

Volume

1

A-WIT TECHNOLOGIES INC.

... a passion for execution ...

Using the BOL-BOT with a Radio Controller System

Version 1.0

A-WIT TECHNOLOGIES INC.

Using the BOL-BOT with a Radio Controller System

© A-WIT Technologies Inc.
Phone (800) 985-AWIT • Fax (800) 985-2948

Table of Contents

Notices	1
Getting Support	2
Direct Control of the BOL-BOT Servos	2
C Stamp Control of the BOL-BOT Servos	5
Terms and Conditions	15

Introduction

This Application Note describes how to control the A-WIT BOL-BOT robot with a Radio Controller. This is not hard to do at all. It can be done in two ways. First, the C Stamp can be bypassed completely, and the radio receiver can control the BOL-BOT servos directly. However, with this paradigm, the relationship of movements of the radio controller levers to the actual movements of the BOL-BOT is cumbersome. A second approach has the radio receiver interface to the C Stamp, and the C Stamp in turn controls the servos, such an accurate relationship between the radio controller levers movements and BOL-BOT movements is achieved.

Notices

CSTAMP™ and CSTAMP™ Related Hardware Products, Software Products and Documentation are developed and distributed by A-WIT Technologies, Inc. All rights reserved by A-WIT Technologies, Inc. A-WIT SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL A-WIT BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

MPLAB C-18 and MPLAB C-18 Users Guide is reproduced and distributed by A-WIT Technologies, Inc. under license from Microchip Technology Inc. All rights reserved by Microchip Technology Inc. MICROCHIP SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED “AS IS,” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO

EVENT SHALL MICROCHIP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

Getting Support

If possible, please check the C Stamp website www.c-stamp.com under SUPPORT for any updates to documentation, changes, or notices that may have become available since your Installation CD was produced. If you continue to have any issues for which a solution is not found in the aforementioned website, please e-mail tech_support@a-wit.com for help.

Direct Control of the BOL-BOT Servos

This section describes how to configure the BOL-BOT so that it can be controlled using a wireless RC remote control bypassing the C Stamp.

Supplies you will need:

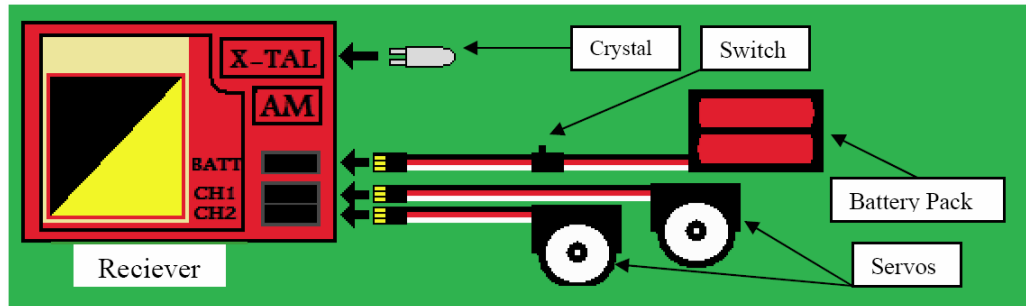
- Ranger II N – Hitec - 2CH AM Radio Control System
- BOL-BOT
- Hitec 2CH AM Receiver
- Hitec Receiver Battery Pack
- Micro Switch Harness
- Crystal
- 8 AA and 4 AAA Batteries
- Extension cables
- Hookup wires

Below is a picture of the transmitter.

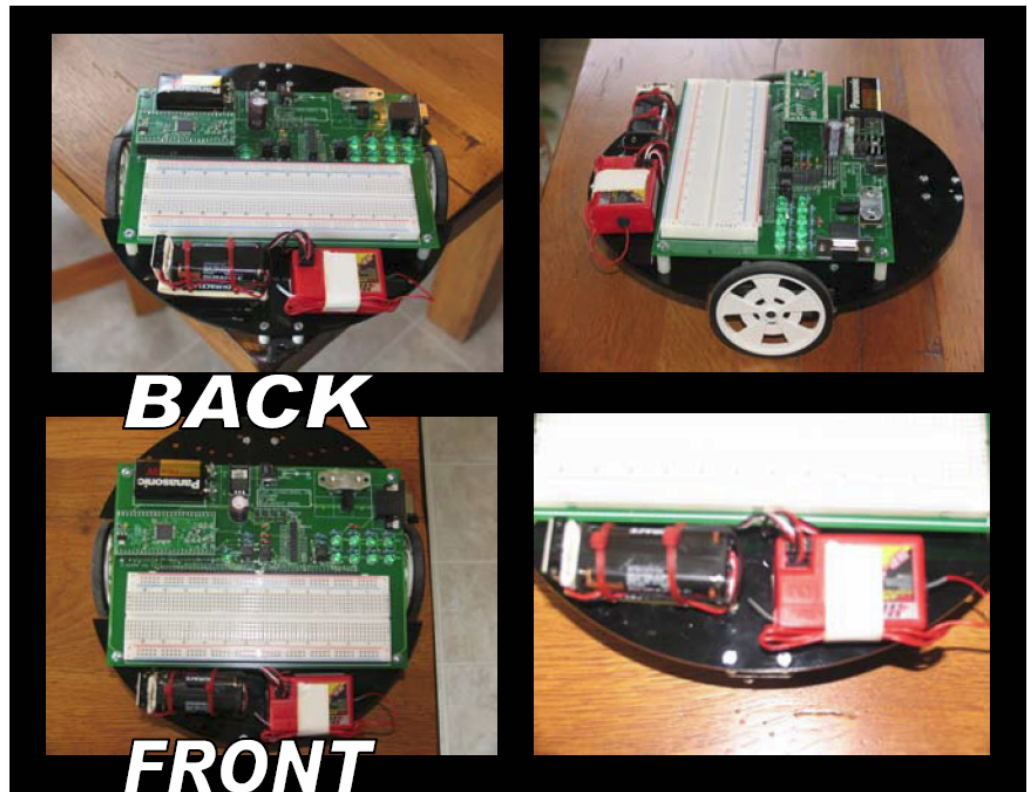


The manual for the transmitter has instructions on how to connect the various items and wires that come with it to allow the robot to be controlled using the remote control. The receiver has three ports on the front, one port on the side, and an antenna. The two ports that are labeled ch 1 and ch 2 are the ports that connect the servos. There is another port right above the other two, labeled BATT, which connects to the receiver's battery pack. The battery pack takes four AAA batteries. There is also a Micro Switch Harness. The battery pack has to be connected to the Micro Switch Harness in order for the receiver to be turned on and off easily. The last item is the crystal. The crystal goes in the side of the receiver in the port marked X-TAL.

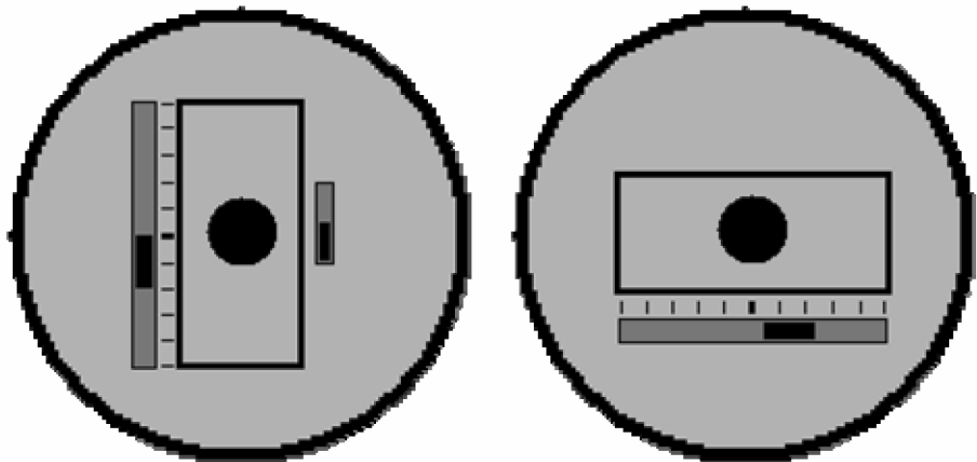
The following diagram shows an example of how to connect the different components to the receiver.



Below are a few pictures of the entire BOL-BOT with the receiver and the battery pack attached to it.



After attaching everything, you can test the transmitter out and see what it makes the BOL-BOT do. Adjusting the switches next to the left and right control sticks affect how fast the wheels move. Bellow is a diagram of what the settings can be to prevent the BOL-BOT from moving when the control levers are not being pushed.



Now that the BOL-BOT is at rest when it is not being moved with the transmitter, you can find out what control does what. After some experimentation, you will find out how each Control Stick moves the BOL-BOT in a different way.

Transmitter Input Table:

Left Control Stick	Right Control Stick	Result
Nothing	Nothing	At Rest
Up	Nothing	Slow Turn Forward To The Right
Back	Nothing	Slow Turn Backward To The Left
Nothing	Right	Slow Turn Forward To The Left
Nothing	Left	Slow Turn Backward To The Right
Up	Left	Sharp Turn To The Right
Up	Right	Go Forward
Back	Right	Sharp Turn To The Left
Back	Left	Go Backward

To test out the project, make the BOL-BOT take different paths. You can do things like making the BOL-BOT go straight and see how far it can go, and make it go behind a wall and see if the BOL-BOT keeps going. Another thing you can try is to put down cups to see if you can control the BOL-BOT to swerve in and out of each one. You will see that you have a lot of control with the BOL-BOT.

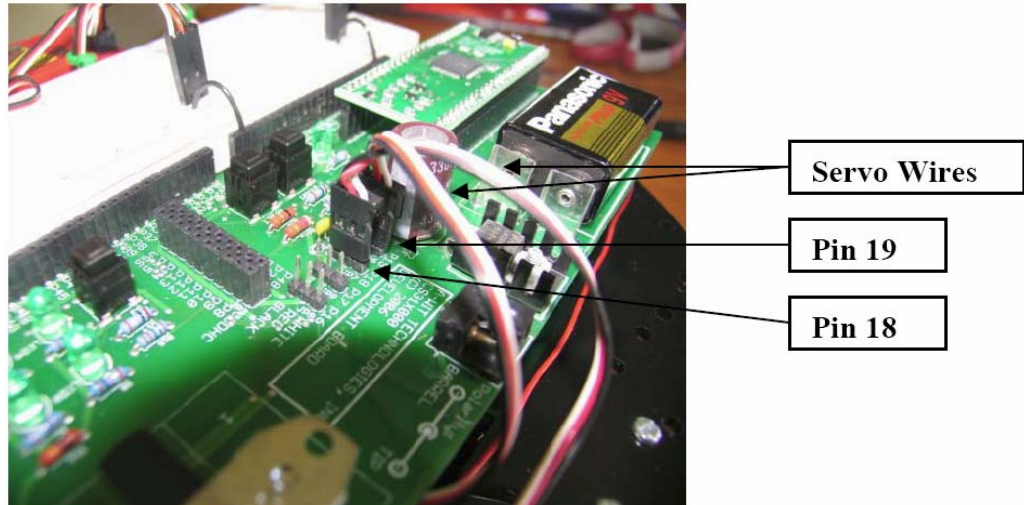
C Stamp Control of the BOL-BOT Servos

This section describes how to configure the BOL-BOT so that it can be controlled using a wireless RC remote control using the C Stamp to control the BOL-BOT servos.

In the previous project, when you use the control sticks to move the robot, each one controlled a separate wheel, so you have to hold down both of the control sticks at the same time in order for it to go in to the direction that you want. Additionally, the directions that you have to push the sticks are not very user friendly, so let us change that. The goal for this project is to program the BOL-BOT so that both of its wheels would simultaneously move when one of the control sticks is moved forward, backward, left, and right.

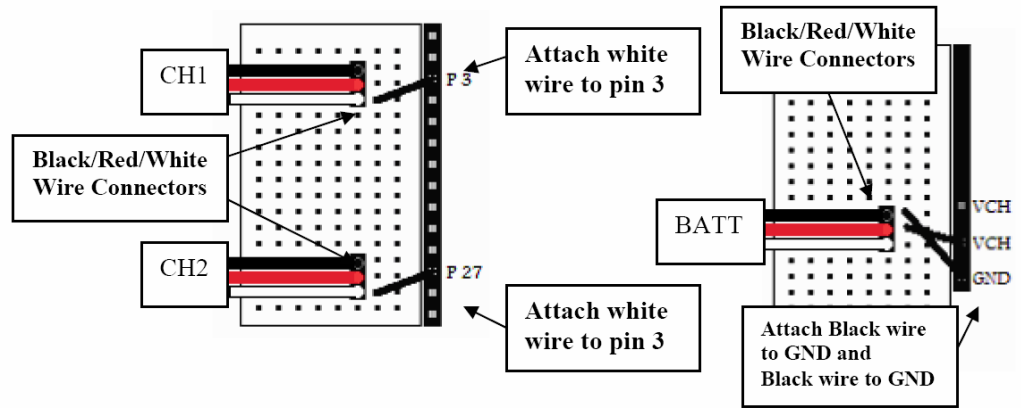
To accomplish this you can use the **PULSIN** command to receive the input from the radio receiver into the C Stamp. You can look this command up in the A-WIT manual if you need additional help. The manual describes how to use **PULSIN** and provides a sample program.

In this project, you no longer needed the receiver's battery pack because now it would be using the power from the BOL-BOT main battery. You should also connect the servo wires to pins 19 and 18, as shown in the figure below.

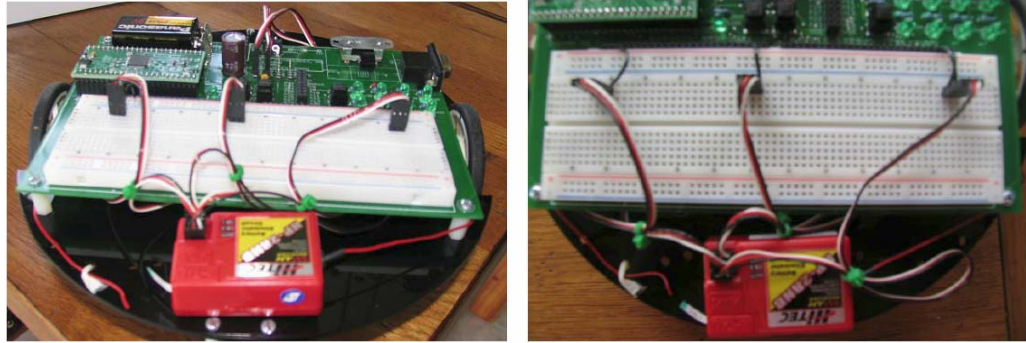


You will have to use the breadboard, which is the big white board made for connecting wires. Connect the BOL-BOT's battery to the receiver in the slot that is labeled BATT for battery. Use extension cables, which look like the servo wires because they are black, red and white, to connect the battery slot to the breadboard. It does not matter where the wires are connected on the breadboard for now. In order for the wire to be connected to the battery, the red wire has to be connected to the VHC port and the black wire has to be connected to the GND port.

Now you have to connect the channels 1 and 2 slots to the breadboard. Use extension cables for both channels 1 and 2 and connect them in the breadboard. When you move one of the control sticks, there is a pulse sent to that channel. What you want to do is to make it so that the pulse from that channel goes through to a wheel servo, as controlled by the C Stamp. Since the white wire is the servo control, use it to connect to the pins. Use hook-up wires to connect the white wire in channel 1 to pin 27 and the white wire in channel 2 to pin 3. Here are diagrams of how to connect them.



Here are pictures of the whole Robot.



Now that everything is connected properly, it is time to write the code for the robot using the **PULSIN** command. Try using **PULSIN** to make LEDs blink first, because it is the easiest way to start. When you press the control stick to go up, one LED blinks, when you press down, another LED blinks, and when you do nothing, a third LED blinks. It is a good way to test out the **PULSIN** command.

Start with channel 1, which has the control stick that goes left to right and is connected to pin 27. You have to **PULSIN** pin 27. The transmitter is constantly sending a pulse of 1.5 mS to the robot, which keeps it at rest. By moving the control sticks, the transmitter sends pulses greater than or less than 1.5 mS. To determine what direction makes what pulse, a pulse of greater than 1.5 mS causes one LED to blink and a pulse of less than 1.5 mS causes another LED to blink. When you test this program out, you will find out what direction you needed to put in each pulse on the first channel. After that, you can do the same thing for the second channel but instead of having to do **PULSIN** on pin 27, you use **PULSIN** on pin 3, which connected to the other channel.

The code looks like this for channel 1:

```

#include "CS110000.h"
void main(void)
{
    BIT ON;
    WORD WIDTH;
    while(1){
        // becomes WIDTH = PULSIN(3, 1, 65000); for ch 2
        WIDTH = PULSIN(27, 1, 65000);
        if((WIDTH<=2500) && (WIDTH >=2187)){
            STPIND(39, HIGH);
        }else{
            if((WIDTH<=1562) && (WIDTH >=1250)){
                STPIND(42, HIGH);
            }else{
                STPIND(46, HIGH);
            }
        }
        }
    PAUSE(1000);
    STPIND(39, LOW);
    STPIND(42, LOW);
    STPIND(46, LOW);
}
}

```

After you find out that this works for both channels, combined both of the programs to make the LEDs blink for both channel 1 and 2. When are able to do this, all that is left to be done is to replace the LED blinks with functions to go forward, backward, left, and right all in the appropriate places. Here was the final code.

```

#include "CS110000.h"

void goback(BYTE lp, BYTE rp, WORD t);
void goforth(BYTE lp, BYTE rp, WORD t);
void turnl(BYTE lp, BYTE rp, WORD t);
void turnr(BYTE lp, BYTE rp, WORD a);
float tr = 93.79; // deg/S

void main(void)
{
    BYTE lp = 19;
    BYTE rp = 18;
    BYTE LED8 = 39;
    BIT ON;

```

```

WORD WIDTH;
while(1){
    if(WIDTH = PULSIN(27, 1, 65000)){
        if((WIDTH<=2500) && (WIDTH >=2187)){
// invoke the function to make the robot turn Right
            turnr(lp, rp, 1);
        }
        else{
            if((WIDTH<=1562) && (WIDTH >=1250)){
// invoke the function to make the robot turn Left
                turnl(lp, rp, 1);
            }
            else{
                STPIND(46, HIGH);
            }
        }
    }
//    PAUSE(1000);
    STPIND(39, LOW);
    STPIND(42, LOW);
    STPIND(46, LOW);
//    PAUSE(1000);
}
    if(WIDTH = PULSIN(3, 1, 65000)){
        if((WIDTH<=2500) && (WIDTH >=2187)){
// invoke the function to make the robot go Straight
            goforth(lp, rp, 1);
        }
        else{
            if((WIDTH<=1562) && (WIDTH >=1250)){
// invoke the function to make the robot go Back
                goback(lp, rp, 1);
            }
            else{
                STPIND(46, HIGH);
            }
        }
    }
    STPIND(39, LOW);
    STPIND(42, LOW);
    STPIND(46, LOW);
}
}
// Function to make the robot turn Left
void turnl(BYTE lp, BYTE rp, WORD a){
    BYTE t, i;

```

```

    t = (a / tr) / 0.02 + 0.5;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 800, 1, 1);
        PULSOUT(rp, 800, 1, 1);
        PAUSE(16);
    }
}
// Function to make the robot turn Right
void turnr(BYTE lp, BYTE rp, WORD a){
    BYTE t, i;
    t = (a / tr) / 0.02 + 0.5;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 400, 1, 1);
        PULSOUT(rp, 400, 1, 1);
        PAUSE(18);
    }
}
// Function to make the robot go Back
void goback(BYTE lp, BYTE rp, WORD t){
    WORD i;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 400, 1, 1);
        PULSOUT(rp, 800, 1, 1);
        PAUSE(17);
    }
}
// Function to make the robot go Straight
void goforth(BYTE lp, BYTE rp, WORD t){
    WORD i;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 800, 1, 1);
        PULSOUT(rp, 400, 1, 1);
        PAUSE(17);
    }
}
}

```

With this code, the robot moves straight using one control stick, backward using one control stick, left using one control stick, and right using one control stick. We can improve on this code by having the robot to be able to move and turn at the same time. You can keep the previous code that made the robot move forward, backward, left, and right, and add the new code that makes the robot move in two directions at one time.

You can do a nested if statement because it is the easiest to set up. Start out by finding out if the user wanted the robot to go forward or backward. Upon finding that out,

write statements for it to find out if the user also pressed the left or right direction at the same time as the other direction. It works!

Here is the final program after the final code is added.

```

#include "CS110000.h"

void goback(BYTE lp, BYTE rp, WORD t);
void goforth(BYTE lp, BYTE rp, WORD t);
void turnl(BYTE lp, BYTE rp, WORD t);
void turnr(BYTE lp, BYTE rp, WORD a);
void turnrr(BYTE lp, BYTE rp, WORD a);
float tr = 93.79; // deg/S

void main(void)
{
  BYTE lp = 19; // left servo pin
  BYTE rp = 18; // right servo pin
  BYTE LED8 = 39; // on board LED8 pin
  BYTE x;

  BIT ON;
  WORD WIDTH;
  while(1){
    if(WIDTH = PULSIN(27, 1, 65000)){
      if((WIDTH<=2500) && (WIDTH >=2187)){
        if(x = 2){
          STPIND(39, HIGH);
          turnr(lp, rp, 1);
        }else
          PAUSE(20);
      }
      if((WIDTH<=1562) && (WIDTH >=1250)){
        if(x = 1){
          STPIND(42, HIGH);
          turnl(lp, rp, 1);
        }else
          PAUSE(20);
      }
    }
    if(WIDTH = PULSIN(3, 1, 65000)){
      if((WIDTH<=2500) && (WIDTH >=2187)){
        if(x = 4){
          STPIND(39, HIGH);
          goforth(lp, rp, 1);
        }
      }
    }
  }
}

```



```

        PULSOUT(rp, 400, 1, 1);
        x = 1;
    }
}
void turnl(BYTE lp, BYTE rp, WORD a)
{
    BYTE t, i, x;
    t = (a / tr) / 0.02 + 0.5;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 800, 1, 1);
        PULSOUT(rp, 800, 1, 1);
        x = 2;
    }
}
void goback(BYTE lp, BYTE rp, WORD t)
{
    WORD i;
    BYTE x;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 400, 1, 1);
        PULSOUT(rp, 800, 1, 1);
        x = 3;
    }
}
void goforth(BYTE lp, BYTE rp, WORD t)
{
    WORD i;
    BYTE x;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 800, 1, 1);
        PULSOUT(rp, 400, 1, 1);
        x = 4;
    }
}
void turnrr(BYTE lp, BYTE rp, WORD a)
{
    BYTE t, i, x;
    for(i = 0; i < t; i = i + 1){
        PULSOUT(lp, 1000, 1, 1);
        PULSOUT(rp, 500, 1, 1);
        PAUSE(15);
    }
}
}

```

Here is a chart of the inputs before the Robot was reprogrammed.

Left Control Stick	Right Control Stick	Result
Nothing	Nothing	At Rest
Up	Nothing	Slow Turn Forward To The Right
Back	Nothing	Slow Turn Backward To The Left
Nothing	Right	Slow Turn Forward To The Left
Nothing	Left	Slow Turn Backward To The Right
Up	Left	Sharp Turn To The Right
Up	Right	Go Forward
Back	Right	Sharp Turn To The Left
Back	Left	Go Backward

Now here is a chart of the inputs after the Robot was reprogrammed.

Input	Result
UP	Robot moved Forward
DOWN	Robot moved Backward
LEFT	Robot moved Left
RIGHT	Robot moved Right
UP/LEFT	Robot moved forward and left
UP/RIGHT	Robot moved forward and right
DOWN/LEFT	Robot moved backward and left
DOWN/RIGHT	Robot moved backward and right

As you see, reprogramming the robot has made controlling it much more user friendly, because it is simpler and a lot easier to control. Now controlling the robot is so much easier!

Terms and Conditions

Quality Assurance

A-WIT has stringent quality control procedures in place to insure the best quality products.

90-Day Limited Warranty

A-WIT Technologies, Inc warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, A-WIT Technologies, Inc. will, at its option, repair, replace, or refund the purchase price. After 90 days, products can still be sent in for repair or replacement, but there will be a \$10.00USD minimum inspection/labor/repair fee (not including return shipping and handling charges).

14-Day Money-Back Guarantee

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a refund. A-WIT will refund the purchase price of the product in the form of a check, excluding shipping/handling costs, once the product is received. This refund does not apply if the product has been altered or damaged. If you decide to return the products after the 14-day evaluation period, a 20% restocking fee will be charged against a credit.

Disclaimer

Warranty does not apply if the product has been altered, modified, or damaged. A-WIT makes no other warranty of any kind, expressed or implied, including any warranty of merchantability, fitness of the product for any particular purpose even if that purpose is known to A-WIT, or any warranty relating to patents, trademarks, copyrights or other intellectual property. A-WIT shall not be liable for any injury, loss, damage, or loss of profits resulting from the handling or use of the product shipped.

How to Return a Product

When returning, you must first e-mail sales@a-wit.com for a Return Merchandise Authorization number. No packages will be accepted without the RMA number clearly marked on the outside of the package. After inspecting and testing, we will return your product, or its replacement using the same shipping method used to ship the product to A-WIT within 30 days. In your package, please include a daytime telephone number and a brief explanation of the problem.

Please contact our Sales Department at sales@a-wit.com if you have any questions regarding our warranty policy or if you are requesting an RMA number.

