

## Engineering Lab

# Remote Control Basics

In this lesson you will:

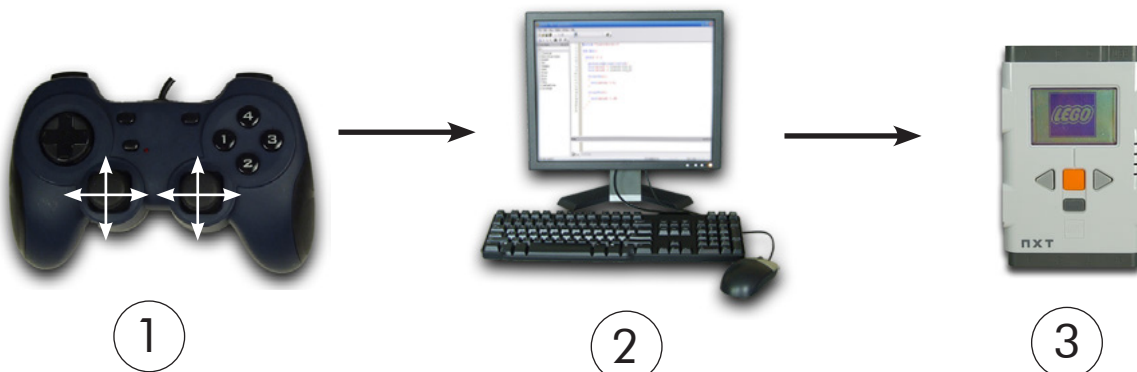
1. Learn how the remote control, computer, and NXT communicate.
2. Open a sample program named "NXT 2 Joystick Drive", drive your robot using that code, and then optimize the code.
3. Be able to:
  - Describe how NXT remote communications works.
  - Describe what an "include file" is.
  - Describe what the "getJoystickSettings" function does.
  - Describe what the variable "joystick.joy1\_y1" is and how what makes it different from "joystick.joy1\_y2".
  - Optimize your program for better joystick control.



## How are the communication signals sent?

Until now, you have written code that allows your robot to operate autonomously. There are situations where a programmer cannot plan every possible behavior using autonomous programming; for these situations it would be nice to be able to use a combination of autonomous and remote control. ROBOTC makes remote control very easy.

How does remote control work with the NXT? There are four elements that work together: the human driver, the joystick controller, the computer with ROBOTC, and the NXT. When the joysticks axes are pushed in one direction or another, a signal is created based on the direction the joystick is pushed. The signal returns a value that ranges from +127 to -127. Once the computer gets the signal, it is sent to the NXT by ROBOTC, where the function "getJoystickSettings" receives that data and places it into a variable that can be used in our program. (i.e. to power the motors) The computer sends the joystick data to the NXT over the Bluetooth or USB link.



When the joysticks axes are pushed in one direction or another, a signal is created based on the location of the joystick; the signal ranges from +127 to -127.

The signal is sent to the computer, where ROBOTC converts the signal information into information the NXT can understand.

ROBOTC then sends the data to the NXT over Bluetooth or USB so the NXT can process and use the data to control its motors.

# Engineering Lab

## Open the remote control sample program

1. Go to "File Menu - Open Sample Program"
2. Select the "Remote Control" folder
3. Open "NXT 2 Joystick Drive.c" program

```

1 #include "JoystickDriver.c"
2
3 task main()
4 {
5     while(1 == 1)
6     {
7         getJoystickSettings(joystick);
8         motor[motorC] = joystick.joy1_y1;
9         motor[motorB] = joystick.joy1_y2;
10    }
11 }
```

Include all of the contents of the file joystickDriver in this program

The "getJoystickSettings(joystick)" function updates the variables that store the joystick controller's position

These are variables that map from specific parts of the joystick controller and will be used to set the power of motorB and motorC

## What is #include?

The `#include` command is known as a directive. A directive is a special type of command that tells the ROBOTC compiler information it needs to know to make your program run correctly. One type of directive that you have already used is the "pragma" directive. The pragma directive is used to tell ROBOTC about the sensors and controllers that are connected to the NXT.

In the case of line 1, the "`#include`" directive is telling ROBOTC to "include" all of the contents of a file named "`JoystickDriver.c`" in this program. Include files make your program easier to read and allows you to use the same code in multiple programs without having to type the code over and over.

The file "`JoystickDriver.c`" contains the functions needed to process and use the joystick controller. This include file must be included in every program where you plan on using remote control. The program "`JoystickDriver.c`" is built into ROBOTC, you just need to include the file at the start of your program.

## What does the file joystickDriver.c do?

Recall how remote control communication works, there is a path from your joystick controller to your computer and from the computer to the NXT. The JoystickDriver include file translates the joystick controller information sent from the PC to the NXT into information our program can use.

## What is getJoystickSetting(joystick)?

The `getJoystickSettings(joystick)` function is a new function that is made available with the JoystickDriver include file. Note that the `getJoystickSettings` function is inside a while loop and that if it was not inside the while loop the joystick controller data would not be updated. The `getJoystickSettings` function reads the data coming from the remote controller and continuously updates variables "`joystick.joy1_y1`" and "`joystick.joy1_y2`". If the function was not inside a loop, then the variables would not be updated and the remote controller data would not be updated.

## Engineering Lab

### The joystick.joy1\_y1 and joystick.joy1\_y2 variables

```

1  #include "JoystickDriver.c"
2
3  task main()
4  {
5      while(1 == 1)
6      {
7          getJoystickSettings(joystick);
8          motor[motorC] = joystick.joy1_y1;
9          motor[motorB] = joystick.joy1_y2;
10     }
11 }

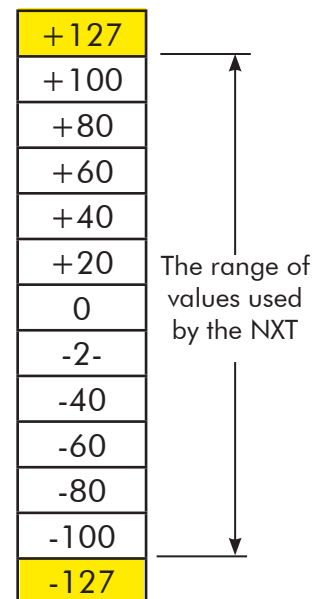
```

Once the programmer identifies which parts of the joystick controller that they want to use feedback from, the values are assigned as motor speeds.

### What does "joystick.joy1\_y1" and joystick.joy1\_y2" mean?

Look at the two motor statements on lines 8 & 9. Instead of setting power levels directly to the motors we are using the variables "joystick.joy1\_y1" and "joystick.joy1\_y2".

1. The first part of the command, "joystick", tells the compiler that we are getting information from our joystick controller.
2. The second part of the command references which joystick controller that we are using. For this program we are only using one joystick and so every command will reference "joy1".
3. And the third part, "y1", references the part of the joystick controller that we are accessing. In this case "y1" is the vertical axis on the left side of the joystick controller.



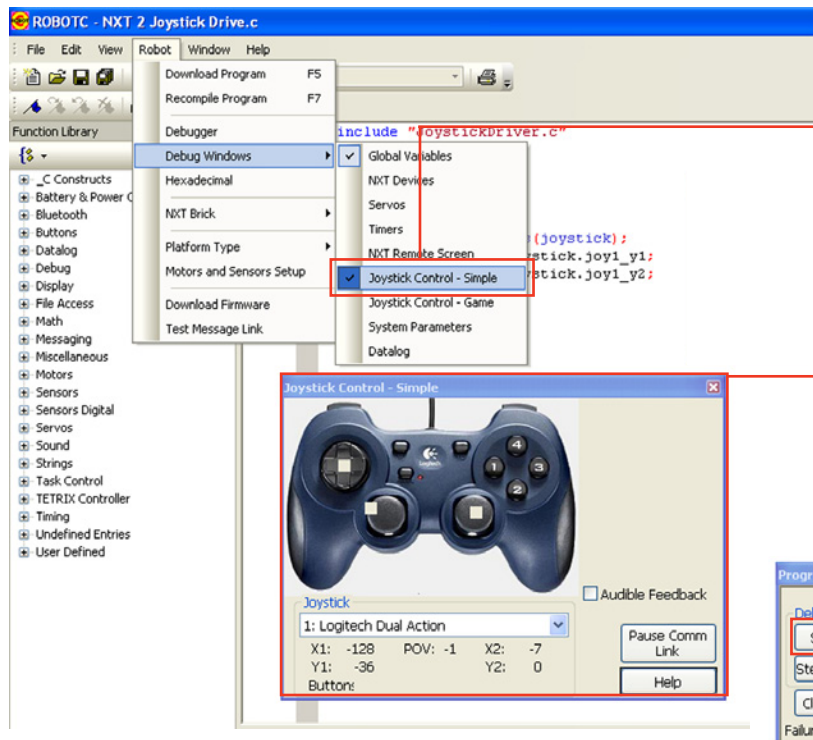
The joystick gives values of +127 through -127. The motor only uses the values from +100 through -100. The functions built into ROBOTC ignore values over the limits of the motor.

Once the programmer identifies which parts of the joystick controller that they want to use, the values are then assigned as motor speeds.

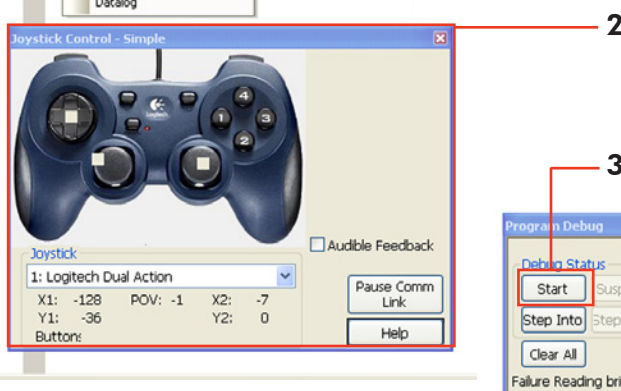
# Engineering Lab

## Testing your program

Download your program to your robot. In order to test the robot you will need a remote control and a Bluetooth adapter. See the section on "Using Bluetooth" if you are experiencing any Bluetooth communication problems.



1. In order to control your robot using the remote control you will need to open up the "Joy stick Control - Simple" menu from the Debug Windows pallet.
2. Once you select "Joystick Control - Simple" this debugger window will appear.
3. Select "Start" to start your program and begin using the remote control.



## Optimizing the Remote Control

Start your program and drive around. You may notice when testing that your robot will drift when you release the joysticks. This occurs because the joystick axes do not always return to center when you let go of the joysticks.

Look at the Joystick Controller debugger window below. This picture of the screen was captured when both joystick axes were released. Note that the Y1 axis still reads -1.

This will occur because the joystick does not always return to the mechanical center of the joystick controller.

Since the joystick axes controller values are continually passed on to the program this will cause your robots to drift even though you are not pressing your joystick axes.

With the joystick axes released, note that the signal from "Y1" is still -1.



# Engineering Lab

## Optimizing the Remote Control (cont.)

### Programming a "Dead Zone" onto your remote control

To avoid the drifting motors problem, we can create a "dead zone" in our program. A "Dead Zone" is the range of motion around the center position of an axis which returned a value of zero.

To create a dead zone we will use programming logic so that the motor power is set to zero if the positional value of the joystick axis is less than 5 or greater than -5. We can do that by setting up a condition so that if the value of the joystick axis is less than 5 and it is greater than a -5 then set the motor to zero.

That code to check that condition looks like this: `if(joystick.joy1_y1 < 5 && joystick.joy1_y1 > -5)` then set the value of the motor to zero. See the code and explanation below to see it work.

<pre> 1  #include "JoystickDriver.c" 2 3  task main() 4  { 5      while(1 == 1) 6      { 7          getJoystickSettings(joystick); 8 9          if(joystick.joy1_y1&lt;5 &amp;&amp; joystick.joy1_y1&gt; -5) 10         { 11             motor[motorC] = 0; 12         } 13         else 14         { 15             motor[motorC] = joystick.joy1_y1; 16         } 17 18         if(joystick.joy1_y2&lt;5 &amp;&amp; joystick.joy1_y2&gt; -5) 19         { 20             motor[motorB] = 0; 21         } 22         else 23         { 24             motor[motorB] = joystick.joy1_y2; 25         } 26     } 27 }</pre>	<ol style="list-style-type: none"> <li>1. Include the file "Joystick Driver.c" in this program</li> <li>2. Loop this block of code forever (Infinite Loop)</li> <li>3. Update the variables that hold the position of the joystick axes</li> <li>4. If the value of joystick axis "Y1" is less than 5 or greater than -5, set motorC's power level to 0.</li> <li>5. Otherwise set motorC's power level to the value of "Y1"</li> <li>6. If the value of joystick axis "Y2" is less than 5 or greater than -5, set motorB's power level to 0.</li> <li>7. Otherwise set motorB's power level to the value of joystick "Y2"</li> </ol>
---	---

# Engineering Lab

## Optimizing Remote Control (cont.)

### Variable Math

In a previous lesson we learned that valid motor power levels range from -100 through +100. Motors that receive values beyond their limits, such as -127 and +127, will default to -100 or +100, respectively.

When using the joystick controller, robots travelling at full speed may have difficulty turning precisely and may be difficult to control. To help control our robot, we may need to slow it down. To do this we will need to adjust the values that we get from the joystick controller. We can do this using something called "Variable Math".

Variable math can be used to modify the value of a variable before we assign it to something. If we take a look at a line of code `motor[motorB] = joystick.joy1_y1;` we see that this line of code reads the value from the variable `joystick.joy1_y1` and assigns it to `motorB`. Since the variable `joystick.joy1_y1` returns a value between -127 and +127 we could manipulate the variable mathematically to change the value assigned to `motorB`. For instance, if we divided the variable by two we get:

$$\text{joystick.joy1\_y1} / 2$$

If the value of `joystick.joy1_y1` was equal to 127, then the value passed to the motors would be 63. If we did that to both motors, then we would be able to slow down our robot and increase our control significantly. Try dividing by 3 and see what happens. Experiment to find what works best for you.

```

1  #include "JoystickDriver.c"
2
3  task main()
4  {
5      while(1 == 1)
6      {
7          getJoystickSettings(joystick);
8          motor[motorC] = joystick.joy1_y1 / 2;
9          motor[motorB] = joystick.joy1_y2 / 2;
10     }
11 }
```

In this example we divided the variable `joystick.joy1_y1` by 2. Experiment with other options to find the best control option for you.



## Engineering Lab

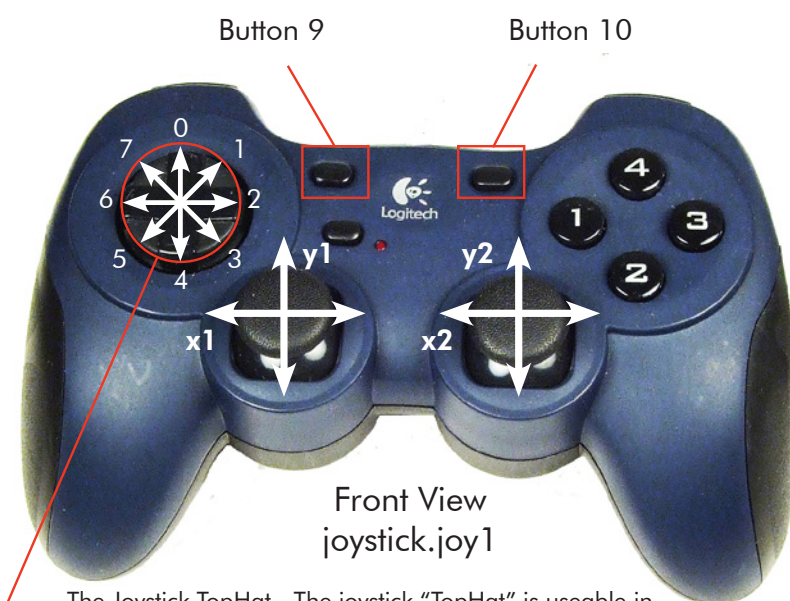
### Remote Control Buttons

In this lesson you will:

1. Learn how to program the buttons on your remote controller.
2. Review the "JoystickDriver" include file.
3. Review the function "getJoystickSettings(joystick)".
4. Identify the names and locations of all buttons on the joystick.

#### Remote Control Button Overview

The joystick remote controller is a very powerful tool that a programmer can use to achieve finite control of their robot. Each button can be programmed to control a specific behavior, for example - goStraight, rightTurn, leftTurn, openGripper, closeGripper - allowing limitless options.



The Joystick TopHat - The joystick "TopHat" is useable in your program and accessed with the reserve word "joystick.joy1\_TopHat". The TopHat returns values from -1 to 7. The -1 value signifies that nothing is pressed.



**Joysticks:** The joystick controller is named joystick.joy1. There are two joystick axes on the joystick.joy1 controller. They are the round knobs that are labeled x1/y1 and x2/y2 on the picture on the left. To access the "y-axis" of joy\_1 the command would be "joystick.joy1\_y1".

The joystick axis names are:

```
joystick.joy1_y1
joystick.joy1_x1
joystick.joy1_y2
joystick.joy1_x2
```

**Note:** ROBOTC has the capability of working with two joystick controllers at a time. There is also a joystick.joy2 remote controller that can be programmed with ROBOTC.

**Buttons:** There are 10 programmable buttons on the joystick controller. They are found in three locations.

Buttons 1-4 are located on the front right side of the joystick controller.

Buttons 5-8 are located on the top of the joystick controller.

Buttons 9 and 10 are above the joysticks.

Joystick buttons values are accessed: joy1Btn(1) through joy1Btn(10)

Note: There is a set of joy2Btn(#) that map to the second joystick controller.

# Engineering Lab

## Remote Control Lesson Background

This lesson assumes you understand how Bluetooth communications works, that you have programmed your robot to move around using joystick control, that you have optimized your code by creating a dead zone, and that you have slowed down your robot's speed through the use of variable math. If you have not completed those lessons, complete them before beginning this one.

## Things to Remember with Every Remote Control Program

1. Every Remote Control Program will need to include the "JoystickDriver.c" include file.
2. The "getJoystickSettings(joystick)" function is built into the "JoystickDriver" include file. The getJoystickSettings function updates the joystick variables so that the program has the current conditions of the joystick.

## Review

<pre> 1  #include "JoystickDriver.c" 2 3  task main() 4  { 5      while(1 == 1) 6      { 7          getJoystickSettings(joystick); 8          motor[motorC] = joystick.joy1_y1; 9          motor[motorB] = joystick.joy1_y2; 10     } 11 }</pre>	<p>Include all of the contents of the file joystickDriver in this program.</p> <p>The "getJoystickSettings(joystick)" function updates the variables that store the joystick controller's position.</p> <p>These are variables that map from specific parts of the joystick controller and will be used to set the power of motorB and motorC.</p>
--	--

## Programming Remote Control Buttons

Joystick buttons behave the same way as the NXT touch sensors do. When the NXT touch sensor is pressed, it returns a value of "1". When it is not pressed, it returns a value of "0". Similarly, joystick buttons receive a value of "1" when pressed and a value of "0" when not pressed.

To introduce programming remote control buttons, we will be using the gripper arm connected to the NXT robot. If you have not built the gripper arm attachment, build it now. You can find directions for the attachment under the using buttons video.



Once we have the robot gripper arm attached, we will want to program two new behaviors. The new behaviors will be "Open the Gripper Arm" and "Close the Gripper Arm". To activate the buttons, we will use two buttons on our joystick controller. There are 10 possible buttons available on the controller:

Buttons 1-4 are on the front right side of the remote control

Buttons 5-8 are on the top of the remote control

And buttons 9 and 10 are located in the middle, on the top of the remote control.

For our gripper program, we are going to program buttons 5 and 6.



# Engineering Lab

## The Remote Control Program to Control Buttons 5 & 6

```

1 #include "JoystickDriver.c"
2
3 task main()
4 {
5     while(1 == 1)
6     {
7         getJoystickSettings(joystick);
8
9         if(joy1Btn(6) == 1)
10        {
11            motor(motorA) = 25;
12        }
13        else if(joy1Btn(5) == 1)
14        {
15            motor(motorA) = -25;
16        }
17        else
18        {
19            motor(motorA) = 0;
20        }
21    }
22 }

```

Joystick controller buttons are programmed using the reserved word:

`joy1Btn(n)`

"joy1" references remote control number one.

"Btn" is short for button

"n" can be replaced by any number from 1 to 10. This references which button you want to read the value from.

Line 9 refers to joystick controller button 6, and line 13 refers to joystick controller button number 5.

What do lines 19 through 22 do? Experiment by commenting that code out and see what happens.



Copy the program above into ROBOTC and download it to your robot. Make sure that the motor on the robot gripper is connected to motorA. When you press buttons 5 and 6, your robot gripper will open and close.



Note: If you download your program, hit the start button on ROBOTC, and find your remote control is not working, it could be because you don't have the Joystick Controller Debug screen open.

The joystick remote controller gives you unlimited control and programming possibilities. You can also call functions using buttons. See the next page for an example using functions for the joystick TopHat.

## Engineering Lab

### Programming the Joystick TopHat

The Joystick TopHat - The joystick TopHat is programmable just like other buttons. The TopHat is accessed with the reserve word:

`joystick.joy1_TopHat`

The TopHat returns values from -1 to 7. The -1 value signifies that nothing is pressed.

See the program example below to see how the Joystick TopHat is programmed.



Front View  
`joystick.joy1`

### Program Example

```

1 void MoveForward()
2 void TurnRight()
3 void TurnLeft()
4 void Stop()
5
6 #include "JoystickDriver.c"
7
8 task main()
9 {
10     while(1 == 1)
11     {
12         getJoystickSettings(joystick);
13         if(joystick.joy1_TopHat == 0)
14         {
15             MoveForward();
16         }
17         if(joystick.joy1_TopHat == 2)
18         {
19             TurnRight();
20         }
21         if(joystick.joy1_TopHat == 6)
22         {
23             TurnLeft();
24         }
25         if(joystick.joy1_TopHat == -1)
26         {
27             Stop();
28         }
29     }
30 }

```

The functions MoveForward, TurnRight, TurnLeft, and Stop are not complete in this example. This is only to show what is possible using buttons, joystick axes, and the TopHat controls.

If the top switch of the TopHat is pressed, then call the function MoveForward.

If the right switch on the TopHat is pressed then call the function TurnRight.

If the left switch on the TopHat is pressed then call the function TurnLeft.

If nothing on the TopHat is pressed then call the function Stop.