
Build the Ultimate Remote Control Part 2 as seen in June 2007 of Servo Magazine

Pick up an issue at
<http://www.servomagazine.com>

Last month we built the bulk of our "Ultimate Remote Control". This month I'm going to show you how to add a couple of Zigbee units to the system. This will allow you to build a true two-way remote that will allow you to send commands to your robot as well as collect telemetry and display it on the controllers LCD.



The VEX Transmitter

I was experimenting with the transmitter module used in the VEX radio and came up with some interesting findings. Before I jump into the Zigbee interface I want to show you how to actually connect the VEX transmitter to your controller.

The small circuit board shown in Figure 2 is, in fact, the transmitter portion of the VEX controller. This was circuit board that was partially covering the two joysticks. You can go ahead and remove it completely from the radio if you wish to duplicate the experiments I am about to show you. For my initial tests I mounted mine on a small piece of plastic as shown in Figure 3.

The board has three connections we are concerned with. The ANT pad connects to the VEX antenna via a small lug. The GND pad connects to VSS on our controller and the VCC pad is connected to VIN. The MOD pad which is the input to the transmitter is connected to the DiosPro port 28 as shown in Schematic 1.

You can wire the transmitter temporarily to the controller so make sure the wires you connect are at least 12" long. You can trim them down later if you decide to mount the transmitter permanently to the controller.

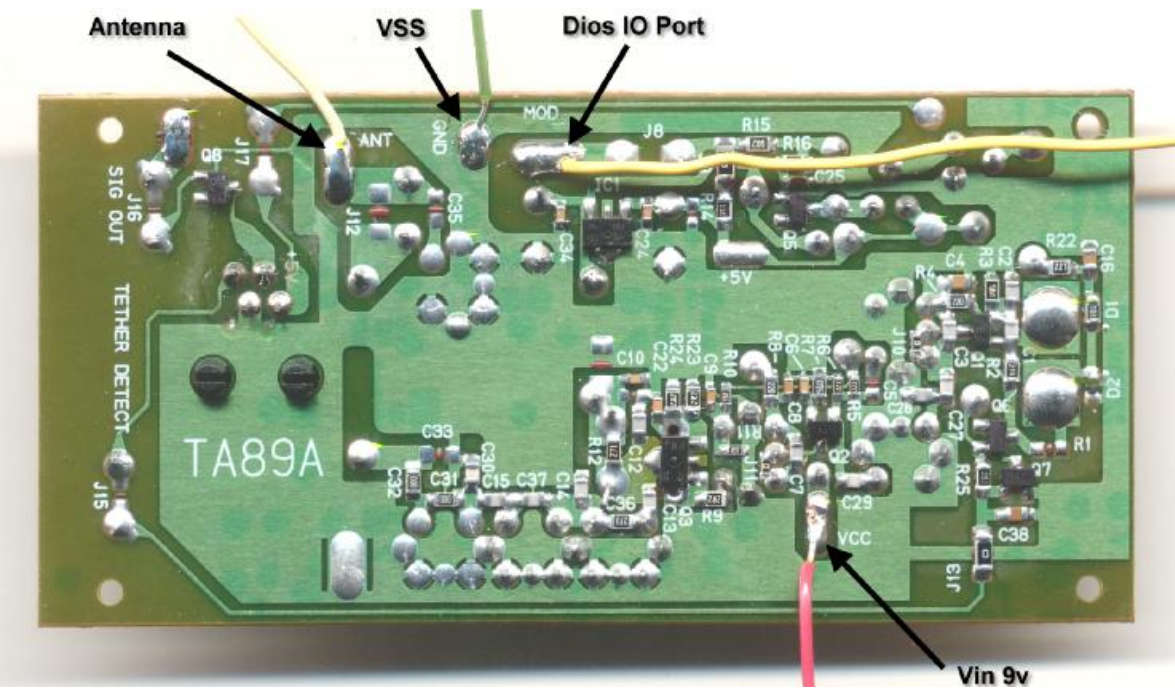
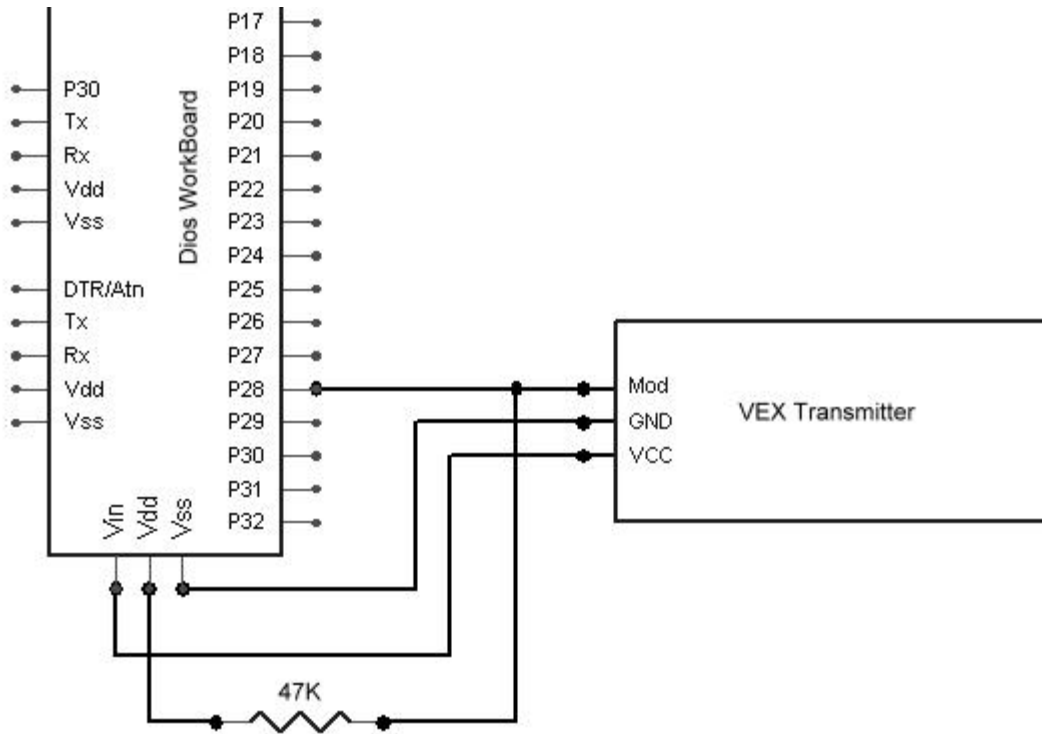


Figure 2



Schematic 1

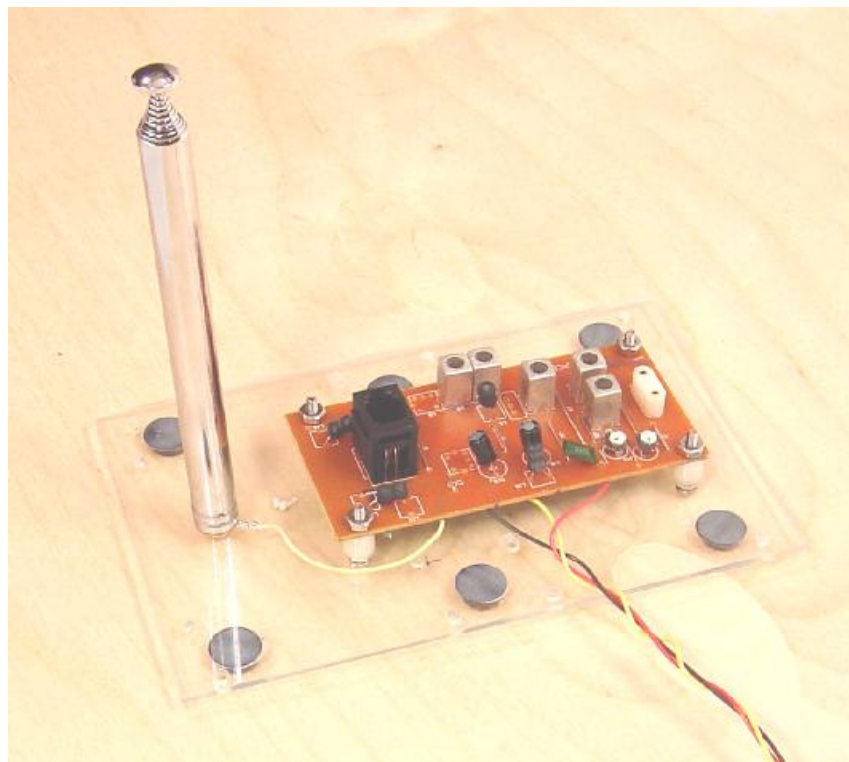


Figure 3

We modulate the transmitter by applying a ground to the mod lead on the transmitter. The timing is a bit restrictive, but by experimenting I have been able to come up with a reliable interface.

Take a look at program **VEXTXanalog.txt**. This program has a simple loop where all 8 of the analog channels on your remote are read along with the keypad and 6 buttons. The program then pulls the modulator port low for the number of microseconds corresponding to the AtoD reading for the particular channel. The same is done for the keypad and each of the 6 buttons.

In order to allow you to trim each of the channels, an offset is added to each of the AtoD channels. I added a SPDT switch to our remote to allow us to place it in program mode. Tie the center lead on the switch to Port 11 and the other leads to VSS and VDD. When Port 11 is high, the program allows you to trim the offset values of the various joystick channels. You can also reverse each axis. The readings can be saved to the DiosPro's EEPROM so that you need only set these values once. The **VEXTXanalog.txt** program has instructions as to what keypad buttons change which values.

To test the transmitter, connect it to the small two-wheeled AX12 robot we built a few months ago. This bot is a good example because it's simple enough that you should be able to make modifications to any robot application.

The program **VEXBOTanalog.txt** is meant to be programmed into the actual robot and mates up with the controller program called **VEXTXanalog.txt**. The VEX receiver comes with all the mounting hardware needed to mount both the antenna and a receiver as shown in Figure 4. I used the small cable that came with the VEX receiver as described last month. You can mount the 22K resistor on the inside or outside of the radio. Schematic 2 shows the complete hookup of the AX-12 servos and VEX radio.

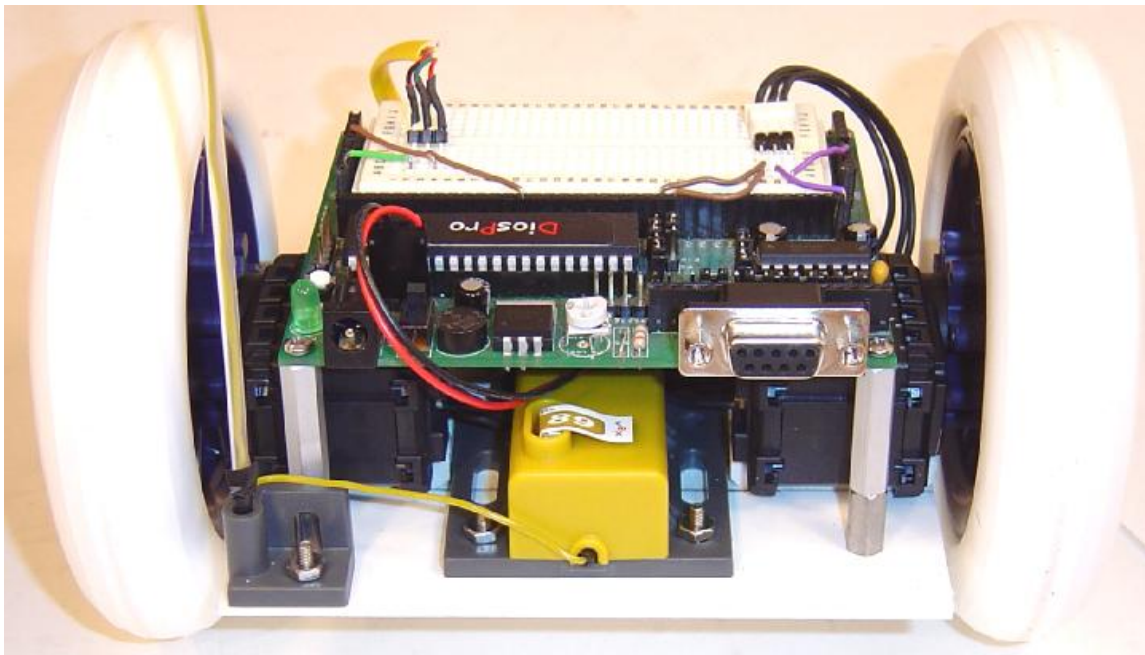
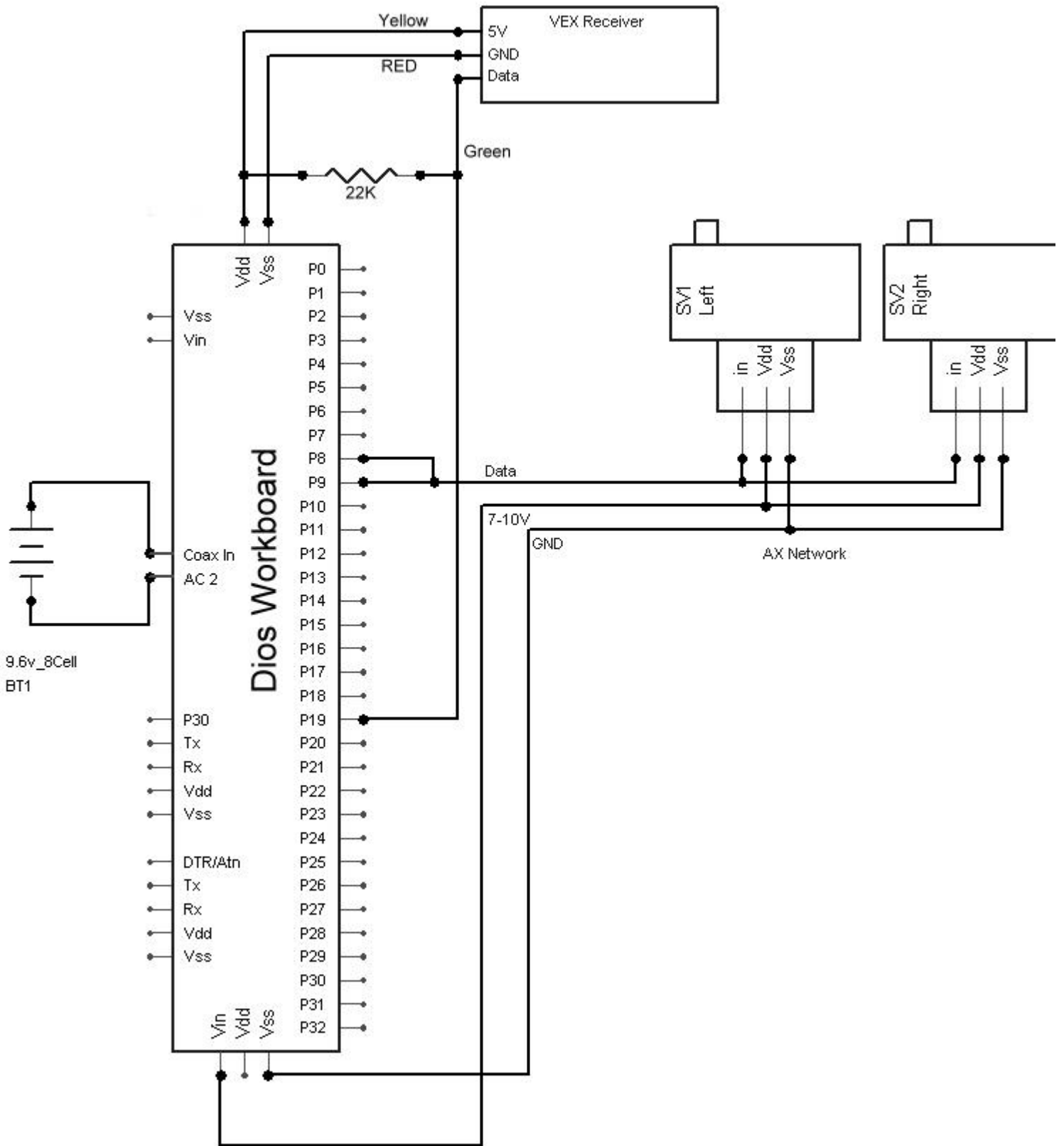


Figure 4



Schematic 2

Load the programs take the robot out for a spin. You have several channels left on the receiver so you can use the buttons and keypad to send special instructions to the robot. For instance, when ALT key (Key 11) is hit, the robot will calibrate itself to the transmitter.

If you decide to use the VEX transmitter in your remote you need to keep a few things in mind. The original 7.2v battery pack that I suggested last month cannot be used to power the system. You will need to upgrade to a 9.6v or larger pack. The problem is there is not enough room inside the remote to do this. You have a couple of options though.

1. Install some Velcro straps to the underside of the remote to hold a larger pack in place.
2. Add some additional spacers to the upper and lower bases.
3. Take the battery holder portion of the original VEX transmitter enclosure and attach that to your remote.

As you can see in Figure 1, I chose the third option. I cut the enclosure off so that all the standoffs were intact. I then used some #4 machine screws to attach the enclosure to my base. I drilled a hole to pass the battery connector (also retrieved from the VEX radio) then attached a small header in order to connect to the Coax and AC2 pins on the power header on the Dios WorkBoard.

There is space to mount the transmitter just over the Serial LCD. It was attached to the rear base between the buttons. I used a small right angle bracket to attach the antenna to the base as well. I also cut out a small square hole so I could change the transmitter crystal.

All in all the radio works pretty well. I also included a Bot program called **VEXBOTanalog2.txt**. This program works the same as the VEXBOTanalog.txt except a custom assembly language decoder was built. It does not have the resolution that the original program does, but it is more stable and has a feature where it can detect the loss of the transmitter signal.

Final Thoughts on the VEX Interface

The world of analog RC signals is not a perfect one. There is always a certain amount of drift the transmitter pulse width. When I created this interface I added a new command to the DiosPro called fuzzy that works very much the same as the lookdown command except that lookup indexes are supplied in pairs. These pairs define the high range and low range that is expectable for the various signal pulses used by both the keypad and buttons.

I have included a couple of extra VEX interface programs that let you experiment with creating a full digital interface. The program **VEXTXdig.txt** encodes the raw AtoD signals into a 10bit + parity value that is transmitted to the receiver. The **VEXBOTdig.txt** is used to decode these signals. The programs are only in the experimental stage and need a bit more work for prime time.

While there is a lot you can do with the VEX transmitter interface I only show it here so that you can perform your own experiments. While we don't make any changes to the actual transmitter, I'm sure these would be considered modifications and may or may not be allowed by FCC regulations.

Zigbee Interface

Now we are going to open up the real power of our remote control system. By using Zigbee modules you will have true two way communications with your robot. This will allow you to transmit the equivalence of several channels, as well as receive telemetry data from the robot. As a bonus the Zigbee modules require less battery power. This allows us to use the original 6-Cell battery pack described back in Part 1. I also found the Zigbee modules easier to work with and because of their size I had more options when mounting them in the enclosure. I will show you how to use two different modules from different manufacturers so you can decide which will best suite your needs.

ZIG-100 Module

The Robotis ZIG-100 module shown in Figure 5 is the easiest Zigbee module I have ever had the pleasure to get my hands on. You purchase them in pairs already configured for peer-to-peer operation. This means you don't have to have any understanding of the underlying Zigbee operations. As if the modules could not be any easier to work with, they come with .1" headers installed so that you can plug them right into your breadboard like the module shown in Figure 6. I took these modules right out of the box and wired one into the controller and the other in the AX robot and was up and running in just a few minutes.



Figure 5

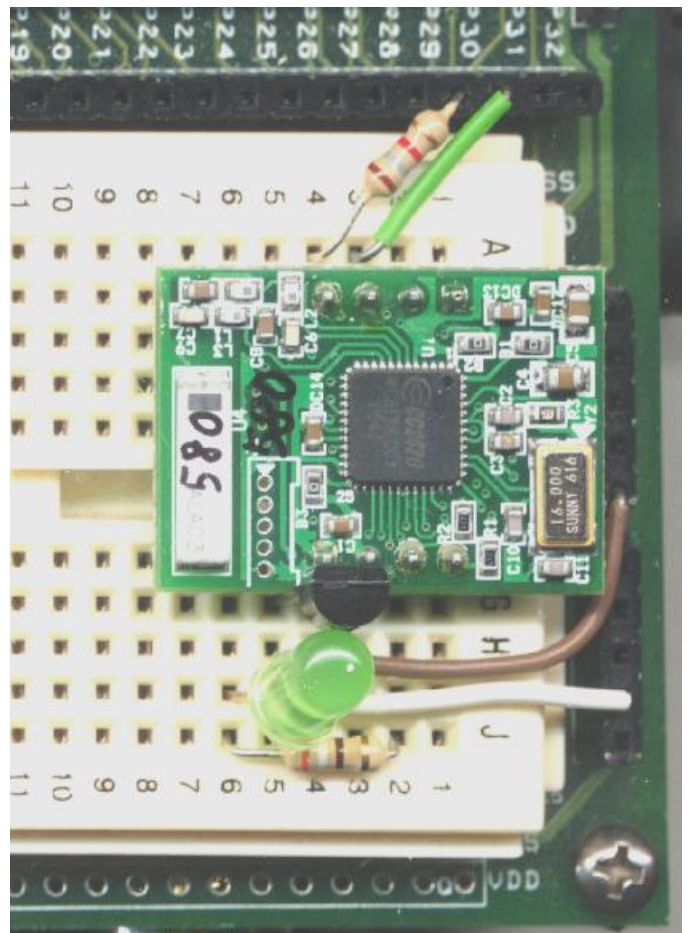


Figure 6

MaxStream XBee Module

The Maxstream XBee module shown in Figure 7 is a very serious Zigbee module. And while they are not plug and play like the ZIG-100 modules, they do have over 50 configuration parameters that will allow you to create different network configurations, as opposed to the ZIG-100's 3 parameters. This flexibility does come, however, at the cost of more complexity. You must at least have a basic understanding of Zigbee networking and the XBee command set. The XBee module has 2, 10-pin headers, the down side is that they are 2mm on center. This adds to a more complicated interface. If you want to use this module you have to build some sort of interface board.

Zigbee Interface Board

Zigbee modules operate at 2.7-3.6V. In order to interface 5v circuits to these modules and to have the ability to mount the modules we need to build some sort of board. For the ZIG-100 module this is pretty easy considering the headers are .1" on center so they will plug into just about any PCB you can pick up at your local Radio Shack. Not So with the MaxStream XBee modules.

MaxStream Xbee Interface Board

For the Xbee you have a couple of choices; Figures 8a through 8d show the various application notes I have created. They can be seen at:

<http://www.kronosrobotics.com/Projects/MaxStreamInterface1.shtml>

<http://www.kronosrobotics.com/Projects/MaxStreamInterface2.shtml>

<http://www.kronosrobotics.com/Projects/MaxStreamInterface2.shtml>

<http://www.kronosrobotics.com/Projects/MaxStreamInterface4.shtml>

I wanted to create a common interface for using both types of modules, so the controller and robot interface are the same regardless of the type of Zigbee module you use. Figure 9 shows the wired Zig100 module. Schematics 3 and 4 show the schematics for both the XBee and ZIG-100 module interfaces. You may omit the LED's if you don't want a connection indicator. On my controller I mounted the LED on the top base. The 1.8K/3.3K resistor pair on the ZIGRX lead is used to drop the 5v IO port to 3V in order not to damage the module. The 100 Ohm resistor on the ZIGTX is not needed but is there in order to keep a short or other misconnection from damaging the module. If you are going to mount one of the modules on a noisy robot you may want to add some 100uf caps to both sides of the regulator.



Figure 7



Figure 8a



Figure 8b

The ZIGREST lead is only needed for configuration and is not used during normal operation.

Module Status LED's

The way the status LED works is different for both modules. On the ZIG-100, the LED flashes when the unit is not connected to another module and goes solid when a connection is achieved. On the XBee, the LED lights when the module receives data and the amount of brightness indicates signal strength.

I should also note that the XBee needs almost 10 seconds from the point both controller and robot modules are turned on to sync up. On the ZIG-100 it takes about 1 second.

ZIG-100 Library

The DiosPro has libraries for configuring both modules and I have included programs called XBsetup.txt and ZIG100setup.txt to get you up and running as soon as possible. While the ZIG-100 modules can be used immediately without configuration changes, you will need to make changes to the default XBee module settings. To configure either module connect it to the DiosPro as shown in Schematic 5. Make sure you add the reset lead to Port 10 as it is not shown in the schematic. Next then run the included XBsetup.txt or Z100setup.txt program on both the controller and robot modules.

You set the destination address to match the source address to the other module as shown in Figure 11. For example module 500 sets its destination to 501 and module 501 sets its destination to 500. On the ZIG-100 modules the source is permanently set and called ID. On the XBee unit you will need to make up an ID.



Figure 8c



Figure 8d

Peer to Peer Network Configuration

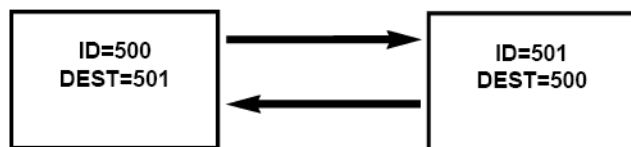


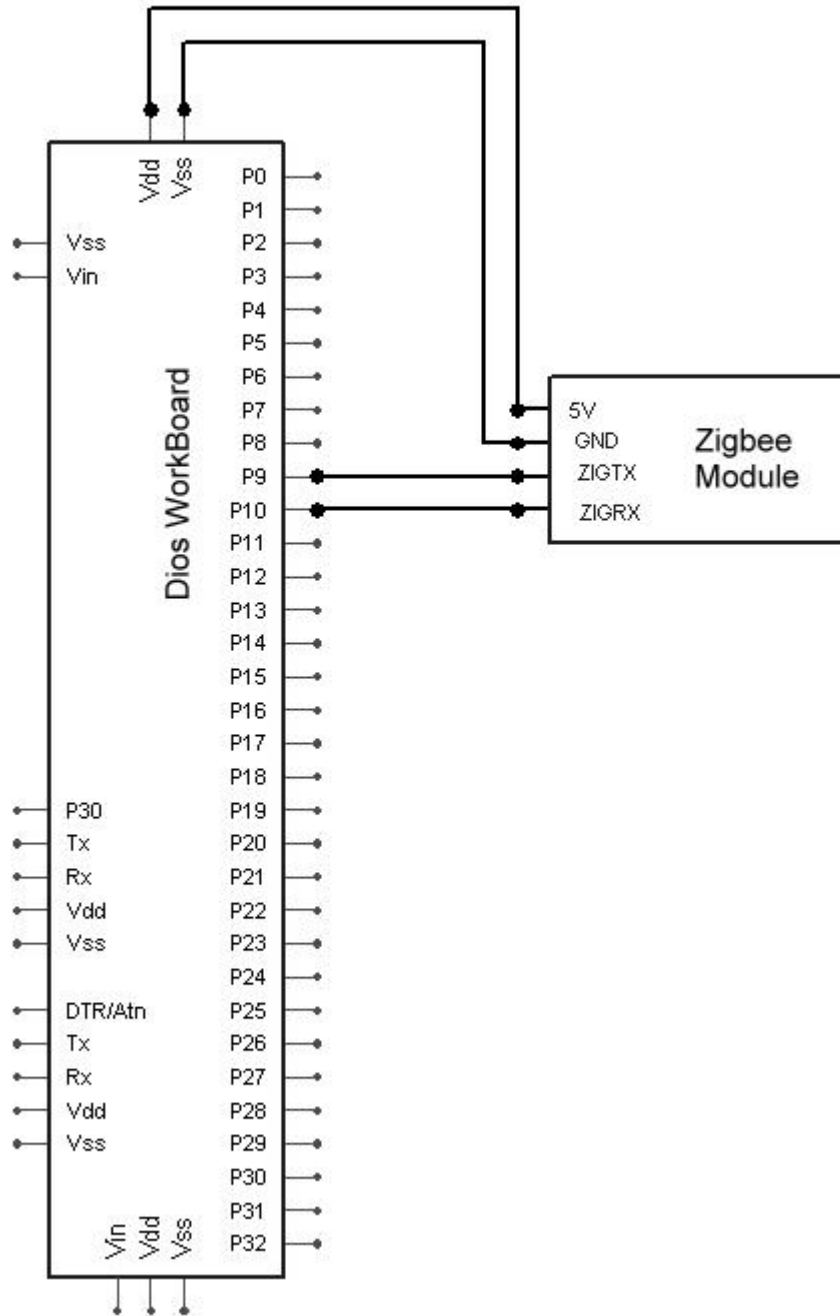
Figure 11

Zigbee Controller Hookup

Schematic 5 shows the connections needed to add either module to the remote control system.



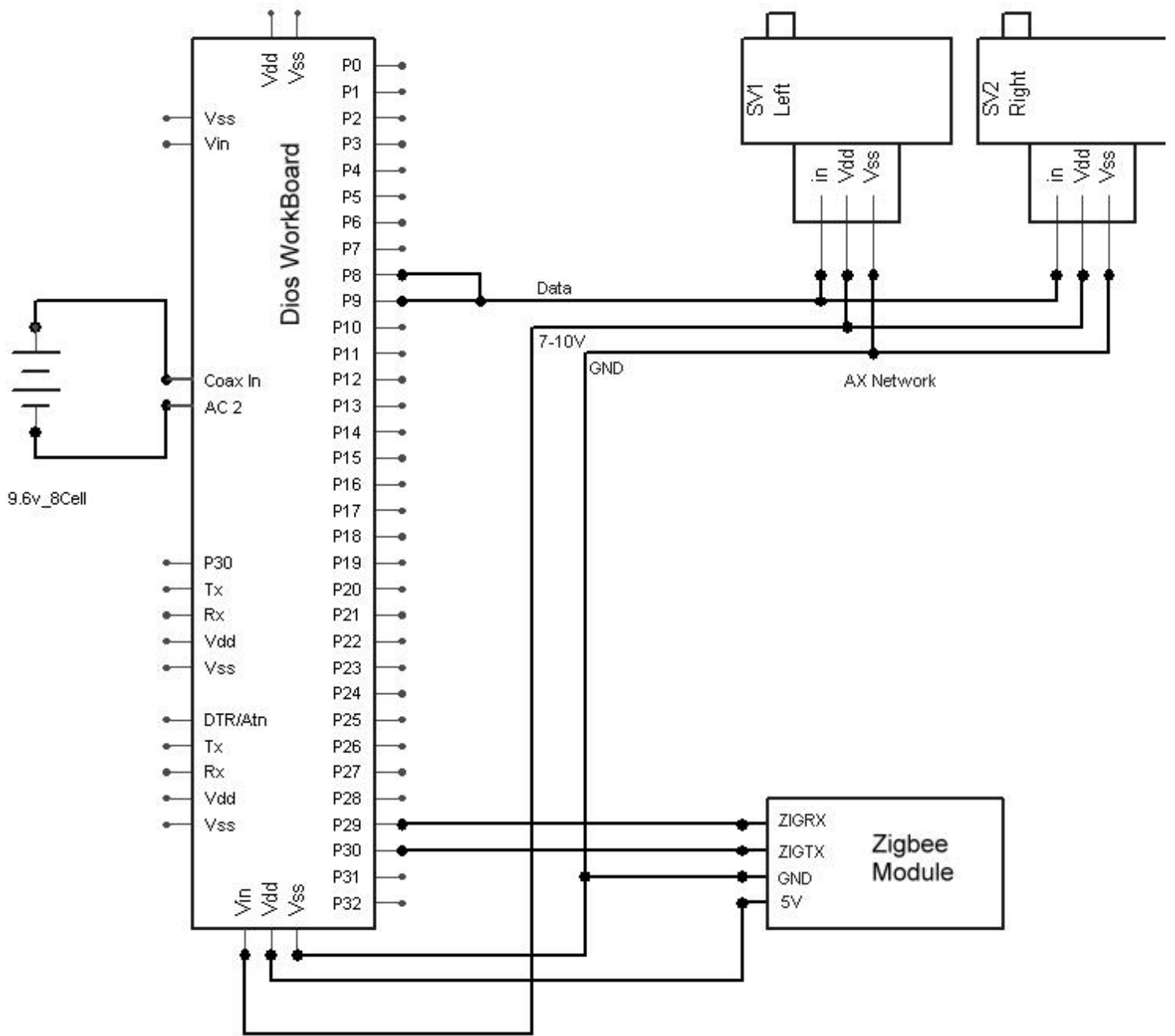
Figure 9



Schematic 5

Zigbee Robot Connection

The Zigbee module connection on the robot is exactly the same as on our remote. The only difference is that the hardware UART is occupied by the AX-12 Smart servos so we are going to use ports 29 and 30 on the DiosPro as shown in Schematic 6.



Schematic 6

Testing the ZIG100 Interface

Let's start by loading up the ZIGTX.txt program and programming it into the controller. This program runs much different than the VEX version. The remote actually becomes a slave and monitors the hardware UART port for Zigbee activity. The programmer section has also been upgraded a bit as well. If the program switch on Port 11 is high, it will check the keypad and make changes to the trimmer and reverse options on the controller.

If Zigbee activity is detected, the appropriate command sent by the robot is processed and serviced. The remote is designed to send out a value of 1500 when in the joystick is in the center position. The keypad sends a range of 0-12 when requested and the buttons will send a single byte with each of the 6 buttons representing a single bit.

Load up program ZIGBOT0.txt into a remote DiosPro or robot. In this first program, the program requests data from all the channels on the controller. Use this as a template for adding the code to your own bots. I have created a DiosPro library called URBOT.lib. Just making the call to the URserinit will load the library so that you can access the four main functions; URsergetchannel, URsergetkey, URsergetbut, URsersendLCDchr and URsersendLCDstring.

The UR (Ultimate Remote) interface is a simple one. The remote robot makes a request via the Zigbee interface to the controller. The controller looks at the appropriate command request and services the request. Table 1 shows the current list of valid commands. You can add your own. Add a new label with the code that handles the request then add that label to the branch command.

| Command | Description |
|---------|--|
| 1 | Bot is requesting analog channel 1 data. It is sent back in 2 bytes. |
| 2 | Bot is requesting analog channel 2 data. It is sent back in 2 bytes. |
| 3 | Bot is requesting analog channel 3 data. It is sent back in 2 bytes. |
| 4 | Bot is requesting analog channel 4 data. It is sent back in 2 bytes. |
| 5 | Bot is requesting analog channel 5 data. It is sent back in 2 bytes. |
| 6 | Bot is requesting analog channel 6 data. It is sent back in 2 bytes. |
| 7 | Bot is requesting analog channel 7 data. It is sent back in 2 bytes. |
| 8 | Bot is requesting analog channel 8 data. It is sent back in 2 bytes. |
| 9 | Bot is requesting keypad data. It is sent back in a single byte 0-12 |
| 10 | Bot is requesting button data. It is sent back in a single byte 0-63 with each bit representing one of the buttons. |
| 11 | Bot is sending a complete LCD string. First the length is sent then each of the bytes that are meant to be sent to the Serial LCD. This data is just a pass through. |
| 12 | Bot is sending a single LCD character. |
| 13 | Bot is sending a single LCD character that is to be routed to the debug port on the controller. |

Table 1

Load up program ZIGBOT1.txt into your robot. This program will allow you to control the two AX-12 servos using channels 2 and 4 on the controller. When the program starts it takes a reading of the two channels and that is used as a reference for the center position of the joysticks. The program will also stop the robot if the link between the Zigbee units are lost. This is a safety feature.

Program ZIGBOT2.txt goes to the next level and takes a voltage reading from one of the AX-12's and sends it to the LCD. This allows you to monitor the battery voltage on your robot directly on the remotes LCD display.

Program ZIGBOT3.txt adds even more display data taken from the AX-12 smart servos. The current load and temperature is taken from the servos and sent to the LCD.

Final Thoughts

The real power of the Ultimate Remote will really come into its own once you start building very complicated robots. When I first started using the AX-12 smart servos I built a very sophisticated biped robot. It had so much functionality I had no way of controlling the thing. Well, the cool part is now I do.

I have played with the range and have been able to control my robots at well over 60 feet. This is good considering I have several other technologies including WiFi operating on the same frequencies.

Going Further

There are quite a few enhancements you could make to the control system. Currently we are only saving a handful of configuration parameters to the DiosPros's EEPROM. You could save information for five or six different robots.

You could also add a base set of setup commands to allow you to change the ID and channels on the units. On the XBee, changing the channels changes the actual frequency the module is operating on.

You could make the modules accessible on the controller so that it could be used to configure all your modules.

All the example programs as well as the source are available for download at:
<http://www.kronosrobotics.com/Projects/remote.shtml>

Parts

Available from Kronos Robotics - www.kronosrobotics.com

- DiosPro Chip #16428
- Dios Workboard Deluxe #16452
- DiosCompiler Free Download from www.kronosrobotics.com

All Electronics - www.allelectronics.com

- Two 6-Channel Transmitter Receivers #JS-6

Crustcrawler - www.crustcrawler.com

- ZIG-100 Kit