

16-311: Getting Started with ROBOTC and the LEGO Mindstorms NXT

Aurora Qian, Billy Zhu

May, 2016

Table of Contents

1. Download and Install
2. License Activation
3. Wireless Connection
4. Running Programs
5. Sample Program
6. Important API
 - a. Motor & Sensor Setup
 - b. Motor
 - c. Sensor
 - d. Waiting
 - e. Timer
 - f. Encoder
 - g. Printing
 - h. Joystick
7. Troubleshooting
8. References

Download & Install

Step 1. Download ROBOTC from <http://robotc.net>

- a. Select **Free Trial Download**

Free Trial Download

- b. Select **NXT**



- c. Click **Download version 4.xx**

Download version 4.52
(includes free trial)

[Download .msi file](#)

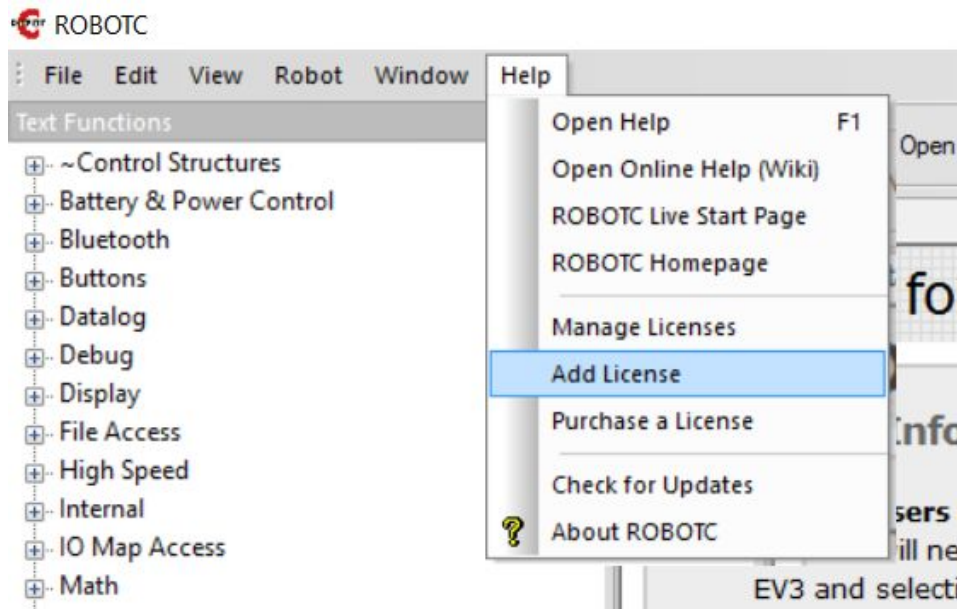
- d. Run the installer, and the program will look like this



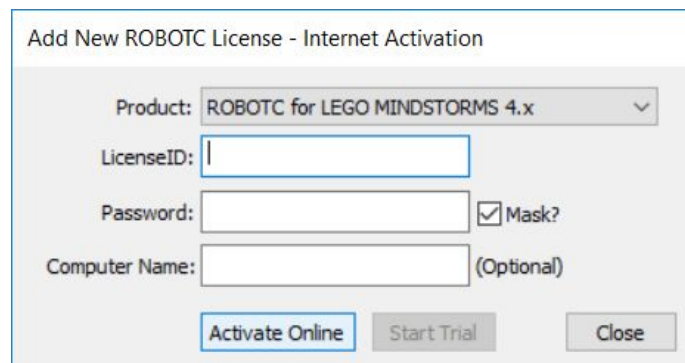
License Activation

Step 1. Open the “Add License” window

Help > Add License



Step 2. Enter the License ID and Password provided to you

A screenshot of the 'Add New ROBOTC License - Internet Activation' dialog box. The title bar reads 'Add New ROBOTC License - Internet Activation'. The dialog contains the following fields and controls:

- 'Product:' dropdown menu set to 'ROBOTC for LEGO MINDSTORMS 4.x'.
- 'LicenseID:' text input field.
- 'Password:' text input field with a checked 'Mask?' checkbox.
- 'Computer Name:' text input field with '(Optional)' text to its right.
- Three buttons at the bottom: 'Activate Online' (highlighted in blue), 'Start Trial', and 'Close'.

Step 3. Click “Activate Online”

Wireless Connection

On your brick

When you first try to connect your brick to your computer through bluetooth, you need to enable bluetooth connection on your brick.

Step 1. Switch the brick on

Press the orange button on the brick



Step 2. Enable Bluetooth on the brick

Bluetooth > On/Off > On

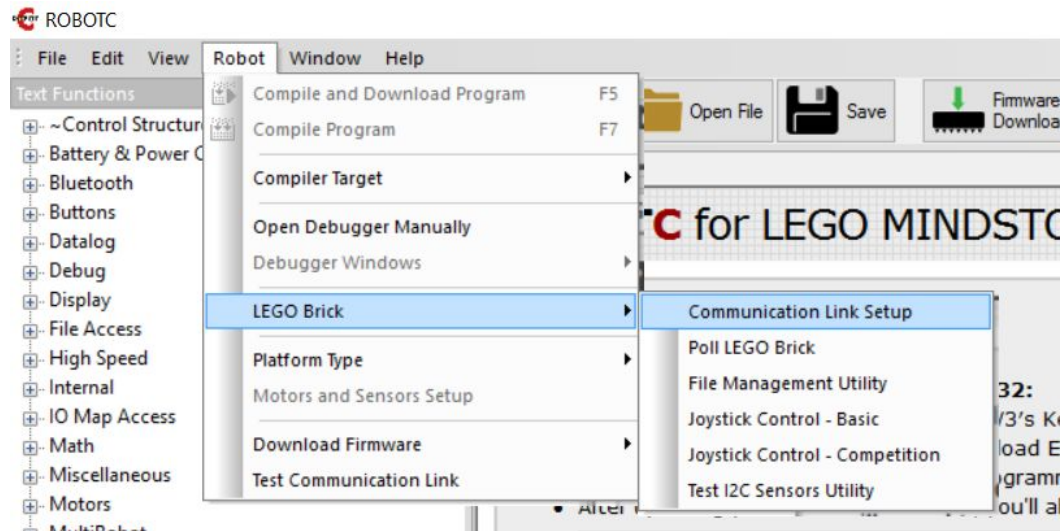
Step 3. Set Bluetooth to visible

Bluetooth > Visibility > Visible

In ROBOTC

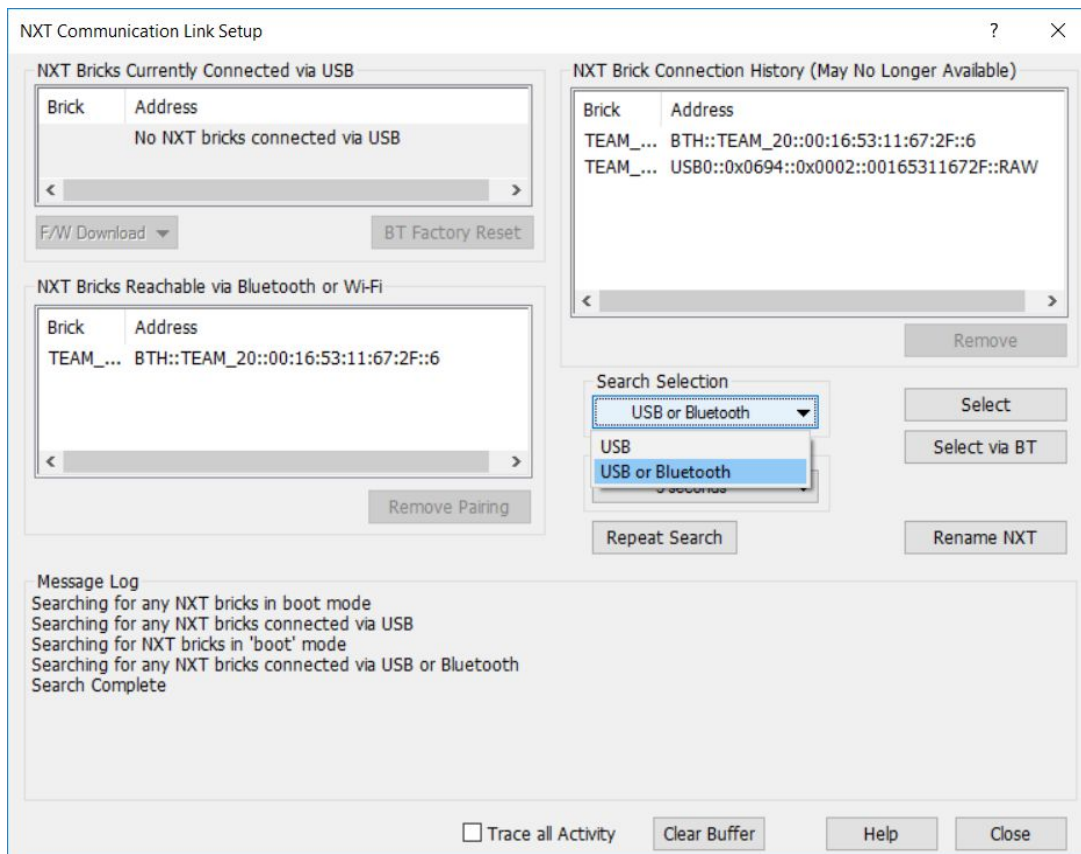
Step 1. In the top left menu, open “communication link setup” window

Robot > NXT Brick > Communication Link Setup



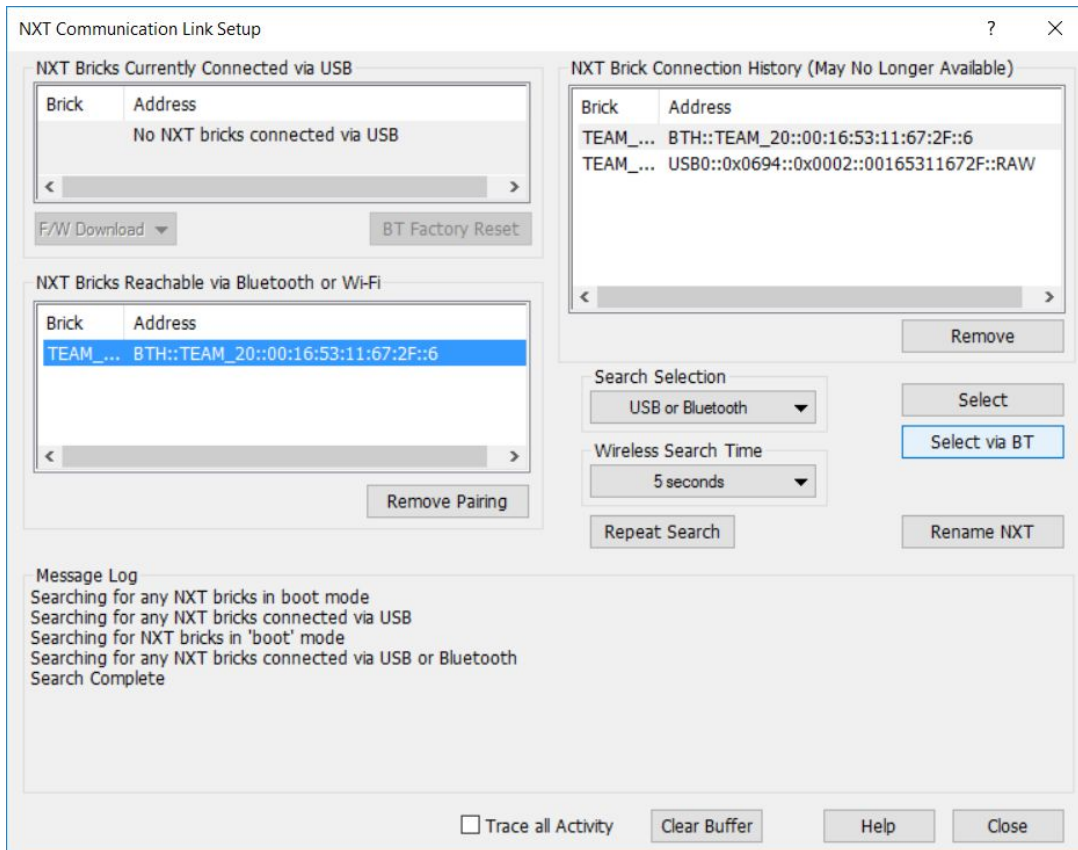
Step 2. Search your NXT brick

In the “NXT Brick Link Selection” screen, change the "Search Selection" drop menu to "USB or Bluetooth" and click "Repeat Search".

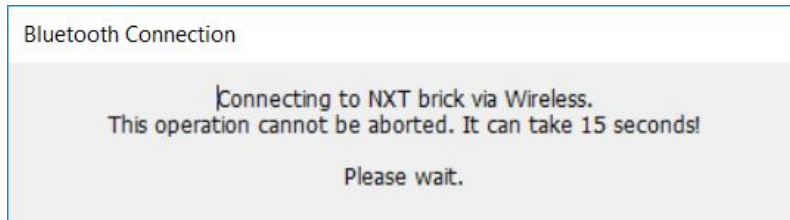


Step 3. Select the brick you want to connect to.

Click the name of the brick under the "NXT Bricks Reachable via Bluetooth Wireless or Wifi" menu and then click the "Select via BT" button on the right.



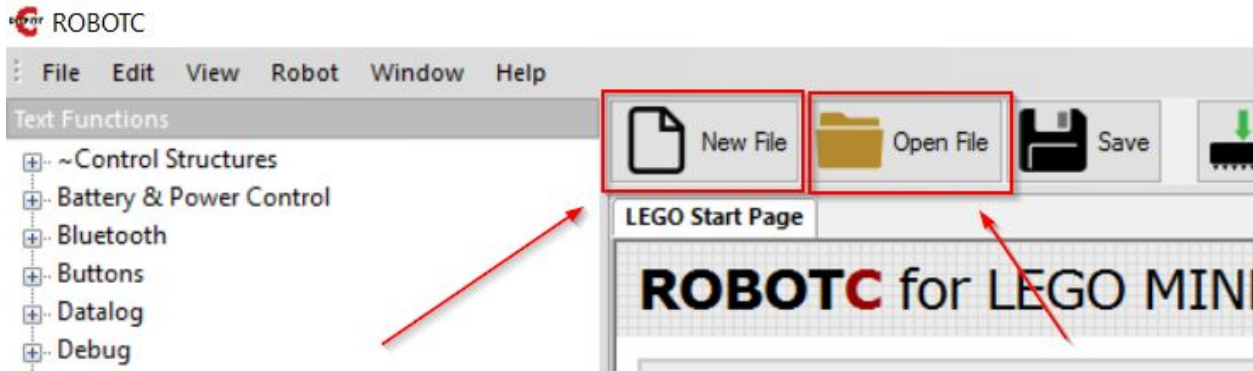
Step 4. ROBOTC will start to pair with the NXT.



Step 5. Once the "Bluetooth Connection" window disappear you have successfully connected your brick with ROBOTC through bluetooth.

Running Programs

Step 1. Create a new source code file (OR if you already have one, open it)

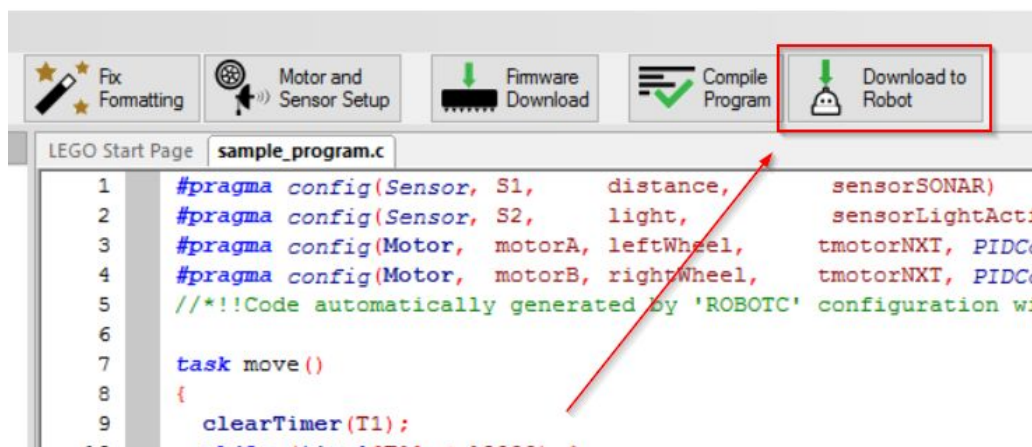


Step 2. Write code

- You can refer to the sample program below. Take a look at the “Important API” section for detailed documentation of common functions.

Step 3. Download code to robot

1. Make sure your robot is on and connected via Bluetooth.
2. Download code to your robot by clicking “Download to Robot”.



- If the program reports errors in your code, fix the problems and retry “Download to Robot”.
3. Run the code by clicking “start” on the pop-up window. To abort, click “stop”.
 4. Each time you modify your code, you need to click “Download to Robot” again so the robot can run the new code.

Sample Program

Below is a sample program. We first create a task called **move**. A task in ROBOTC is like a concurrent program that could run in parallel with other instructions in main task. In task **move**, we let the robot move forward for 10000 ms. If during this time the Ultrasonic sensor value is ever lower than 20, we stop and set both motor's velocity to 0. It is important to add line 21 and 22 because if you don't tell the motors to stop, the motors might still go after you break the while loop.

In the main task we first start the task **move**, then we give other instructions to robot (line 29 to 33). As a result, both routines (lines 9-22 and lines 29-33) runs concurrently. It is important to add the wait1Msec line in every while loop (line 17 and line 32) to avoid tight loops.

```
1  #pragma config(Sensor, S1,    distance,    sensorSONAR)
2  #pragma config(Sensor, S2,    light,      sensorLightActive)
3  #pragma config(Motor,  motorA, leftWheel,  tmotorNXT, PIDControl, encoder)
4  #pragma config(Motor,  motorB, rightWheel, tmotorNXT, PIDControl, encoder)
5  /**Code automatically generated by 'ROBOTC' configuration wizard  !**//
6
7  task move()
8  {
9      clearTimer(T1);
10     while (time1[T1] < 10000) {
11         motor[leftWheel] = 10;
12         motor[rightWheel] = 10;
13
14         if (SensorValue(distance) < 20)
15             break;
16
17         wait1Msec(5);
18     }
19
20     // stop all wheels
21     motor[leftWheel] = 0;
22     motor[rightWheel] = 0;
23 }
24
25 task main()
26 {
27     startTask(move);
28
29     clearTimer(T2);
30     while (time1[T2] < 5000) {
31         writeDebugStreamLine("Light Value: %d", SensorValue(light));
32         wait1Msec(5);
33     }
34 }
```

Important API

- 1. Motor & Sensor Setup**
- 2. Motor**
- 3. Sensor**
- 4. Waiting**
- 5. Timer**
- 6. Encoder**
- 7. Printing**
- 8. Joystick**

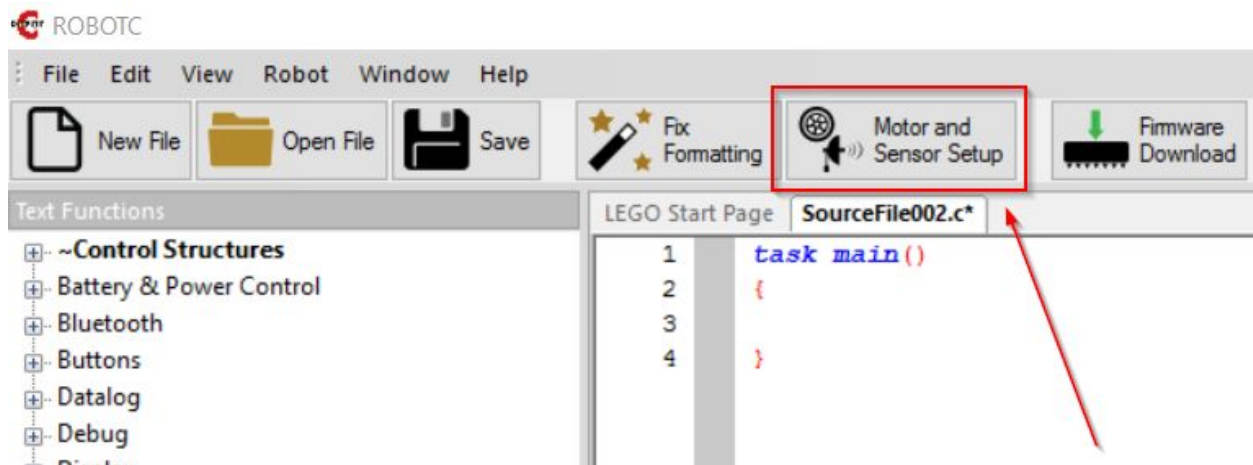
Motor & Sensor Setup

Before writing code for your robot, you need to set up the program so it knows what motors and sensors are connected to each of the ports.

Step 1. Connect the motors and sensor to your NXT brick with the provided cables.

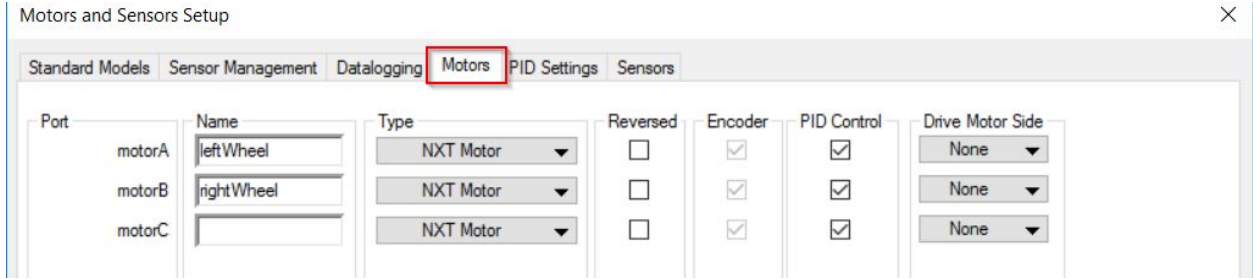
- Cables are universal. The same can be used with motors and sensors
- Motors use ports A, B, or C. Sensors use ports 1, 2, 3, or 4.

Step 2. Open your source code file in the ROBOTC program, and click on “Motor and Sensor Setup”



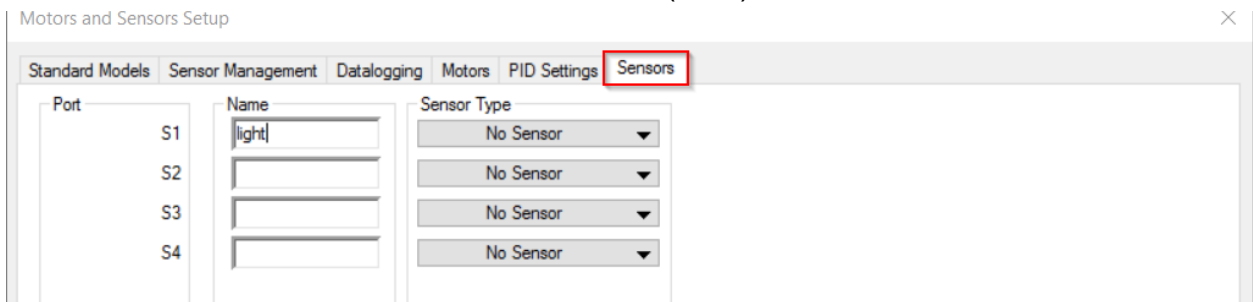
Step 3. Open the “Motors” tab to configure motors

- Fill in names only for the used ones. Unused ports can be left blank
- Explanations for the input fields:
 - Name: An alias for the motor so you can refer to in the code
 - Type: Always choose “NXT Motor”
 - Reversed: If checked, the motor will spin in the reverse direction when commanded to spin forwards
 - PID Control: If checked, the motor will always try to spin at the speed commanded by you (e.g. it will over compensate when it senses larger friction). This option should remain checked most of the time.
 - Drive Motor Side: usually “None”



Step 4. Open the “Sensors” tab to configure sensors

- Fill in names only for the used ones. Unused ports can be left blank
- Explanations for the input fields:
 - Name: An alias for the sensors so you can refer to it in the code
 - Sensor Type: The type of sensor connected to this port. The common ones used in this course are:
 - Light Sensor: “Light Sensor (NXT)” > “Light-Reflected (NXT)”
 - Ultrasonic Sensor: “Sonar (NXT)”



Step 5. When you are done configuring, click “OK”. Some configuration code will appear at the top of your source code.

```

LEGO Start Page SourceFile002.c*
1  #pragma config(Sensor, S1, light, sensorNone)
2  #pragma config(Motor, motorA, leftWheel, tmotorNXT, PIDControl, encoder)
3  #pragma config(Motor, motorB, rightWheel, tmotorNXT, PIDControl, encoder)
4  /**!!Code automatically generated by 'ROBOTC' configuration wizard !!*/
5
6  task main()
7  {
8
9  }

```

Motor

Motors are controlled by the `motor[]` command.

- Possible power values for a motor ranges from -100 to 100 (inclusive).
- Negative power spins the motor in reverse direction, while positive power spins the motor forwards.
- A power of 0 stops the motor completely.
- The argument inside the square brackets is the motor name.
- Once a motor power is set, the motor remains at that power until a new value for that motor is set.

e.g.

```
1 motor[leftWheel] = 90; // The left wheel spins forwards at power level 90
2 motor[rightWheel] = -30; // The right wheel spins backwards at power level 30
```

Sensor

Sensor values are read with the `SensorValue[]` command.

- The argument inside the square brackets is the sensor name.

e.g.

```
1 reading = SensorValue[light]; // The variable "reading" is set to
2 // the light sensor reading
```

Waiting

You can delay actions (pause for a certain amount of time before performing the next action) by calling the `wait1Msec()` function.

- The argument inside the parenthesis is the amount of time to wait for, in milliseconds (1000 milliseconds = 1 second).
- You can also call a similar function `wait10Msec(n)`, which pauses the program for $10 \times n$ milliseconds.

e.g.

```
1 wait1Msec(1000); // Pause code execution for 1000 milliseconds (1 second)
2 wait10Msec(100); // Pause code execution for 10 * 100 milliseconds (1 second)
```

Timer

You can use timers to calculate elapsed time. This can be useful for performing a task for a certain amount of time.

- The program provides 4 global timers: **T1**, **T2**, **T3**, and **T4**.
- A timer, e.g. **T1**, can be cleared to 0 with a call to `clearTimer(T1)`
- A timer's current reading can be obtained with a call to `time1[T1]`. The returned value is in milliseconds.

e.g.

The following code performs some task for 3 seconds.

```
1 clearTimer(T1);
2 while (time1[T1] < 3000) {
3     // Some task is being performed
4 }
```

Encoder

The easiest and most accurate way of finding out how much a motor has spinned is by reading from motor encoders. All motors are equipped with sensors that monitor how much it has turned. These are called encoders, and they can be used to accurately figure out how much a motor has turned. This is especially useful in dead reckoning tasks.

- Each motor has one encoder.
- Encoder values are set and read using the same `nMotorEncoder[]` command. The unit is degrees.
- The argument inside the square brackets is the motor name.

e.g.

The following code spins leftWheel for 360 degrees (1 full spin)

```
1 nMotorEncoder[leftWheel] = 0; // set the motor encoder back to 0
2 while (nMotorEncoder[leftWheel] < 360) {
3     motor[leftWheel] = 10;
4 }
5 motor[leftWheel] = 0; // stop the wheel
```

Printing

Printing can be useful for debugging and providing insights on the program's state. There are two main ways of debugging: printing to console and printing to the NXT brick's display.

- To show what the `writeDebugStreamLine` print to console, open Robot > Debugger Windows > Debug Stream. Then a window will pop up, showing the debugging information you print.
- To write to nxt brick, call `nxtDisplayTextLine(n, s, ...)` where `n` is the line number, and `s` is the string to display. A second function call with the same line number will erase the text already on that line.
- Both functions accept fprintf-style string arguments.

e.g.

```
1 writeDebugStreamLine("Time is %d", time1[T1]); // prints out the reading of
2 // timer T1 to debug stream
3
4 nxtDisplayTextLine(0, "Timer T1 value:"); // writes the value of T1 to
5 nxtDisplayTextLine(1, "%d", time1[T1]); // nxt brick in 2 separate lines
```

Joystick

The default model of joystick in ROBOTC is logitech f310.



ROBOTC supports using joystick controllers to drive your NXT. This allows the user to send commands to their robot in real time.

After you plug your usb joystick into the computer, the joystick is ready to use.

Programming with joystick

- Before we write the code to control joystick we need to include a file called "JoystickDriver.c".
- Before you can use these variables, you have to "update" the variables by getting the newest packet of data from the joysticks. Because the joystick station may only send updates every 50-100ms, you should update your joystick values as often as possible to get the most up to date joystick data. To "update" your joystick variables, use this function `getJoystickSettings(joystick)`.

e.g.

```
1 // Tells ROBOTC to include the driver file for the joystick.
2 #include "JoystickDriver.c"
3
4 task main()
5 {
6     while(true)
7     {
8         getJoystickSettings(joystick); // Update Buttons and Joysticks
9         motor[motorC] = joystick.joy1_y1;
10        motor[motorB] = joystick.joy1_y2;
11    }
12 }
```

- The program provides 4 variables to control a joystick:
 - Joy1_x1: Value of the X Axis on the Left Joystick on Controller #1. Ranges in values between -128 to +127.
 - Joy1_x2: Value of the X Axis on the Right Joystick on Controller #1. Ranges in values between -128 to +127.
 - Joy1_y1: Value of the Y Axis on the Left Joystick on Controller #1. Ranges in values between -128 to +127.
 - Joy1_y2: Value of the Y Axis on the Right Joystick on Controller #1. Ranges in values between -128 to +127.

- Below is a sample program

```
1 while(true) // infinite loop:
2 {
3     getJoystickSettings(joystick); // update buttons and joysticks
4     motor[motorB] = joystick.joy1_x1; // motorB's powerlevel is set to the
5                                     // left stick's current x-value
6 }
```


Troubleshooting

1. My computer got stuck after trying to connect to the NXT brick.

It is common for the ROBOTC program to be stuck after a failed connection attempt. If it no longer responds to you, simply force quit the program and start again. To avoid this hassle, always check to make sure the brick is on (and bluetooth on the brick is on) before trying to connect via Bluetooth.

2. My NXT brick is stuck in the program and I cannot control it anymore.

If your NXT is stuck halfway in the program (stuck at one command) and you can no longer control it via Bluetooth nor the on-brick buttons, you will have to take out the battery (force shut down). Put in the batteries again and it should boot back up as usual.

3. My NXT brick keeps making a “tic” noise and doesn’t show anything on the screen. Pressing reset and replacing the battery doesn’t work either.

- Plug the NXT into your computer and power it on (so that it is clicking)
- Open Device Manager
- Find 'Bossa Programming Port' under COM ports.
- Right click and select Properties.
- Go to 'Driver' tab.
- Select 'Update Driver'
- Select 'Browse My Computer for Driver Software'
- Search in the 'C:\Windows\INF' folder
- Select LEGO Driver
- Try downloading firmware again from RobotC as explained in lecture.

References

"NXT." - *ROBOTC API Guide*. Web. 05 May 2016.

"ROBOTC for LEGO MINDSTORMS." *ROBOTC.net*. Web. 05 May 2016.

16-311 TAs