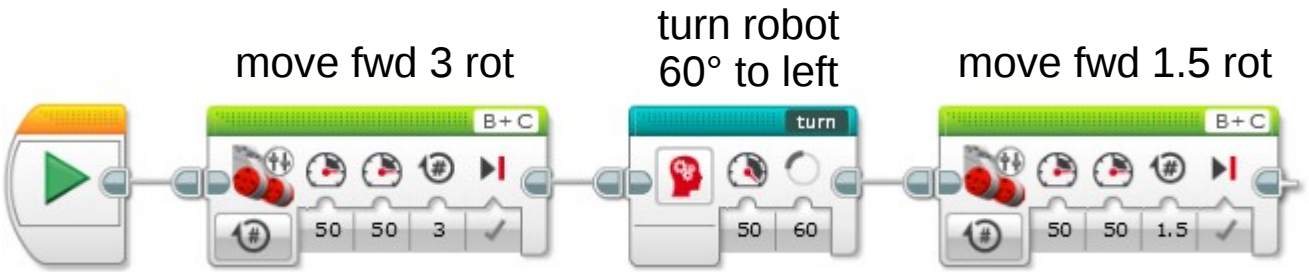


Creating a My Block

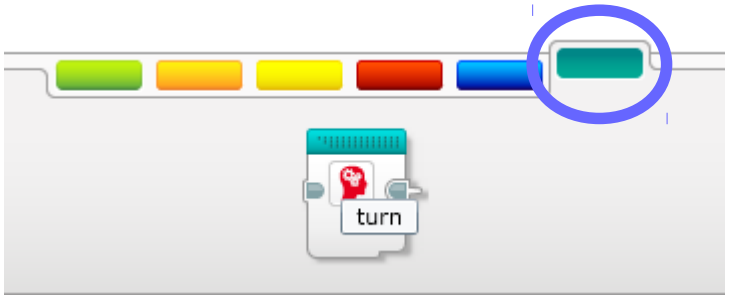
We now have a “My Block” for turns



Our original program has the three-block sequence replaced with our “turn” block

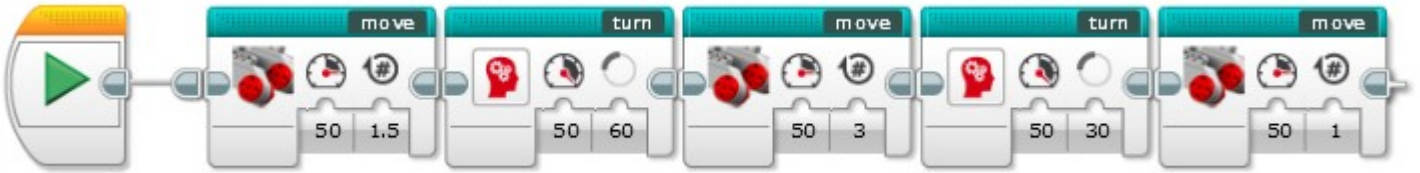


Our turn block appears in the My Blocks palette



My Blocks

Easier to read and build program



Suggestions:

- Create a My Block for each “mission”
- Combine mission My Blocks into “trip” My Blocks
- Create standard My Blocks for
  - Moving forward and backward, turning
  - Stopping
  - Motorized attachments
  - Initialization

Stop stop stop

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, so the motors have to back up a bit

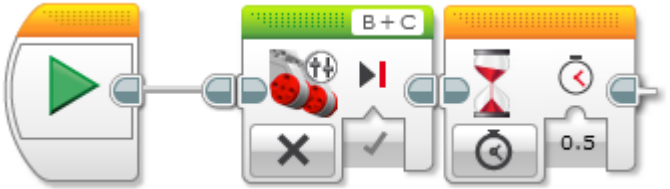
This takes a little time

If your program immediately goes to next action, robot will be inconsistent

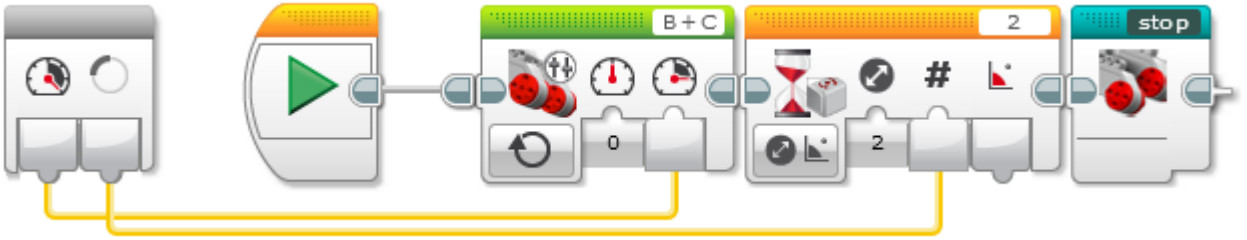
Create a “stop” My Block

Add a short wait after every movement

A simple “stop” My Block:



Can use this whenever you want to be sure the robot is stopped:



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

Thank you!

Questions?

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

Join the NorthTexasFLL group!