

V5 Coding Studio: Movement

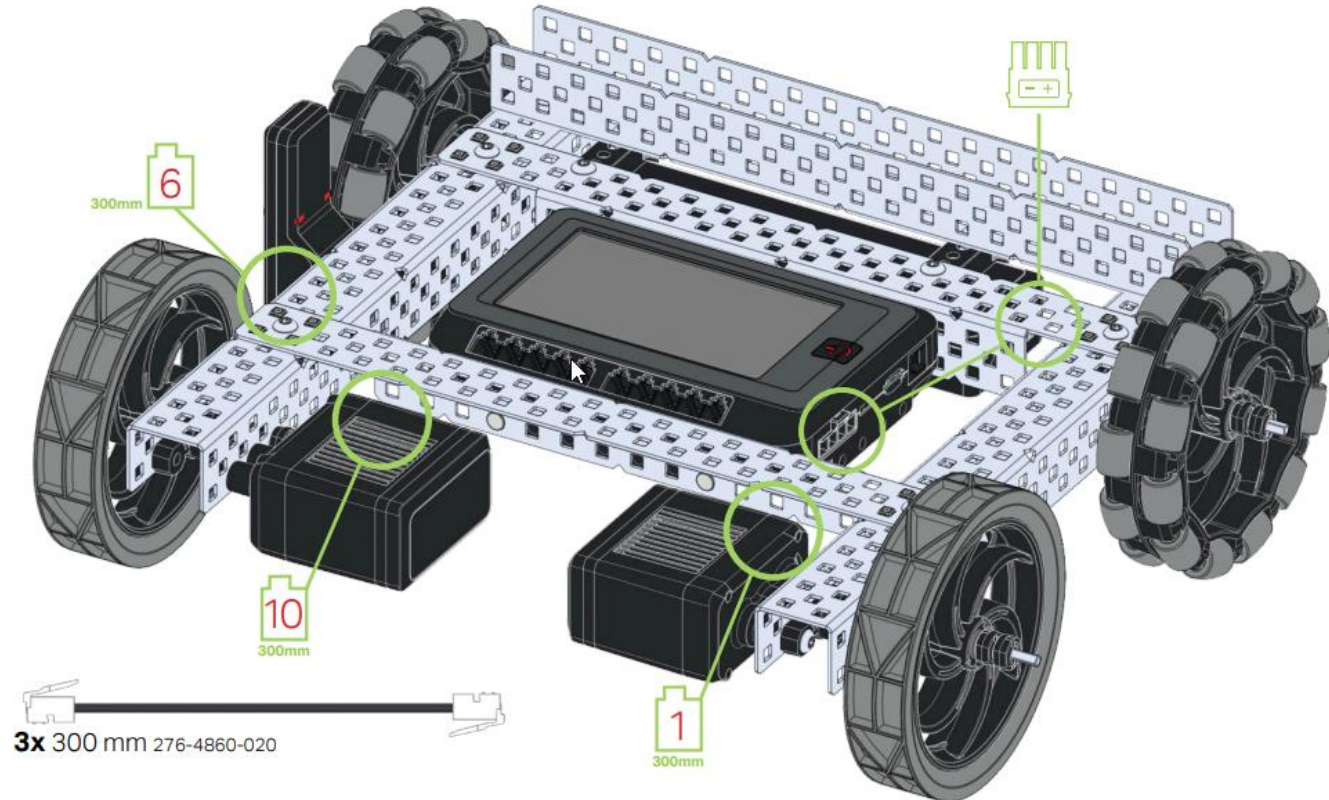
Motor Setup

rotateFor()

Turn

Using the Clawbot

- Clawbot default values
 - RightMotor = Port 10
 - LeftMotor = Port 1
- Motors come with the green (18:1 or 200 RPM) gear cartridge

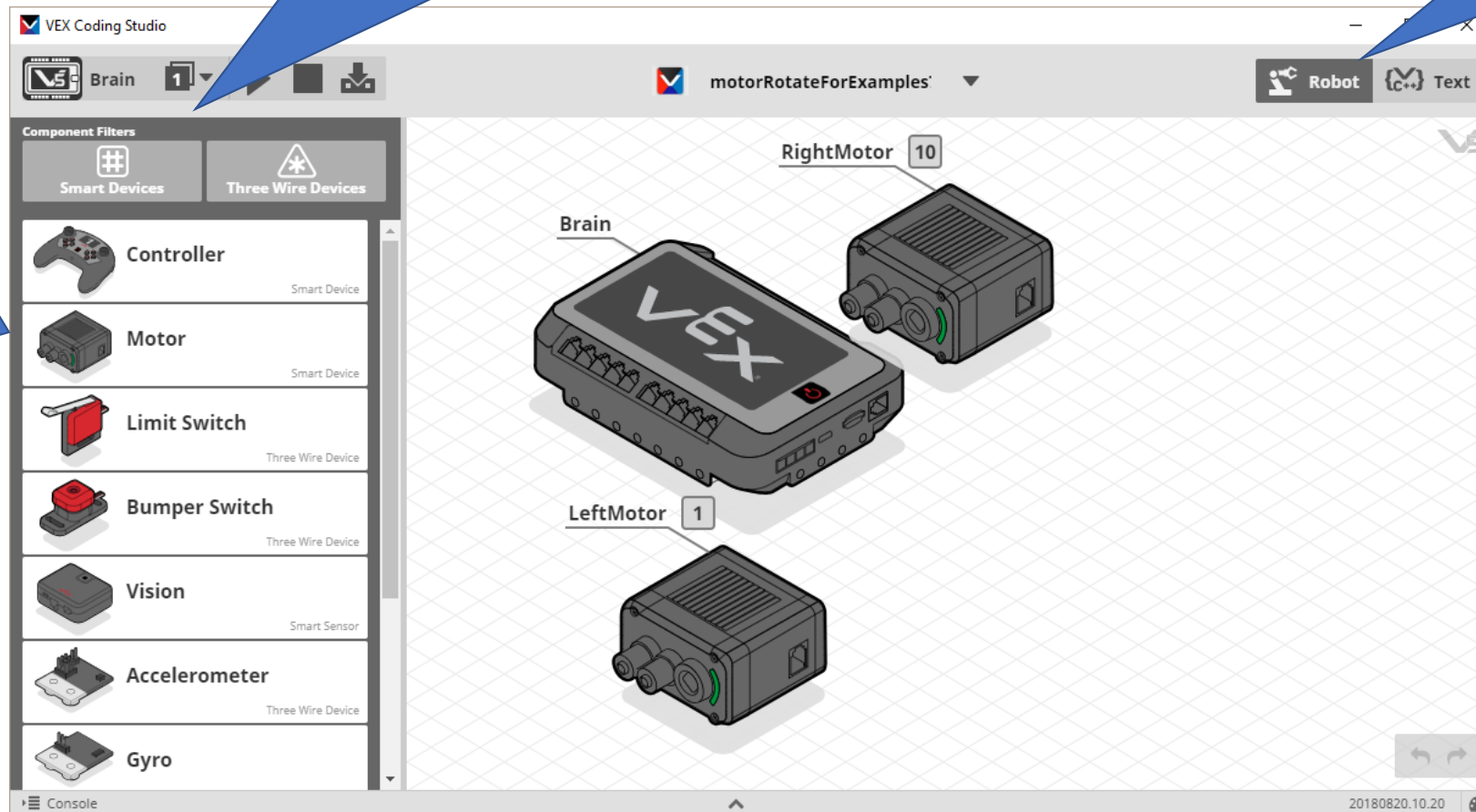


Need to configure the robot in V5 Coding Studio

2) If you don't see the 'Motor' select the 'Smart Devices' buttons.

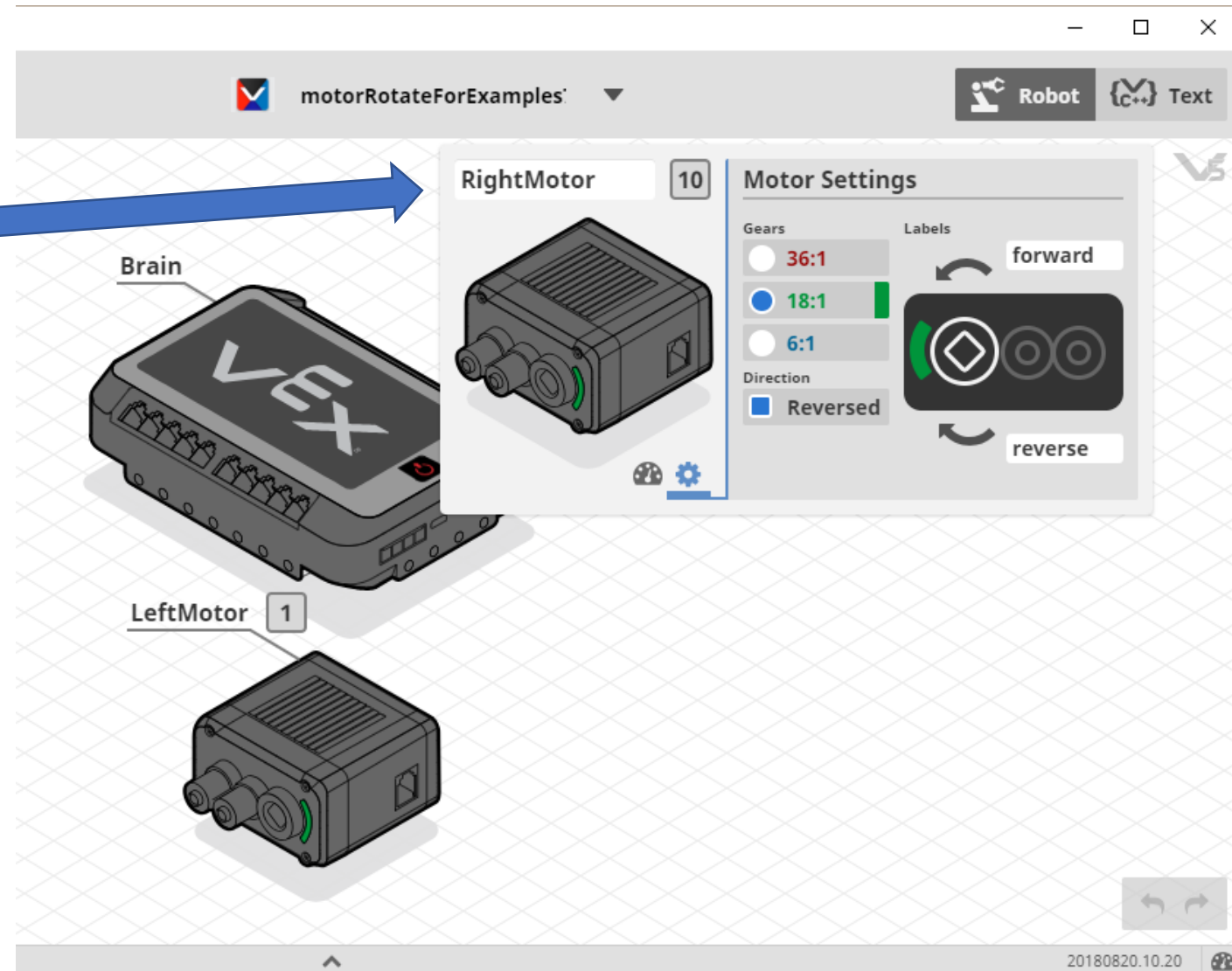
1) Select 'Robot' to get to the configuration window

3) Drag and Drop the parts for the robot.



Motor Setup

- Name the Motor
 - Start with a letter
 - Can use letters, umbers and underscores
 - Not a VEX C++ reserved word
 - Describes the device. (LeftMotor, ArmMotor, FrontSonarSensor, ...)



Set the Motor Port

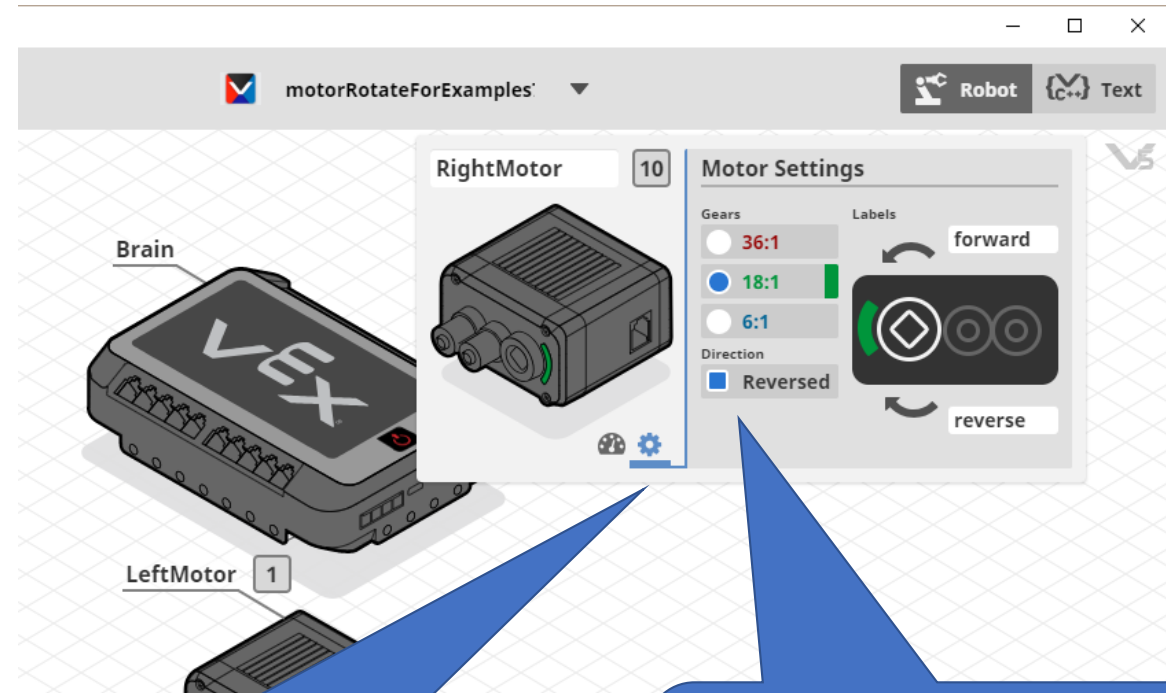
1) Click on the number to open the port selection window.

The screenshot shows the VEX Coding Studio interface. On the left is a 'Component Filters' sidebar with categories like 'Smart Devices' and 'Three Wire Devices'. The main workspace contains a 'Brain' component and two 'Motor' components labeled 'LeftMotor' and 'RightMotor'. A 'Select Port' dialog box is open over the 'RightMotor' component, displaying a grid of port numbers from 1 to 16. The number '10' is highlighted in the grid. A blue callout bubble points to the number '10' in the dialog box.

2) Click on the port to select it.
For the clawbot:
Right Motor in Port 10
Left Motor in Port 1

Set Gear Cartridge and Reverse as needed

- For the Clawbot the motors will need to mirror each other so to keep it from spinning in circles when it is supposed to go straight, **reverse the Right Motor.**



1) Click on the gear to open the Motor Settings Window.

Reversed button.

Programming the Motor

- We will focus on the `Motor.rotateFor()` command for movement.
- There are several movement commands.

```
Motor.spin(directionType::fwd);  
Motor.spin(directionType::fwd,50,velocityUnits::rpm);  
Motor.rotateTo(90,rotationUnits::deg,50,velocityUnits::pct);  
Motor.rotateTo(90,rotationUnits::deg);  
Motor.rotateFor(90,rotationUnits::deg,50,velocityUnits::pct);  
Motor.rotateFor(90,rotationUnits::deg);  
Motor.rotateFor(2.5,timeUnits::sec,50,velocityUnits::pct);
```

Console

```
Motor.rotateFor(2.5,timeUnits::sec);  
Motor.startRotateTo(90,rotationUnits::deg,50,velocityUnits::pct);  
Motor.startRotateTo(90,rotationUnits::deg);  
Motor.startRotateFor(90,rotationUnits::deg,50,velocityUnits::pct);  
Motor.startRotateFor(90,rotationUnits::deg);  
Motor.stop();  
Motor.stop(brakeType::coast);
```

Motor.rotateFor() Example

... at 50% velocity.

...rotateFor()...

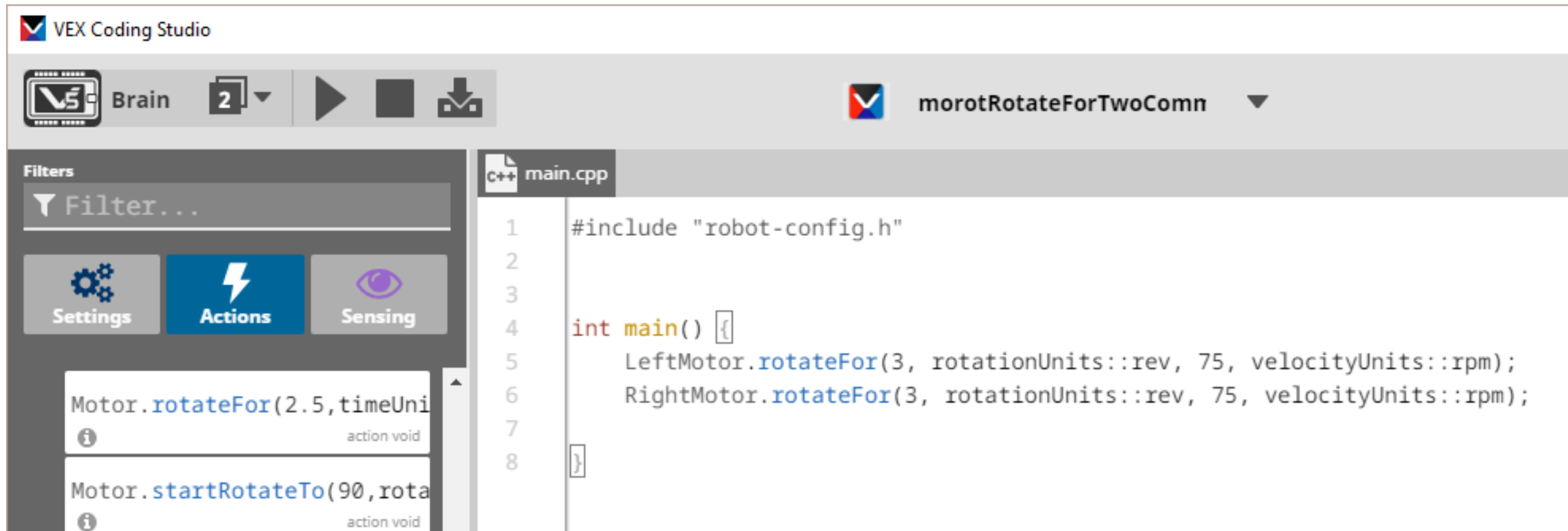
```
1 #include "robot-config.h"
2
3
4 int main() {
5     LeftMotor.rotateFor(90, rotationUnits::deg, 50, velocityUnits::pct);
6 }
7
```

The LeftMotor will...

... 90 degrees ...

Test it.

- Create, save, download and run this program.
- What do you notice?



VEX Coding Studio

Brain 2

morotRotateForTwoComn

Filters

Filter...

Settings Actions Sensing

Motor.`rotateFor`(2.5, timeUni
action void

Motor.`startRotateTo`(90, rota
action void

main.cpp

```
1 #include "robot-config.h"
2
3
4 int main() {
5     LeftMotor.rotateFor(3, rotationUnits::rev, 75, velocityUnits::rpm);
6     RightMotor.rotateFor(3, rotationUnits::rev, 75, velocityUnits::rpm);
7
8 }
```

Motor Command rotateFor() with all its options

A double (real) value that describes **how many units** will be completed. (10, 4.5, -20,...)

Unit Description

deg A rotation unit that is measured in **degrees**.

rev A rotation unit that is measured in **revolutions**.

raw A rotation unit that is measured in **raw data form**.

Motor.**rotateFor**(double rotation, rotationUnits units, double velocity, velocityUnits units_v, bool waitForCompletion=true)

A double (real) value that describes the velocity. (10, 4.5, -20,...)

Unit Description

pct A velocity unit that is measured in **percentage**.

rpm A velocity unit that is measured in **rotations per minute**.

dps A velocity unit that is measured in **degrees per second**

Optional: If left off then it will complete this command before starting the next command.
false = It will start the next command immediately after starting this command.

'rotateFor' Examples

```
int main() {
```

```
LeftMotor.rotateFor(3.5, rotationUnits::rev, 75, velocityUnits::rpm);
```

The LeftMotor will rotate 3.5 revolutions at 75 revolutions per minute (rpm).

```
LeftMotor.rotateFor(360, rotationUnits::deg, 80, velocityUnits::dps, false);
```

The LeftMotor will rotate 360 degrees at 80 degrees per second (dps) and will not wait until the command is finished before going to the next command.

```
RightMotor.rotateFor(720, rotationUnits::deg, 80, velocityUnits::dps);
```

The RightMotor will rotate 720 degrees at 80 degrees per second (dps) and will complete this command before going to the next command.

```
LeftMotor.rotateFor(3.5, rotationUnits::rev, false);
```

```
RightMotor.rotateFor(3.5, rotationUnits::rev); //3.5 revolutions
```

The LeftMotor will rotate 3.5 revolutions at the default speed or speed set by the Motor.setVelocity() command and will not wait until the command is finished before going to the next command.

```
RightMotor.rotateFor(3500, timeUnits::msec, 70, velocityUnits::pct); //3.5 seconds
```

```
RightMotor.rotateFor(3.5, timeUnits::sec);
```

The RightMotor will rotate 3500 milliseconds (ms) and 70% of the maximum speed.

```
}
```

The RightMotor will rotate 3.5 seconds (ms) at the default speed.

Proportional Movement

- With the rotateFor() command you can use the built-in motor encoder to control how far the robot moves.
- We know
 - 360 degrees = 1 revolution
 - Since the wheel is connected directly to the motor.
 - 1 motor revolution = 1-wheel revolution
 - 1 - wheel revolution = circumference of the wheel
 - Circumference of the wheel = $\text{PI} * \text{wheel diameter}$

- **Mini Challenge 1:**

- Write a program to have the robot move exactly **one yard** without guessing.
- No testing on the course
- Check answers on the floor.

Example: Finding the revolutions needed to travel 5 feet with a 4-inch diameter wheel directly connected to the motor.

$$5 \text{ feet} * (12 \text{ inches} / 1 \text{ foot}) * (1 \text{ Revolution} / \text{PI} * 4 \text{ inches}) = 4.77 \text{ revolutions}$$

Mini Challenge 2: 90 degree turns

- Write a program
 - 90 degree turn in each direction
 - Save it in slot 2
 - Give a short description
 - Try to calculate
 - Check and modify as needed

Programming Arms: Handling a range of motion limitation.

```
#include "robot-config.h"
```

```
int main() {  
    ArmMotor.setTimeout(5, timeUnits::sec);  
    ArmMotor.setStopping(brakeType::hold);  
    ArmMotor.rotateTo(90, rotationUnits::deg);  
}
```

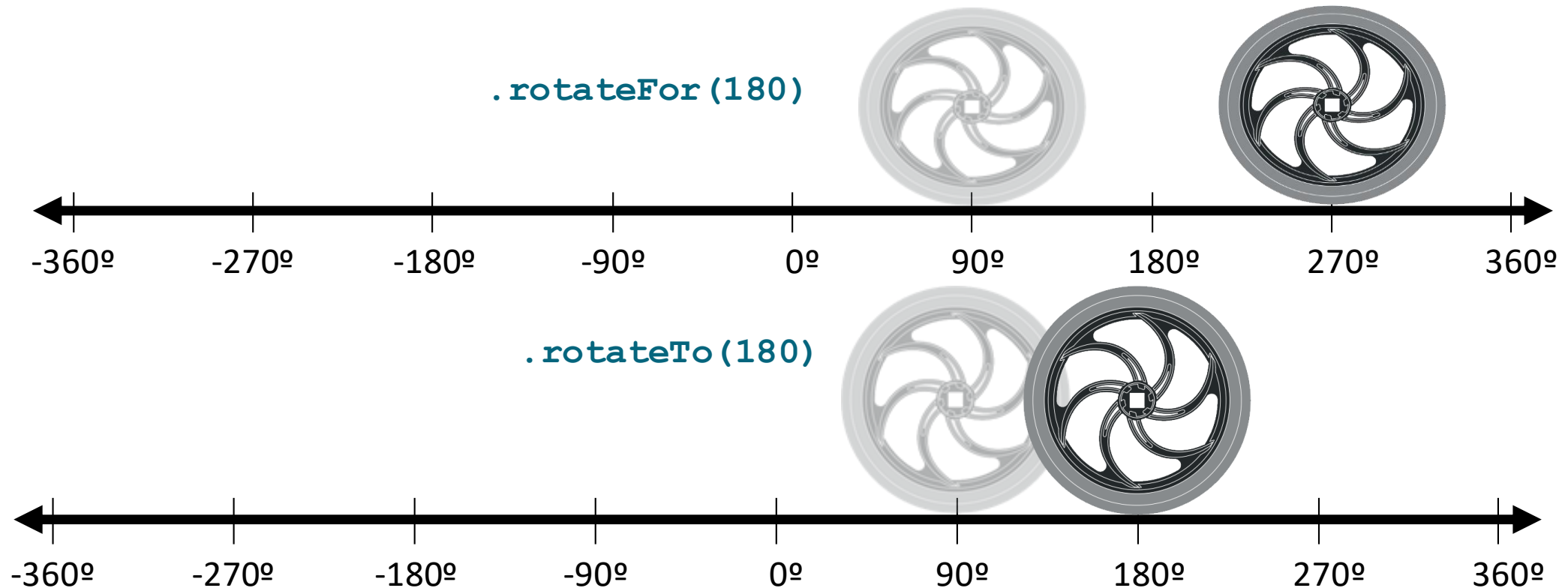
Sets the motor to stop spinning if it gets stuck for 5 seconds

Sets the motor to maintain its position after the movement is complete.

Rotates to the **absolute** position of 90 degrees

rotateTo() vs rotateFor()

- rotateFor() rotates a motor the full specified amount regardless of the current motor encoder reading.
- rotateTo() rotates a motor to a specific motor encoder reading.



Programming Hands/Grabbers

```
int main() {  
    ClawMotor.setTimeout(2, timeUnits::sec);  
    ClawMotor.setStopping(brakeType::hold);  
    ClawMotor.setMaxTorque(30, percentUnits::pct);  
    ClawMotor.rotateTo(90, rotationUnits::deg);  
}
```

Sets the motor to stop spinning if it gets stuck

Sets the motor to maintain its position after the movement is complete.

Sets the maximum amount of torque the motor will use.

Rotates to the **absolute** position of 90 degrees

rotateFor() Summary

- **Motor.rotateFor(double rotation , rotationUnits units, boolean waitForCompletion = true);**
- Used to rotate the left and right motors for a specific target rotational distance.
- Motor.rotateFor can be used either as a [blocking](#) or non-blocking command.
- Can be a non-blocking by including 'false' as the waitForCompletion .
- This command can be used not only with wheel motors but also with arm or claw motors, allowing them to be moved specific distances while safely avoiding overextensions without the need for a limit switch.

```
1  #include "robot-config.h"
2
3
4  int main() {
5      LeftMotor.rotateFor(90, rotationUnits::deg, 50, velocityUnits::pct);
6  }
```

...rotateFor()...

... at 50% velocity.

The LeftMotor will...

... 90 degrees ...

Motor.rotateTo() Summary:

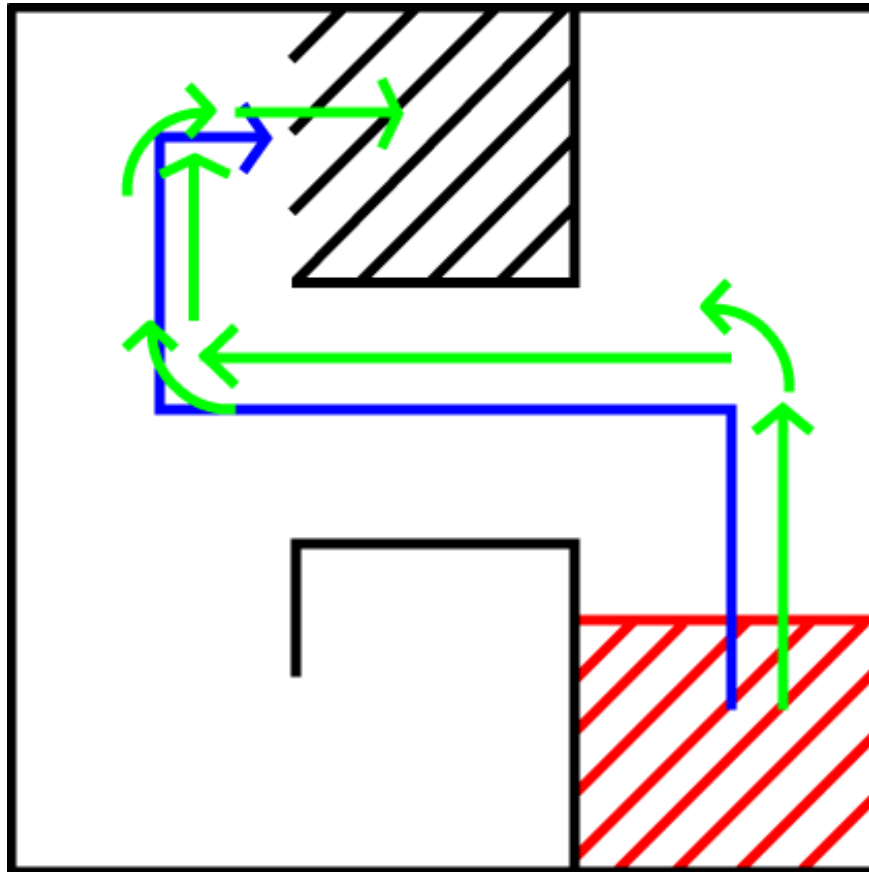
- Turns on the motor and spins it to an absolute target rotation value
- `rotateTo(double rotation, rotationUnits units, bool waitForCompletion=true);`
- `rotateTo(double rotation, rotationUnits units, double velocity, velocityUnits units_v, bool waitForCompletion=true);`

Programming Strategy: Behaviors

- Behavior: Anything the robot does.
- Basic Behaviors: Single Commands
- Simple Behaviors: Simple tasks (go forward)
 - Can be broken down into single commands
- Complex Behaviors: Complex Tasks (Solve a maze)
 - Can be broken down into simple behaviors

Functions: Judson

Movement Challenges



1. Complete the Labyrinth Challenge
2. Programming Skills Challenge
 - With your teammate read the rules for Programming Skills Challenge (Posted on the website)
 - Score as many points as possible
 - Look up other movement commands as needed to help.