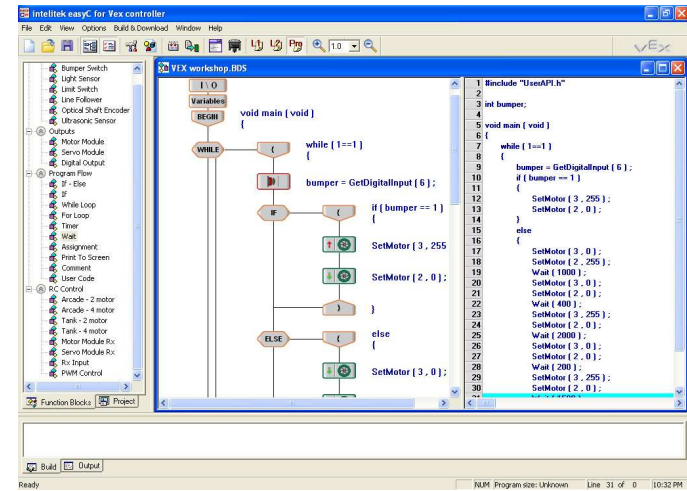
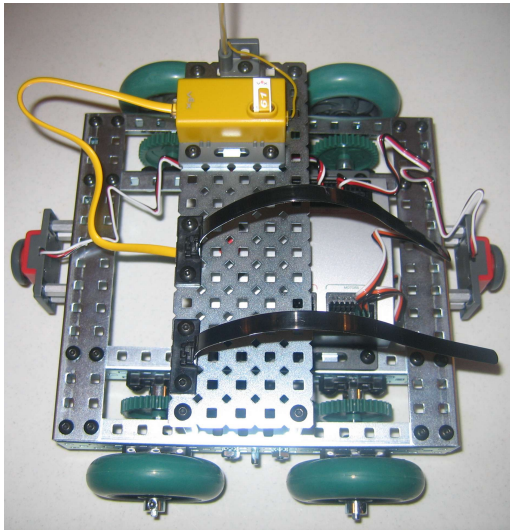
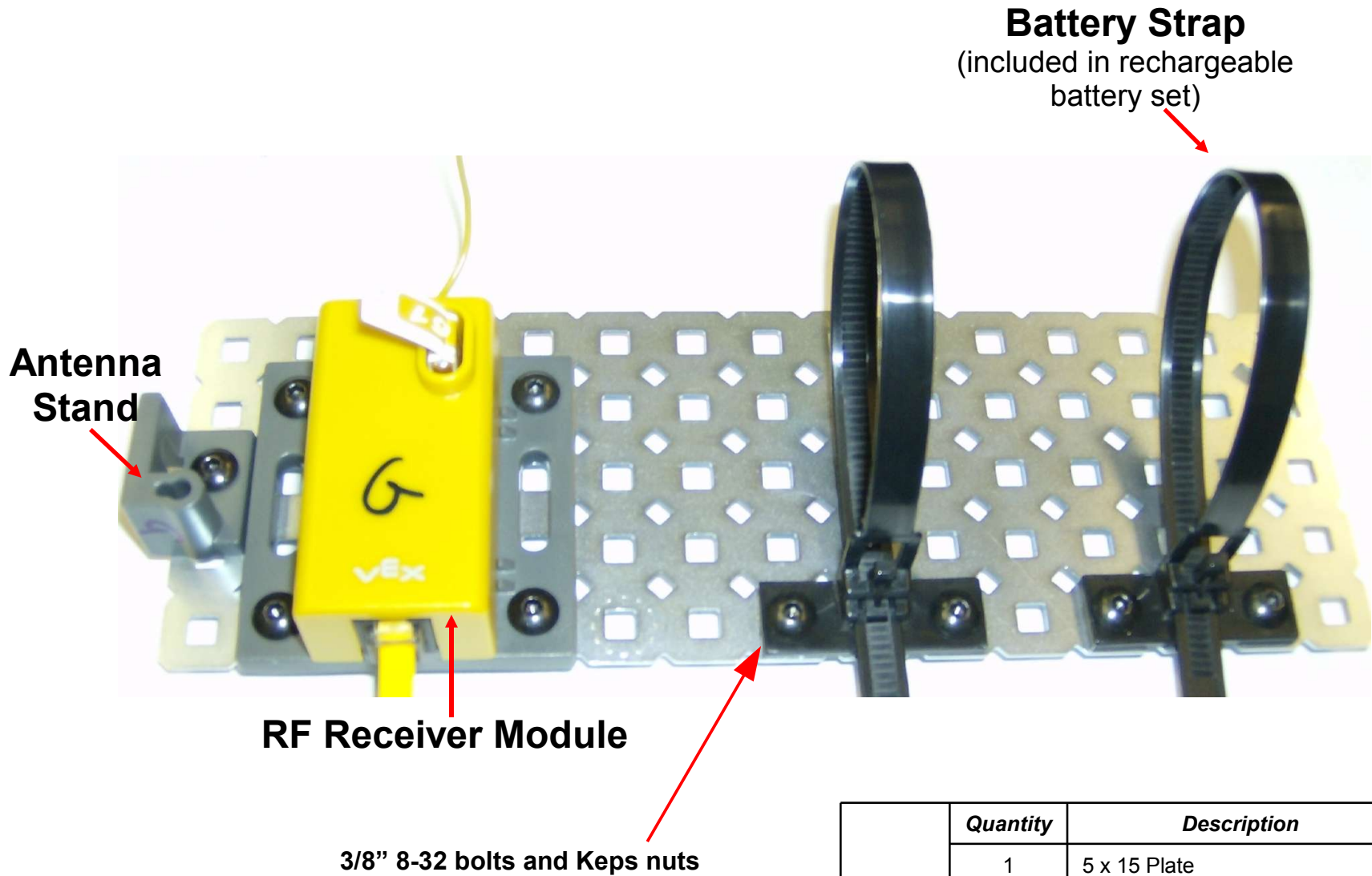


# VEX Robotics Workshop



Supported By:





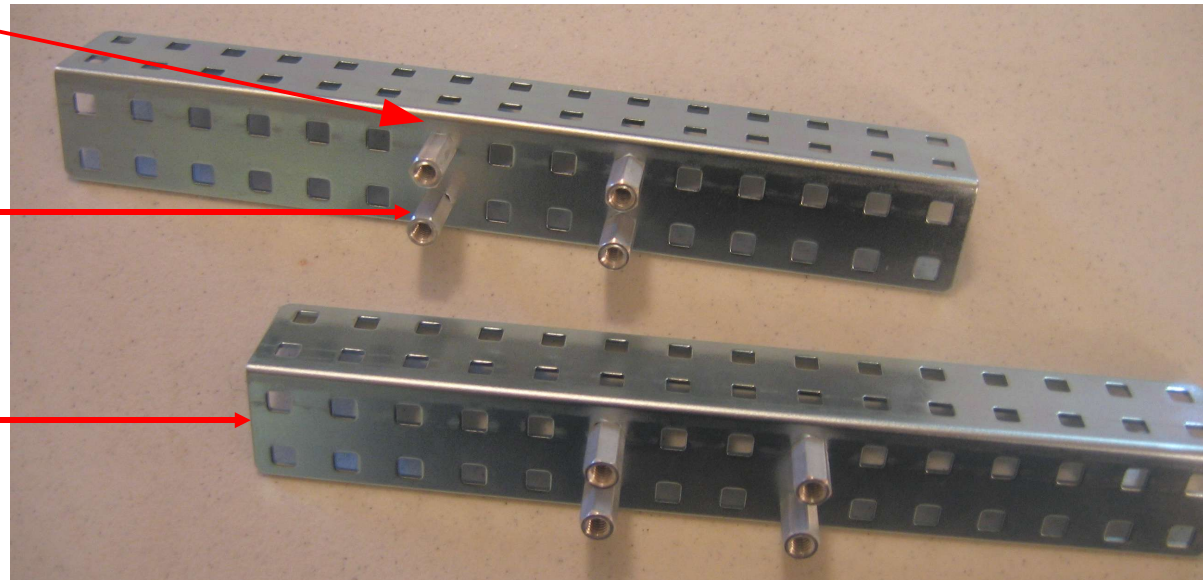
	<b>Quantity</b>	<b>Description</b>
<b>STEP 1</b>	1	5 x 15 Plate
	2	Black Battery Straps
	1	Yellow RF Receiver Module
	1	Grey Antenna Stand
	9	3/8" 8-32 Bolts
	9	Keps Nuts

# Building 2 Bumpers

(4) 3/8" 8-32 bolts per bumper on back side

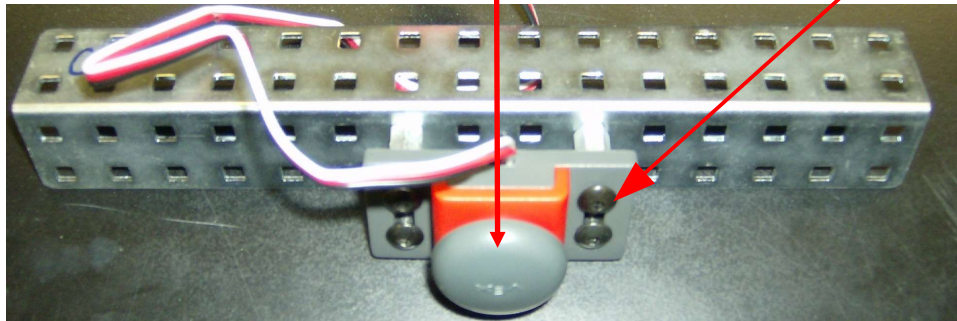
1/2" Threaded Riser

"L" Brackets

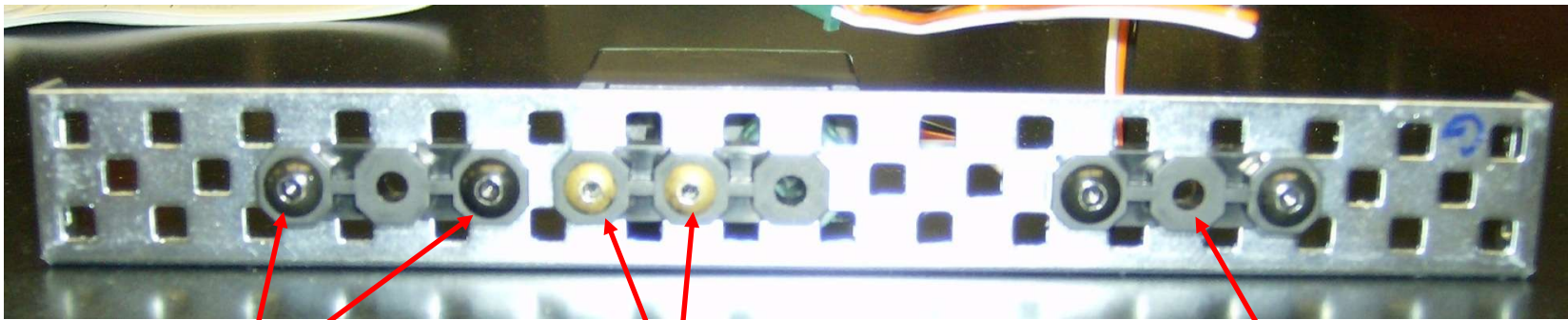


Bumper

(4) 1/4" 8-32 bolts



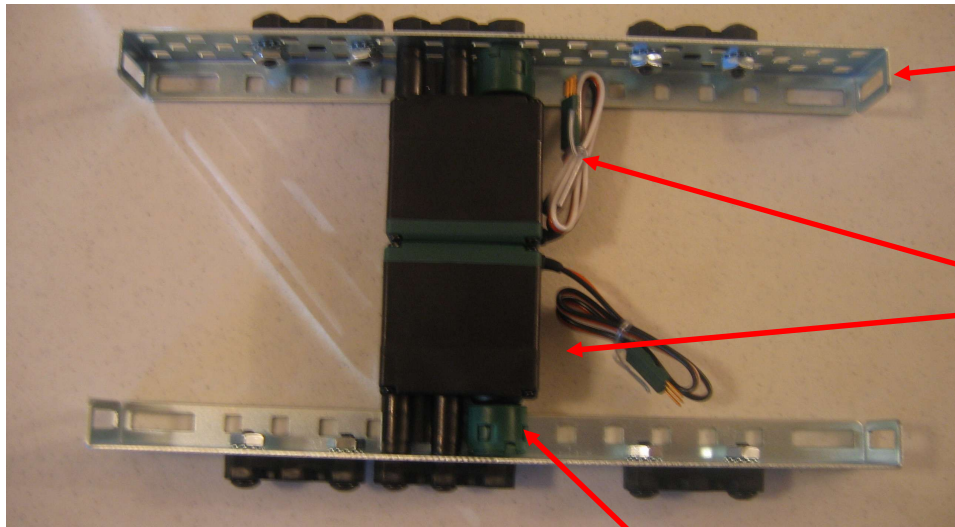
	Quantity	Description
STEP 2	8	3/8" 8-32 Bolts
	8	1/4" 8-32 Bolts
	2	Bumpers
	2	15 x 2 "L" Brackets
	8	1/2" Threaded Riser



**(8) 1/2" 8-32 Bolts  
for Bearings**

**(4) 1/2" 6-32 Bolts  
for Motors  
*(smaller diameter than most)***

**Bearings**



**Chassis Rail**

**Motors NOT Servos**

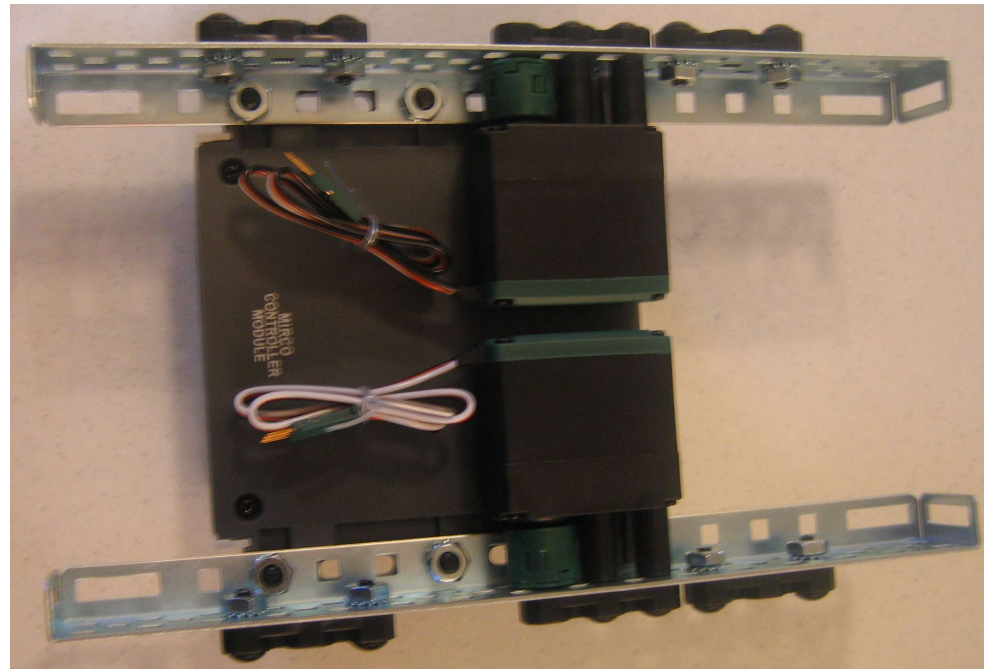
**Make sure clutch is  
present**

	<b>Quantity</b>	<b>Description</b>
<b>STEP 3</b>	2	Brackets
	2	Motors
	6	Bearings
	4	1/2" 6-32 Bolts
	8	1/2" 8-32 Bolts
	8	Keeps Nuts

Top

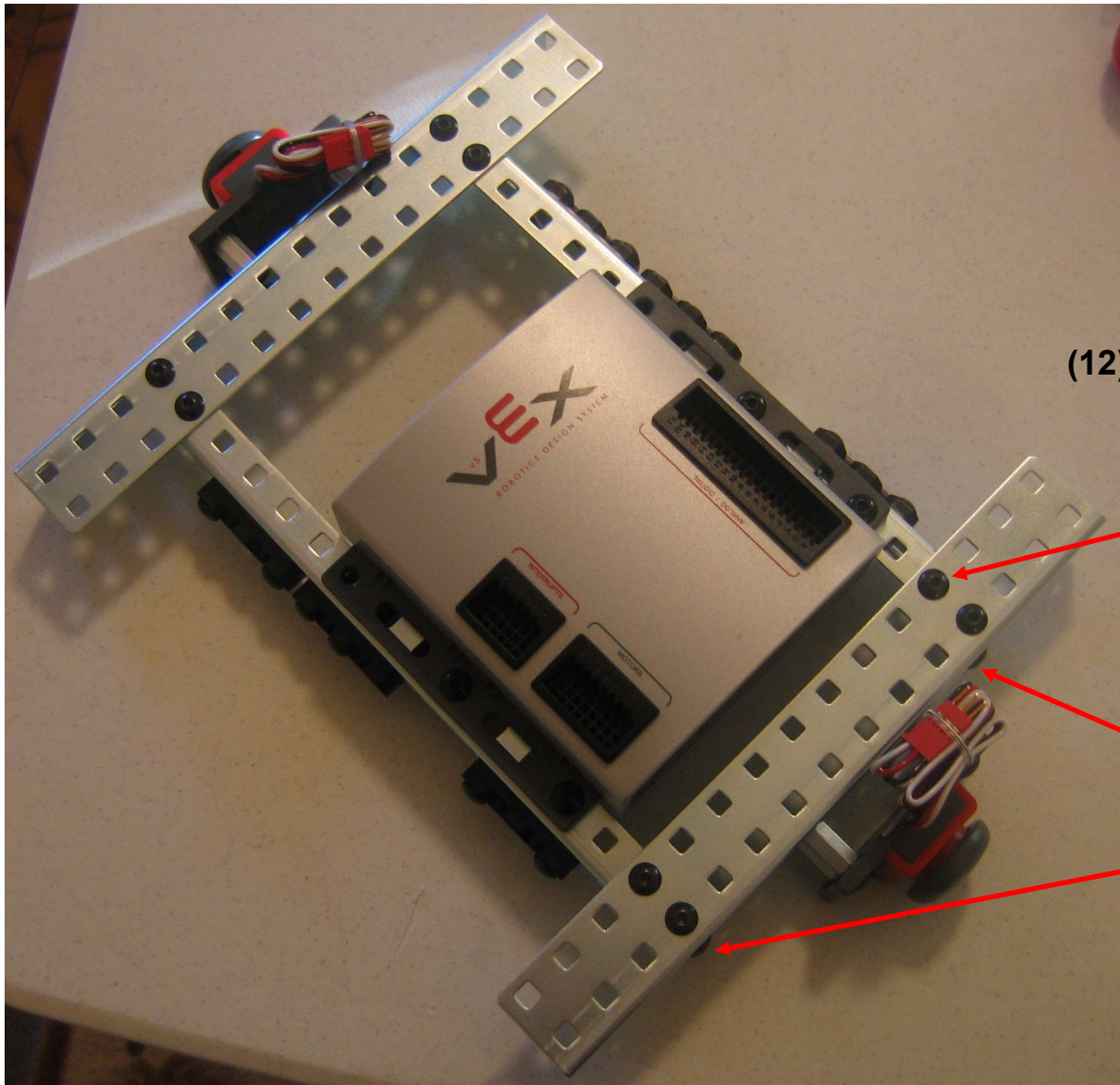


Bottom



(4) 1/2" 8-32 bolts and nuts

	<b>Quantity</b>	<b>Description</b>
<b>STEP 4</b>	1	Microcontroller Module
	4	1/2" 8-32 Bolts
	4	Keys Nuts



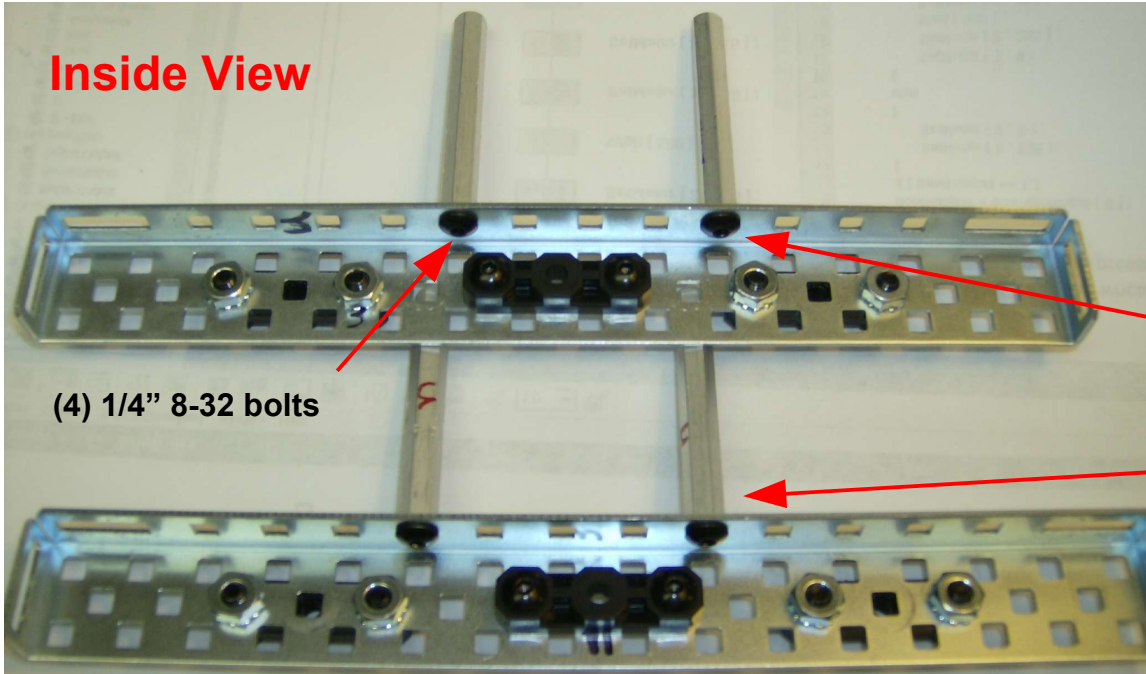
**(12) 1/4" 8-32 bolts and Keps nuts**

**8 Bolts on top  
(2 at each intersection)**

**4 Bolts on the ends  
(1 at each end)**

	<b>Quantity</b>	<b>Description</b>
<b>STEP 5</b>	12	1/4" 8-32 Bolts
	12	Keps Nuts

## Inside View



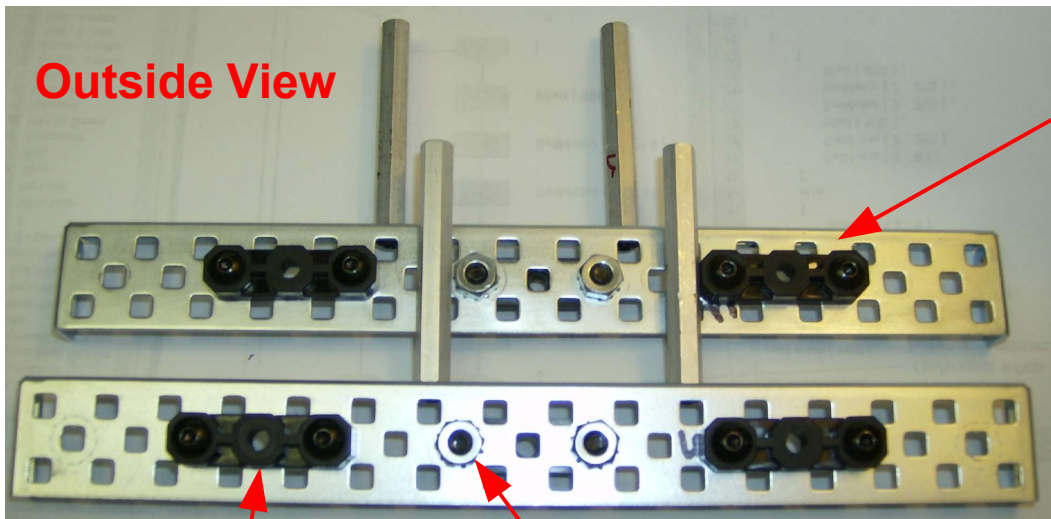
(4) 1/4" 8-32 bolts

Notice that the posts are **NOT** put in the same place for both sides. They are mirror images of each other.

This post is put in on the 3<sup>rd</sup> hole from the right

This post is put in on the 4<sup>th</sup> hole from the right

## Outside View

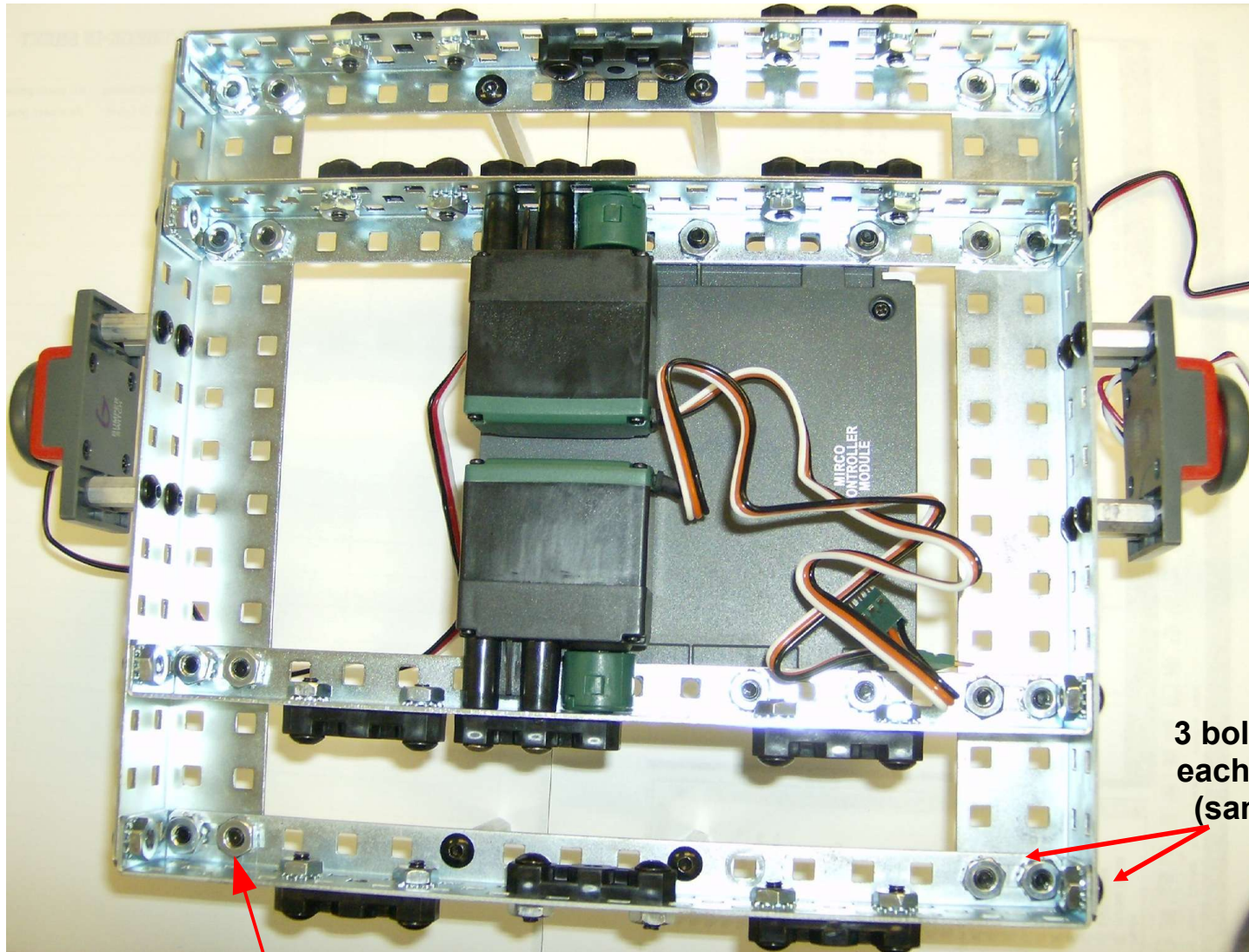


(6) Bearings

(12) 1/2" 8-32 bolts and nuts

Notice that the bearings are put exactly the same for both sides.

	Quantity	Description
STEP 6	2	Chassis Rails
	4	2" Standoffs
	4	1/4" 8-32 Bolts
	6	Bearings
	12	1/2" 8-32 Bolts
	12	Keps Nuts



3 bolts and nuts at each end - 12 total (same as Step 5)

(12) 1/4" 8-32 bolts and nuts

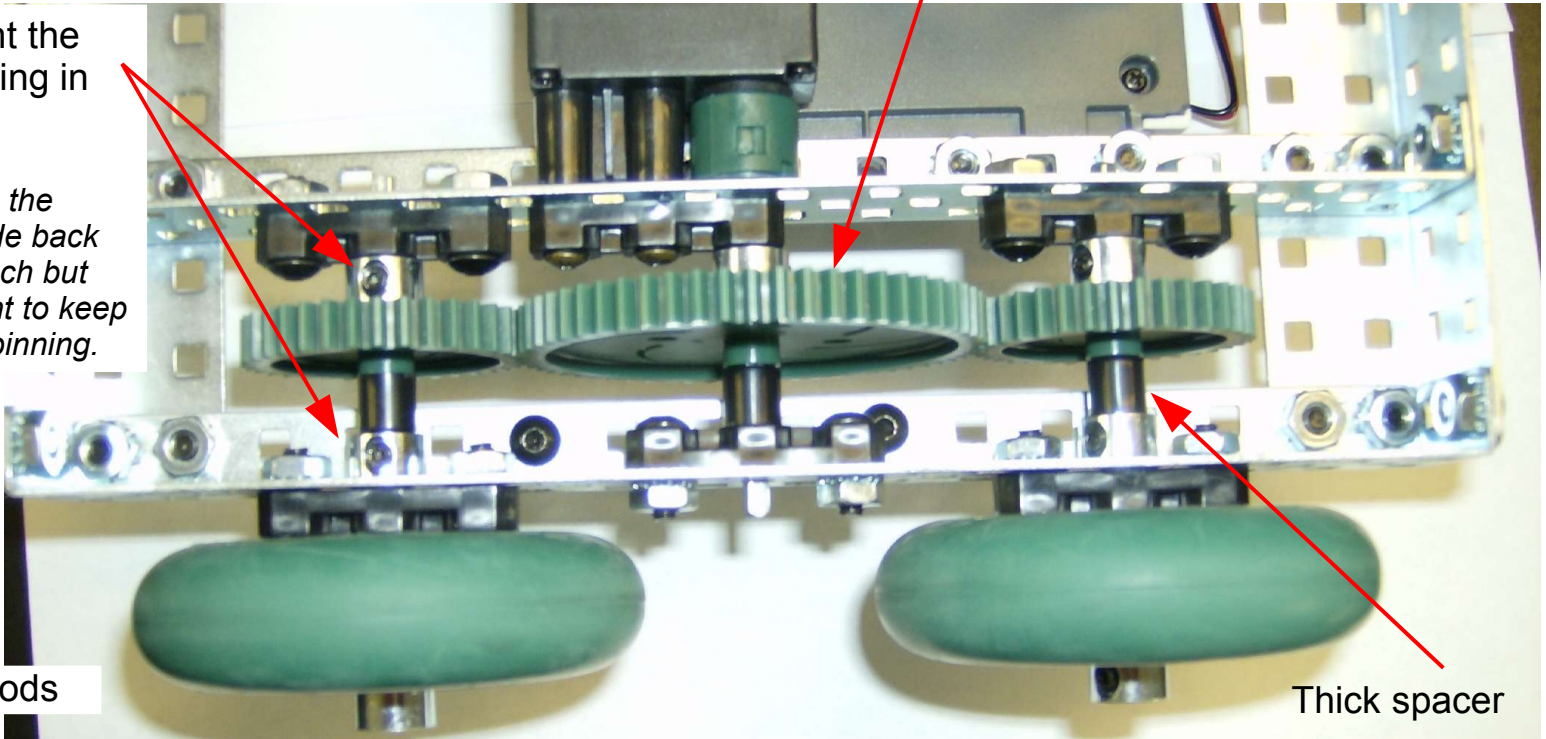
	<i>Quantity</i>	<i>Description</i>
<b>STEP 7</b>	12	1/4" 8-32 Bolts
	12	Keys Nuts



Collars prevent the shaft from sliding in or out.

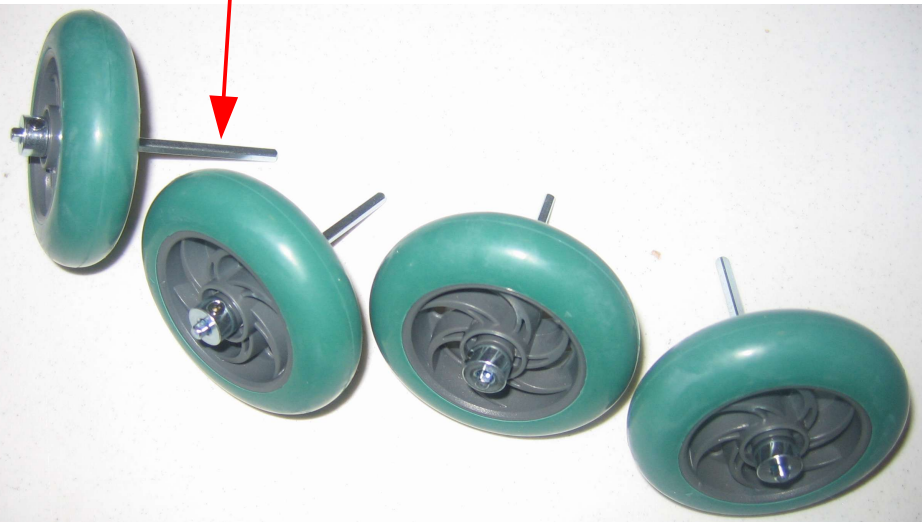
*Adjust collars so the shaft doesn't slide back and forth too much but also isn't too tight to keep the shaft from spinning.*

Large Gear - 60pt



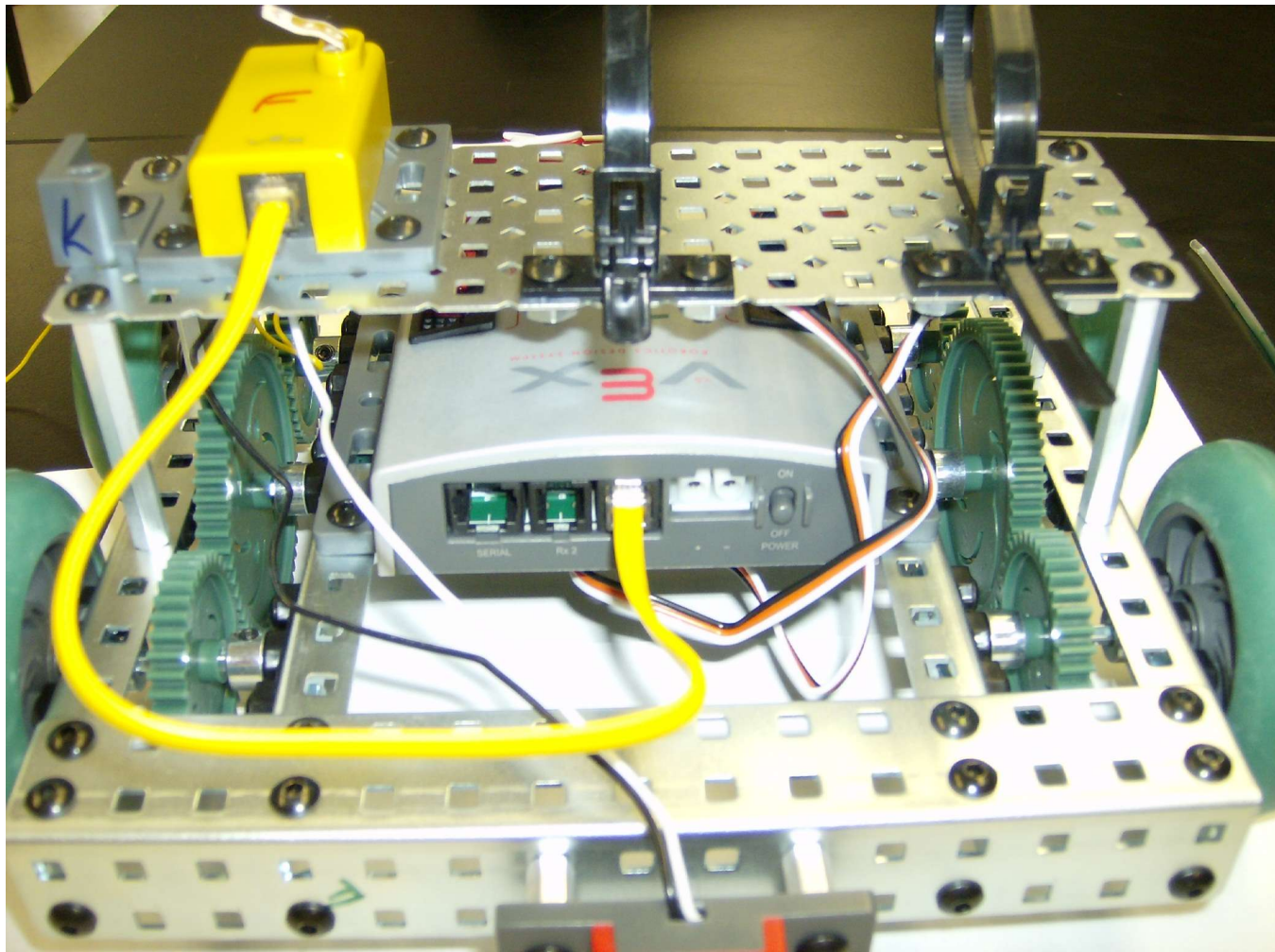
3" Square Rods

Thick spacer



	<b>Quantity</b>	<b>Description</b>
<b>STEP 8</b>	4	3" Square Rods
	2	2" Square Rods
	14	Set-screw Collars
	4	Wheels
	4	Medium Gears – 36 pt
	2	Large Gears – 60 pt
	6	Thick spacers

## Install Top Plate – 1 screw at each corner



STEP 9	Quantity	Description
	4	1/4" 8-32 Bolts

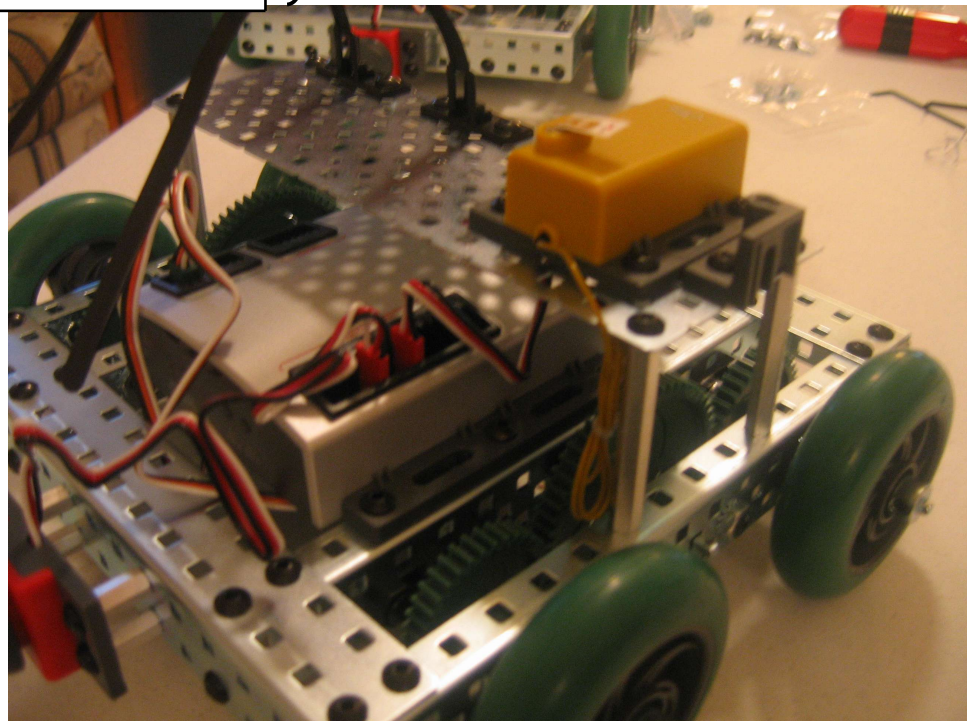
**Plug wires into the microcontroller – be CAREFUL, they only go in one way and be careful not to bend the little pins.**

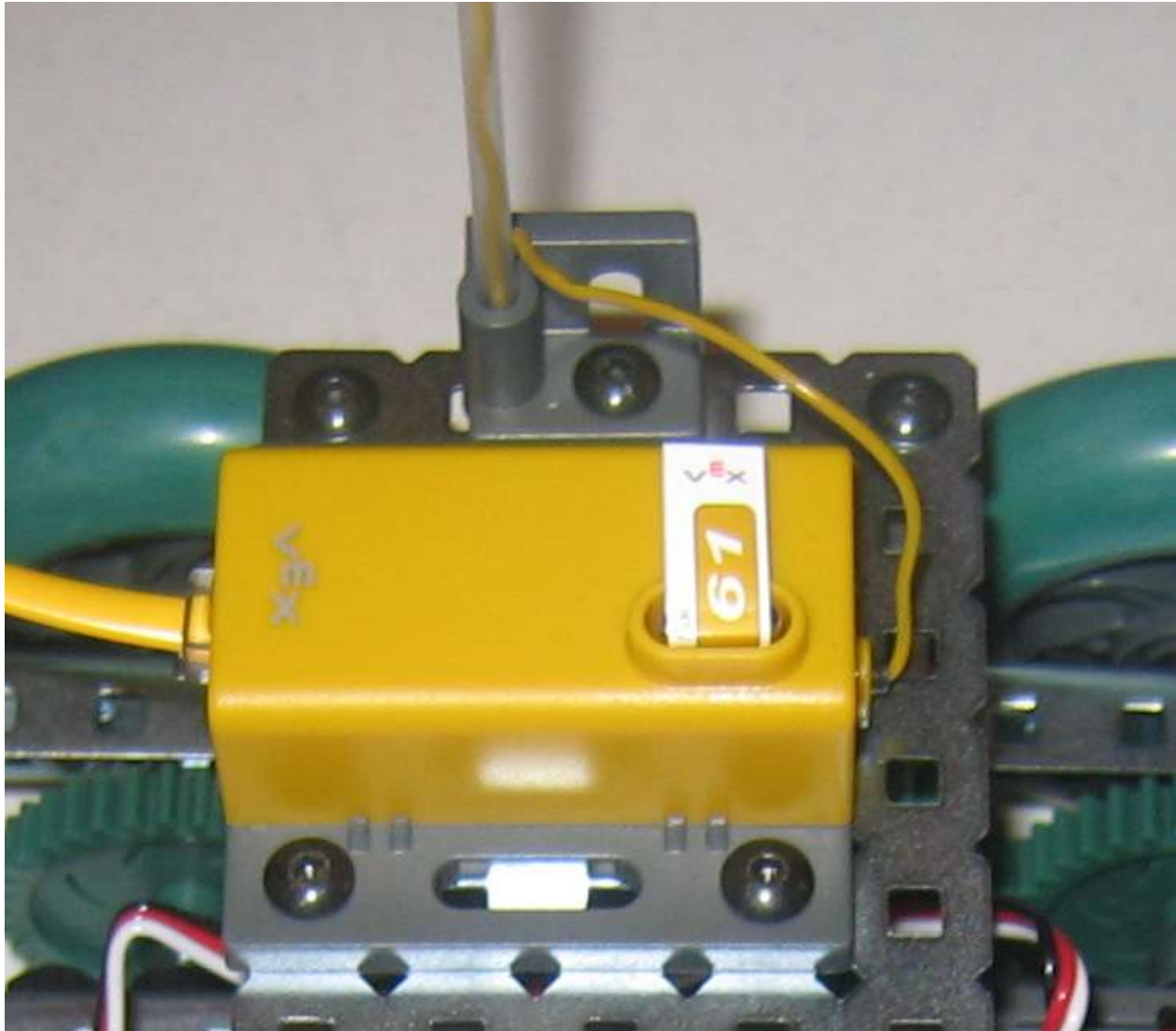
<b><i>Cable from...</i></b>	<b><i>Plugs into...</i></b>
Right Drive Motor	Motor Port 3
Left Drive Motor	Motor Port 2
Front Bumper	Analog/Digital Port 6
Back Bumper	Analog/Digital Port 10

*Port is another word for socket.*

Go into motor ports

Go into analog/digital ports





Slide the antenna wire into antenna tube and mount as shown – the antenna wire slides into the little groove near the hole for the mast.

# RC Control

The screenshot displays the intelitek easyC for Vex controller software interface. The main workspace shows a program flow diagram for a file named "rc control.BDS". The diagram starts with a "BEGIN" block, followed by a "WHILE" loop. Inside the loop, there is a block containing the number "2" with a small robot icon, representing a call to the Tank2 function. The loop ends with an "END" block. The C code on the right side of the workspace is as follows:

```
1 #include "UserAPI.h"
2
3 void main ( void )
4 {
5     while ( 1==1 )
6     {
7         Tank2 ( 0 , 3 , 2 , 3 , 2 , 0 , 0 );
8     }
9 }
```

Annotations with arrows point from the code to the diagram and vice versa:

- "Rx #: Auto" points to the first parameter "0" in the Tank2 function call.
- "Left Channel" points to the second parameter "3".
- "Right Channel" points to the third parameter "2".
- "Left Motor" points to the fourth parameter "3".
- "Right Motor" points to the fifth parameter "2".
- "Invert Left: No" points to the sixth parameter "0".
- "Invert Right: No" points to the seventh parameter "0".

# Drive Forward and Stop

intelitek easyC for Vex controller

File Edit View Options Build & Download Window Help

drive forward and stop.BDS

The screenshot displays the 'drive forward and stop.BDS' project in the intelitek easyC for Vex controller. The interface is divided into three main sections: a left-hand component palette, a central block diagram, and a right-hand code editor.

**Component Palette (Left):** Lists various hardware and control components such as Bumper Switch, Light Sensor, Limit Switch, Line Follower, Optical Shaft Encoder, Ultrasonic Sensor, Outputs (Motor Module, Servo Module, Digital Output), Program Flow (If-Else, If, While Loop, For Loop, Timer, Wait, Assignment, Print To Screen, Comment, User Code), RC Control (Arcade - 2 motor, Arcade - 4 motor, Tank - 2 motor, Tank - 4 motor, Motor Module Rx, Servo Module Rx), and I/O (Variables).

**Block Diagram (Center):** A vertical sequence of blocks starting with 'I/O' and 'Variables', followed by a 'BEGIN' block. The main logic consists of:

- A 'SetMotor' block with parameters (3, 255), annotated with 'Motor 3 Full Forward'.
- A 'SetMotor' block with parameters (2, 0), annotated with 'Motor 2 Full Forward'.
- A 'Wait' block with a value of 3000, annotated with 'Run for 3 seconds'.
- A 'SetMotor' block with parameters (3, 127), annotated with 'Motor 3 Stop'.
- A 'SetMotor' block with parameters (2, 127), annotated with 'Motor 2 Stop'.

The sequence ends with an 'END' block.

**Code Editor (Right):** Shows the C code corresponding to the block diagram:

```
1 #include "UserAPI.h"
2
3 void main ( void )
4 {
5     SetMotor ( 3 , 255 ) ;
6     SetMotor ( 2 , 0 ) ;
7     Wait ( 3000 ) ;
8     SetMotor ( 3 , 127 ) ;
9     SetMotor ( 2 , 127 ) ;
10 }
```

# Drive Forward, Turn and Stop

The screenshot displays the intelitek easyC for Vex controller software interface. The main workspace shows a block diagram for a program titled "drive forward turn and stop.BDS". The diagram consists of the following blocks in sequence:

- I/O
- Variables
- BEGIN
- void main { void } {
- SetMotor ( 3 , 255 );
- SetMotor ( 2 , 0 );
- Wait ( 3000 );
- SetMotor ( 3 , 255 );
- SetMotor ( 2 , 255 );
- Wait ( 400 ); Run for 0.4 seconds
- SetMotor ( 3 , 127 );
- SetMotor ( 2 , 127 );
- END
- }

Annotations with arrows point to specific blocks:

- "Motor 3 Full Forward" points to the SetMotor ( 3 , 255 ); block.
- "Motor 2 Full Reverse" points to the SetMotor ( 2 , 255 ); block.

The right-hand pane shows the corresponding C code:

```
1 #include "UserAPI.h"
2
3 void main ( void )
4 {
5     SetMotor ( 3 , 255 );
6     SetMotor ( 2 , 0 );
7     Wait ( 3000 );
8     SetMotor ( 3 , 255 );
9     SetMotor ( 2 , 255 );
10    Wait ( 400 );
11    SetMotor ( 3 , 127 );
12    SetMotor ( 2 , 127 );
13 }
```

# Drive Forward, Turn, Drive Forward

intelitek easyC for Vex controller

File Edit View Options Build & Download Window Help

Function Blocks Project

drive forward turn drive forward.BDS

SetMotor [ 2 , 0 ] ;

Wait [ 3000 ] ;

SetMotor [ 3 , 255 ] ;

SetMotor [ 2 , 255 ] ;

Wait [ 800 ] ;

SetMotor [ 3 , 255 ] ;

SetMotor [ 2 , 0 ] ;

Wait [ 3000 ] ;

SetMotor [ 3 , 127 ] ;

SetMotor [ 2 , 127 ] ;

END }

```
1 #include "UserAPI.h"
2
3 void main ( void )
4 {
5     SetMotor [ 3 , 255 ] ;
6     SetMotor [ 2 , 0 ] ;
7     Wait [ 3000 ] ;
8     SetMotor [ 3 , 255 ] ;
9     SetMotor [ 2 , 255 ] ;
10    Wait [ 800 ] ;
11    SetMotor [ 3 , 255 ] ;
12    SetMotor [ 2 , 0 ] ;
13    Wait [ 3000 ] ;
14    SetMotor [ 3 , 127 ] ;
15    SetMotor [ 2 , 127 ] ;
16 }
```



# Bumper Test

intelitek easyC for Vex controller

File Edit View Options Build & Download Window Help

BUMPERTEST.BDS

I/O  
Variables  
BEGIN

// Connect Bumper Switch to DI 6

// Switch Open = 1, Switch Closed = 0

WHILE { bumper = GetDigitalInput ( 6 ); PrintToScreen ( "Bumper Switch = %d\n" , (int) bumper ); }

END }

```
1 #include "UserAPI.h"
2
3 int loop = 1;
4 int bumper;
5
6 void main ( void )
7 {
8     //Connect Bumper Switch to DI 6
9     //Switch Open = 1, Switch Closed = 0
10    while ( loop == 1 )
11    {
12        bumper = GetDigitalInput ( 6 );
13        PrintToScreen ( "Bumper Switch = %d\n" , (int) bumper );
14    }
15 }
```

Left sidebar categories:

- Bumper Switch
- Light Sensor
- Limit Switch
- Line Follower
- Optical Shaft Encoder
- Ultrasonic Sensor
- Outputs
  - Motor Module
  - Servo Module
  - Digital Output
- Program Flow
  - If - Else
  - If
  - While Loop
  - For Loop
  - Timer
  - Wait
  - Assignment
  - Print To Screen
  - Comment
  - User Code
- RC Control
  - Arcade - 2 motor
  - Arcade - 4 motor
  - Tank - 2 motor
  - Tank - 4 motor
  - Motor Module Rx
  - Servo Module Rx
  - Rx Input
  - PWM Control

# Bumper Driving (part 1)

The screenshot displays the intelitek easyC for Vex controller software interface. On the left is a component palette with categories: Inputs (Bumper Switch, Light Sensor, Limit Switch, Line Follower, Optical Shaft Encoder, Ultrasonic Sensor), Outputs (Motor Module, Servo Module, Digital Output), Program Flow (If-Else, If, While Loop, For Loop, Timer, Wait, Assignment, Print To Screen, Comment, User Code), and RC Control (Arcade - 2 motor, Arcade - 4 motor, Tank - 2 motor).

The main workspace shows a project named "bumper.BDS" with a flowchart and corresponding C code. The flowchart starts with an I/O block, followed by a Variables block, and a BEGIN block. A WHILE loop contains an empty block, a block for "Set Bumper Value" (frontbumper = GetDigitalInput ( 6 );), and an IF block. The IF block has two paths: one for "if { frontbumper == 1 }" which sets "SetMotor ( 3 , 255 );" and "SetMotor ( 2 , 0 );", and another for "else" which sets "SetMotor ( 3 , 0 );", "SetMotor ( 2 , 255 );", "Wait ( 500 );", "SetMotor ( 3 , 255 );", "SetMotor ( 2 , 255 );", and "Wait ( 800 );".

The C code on the right is as follows:

```
1 #include "UserAPI.h"
2
3 int frontbumper;
4
5 void main ( void )
6 {
7     while ( 1==1 )
8     {
9         frontbumper = GetDigitalInput ( 6 );
10        if ( frontbumper == 1 )
11        {
12            SetMotor ( 3 , 255 );
13            SetMotor ( 2 , 0 );
14        }
15        else
16        {
17            SetMotor ( 3 , 0 );
18            SetMotor ( 2 , 255 );
19            Wait ( 500 );
20            SetMotor ( 3 , 255 );
21            SetMotor ( 2 , 255 );
22            Wait ( 800 );
23        }
24    }
25 }
```

Initialize variable "frontbumper"

Set Bumper Value

Bumper not pushed in

# Bumper Driving (part 2)

intelitek easyC for Vex controller

File Edit View Options Build & Download Window Help

Inputs

- Bumper Switch
- Light Sensor
- Limit Switch
- Line Follower
- Optical Shaft Encoder
- Ultrasonic Sensor

Outputs

- Motor Module
- Servo Module
- Digital Output

Program Flow

- If - Else
- If
- While Loop
- For Loop
- Timer
- Wait
- Assignment
- Print To Screen
- Comment
- User Code

RC Control

- Arcade - 2 motor
- Arcade - 4 motor
- Tank - 2 motor
- Tank - 4 motor
- Motor Module Rx
- Servo Module Rx
- Rx Input

bumper.BDS

ELSE { else {

SetMotor ( 3 , 0 );

SetMotor ( 2 , 255 );

Wait ( 500 );

SetMotor ( 3 , 255 );

SetMotor ( 2 , 255 );

Wait ( 800 );

}

}

END }

```
1 #include "UserAPI.h"
2
3 int frontbumper;
4
5 void main ( void )
6 {
7     while ( 1==1 )
8     {
9         frontbumper = GetDigitalInput ( 6 );
10        if ( frontbumper == 1 )
11        {
12            SetMotor ( 3 , 255 );
13            SetMotor ( 2 , 0 );
14        }
15        else
16        {
17            SetMotor ( 3 , 0 );
18            SetMotor ( 2 , 255 );
19            Wait ( 500 );
20            SetMotor ( 3 , 255 );
21            SetMotor ( 2 , 255 );
22            Wait ( 800 );
23        }
24    }
25 }
```

Only runs when bumper is pressed