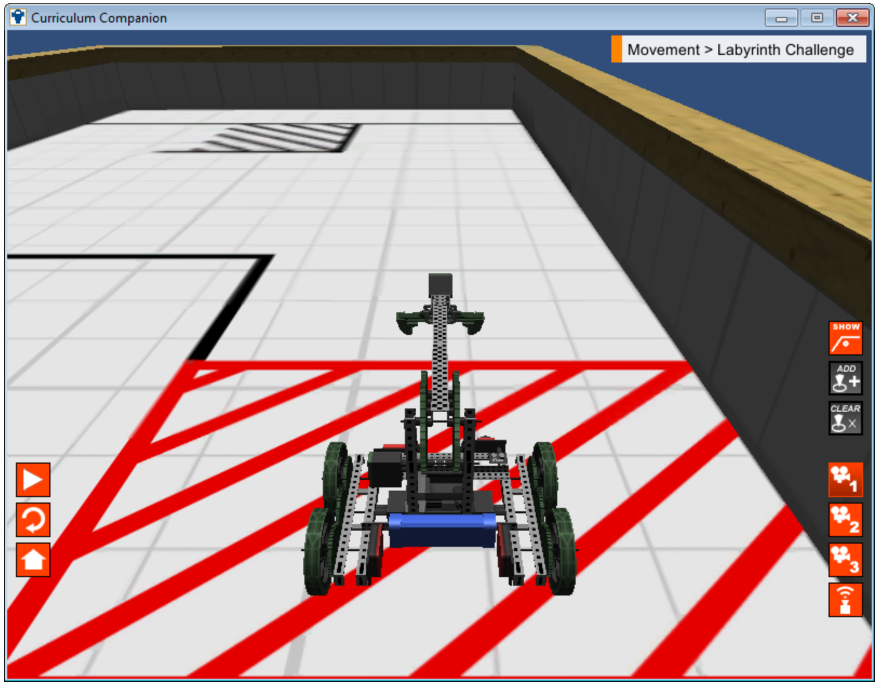
Getting Started in RobotC

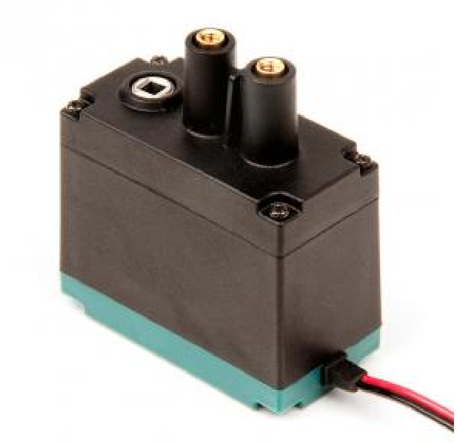


- // Comment
- task
- main()
- motor[]
- {}
- wait1Msec()
- ;
- =
- Header
- Code
- Compile
- Download
- Run

Learning Objectives

- Understand Motion
  - Motors: How they work and respond.
  - Fuses: Understand why they keep blowing
- Understand how to control Motors with a program including
  - Setting up the motors
  - Reading the basic outline of a program
  - Using commands for controlling motors
    - motor[port1] = …
    - motor[rightMotor] = …
    - wait1Msec();
- Be able to write programs for a Robot to complete r virtual challenges.

VEX Motion: Motors



- 2-Wire Motor 393
  - 100 RPM
    - No load
  - Torque peaks at 13.5 in-lbs at
    - 0 RPM
    - **<u>3.6 amp draw</u>**
  - Continually at 3.375 in-lbs
    - +/- 77 RPM
    - 0.9 amp draw

High Speed Gears

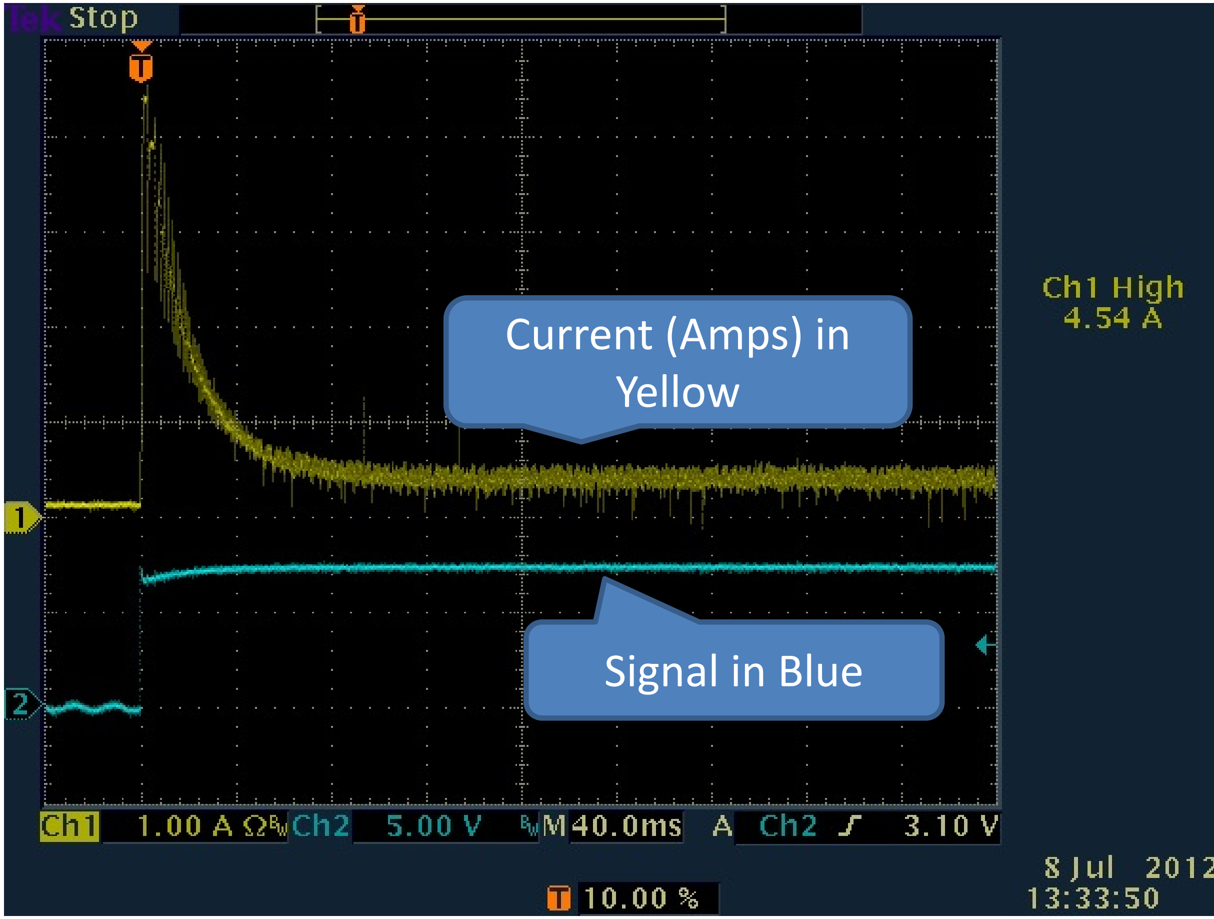
- High Speed Gearing: **60% faster**
- Unscrew the motor and replace internal gearing.
  - 160 RPM
    - No Load
  - Torque 8.4 in-lb in bursts
    - 0 RPM
    - 3.6 AMP
  - Continually at 2.1 in-lbs
    - +/- 123 RPM
    - 0.9 amp draw

Motor Controller

- Motor Controller: 2-Wire to 3-Wire
- Not needed for motor ports 1 and 10

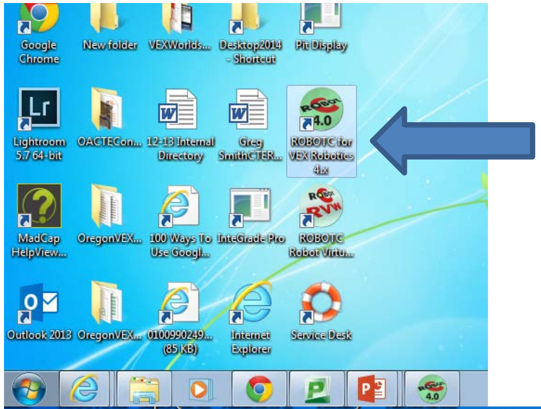
What happens when you floor it?

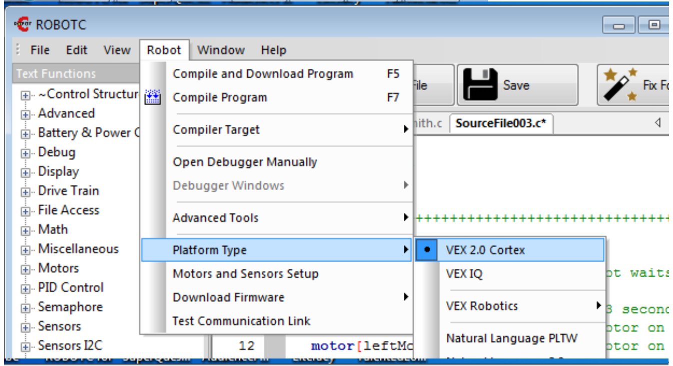
- Fuses you can blow
- Motor: 3.6 Amp
  - One Motor Stops
- Controller: 3 Amp
  - One motor stops
- Cortex Port: 4 amps combined with four other ports. Robot Stops

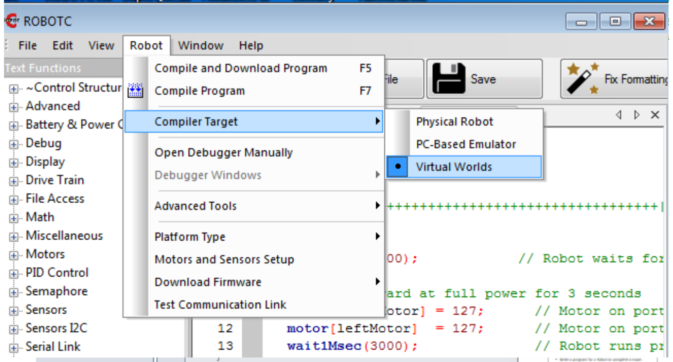


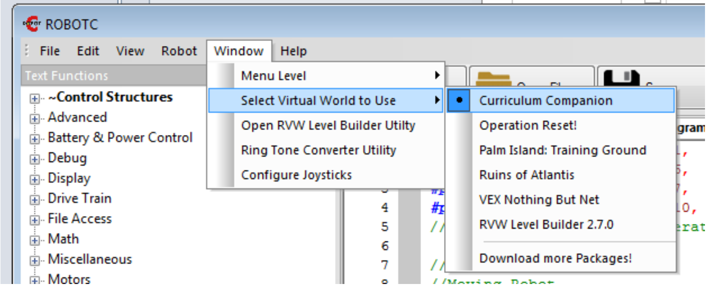
Current (Amps) in Yellow

Signal in Blue





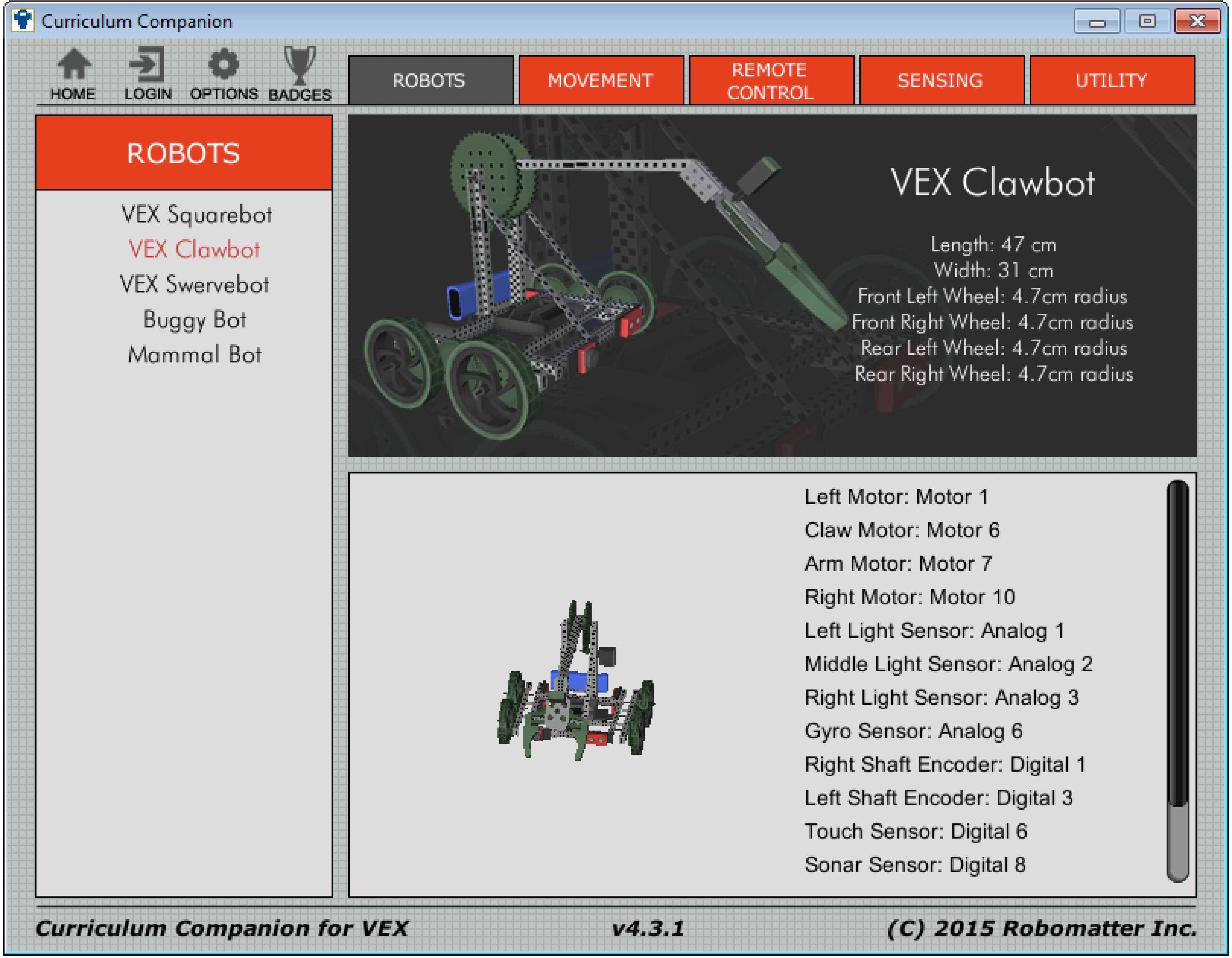




Getting Started

- Open RobotC
- Select VEX 2.0 Cortex Platform
  - Robot-> Platform ->VEX 2.0 Cortex
- Make the robot compile to Virtual Worlds
  - Robot-> Compiler Target -> Virtual Worlds
- Select Virtual World
  - Window->Select Virtual World to Use -> Curriculum Companion

Your Robot



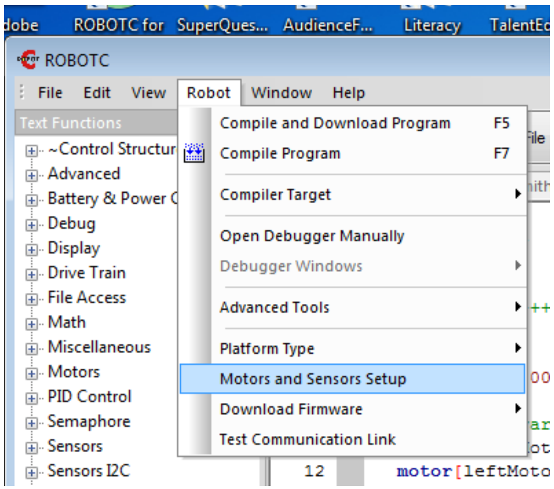
VEX Clawbot

Length: 47 cm
Width: 31 cm
Front Left Wheel: 4.7cm radius
Front Right Wheel: 4.7cm radius
Rear Left Wheel: 4.7cm radius
Rear Right Wheel: 4.7cm radius

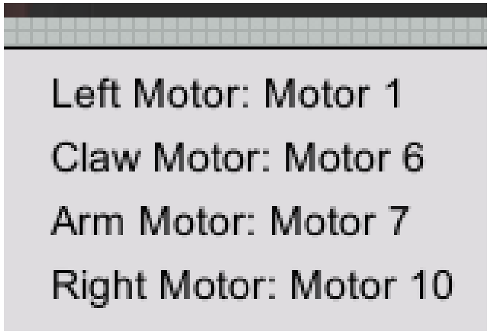
Left Motor: Motor 1
Claw Motor: Motor 6
Arm Motor: Motor 7
Right Motor: Motor 10
Left Light Sensor: Analog 1
Middle Light Sensor: Analog 2
Right Light Sensor: Analog 3
Gyro Sensor: Analog 6
Right Shaft Encoder: Digital 1
Left Shaft Encoder: Digital 3
Touch Sensor: Digital 6
Sonar Sensor: Digital 8

Curriculum Companion for VEX **v4.3.1** ***(C) 2015 Robomatter Inc.***

Configuring the Robot: Focus on Motors



- Robot -> Motors and Sensors Setup
- Select the motor
  - Currently can only purchase 393 Motors, also modify for internal gearing (high speed, turbo speed)
- Naming Convention
  - Rules
    - Start with a letter
    - No spaces, punctuation or reserved words (blue)
  - Style
    - Describes what it represents
    - First letter is lowercase
    - otherWordsStartWithUppercaseLetters
  - For these motors
    - leftMotor
    - clawMotor
    - armMotor
    - rightMotor



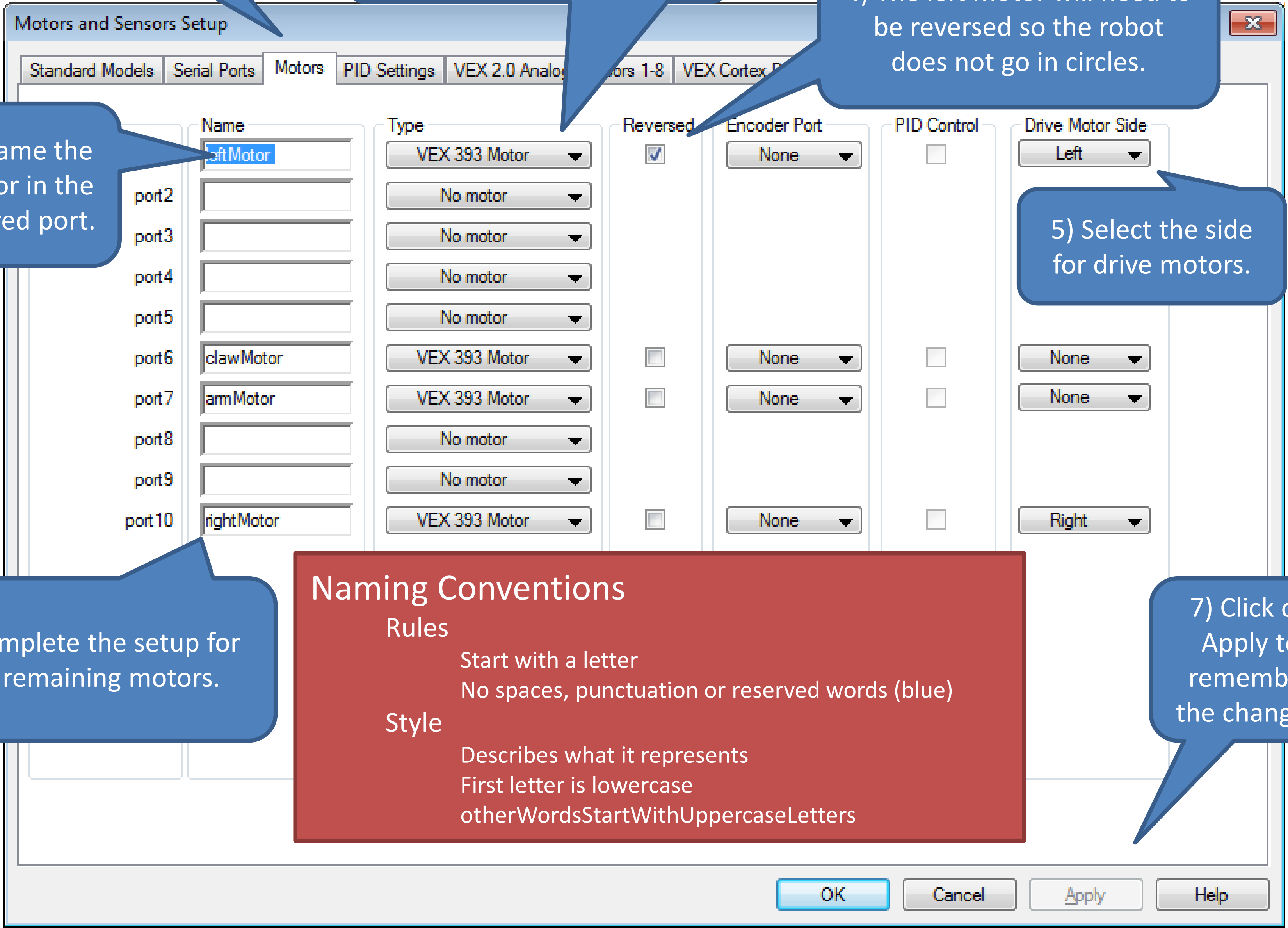
Left Motor: Motor 1

Claw Motor: Motor 6

Arm Motor: Motor 7

Right Motor: Motor 10

Motors and Sensors Setup Page



	Name	Type	Reversed	Encoder Port	PID Control	Drive Motor Side
	eftMotor	VEX 393 Motor	☑	None	□	Left
port2		No motor				
port3		No motor				
port4		No motor				
port5		No motor				
port6	clawMotor	VEX 393 Motor	□	None	□	None
port7	armMotor	VEX 393 Motor	□	None	□	None
port8		No motor				
port9		No motor				
port10	rightMotor	VEX 393 Motor	□	None	□	Right

1) Select the 'Motors' tab.

2) Name the motor in the desired port.

3) Use the pull down menus to select the motor.

4) The left motor will need to be reversed so the robot does not go in circles.

5) Select the side for drive motors.

6) Complete the setup for the remaining motors.

Naming Conventions

Rules

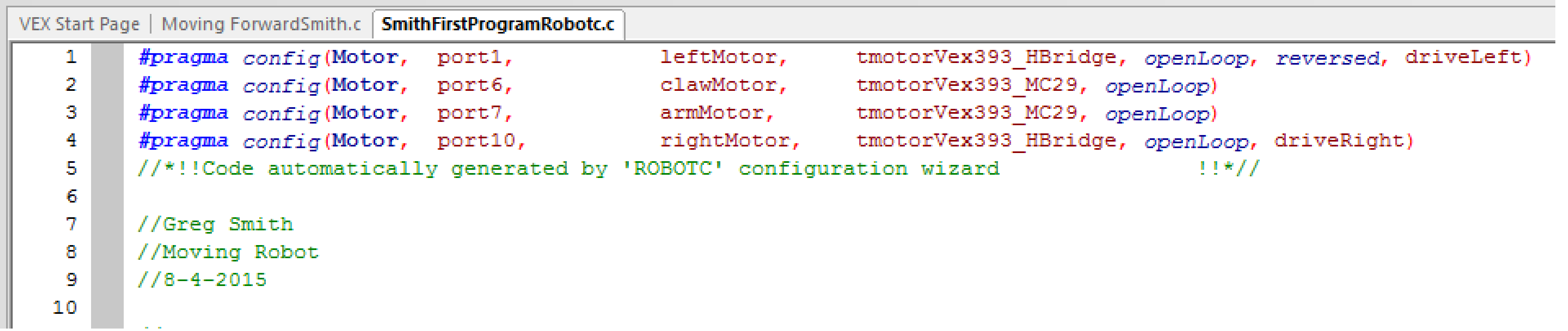
- Start with a letter
- No spaces, punctuation or reserved words (blue)

Style

- Describes what it represents
- First letter is lowercase
- otherWordsStartWithUppercaseLetters

7) Click on Apply to remember the changes.

Code the setup creates ‘pre-processor directives’



```
#pragma config(Motor,  port1,           leftMotor,     tmotorVex393_HBridge, openLoop, reversed, driveLeft)
#pragma config(Motor,  port6,           clawMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           armMotor,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          rightMotor,    tmotorVex393_HBridge, openLoop, driveRight)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//Greg Smith
//Moving Robot
//8-4-2015

```

Now we can start looking at RobotC

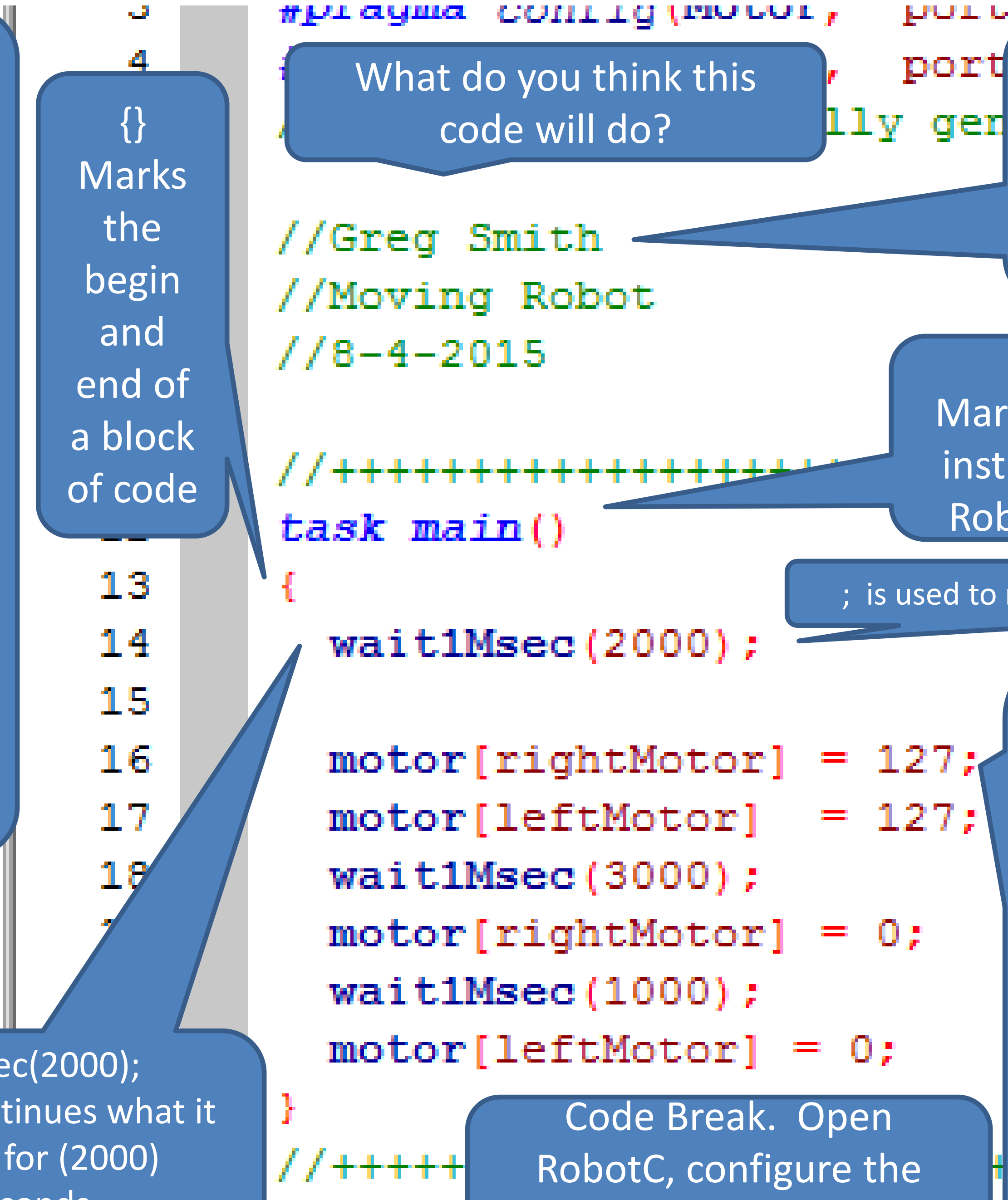
- motor[motorName] = motorPower;
- wait1Msec(milliseconds);
- wait10Msec();

Vocabulary
//
Comment
task
main()
motor[]
motorB.
motorC
{}
wait10Msec()
;
=
Header
Code
Compile
Download
Run

{}
Marks the begin and end of a block of code

What do you think this code will do?

The Header
// In front of the line makes this line a comment
/*   */ for multiple line comments.



```
#pragma config(Motor,      port
#                   ,      port
                         lly gen


//Greg Smith
//Moving Robot
//8-4-2015

//+++++++++++++++++++
task main()
{
  wait1Msec(2000);

  motor[rightMotor] = 127;
  motor[leftMotor]  = 127;
  wait1Msec(3000);
  motor[rightMotor] = 0;
  wait1Msec(1000);
  motor[leftMotor] = 0;
}
//+++++
```

task main()
Marks the beginning of the instructions for the Robot.
RobotC Is CaSe SeNsItIvE!

; is used to mark the end of a command.

motor[motorB] = 127;
motor[] Used to select the motor.
rightMotor = This represents the place where the motor is attached.
motor[port10] = 127;
does the same thing.
= 127;
127 = full power
-127 = Reverse
0 = stop

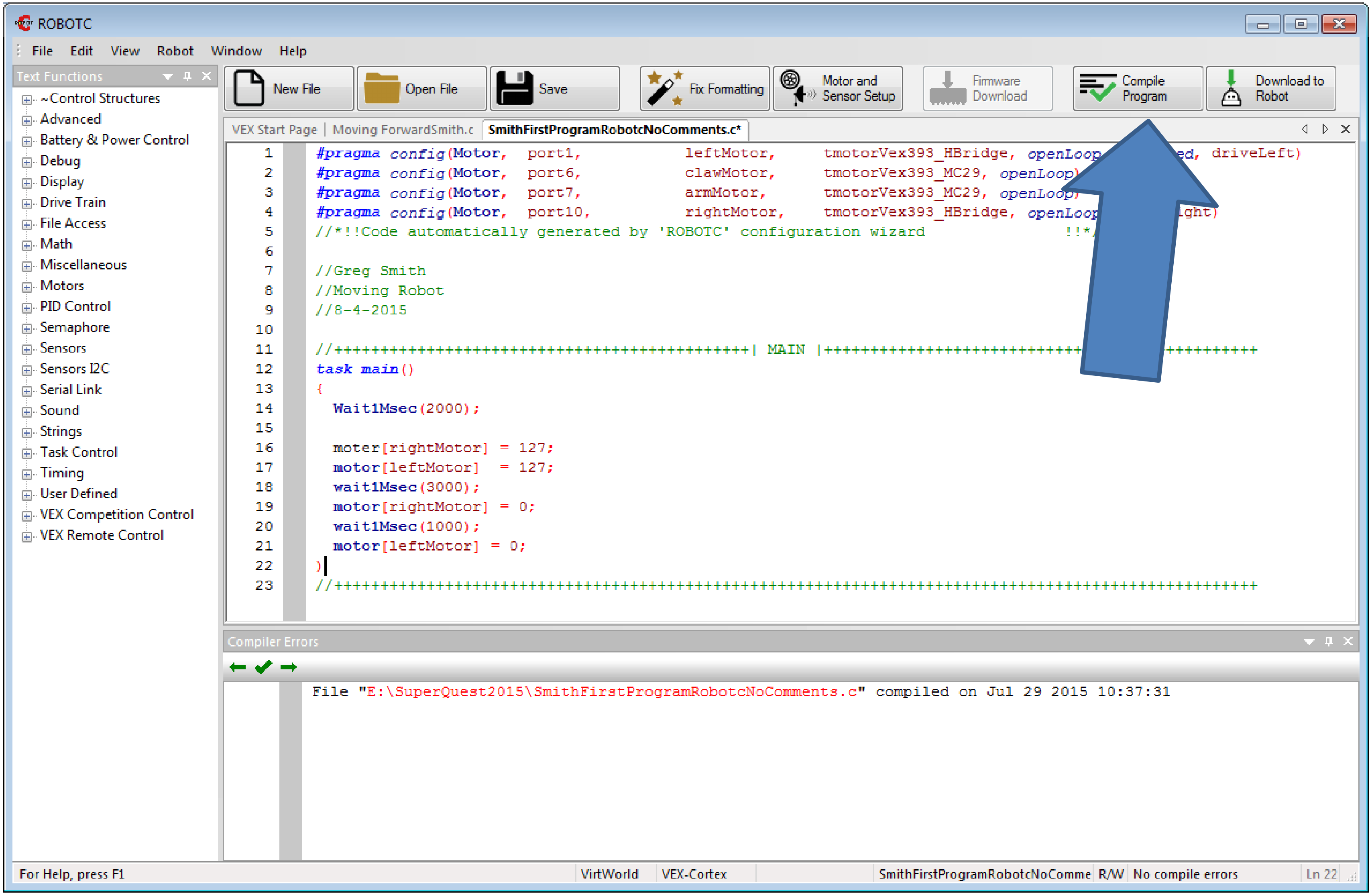
wait1Msec(2000);
The robot continues what it was doing for (2000) milliseconds.
Two seconds in this case.

Code Break. Open RobotC, configure the motors and enter the above code.

Testing the Program

- Compile the program
  - Changes into machine code that the robot understands.
- Download the program
  - Moving the machine language to your Virtual or Physical Robot
- Virtual Robot
  - Log in
  - Select Robot
  - Select Challenge
  - Start Activity

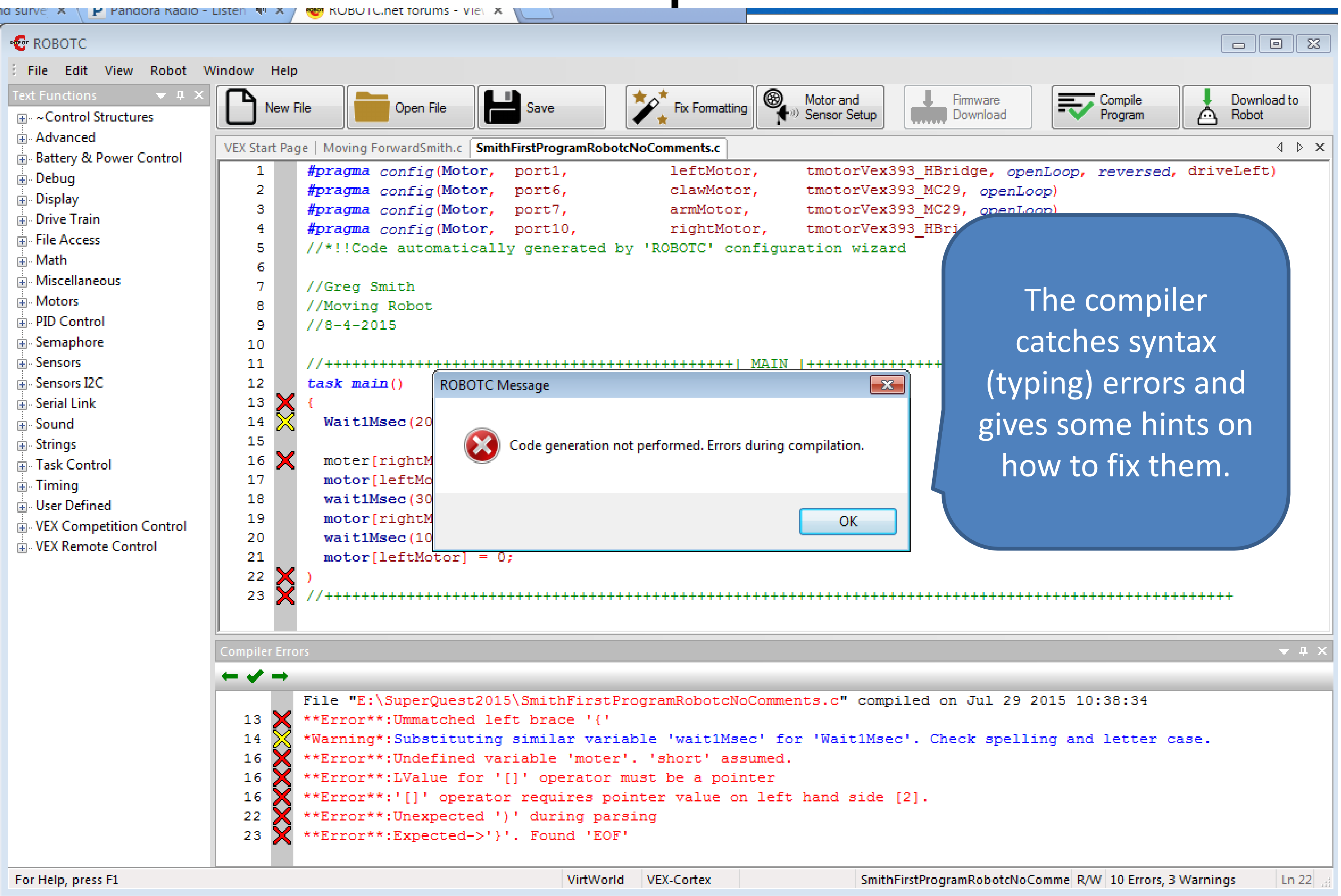
Compiling the Program



```
#pragma config(Motor,  port1,           leftMotor,     tmotorVex393_HBridge, openLoop        ed, driveLeft)
#pragma config(Motor,  port6,           clawMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           armMotor,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          rightMotor,    tmotorVex393_HBridge, openLoop        ight)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*

//Greg Smith
//Moving Robot
//8-4-2015

//+++++++++++++++++++++++++++++++++++++++++++++| MAIN |+++++++++++++++++++++++++++++         +++++++++++
task main()
{
  Wait1Msec(2000);

  moter[rightMotor] = 127;
  motor[leftMotor]  = 127;
  wait1Msec(3000);
  motor[rightMotor] = 0;
  wait1Msec(1000);
  motor[leftMotor] = 0;
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
File "E:\SuperQuest2015\SmithFirstProgramRobotcNoComments.c" compiled on Jul 29 2015 10:37:31
```

Oops!



```
#pragma config(Motor,  port1,           leftMotor,     tmotorVex393_HBridge, openLoop, reversed, driveLeft)
#pragma config(Motor,  port6,           clawMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           armMotor,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          rightMotor,    tmotorVex393_HBri
//*!!Code automatically generated by 'ROBOTC' configuration wizard

//Greg Smith
//Moving Robot
//8-4-2015

//+++++++++++++++++++++++++++++++++++++++++++++| MAIN |+++++++++++++++
task main()
{
  Wait1Msec(20

  moter[rightM
  motor[leftMo
  wait1Msec(30
  motor[rightM
  wait1Msec(10
  motor[leftMotor] = 0;
)
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

ROBOTC Message

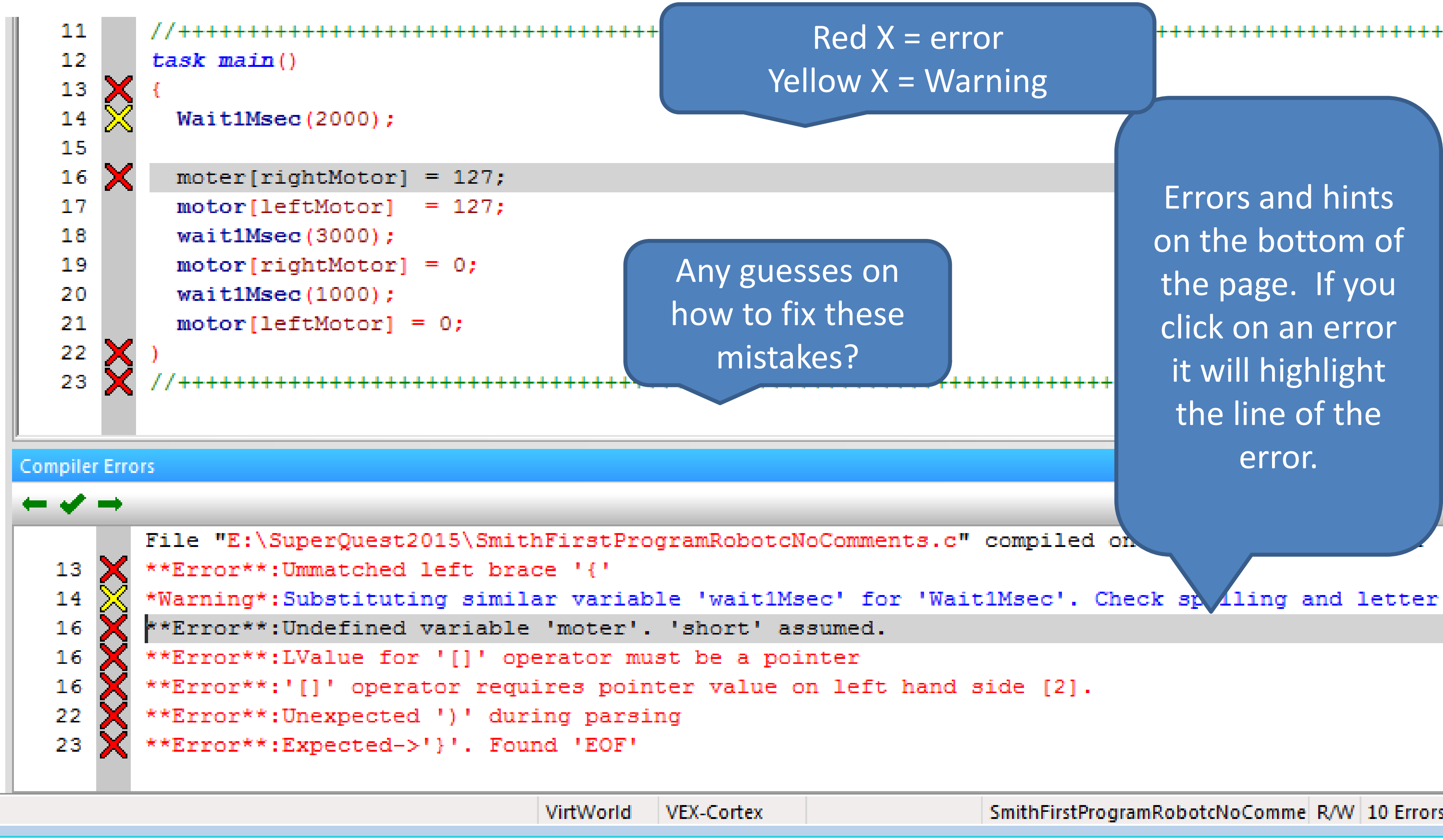
Code generation not performed. Errors during compilation.

OK

```
     File "E:\SuperQuest2015\SmithFirstProgramRobotcNoComments.c" compiled on Jul 29 2015 10:38:34
13   **Error**:Unmatched left brace '{'
14   *Warning*:Substituting similar variable 'wait1Msec' for 'Wait1Msec'. Check spelling and letter case.
16   **Error**:Undefined variable 'moter'. 'short' assumed.
16   **Error**:LValue for '[]' operator must be a pointer
16   **Error**:'[]' operator requires pointer value on left hand side [2].
22   **Error**:Unexpected ')' during parsing
23   **Error**:Expected->'}'. Found 'EOF'
```

The compiler catches syntax (typing) errors and gives some hints on how to fix them.

Errors



```
11  //++++++++++++++++++++++++++++++++++
12  task main()
13  {
14    Wait1Msec(2000);
15
16    moter[rightMotor] = 127;
17    motor[leftMotor]  = 127;
18    wait1Msec(3000);
19    motor[rightMotor] = 0;
20    wait1Msec(1000);
21    motor[leftMotor] = 0;
22  )
23  //++++++++++++++++++++++++++++++++++
```

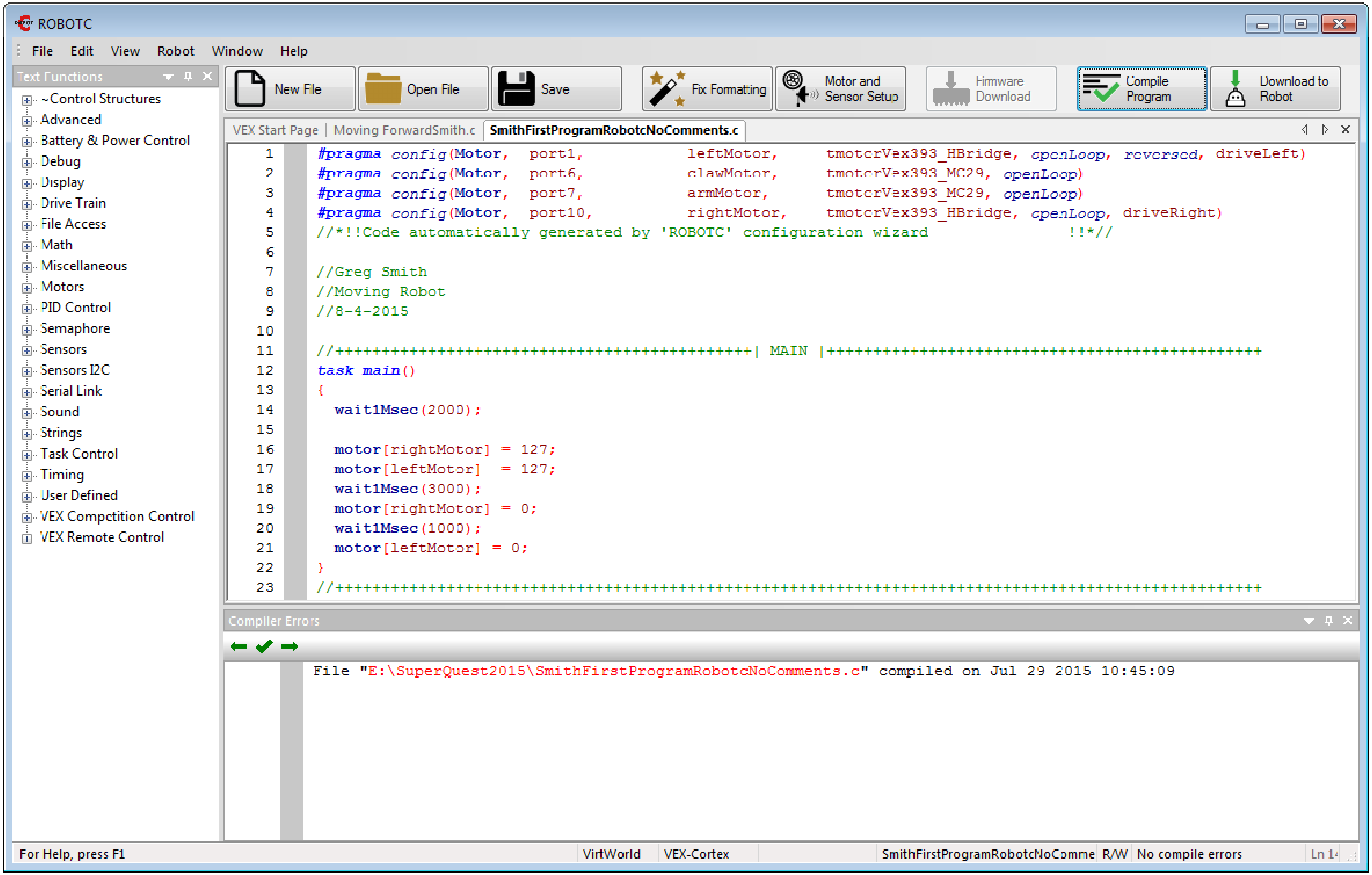
```
    File "E:\SuperQuest2015\SmithFirstProgramRobotcNoComments.c" compiled on
13  **Error**:Unmatched left brace '{'
14  *Warning*:Substituting similar variable 'wait1Msec' for 'Wait1Msec'. Check sp lling and letter
16  **Error**:Undefined variable 'moter'. 'short' assumed.
16  **Error**:LValue for '[]' operator must be a pointer
16  **Error**:'[]' operator requires pointer value on left hand side [2].
22  **Error**:Unexpected ')' during parsing
23  **Error**:Expected->'}'. Found 'EOF'
```

Red X = error
Yellow X = Warning

Any guesses on how to fix these mistakes?

Errors and hints on the bottom of the page. If you click on an error it will highlight the line of the error.

Corrected and Compiled

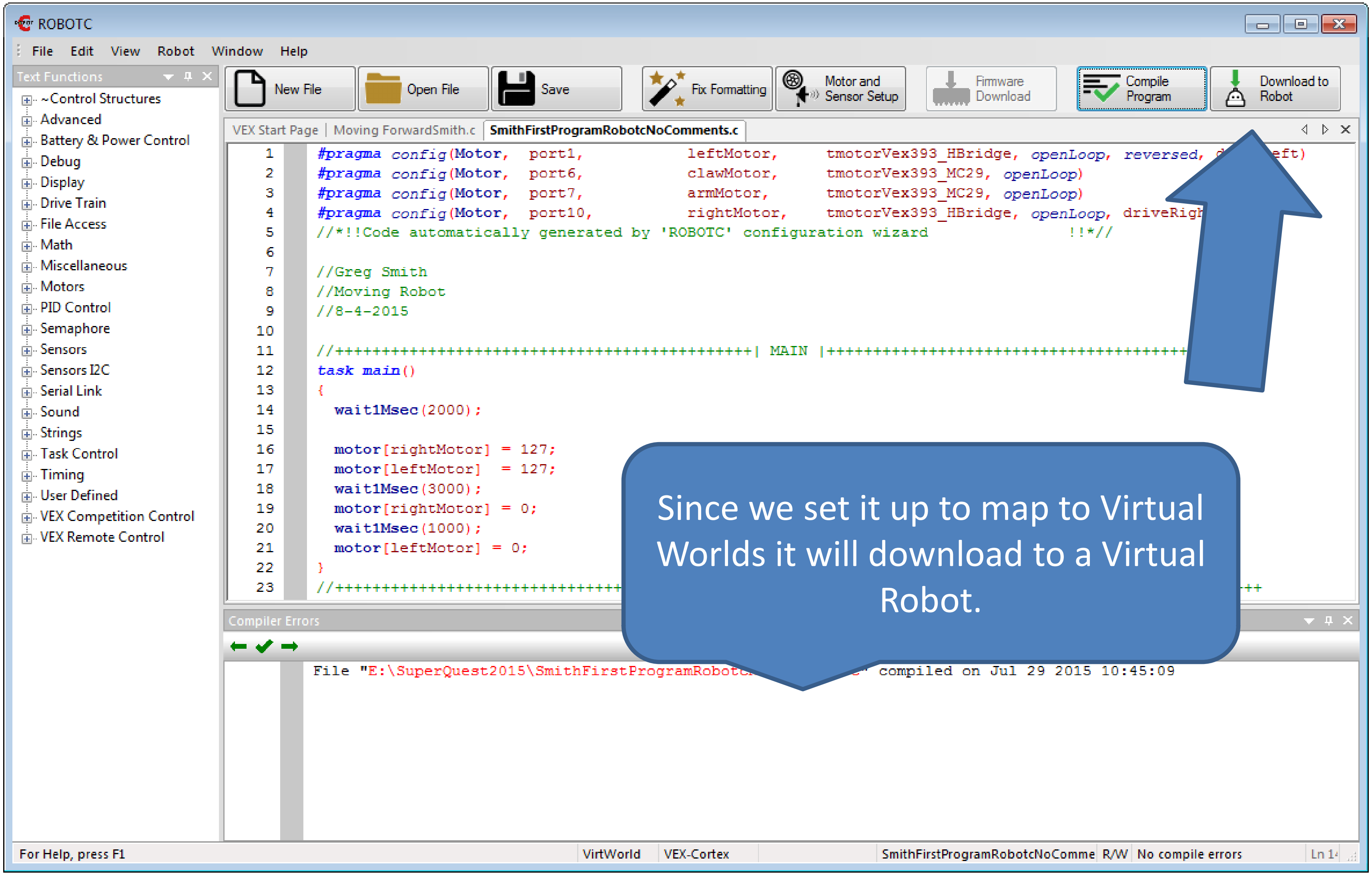


```
#pragma config(Motor,  port1,           leftMotor,     tmotorVex393_HBridge, openLoop, reversed, driveLeft)
#pragma config(Motor,  port6,           clawMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           armMotor,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          rightMotor,    tmotorVex393_HBridge, openLoop, driveRight)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//Greg Smith
//Moving Robot
//8-4-2015

//+++++++++++++++++++++++++++++++++++++++++++++| MAIN |+++++++++++++++++++++++++++++++++++++++++++++++
task main()
{
  wait1Msec(2000);

  motor[rightMotor] = 127;
  motor[leftMotor]  = 127;
  wait1Msec(3000);
  motor[rightMotor] = 0;
  wait1Msec(1000);
  motor[leftMotor] = 0;
}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
File "E:\SuperQuest2015\SmithFirstProgramRobotcNoComments.c" compiled on Jul 29 2015 10:45:09
```

For Help, press F1 | VirtWorld | VEX-Cortex | SmithFirstProgramRobotcNoComme | R/W | No compile errors | Ln 14

Download the program

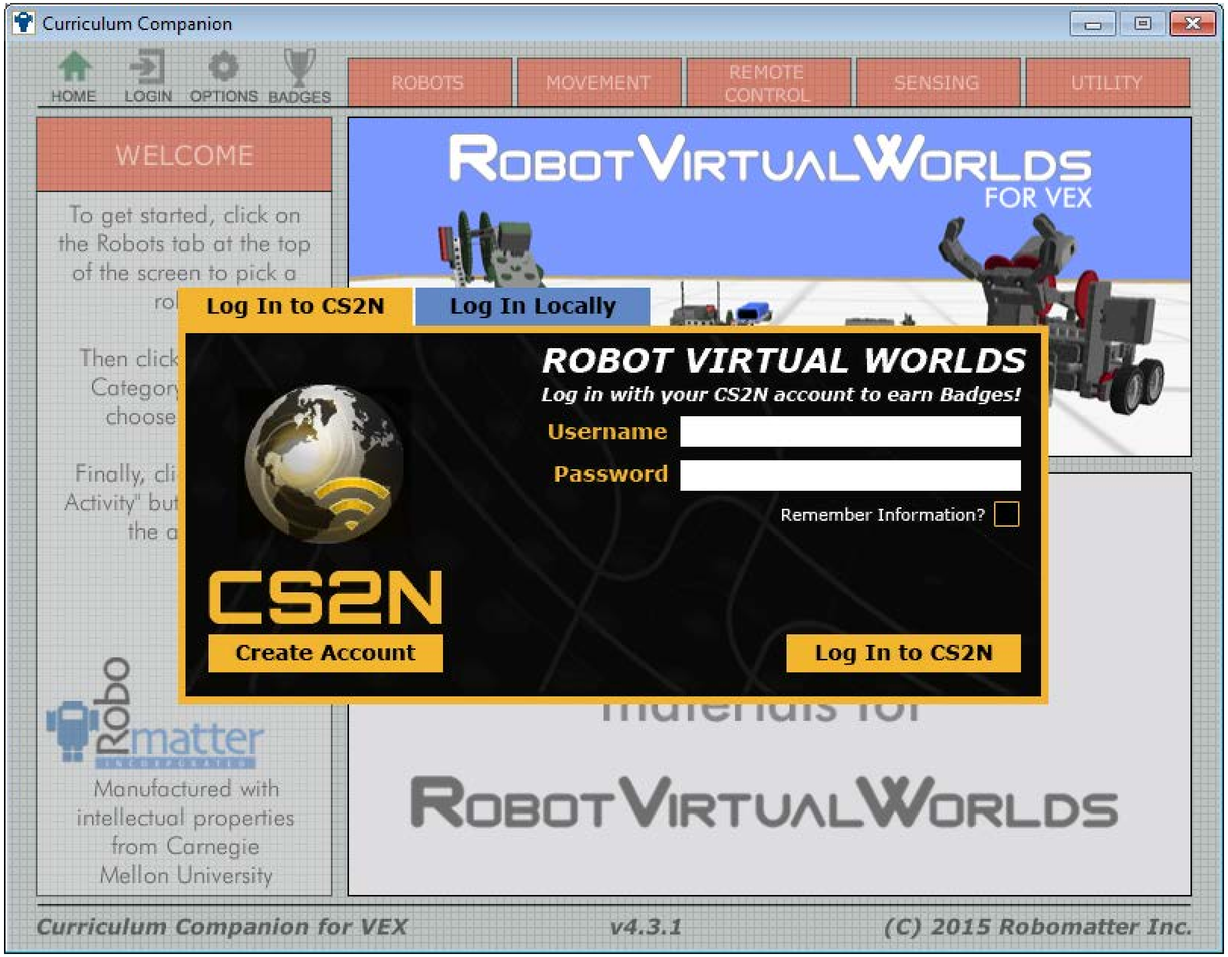


```
#pragma config(Motor,  port1,           leftMotor,     tmotorVex393_HBridge, openLoop, reversed, d     eft)
#pragma config(Motor,  port6,           clawMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port7,           armMotor,      tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port10,          rightMotor,    tmotorVex393_HBridge, openLoop, driveRigh
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//Greg Smith
//Moving Robot
//8-4-2015

//++++++++++++++++++++++++++++++++++++++++++++| MAIN |++++++++++++++++++++++++++++++++++++++++
task main()
{
  wait1Msec(2000);

  motor[rightMotor] = 127;
  motor[leftMotor]  = 127;
  wait1Msec(3000);
  motor[rightMotor] = 0;
  wait1Msec(1000);
  motor[leftMotor] = 0;
}
//++++++++++++++++++++++++++++++++
```

```
File "E:\SuperQuest2015\SmithFirstProgramRobotc           " compiled on Jul 29 2015 10:45:09
```

Since we set it up to map to Virtual Worlds it will download to a Virtual Robot.

Set up and Account with CS2N. It will track progress. Can log in locally as a guest without tracking.



To get started, click on the Robots tab at the top of the screen to pick a ro

Then click Categor choose

Finally, cli Activity" but the a

ROBOT VIRTUAL WORLDS

Log in with your CS2N account to earn Badges!

Username

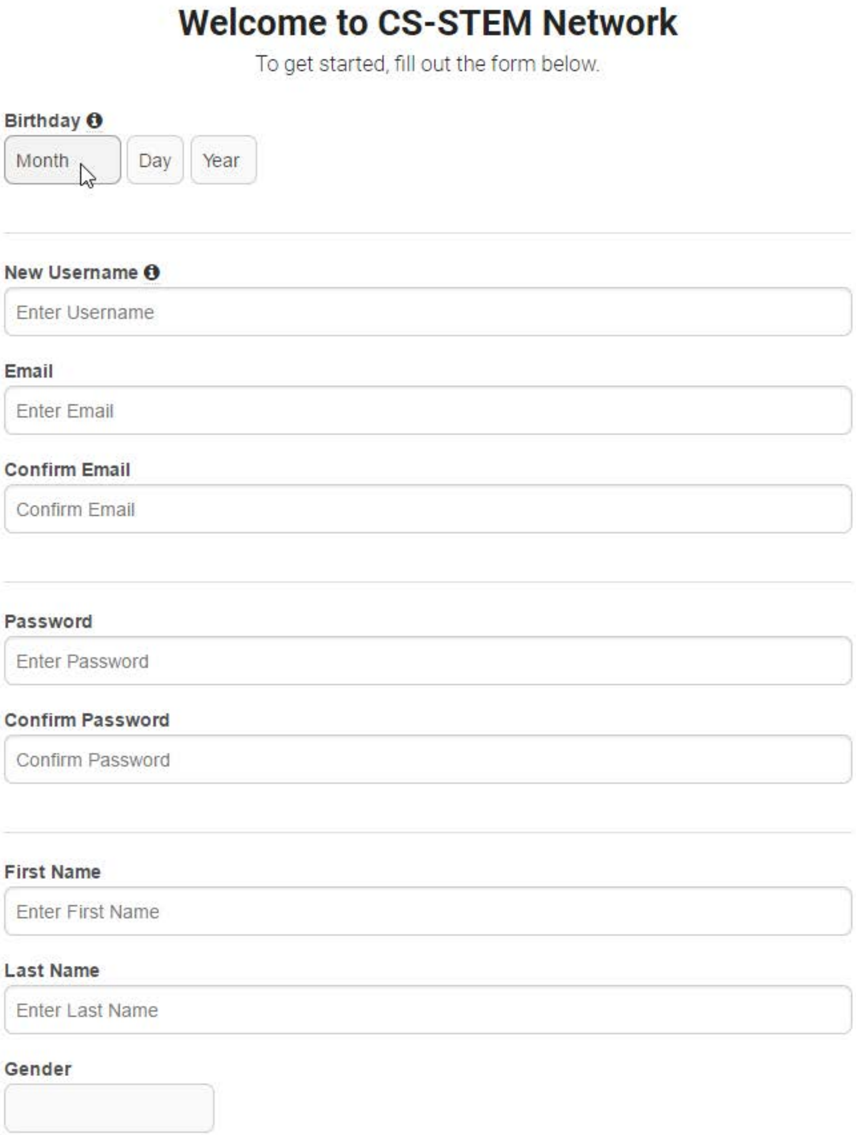
Password

Remember Information? □

Manufactured with intellectual properties from Carnegie Mellon University

Curriculum Companion for VEX v4.3.1 (C) 2015 Robomatter Inc.

Information for CS2N



Welcome to CS-STEM Network

To get started, fill out the form below.

Birthday ℹ

Month | Day | Year

New Username ℹ

Enter Username

Email

Enter Email

Confirm Email

Confirm Email

Password

Enter Password

Confirm Password

Confirm Password

First Name

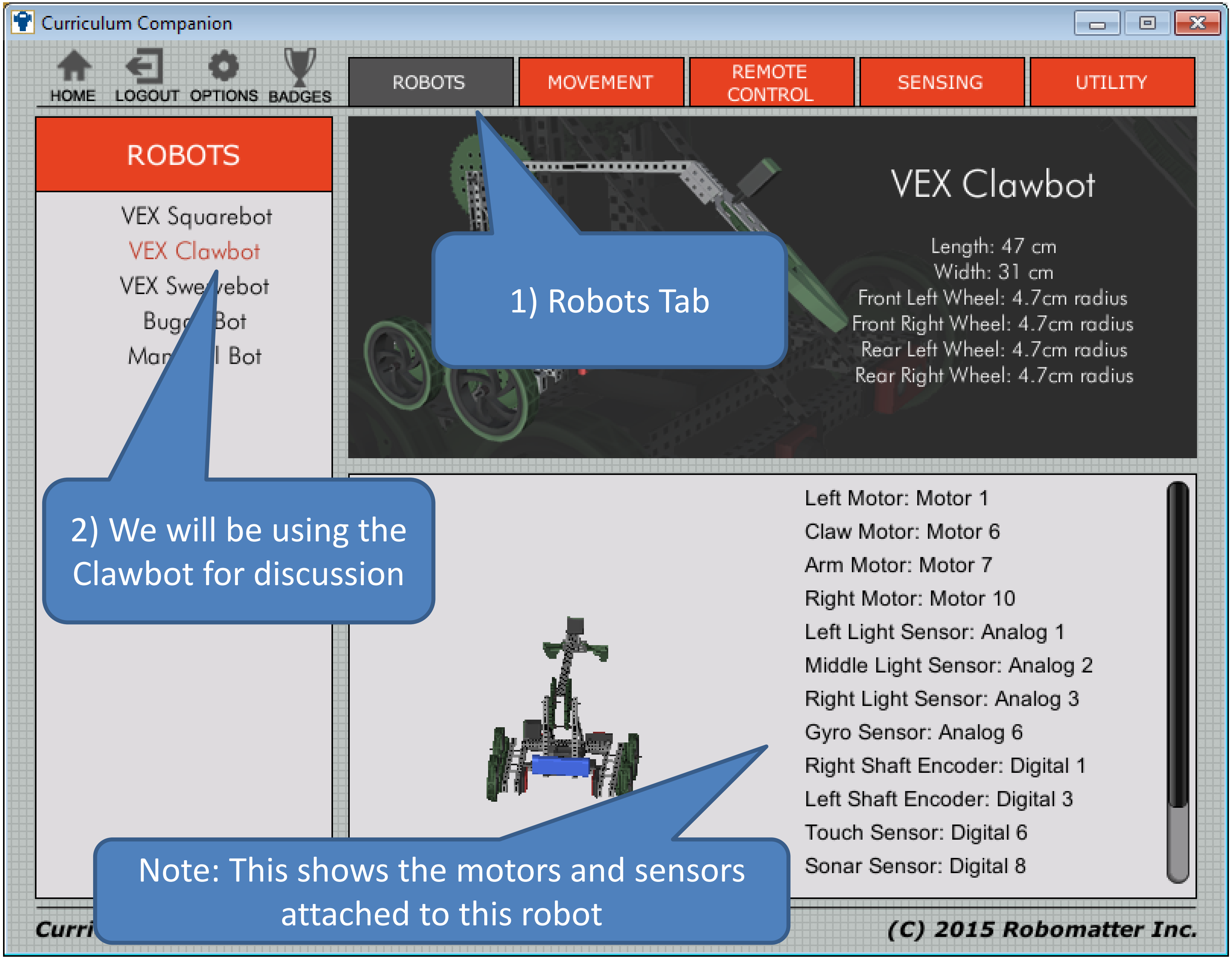
Enter First Name

Last Name

Enter Last Name

Gender

Select Your Robot



VEX Clawbot

Length: 47 cm
Width: 31 cm
Front Left Wheel: 4.7cm radius
Front Right Wheel: 4.7cm radius
Rear Left Wheel: 4.7cm radius
Rear Right Wheel: 4.7cm radius

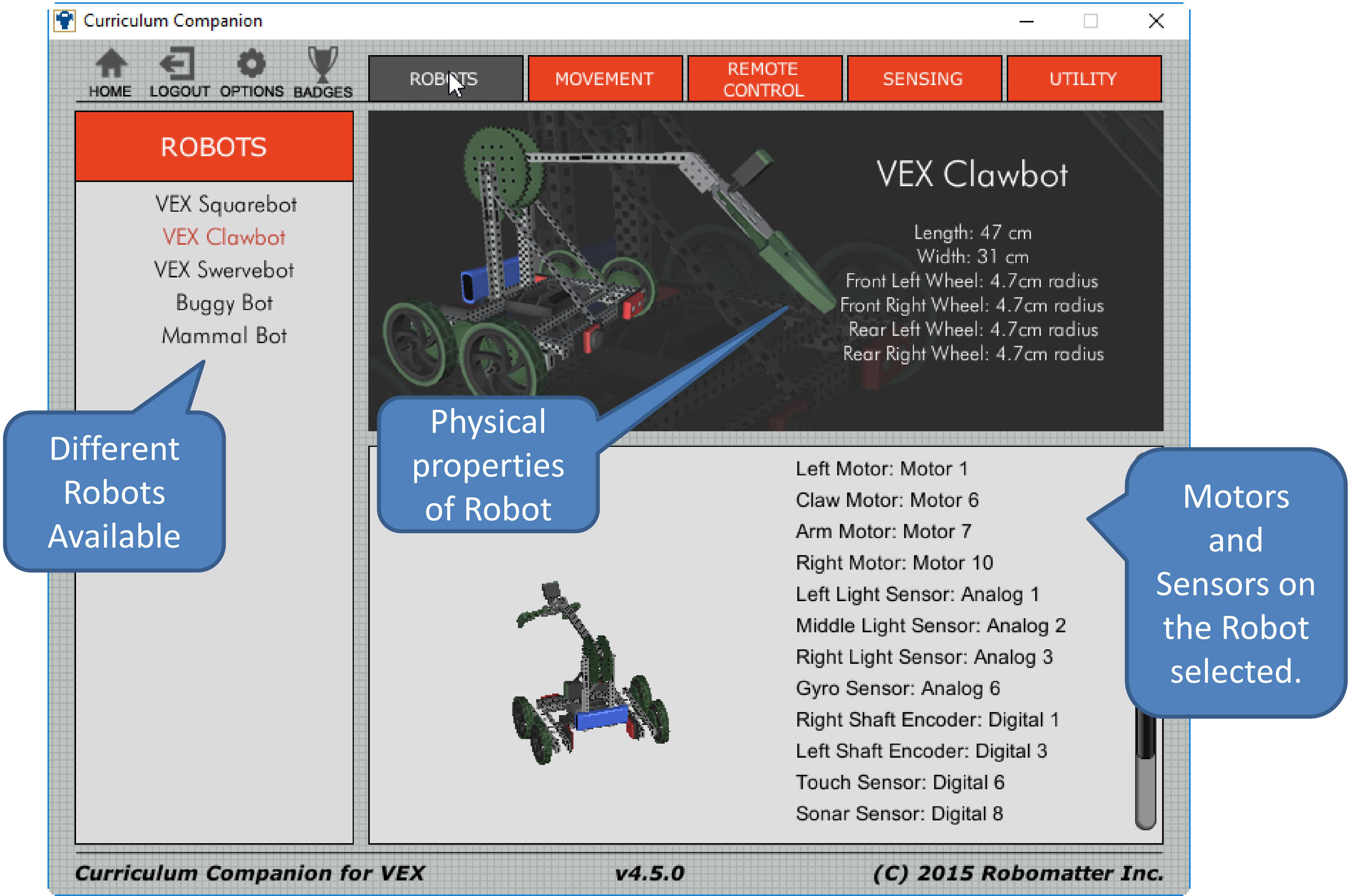
Left Motor: Motor 1
Claw Motor: Motor 6
Arm Motor: Motor 7
Right Motor: Motor 10
Left Light Sensor: Analog 1
Middle Light Sensor: Analog 2
Right Light Sensor: Analog 3
Gyro Sensor: Analog 6
Right Shaft Encoder: Digital 1
Left Shaft Encoder: Digital 3
Touch Sensor: Digital 6
Sonar Sensor: Digital 8

1) Robots Tab

2) We will be using the Clawbot for discussion

Note: This shows the motors and sensors attached to this robot

Robots Tab



ROBOTS

VEX Squarebot
VEX Clawbot
VEX Swervebot
Buggy Bot
Mammal Bot

VEX Clawbot

Length: 47 cm
Width: 31 cm
Front Left Wheel: 4.7cm radius
Front Right Wheel: 4.7cm radius
Rear Left Wheel: 4.7cm radius
Rear Right Wheel: 4.7cm radius

Left Motor: Motor 1
Claw Motor: Motor 6
Arm Motor: Motor 7
Right Motor: Motor 10
Left Light Sensor: Analog 1
Middle Light Sensor: Analog 2
Right Light Sensor: Analog 3
Gyro Sensor: Analog 6
Right Shaft Encoder: Digital 1
Left Shaft Encoder: Digital 3
Touch Sensor: Digital 6
Sonar Sensor: Digital 8

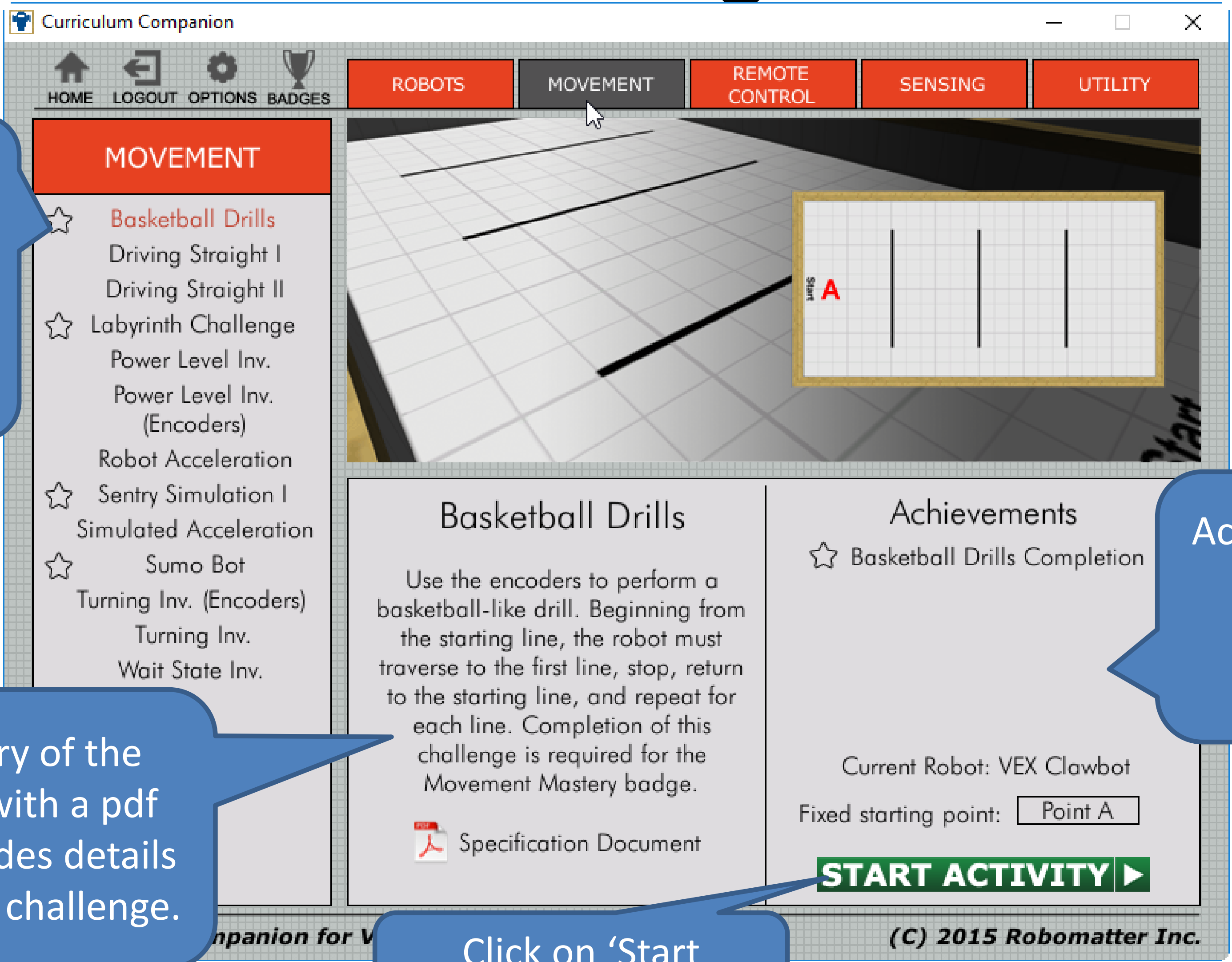
Curriculum Companion for VEX ***v4.5.0*** ***(C) 2015 Robomatter Inc.***

Different Robots Available

Physical properties of Robot

Motors and Sensors on the Robot selected.

Challenges



Basketball Drills

Use the encoders to perform a basketball-like drill. Beginning from the starting line, the robot must traverse to the first line, stop, return to the starting line, and repeat for each line. Completion of this challenge is required for the Movement Mastery badge.

Current Robot: VEX Clawbot

Fixed starting point: Point A

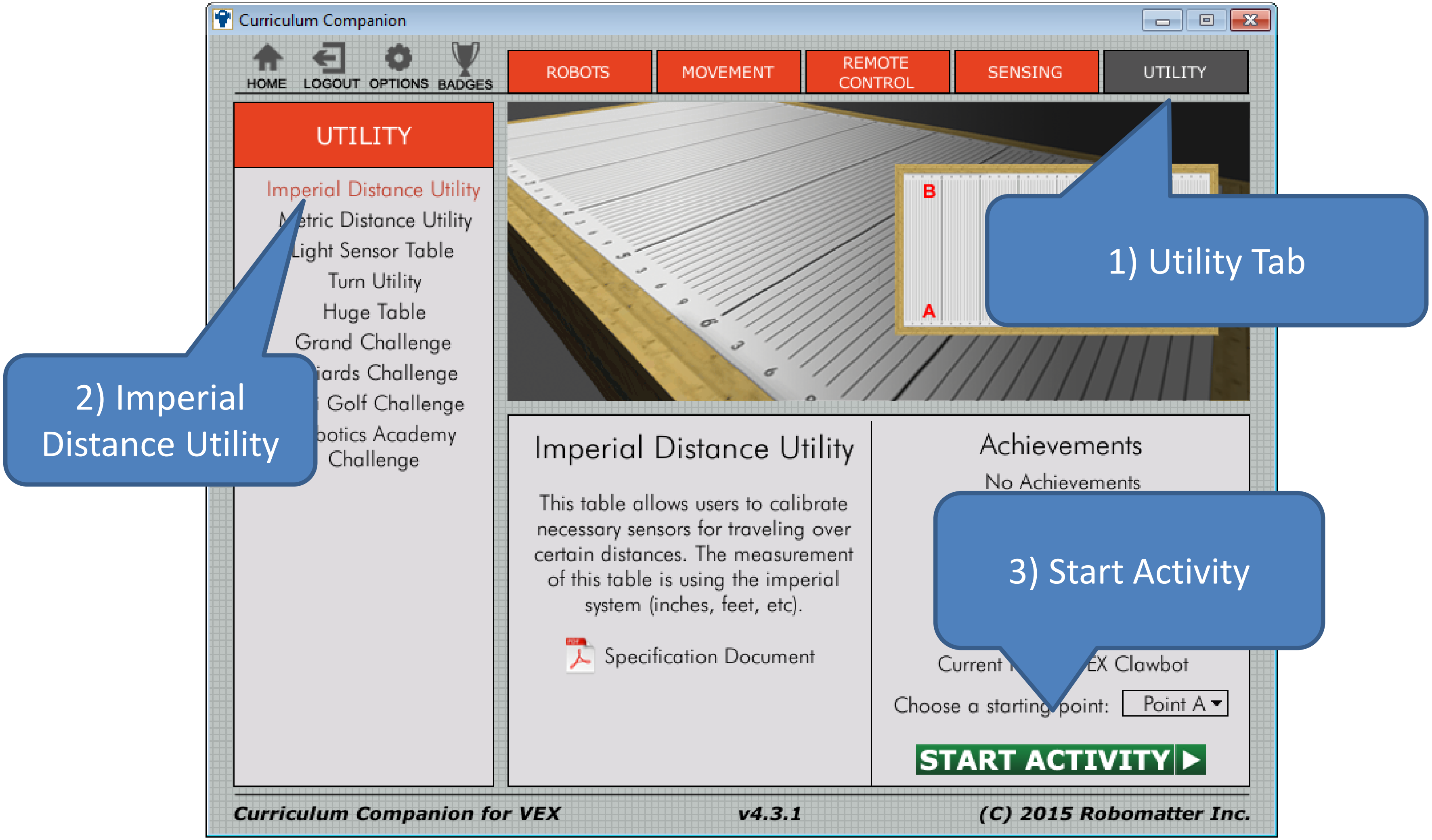
Click to Select Different Challenges

Achievements Some activities allow you

Summary of the activity with a pdf that includes details about the challenge.

Click on 'Start Activity' to begin

We'll Test Our Program in the Utility -> Imperial Distance Utility



Imperial Distance Utility

This table allows users to calibrate necessary sensors for traveling over certain distances. The measurement of this table is using the imperial system (inches, feet, etc).

Achievements

No Achievements

Current EX Clawbot

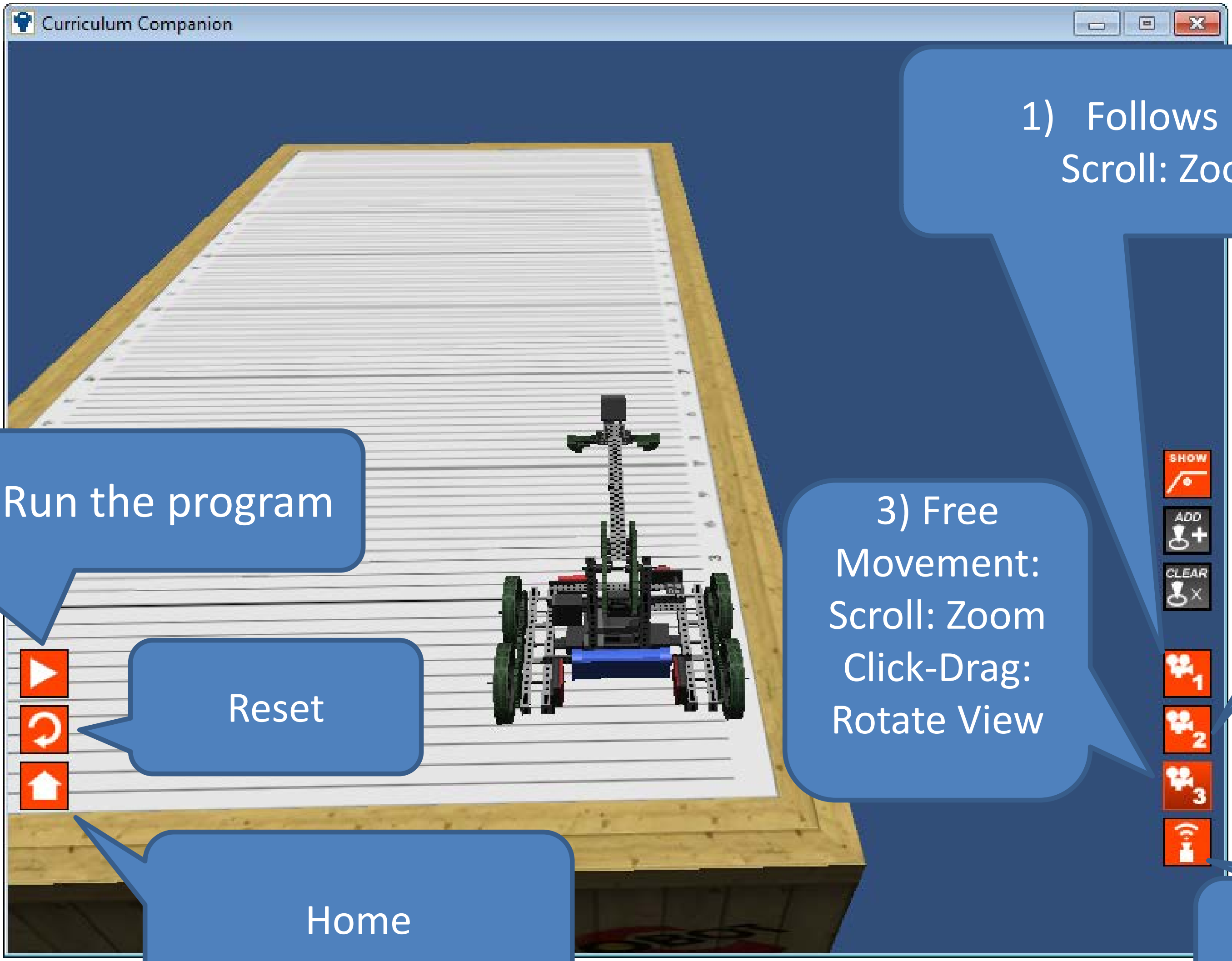
Choose a starting point: Point A

1) Utility Tab

2) Imperial Distance Utility

3) Start Activity

Select Camera and Go



1) Follows Right
Scroll: Zoom

2) Top Down View

3) Free Movement:
Scroll: Zoom
Click-Drag: Rotate View

Play: Run the program

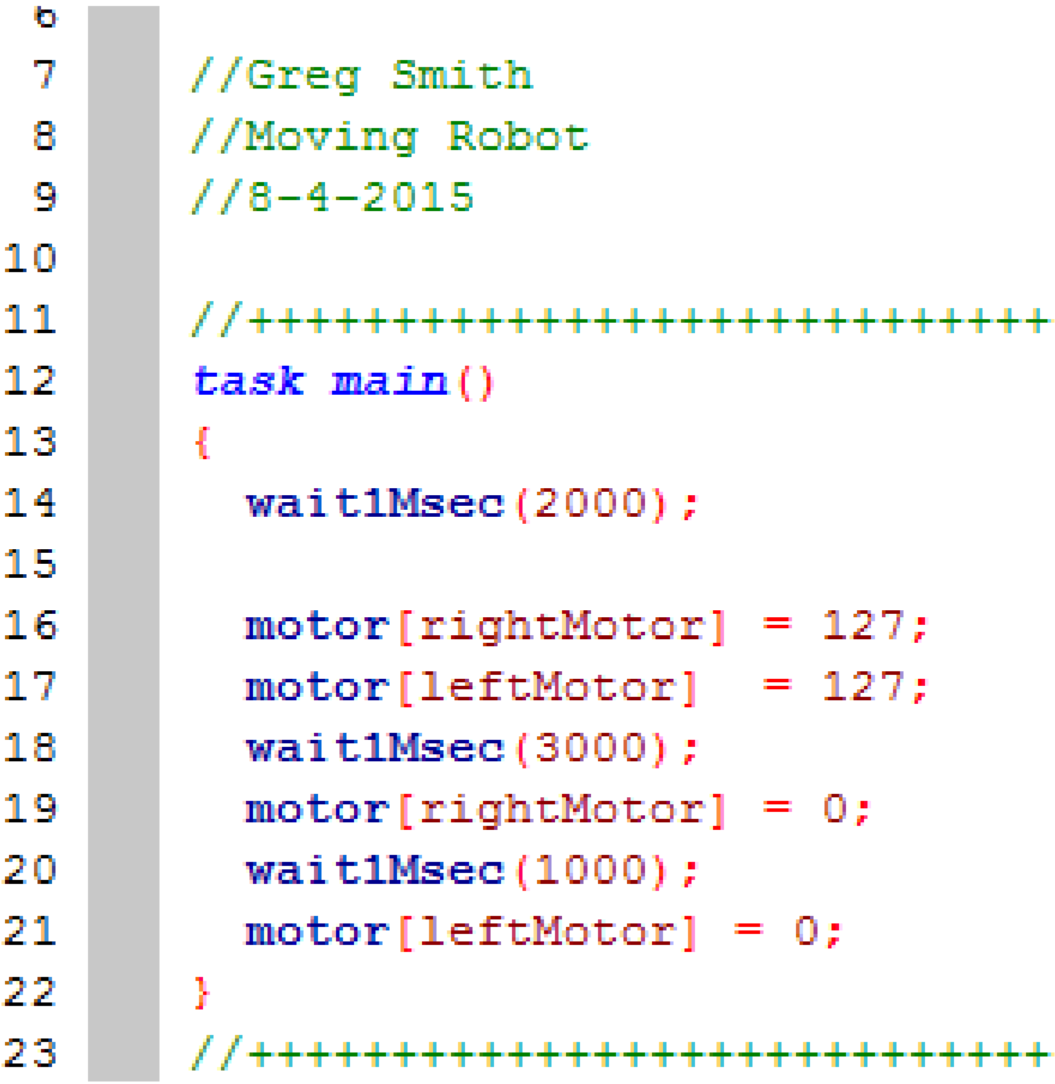
Reset

Home

Show Sensors Toggle

Your Turn

- Enter the Sample program
  - Motors and Sensors Setup
  - task main() and code
- Compile and correct errors
- Download to the virtual robot
- Run the program
- Can you modify this program to…
- Write the letter Z? S?



```
//Greg Smith
//Moving Robot
//8-4-2015

//++++++++++++++++++++++++++++
task main()
{
  wait1Msec(2000);

  motor[rightMotor] = 127;
  motor[leftMotor]  = 127;
  wait1Msec(3000);
  motor[rightMotor] = 0;
  wait1Msec(1000);
  motor[leftMotor] = 0;
}
//++++++++++++++++++++++++++++
```

Reference Website:
http://education.rec.ri.cmu.edu/products/cortex_video_trainer/
Click on Movement for much of the material covered

Teaching Strategy:
Grading Student Programming

Movement: Basketball Drills

Programming (10 Points = 100%)

___ Program compiles (4 points)

___ Header complete with names, description and date (2 points)

___ Code is properly indented (2 points)

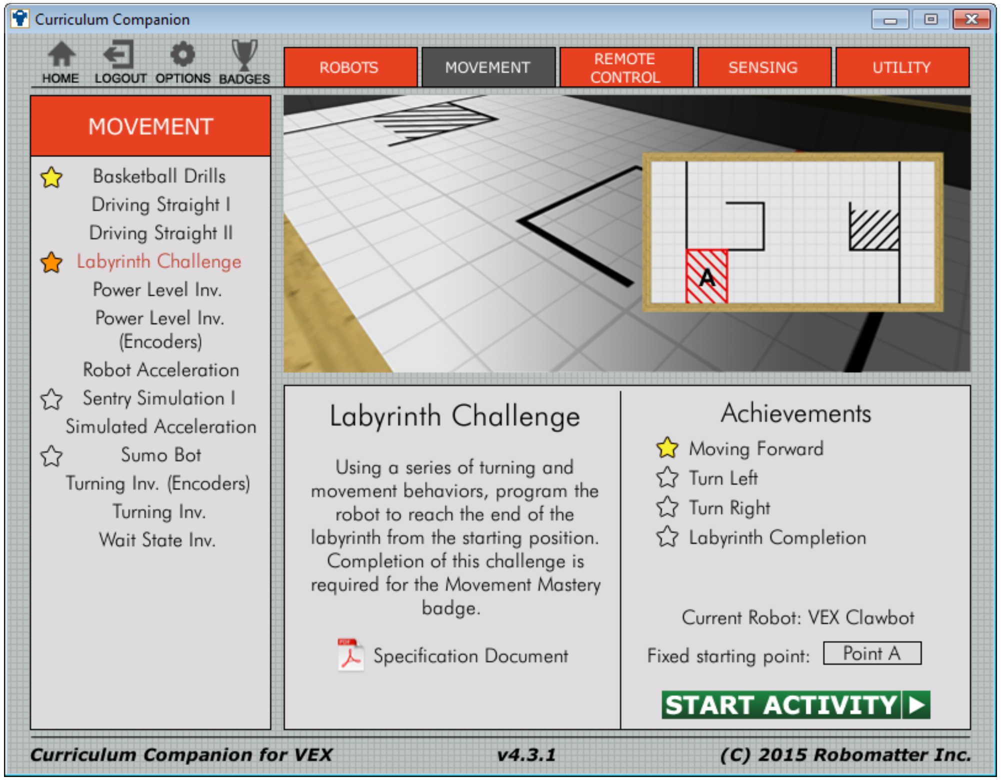
___ Comments in the program describing the code (2 points)

Performance (10 points = 100%)

____ Completed

Online Time: Movement Challenges

- Basketball Drills
- Sentry Simulation 1
- Sumo Bot
- Labyrinth Challenge



Labyrinth Challenge

Using a series of turning and movement behaviors, program the robot to reach the end of the labyrinth from the starting position. Completion of this challenge is required for the Movement Mastery badge.

Specification Document

Achievements

- Moving Forward
- Turn Left
- Turn Right
- Labyrinth Completion

Current Robot: VEX Clawbot

Fixed starting point: Point A

Curriculum Companion for VEX *v4.3.1* *(C) 2015 Robomatter Inc.*

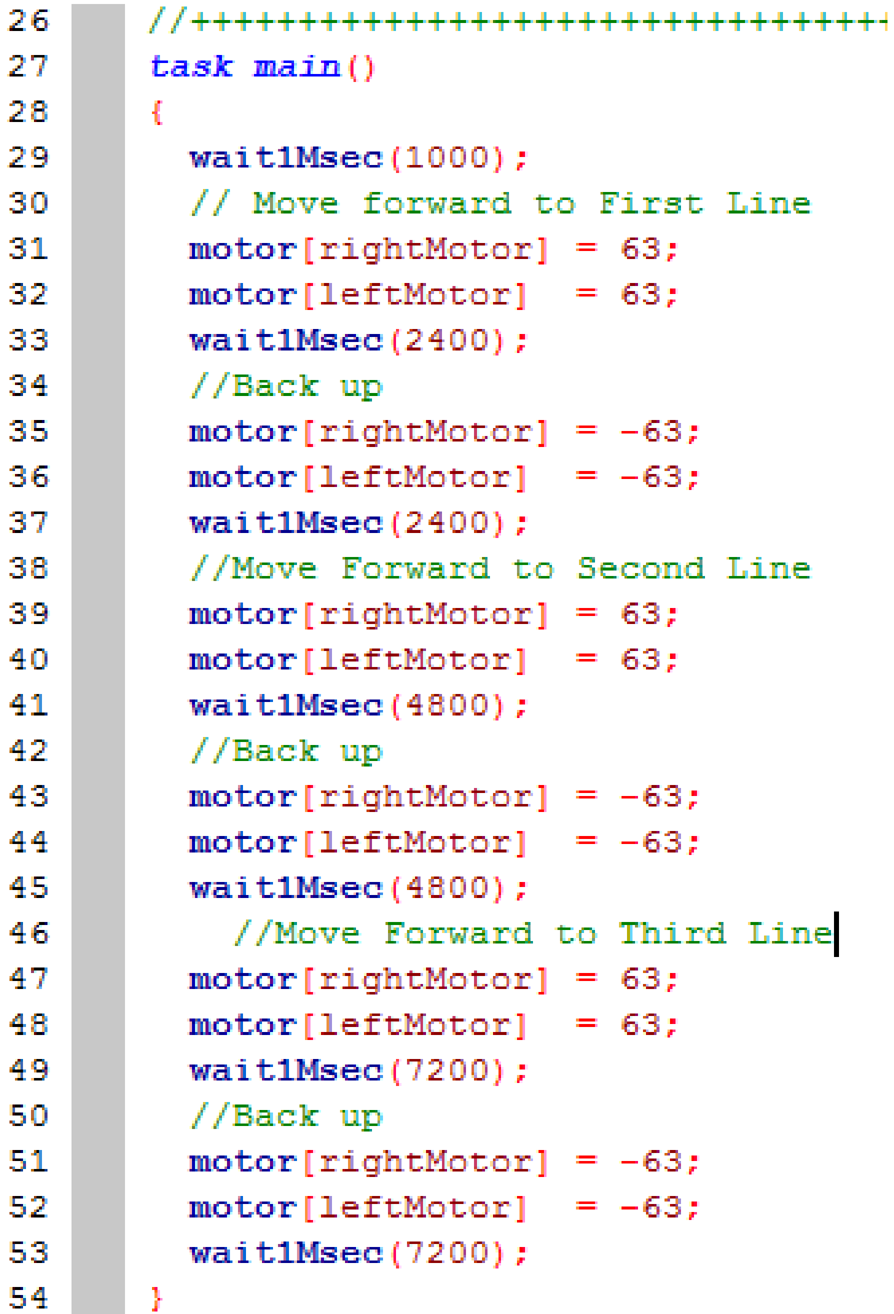
Basketball Drill Programming Alternatives

- Using the Basketball Drills Activity to introduce:
- Variables
- For loop
- Functions

Looking at Potential Solutions to Basketball Drills

- **Pseudo Code**
- Go forward long enough to cross the first line
- Come back
- Go forward long enough to cross the second line
- Come back
- Go forward lone enough to cross the third line
- Come back

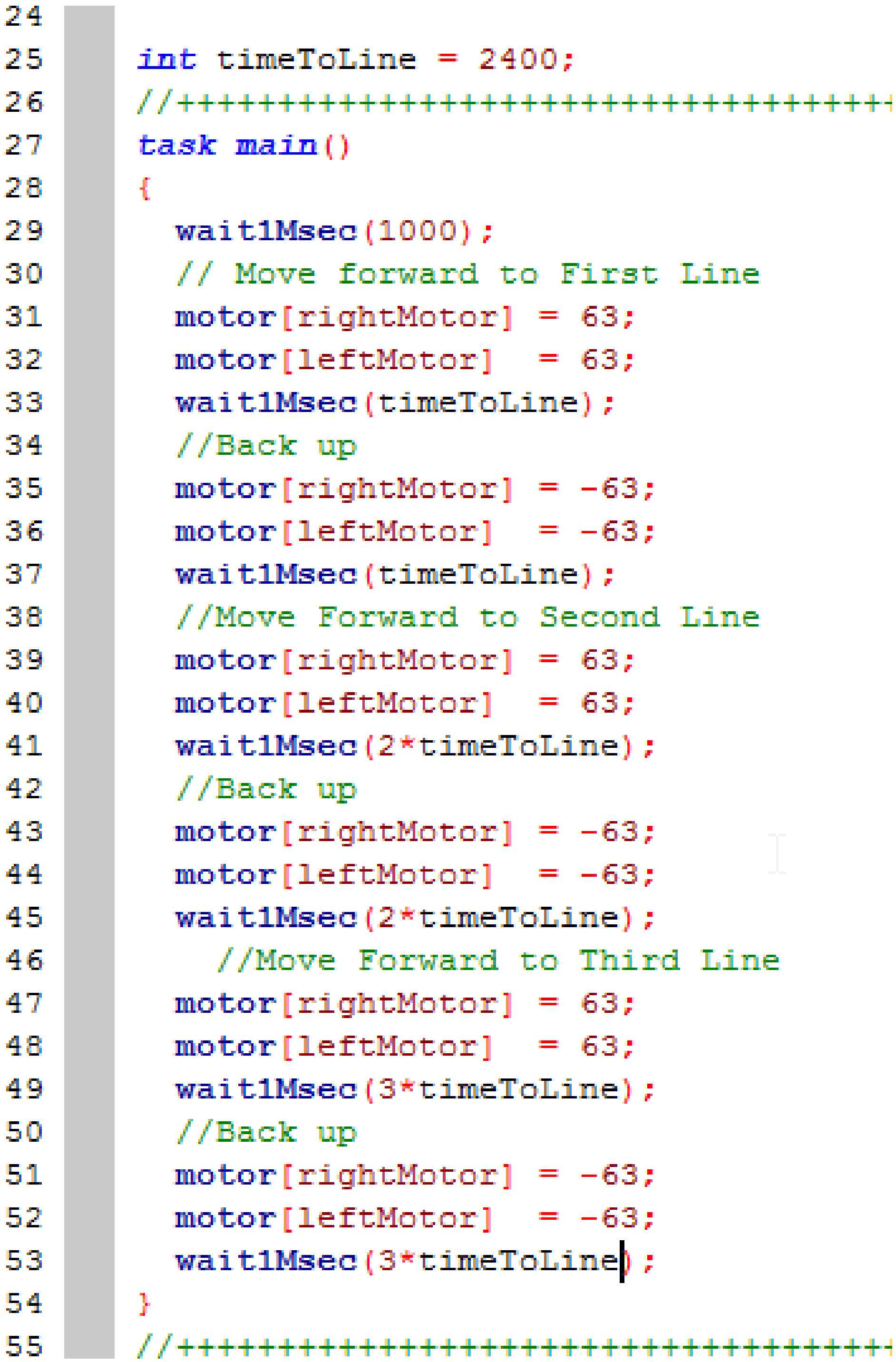
With enough guessing and checking, you can get the correct values for the wait1Msec()



```
//+++++++++++++++++++++++++++++++++
task main()
{
  wait1Msec(1000);
  // Move forward to First Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(2400);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(2400);
  //Move Forward to Second Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(4800);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(4800);
    //Move Forward to Third Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(7200);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(7200);
}
```

Using a Variable to help with changes

If only there was a tool in RobotC that would let the code repeat.



```

int timeToLine = 2400;
//++++++++++++++++++++++++++++++++++++
task main()
{
  wait1Msec(1000);
  // Move forward to First Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(timeToLine);
  //Move Forward to Second Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(2*timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(2*timeToLine);
    //Move Forward to Third Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(3*timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(3*timeToLine);
}
//++++++++++++++++++++++++++++++++++++
```

For loop in RobotC

- When to use it
  - When you want to repeat something a set number of times
- Syntax

Declares an integer variable called **line** and gives it an initial value of 1

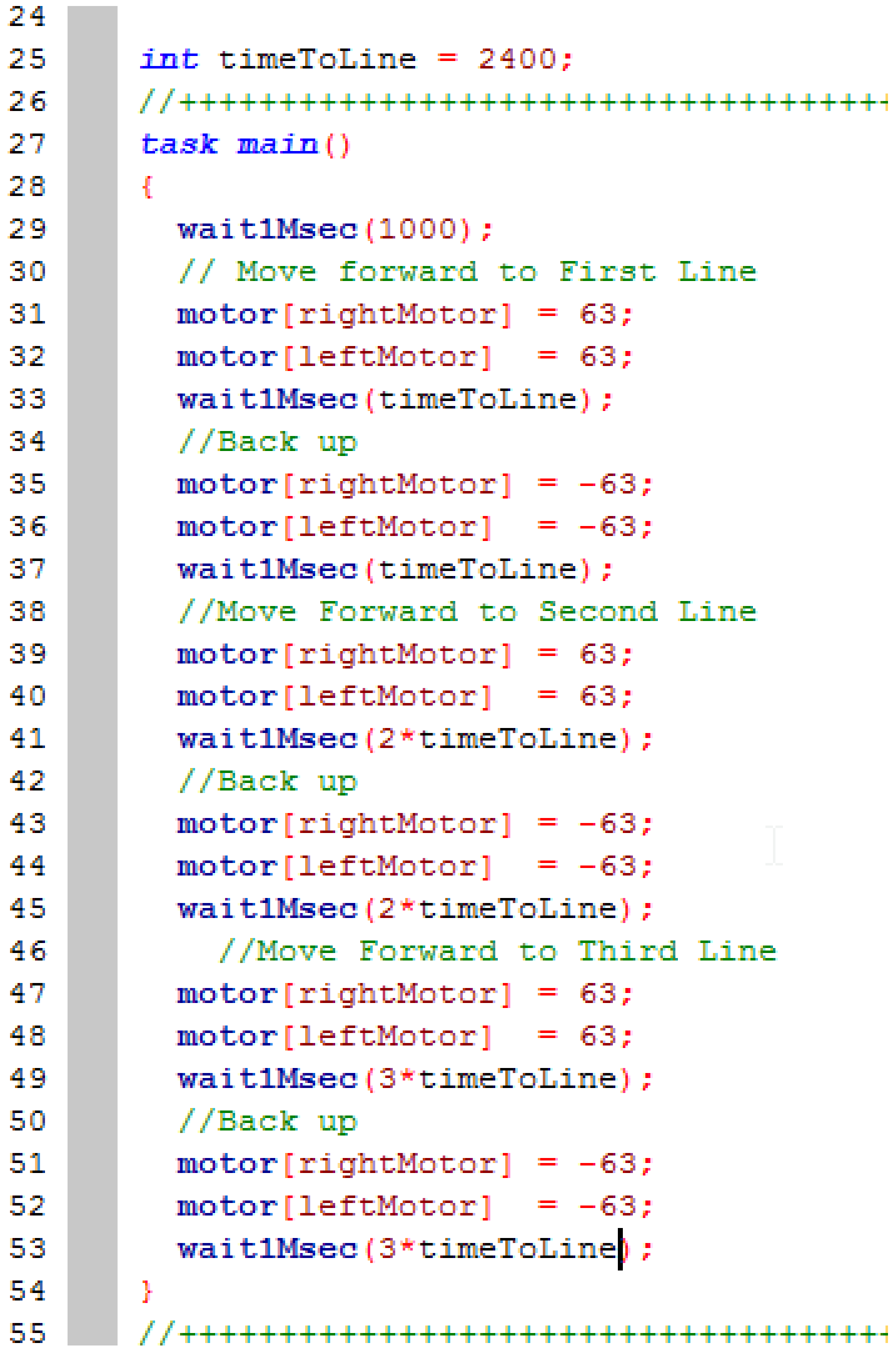
If the **line** variable is **less than or equal to 3** when it reaches this, it will do the loop another time.

After completing the loop, it will add **add 1** to the variable **line**.

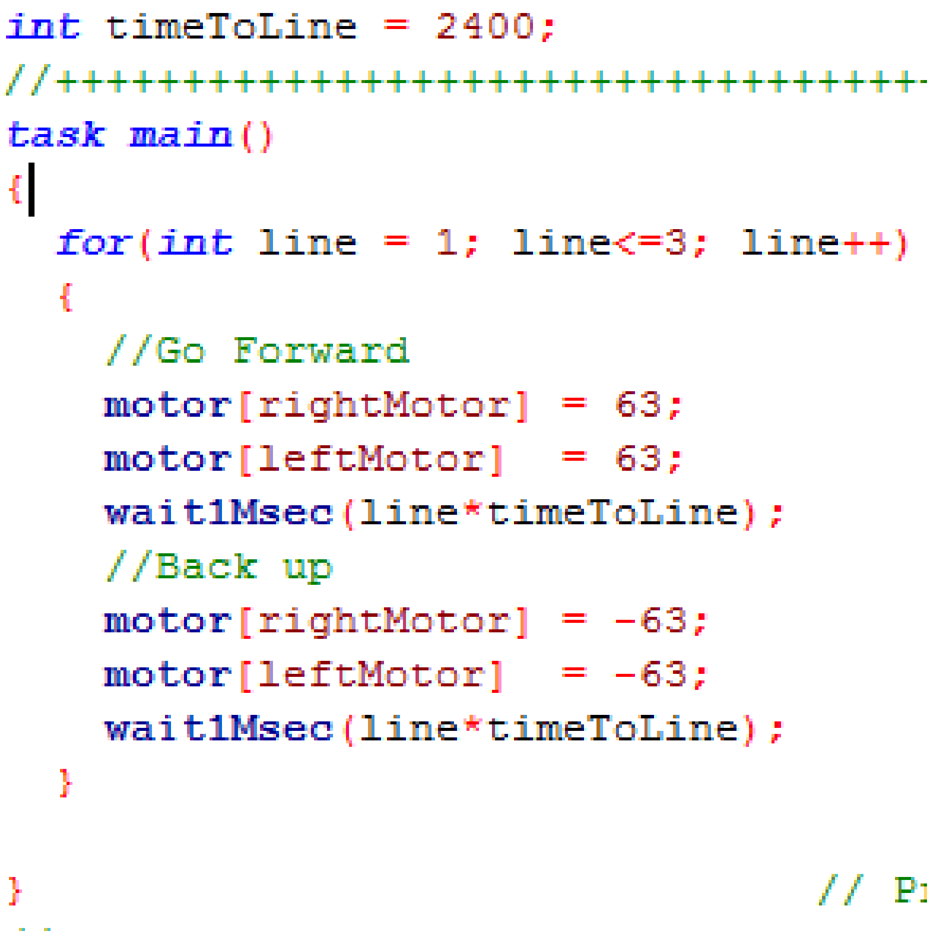
```
for(int line = 1; line<=3; line++)
{
        //Code repeated
}
```

In this example it will repeat the code inside the {} three times.
Once when line = 1
Once when line = 2
And
Once when line = 3

No loop vs. **for loop**

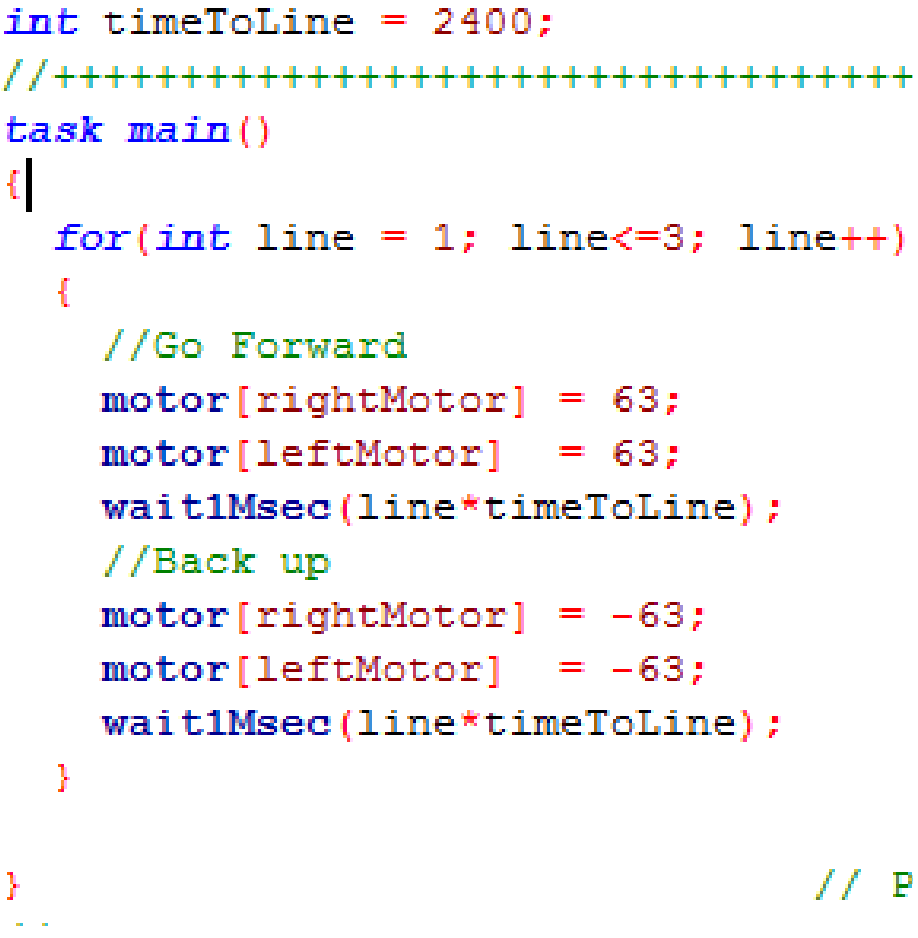


```
int timeToLine = 2400;
//++++++++++++++++++++++++++++++++++++
task main()
{
  wait1Msec(1000);
  // Move forward to First Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(timeToLine);
  //Move Forward to Second Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(2*timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(2*timeToLine);
    //Move Forward to Third Line
  motor[rightMotor] = 63;
  motor[leftMotor]  = 63;
  wait1Msec(3*timeToLine);
  //Back up
  motor[rightMotor] = -63;
  motor[leftMotor]  = -63;
  wait1Msec(3*timeToLine);
}
//++++++++++++++++++++++++++++++++++++
```



```
int timeToLine = 2400;
//++++++++++++++++++++++++++++++++++++++
task main()
{
  for(int line = 1; line<=3; line++)
  {
    //Go Forward
    motor[rightMotor] = 63;
    motor[leftMotor]  = 63;
    wait1Msec(line*timeToLine);
    //Back up
    motor[rightMotor] = -63;
    motor[leftMotor]  = -63;
    wait1Msec(line*timeToLine);
  }

}                                   // P
```

For loop example



```
int timeToLine = 2400;
//++++++++++++++++++++++++++++++++++
task main()
{
  for(int line = 1; line<=3; line++)
  {
    //Go Forward
    motor[rightMotor] = 63;
    motor[leftMotor]  = 63;
    wait1Msec(line*timeToLine);
    //Back up
    motor[rightMotor] = -63;
    motor[leftMotor]  = -63;
    wait1Msec(line*timeToLine);
  }

}                                // P
```

Since line = 1 the first time through this loop

line*timeToLine is the same as

1*2400 = 2400

the first time through this loop.

Then

2*2400 = 4800

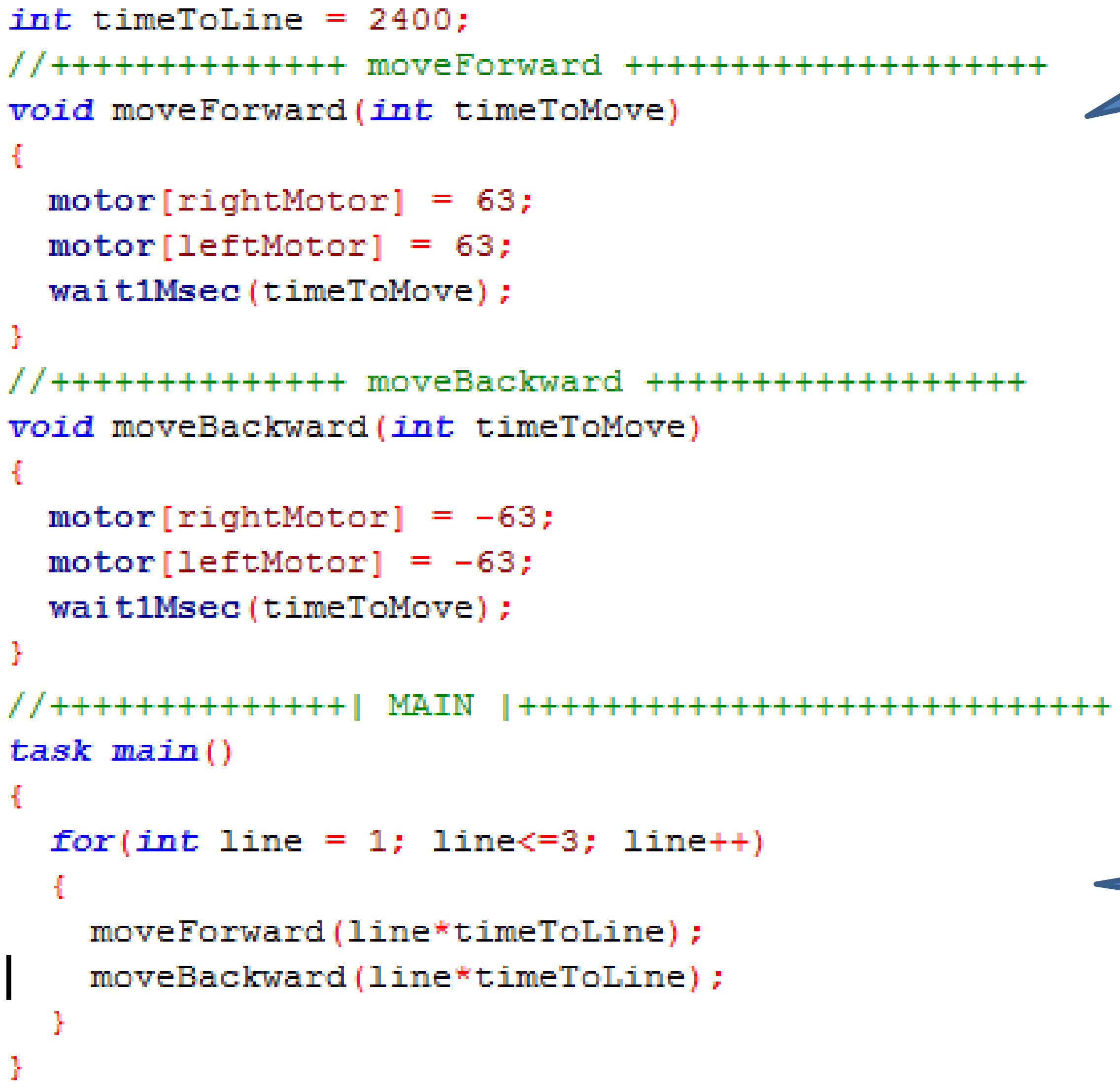
the second time and

3*2400 = 7200

the third time.

RobotC does the math inside the () before executing the wait1Msec() command

Using Functions to make the **main body** easier to read



```
int timeToLine = 2400;
//++++++++++++++ moveForward ++++++++++++++++++++
void moveForward(int timeToMove)
{
  motor[rightMotor] = 63;
  motor[leftMotor] = 63;
  wait1Msec(timeToMove);
}
//++++++++++++++ moveBackward ++++++++++++++++++
void moveBackward(int timeToMove)
{
  motor[rightMotor] = -63;
  motor[leftMotor] = -63;
  wait1Msec(timeToMove);
}
//++++++++++++++| MAIN |++++++++++++++++++++++++++++
task main()
{
  for(int line = 1; line<=3; line++)
  {
    moveForward(line*timeToLine);
    moveBackward(line*timeToLine);
  }
}
```

Define the Functions above the main body.

Main Body

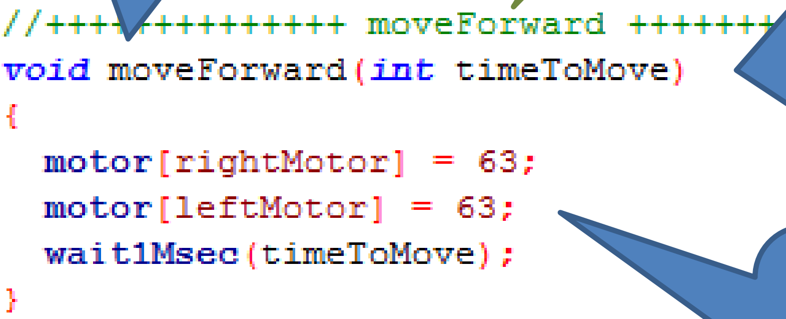
Func
Deta

The function 'Header'
void – It will not return a value
moveForward – The name of this function. You get to pick the name of you function as long as:
-Starts with a letter
-No spaces or punctuation
-Not a reserved Word
And it should describe what it is doing.

//+++++
moveForward
Comments added to make the program easier to read. You can add details, …

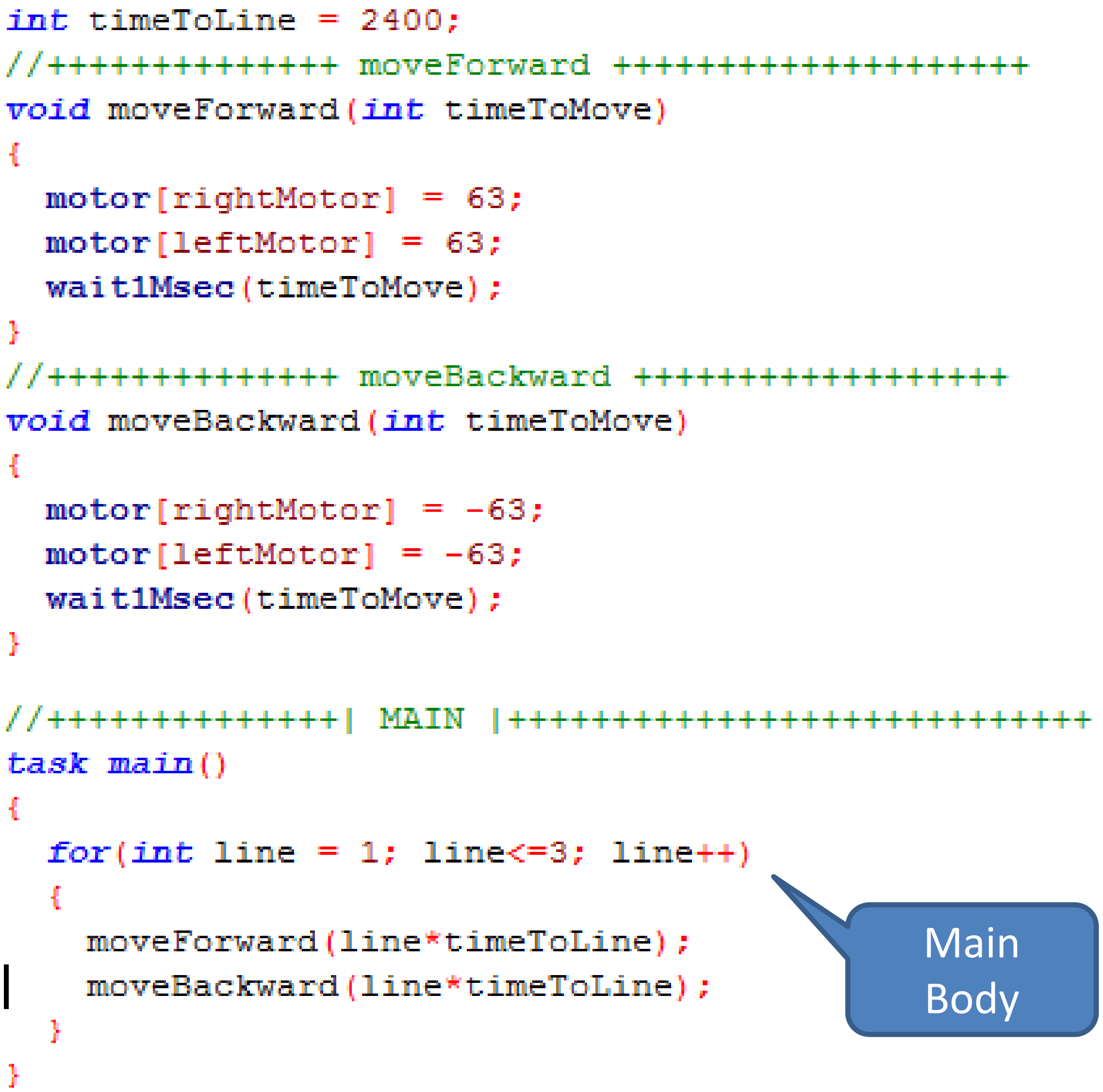
int timeToMove
int – Sets an integer variable
timeToMove – An integer variable that will store the value sent to the function in the call statement.

The code for the function goes between {}. When the function is finished the program will return to the line after the call statement.



```
//+++++++++++++++ moveForward +++++++
void moveForward(int timeToMove)
{
  motor[rightMotor] = 63;
  motor[leftMotor] = 63;
  wait1Msec(timeToMove);
}
```

Dry Run: Reading the Program



```
int timeToLine = 2400;
//+++++++++++++++ moveForward +++++++++++++++++++++
void moveForward(int timeToMove)
{
  motor[rightMotor] = 63;
  motor[leftMotor] = 63;
  wait1Msec(timeToMove);
}
//+++++++++++++++ moveBackward +++++++++++++++++++
void moveBackward(int timeToMove)
{
  motor[rightMotor] = -63;
  motor[leftMotor] = -63;
  wait1Msec(timeToMove);
}

//+++++++++++++++| MAIN |++++++++++++++++++++++++++++
task main()
{
  for(int line = 1; line<=3; line++)
  {
    moveForward(line*timeToLine);
    moveBackward(line*timeToLine);
  }
}
```

Main Body

<u>timeToMove</u>

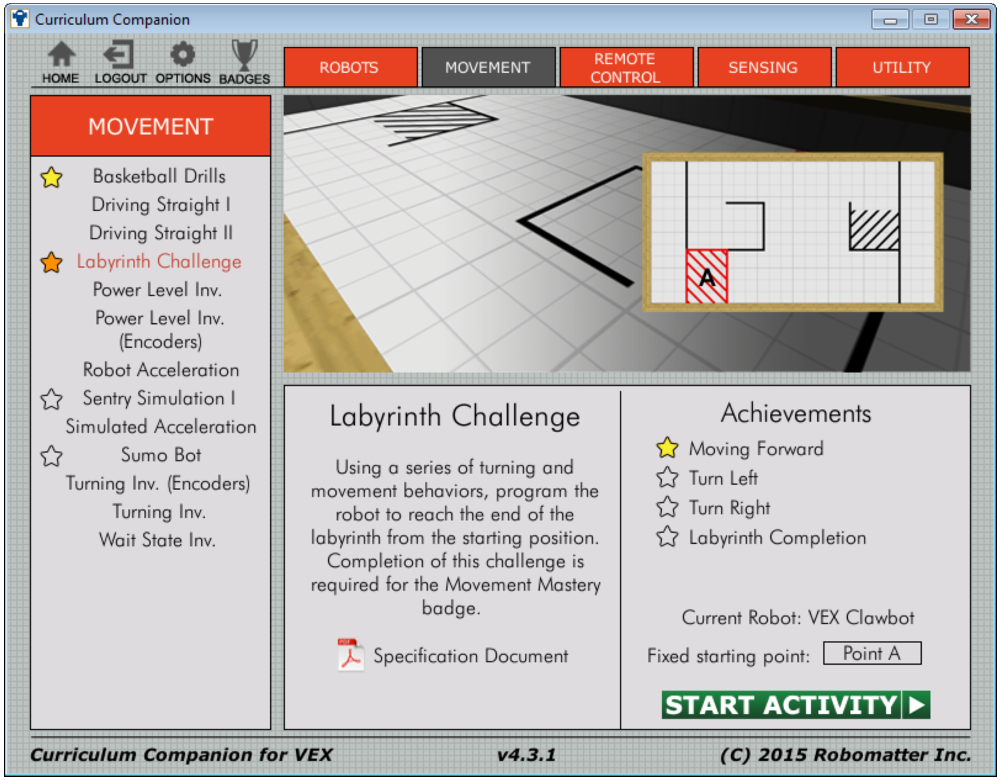
<u>timeToMove</u>

<u>line</u>

Online Time: Movement Challenges

- Basketball Drills
- Sentry Simulation 1
- Sumo Bot
- Labyrinth Challenge

When you complete the activities, incorporate variables, loops, and functions



MOVEMENT

★ Basketball Drills
Driving Straight I
Driving Straight II
★ Labyrinth Challenge
Power Level Inv.
Power Level Inv. (Encoders)
Robot Acceleration
☆ Sentry Simulation I
Simulated Acceleration
☆ Sumo Bot
Turning Inv. (Encoders)
Turning Inv.
Wait State Inv.

Labyrinth Challenge

Using a series of turning and movement behaviors, program the robot to reach the end of the labyrinth from the starting position. Completion of this challenge is required for the Movement Mastery badge.

Specification Document

Achievements

★ Moving Forward
☆ Turn Left
☆ Turn Right
☆ Labyrinth Completion

Current Robot: VEX Clawbot

Fixed starting point: Point A

Curriculum Companion for VEX *v4.3.1* *(C) 2015 Robomatter Inc.*

References

- http://education.rec.ri.cmu.edu/products/cortex_video_trainer/