What is ROBOTC?

- A complete ‘C’ programming language for robotics.
  - Developed for Teaching, Powerful enough for Hobbyists/Professionals
  - Developed in Microsoft Visual C++ 2008
- Extremely easy to use for both beginners and advanced users
  - Most “robot functions” are a single line commands.
- Competition Friendly/Approved – FTC
  - Easy to use template architecture allows students to create basic competition-ready program in only minutes.

What is ROBOTC?

- Developed for Education
  - Students learn in a friendly environment about C-Based programming
  - Friendlier Compiler, GUI Device Configuration, Code Templates
- C-Based programming offers more transferability of job-related skills compared to graphical languages.
- C is the most popular programming language in the world!
  - Used in development of LINUX, Windows and Mac Operating Systems

What is ROBOTC?

- C Language extensions for robotics
  - Built-in variables for robotics devices – motors, sensors, joysticks
  - Example: One line command to drive motors
    - NXT - motor[motorA] = 100; //Turn Motor A @ 100% forward
- User “friendly” compiler
  - Auto-correct from popular programming errors
  - “:” for “;”, . . .
  - Letter case errors in variable names: “playSound” for “PlaySound”.

First Program - "Moving Forward"

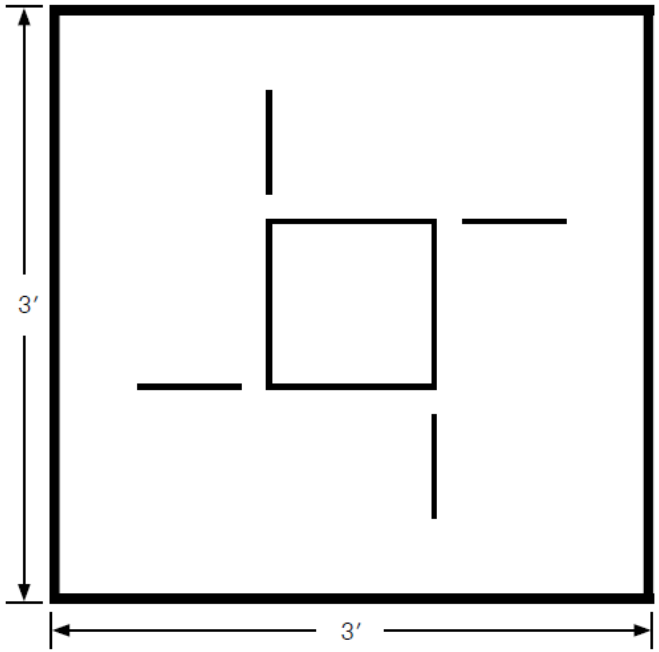
- Live Walkthrough
  - Open ROBOTC's software
  - Download the ROBOTC firmware
  - Open a sample program
  - Download the sample program to the EV3
- Equivalent Lessons
  - Setup – EV3 Setup Section
  - Movement – Moving Forward

Self Paced Lab: Speed & Direction

- Follow along with the videos!
  - Watch the two videos in the "Speed and Direction" chapter of the “Movement” section.
- If finished early…
  - Review the “Speed and Direction” quiz
  - See if you can answer the questions

Robo 500 Challenge

- Notice the Large Book on the tables?
- Complete the "Robo 500 Challenge”
  - First “flowchart” or plan your program
  - Then program your robot
- If finished early…
  - Make sure your code is commented
  - Try to make the robot turn around and complete the Robo500 in reverse.



Robo Rmatter INCORPORATED

Carnegie Mellon Robotics Academy

Speed and Direction

- Any Questions?
- Notes
  - Motor speeds are always between -100 and +100
  - Giving a motor a speed above 100 will not break anything, ROBOTC will just ignore it and set the speed to 100.
  - Motors may have difficulty moving at speeds between -5 and +5
    - This is because of internal resistance from the gears inside the EV3's motor

Planning and Behaviors

- What is the role of the programmer?
- How does the connection between Robots and Programmer work?
  - Humans: Create Plans
  - Robots: Follow out Plans (exactly as told)
    - Sometimes a little too well...

Planning and Behaviors

- What is the best method to create a plan?
- Behavior based planning
  - Each behavior then breaks down into multiple "steps"
  - Each "step" usually correlates to a single line of code in ROBOTC
- Give an example of a daily activity and the behaviors involved

ROBOTC Rules/Syntax

- What does every program have to have?
  - task main()
  - Set of "Curly Braces“ **{}**
- In what order does ROBOTC run a program?
  - Sequentially – Top to Bottom
- What is whitespace?
  - Space in a program to make it easier for a human to read.
- Simple Robot Commands (statements) always end with a…?
  - Semicolon **;**

ROBOTC Rules/Syntax

- Paired Punctuation
  - Some functions have "square brackets“ **[]** , other have "parenthesis“ **()**
  - How do I keep them straight?
    - Memorization
    - Functions Library
- Control Structures
  - Just like task main, they have a set of curly braces defining their beginning and end. **{}**
  - Example?
- Comments
  - **//Necessary Evil**
  - Get into the habit now, or you won't do it ever
  - Your students learn their habits from you…

ROBOTC Rules/Syntax

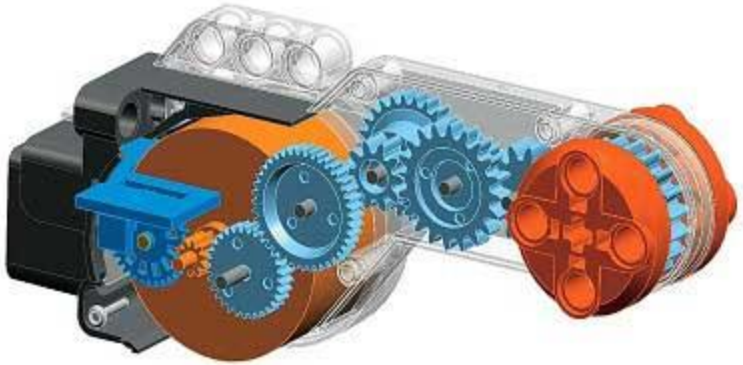
- Syntax is the cause of 80% of programming errors…
- Most important thing to do when troubleshooting: Check Syntax
  - Missing Semicolons
  - Incorrect Braces on Structures
  - Misspelled words
  - Improper ‘case’ – motor vs. MoToR
- ROBOTC is friendly about most of these errors... but can’t catch everything.

Robot Functionality

- wait1Msec(2000)…
  - Timing isn't everything
  - Limited to Battery Power
  - Surface and Traction of the wheel
- There has to be a better way!
  - The NXT motor has one really nice feature…

NXT Motors / Encoders

- Semi-powerful DC motors
  - Equipped with Encoders
  - Can measure the distance traveled
- Returns 360 "counts" per revolution
  - What else has 360 counts?
- NXT Encoders also enable us to "control" the speed of our motors
  - PID Algorithms…
  - It’s okay if you don’t know what that is yet.



NXT Encoders

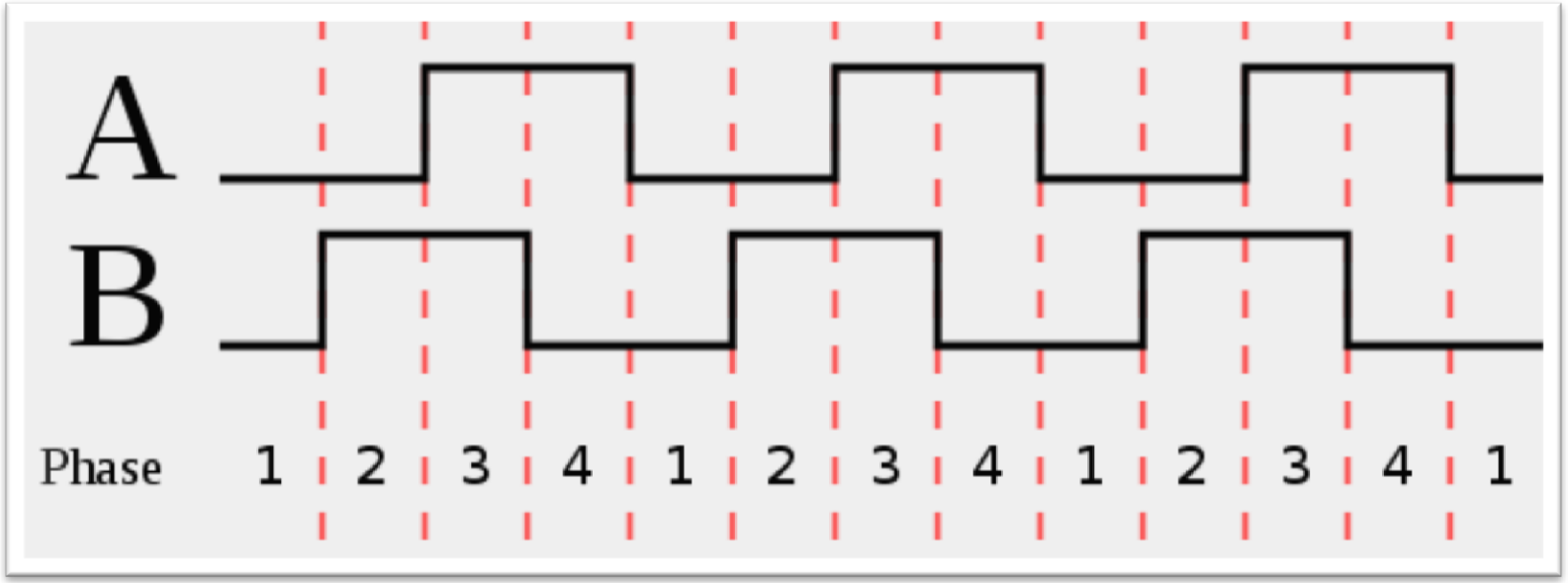
- The NXT Encoder is consider a “Quadrature Encoder”
  - NXT encoders have two outputs called A & B, which are called quadrature outputs, as they are 90 degrees out of phase.
  - This allows the encoder to know distance travel, but also direction traveled.

Robo Rmatter INCORPORATED

Carnegie Mellon Robotics Academy

NXT Encoders

- Quadrature Timing Diagram



Self Paced Lab: Improved Movement

- Follow along with the videos!
  - Watch 3 videos videos in "Improved movement"
  - **Skip "Synchronized Motors”**
- **Follow Along with the videos!**
  - **Write the programs out along side of the videos to try the nMotorEncoderTarget functionality**

Review: Improved Movement

- Any Questions?
- PID Notes
  - PID = Useful for regulating speed
  - Target Distances = Useful for making a motor travel a specific distance
  - PID is on by default for NXT motors
  - PID doesn't work very well at "high" speeds
  - There's no room for the motor to compensate for resistance.

Review: Improved Movement

- nMotorEncoder/nMotorEncoderTarget:
  - nMotorEncoder[motorname]
    - Returns the value of the motor encoder as a value
  - nMotorEncoderTarget[motorname]
    - Sets a relative target for the motor to travel.
    - Positive or negative numbers here don't make a different, it's the total distance you want to travel
  - motor[motorname]
    - Sets the motor speed and direction. Also determines the direct of the Target (either positive or negative)

Encoder Targets

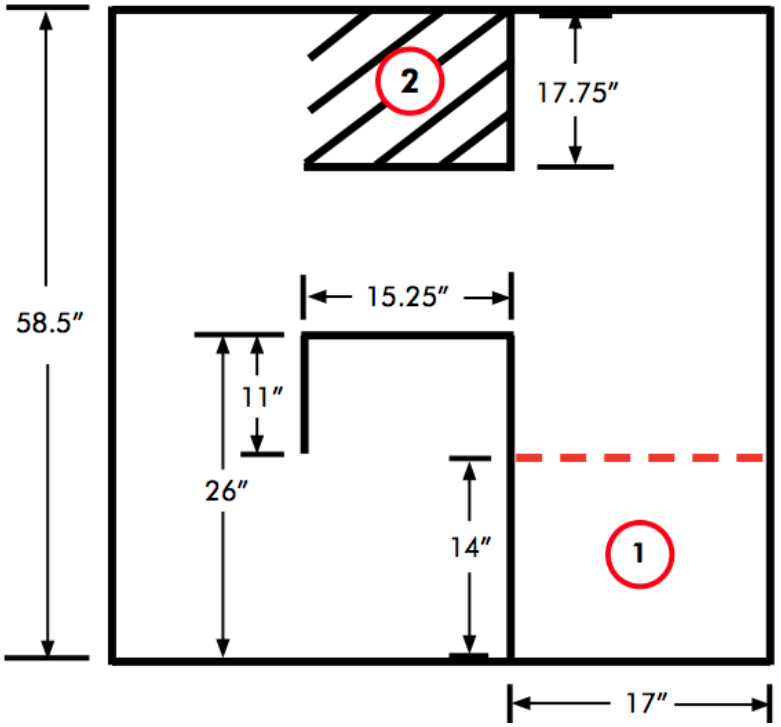
Three steps to move precise distances:

1) Clear the Encoders
2) Set the Encoder Target
3) Move Forward

```
task main()
{

  nMotorEncoder[motorC] = 0;
  nMotorEncoder[motorB] = 0;

  nMotorEncoderTarget[motorC] = 360;
  nMotorEncoderTarget[motorB] = 360;

  motor[motorC] = 30;
  motor[motorB] = 30;

  wait1Msec(5000);
}
```

Challenge: Labyrinth

- Complete the ”Labyrinth Challenge”
  - First “flowchart” or plan your program
  - Then program your robot
- If finished early…
  - Make sure your code is commented
  - Go read the “PID” article on Wikipedia ☺



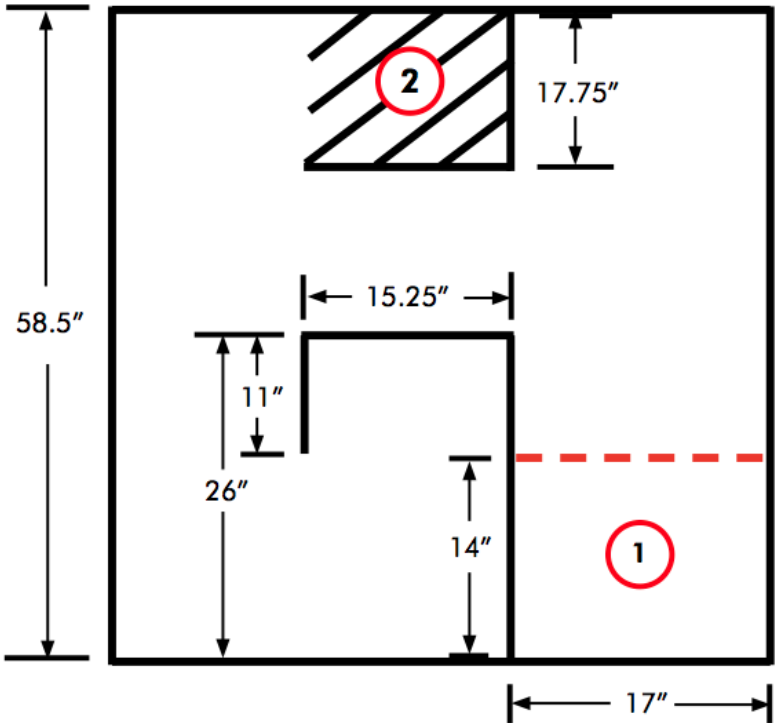
Three steps to move precise distances:

1) **Clear the Encoders**
2) **Set the Encoder Target**
3) **Move Forward**

```
task main()
{

  nMotorEncoder[motorC] = 0;
  nMotorEncoder[motorB] = 0;

  nMotorEncoderTarget[motorC] = 360;
  nMotorEncoderTarget[motorB] = 360;

  motor[motorC] = 30;
  motor[motorB] = 30;

  wait1Msec(5000);
}
```

Challenge: Labyrinth

- Complete the ”Labyrinth Challenge”
  - First “flowchart” or plan your program
  - Then program your robot
- If finished early…
  - Make sure your code is commented
  - Go read the “PID” article on Wikipedia ☺



Three steps to move precise distances:

1) **Clear the Encoders**
2) **Set the Encoder Target**
3) **Move Forward**

```
task main()
{

  nMotorEncoder[motorC] = 0;
  nMotorEncoder[motorB] = 0;

  nMotorEncoderTarget[motorC] = 360;
  nMotorEncoderTarget[motorB] = 360;

  motor[motorC] = 30;
  motor[motorB] = 30;

  wait1Msec(5000);
}
```