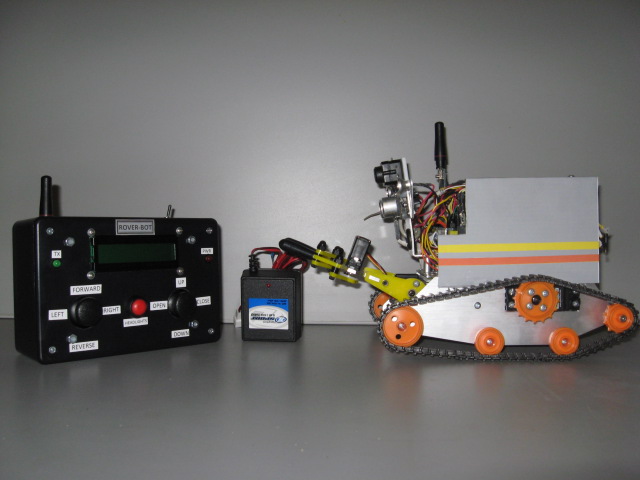
****

**MECHATRONICS PROJECT**

**SPRING 2010**



TEAM [**ROVERBOT**]

BRIAN HOLIDAY

CHARLES HEATH

Table of Contents

**PROJECT DETAILS:**

DESIGN SUMMARY3

STAGE 1: ROBOT 5

STAGE 2: CONTROLLER 6

STAGE 3: ROBOTIC ARM8

ADDITIONAL FEATURES10

DESIGN EVALUATION 11

**APPENDIX:**

**(A)** REQUIRED COMPONENTS USED 12

**(B)** WIRELESS CONTROLLER PROGRAM CODE 13

**(C)** ROBOT PROGRAM CODE 16

**(D)** ROBOT AND CONTROLLER BLOCK DIAGRAM 20

**(E)** PARTS LIST 21

**(F)** SCHEMATICS 22

**(G)** FLOW CHART - BOT 23

**(H)** FLOW CHART - CONTROLLER 24

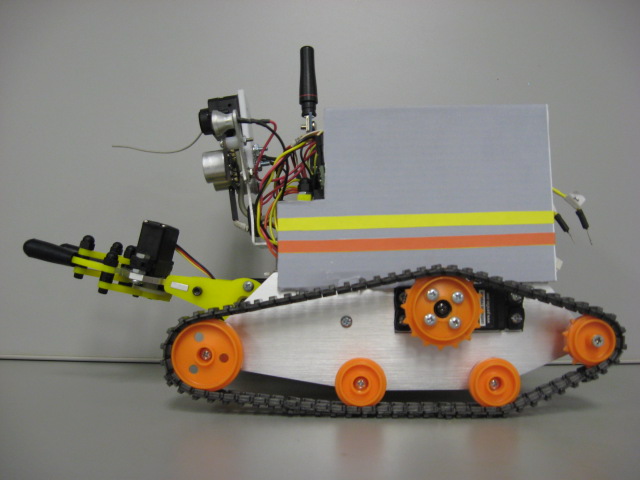
**INSTRUCTION GUIDE** 25

**DESIGN SUMMARY**

For our Mechatronics class, team Rover-Bot was assigned the task of completing a final electronics project. For a list of requirements and what the team incorporated, [see Appendix (A), page 12]. To meet the requirements of the class, team Rover-Bot built a remote control robotic vehicle that has the capability of lifting objects and transporting them to a different location in the room. The principle of operation is based on that of a basic remote control vehicle. Team Rover-Bots project included a drivable robotic vehicle that is controlled by a wireless remote, and has a robotic arm that is attached to the robot. For the project to function as intended, two basic stamps, one AVR microprocessor, and three code programs were incorporated. One basic stamp was used for the robot and one was used for the wireless controller. The AVR Omega microprocessor was used on the robotic arm to provide better control of the up and down motion. A program was written for each based on what was needed to perform the specific operations of the vehicle, wireless remote, and robotic arm.

[See Appendix (B) and(C), pages 13-16 for program codes]

The robot is the main component of the project. It serves the function of creating mobility and contains the majority of the components used. The drive of the robot is obtained from two servo motors which power the tank tracks. A Ping sensor is located on the front for object detection and distance measuring. Above the Ping Sensor is a simple wireless camera, which transfers real time images to a standalone LCD screen. On the back, there are two Infrared Sensors that indicate how close the robot is to objects. If it gets too close while backing up, an attached buzzer on the microprocessor board will sound. There are two LEDs located on the front to function as additional lighting when visibility is low. Additionally, there are two LEDs located in the rear. These indicate which direction the vehicle is turning by flashing each individual side based on the joystick position. Control of the vehicle is achieved by a wireless transceiver which is linked to the wireless controller. There are two LED status indicators which are both green. One indicates power to the robot, and the other is to show that wireless transmission is active. The robot is powered by a 7.2 Volt rechargeable battery pack. Another separate 9 Volt battery provides power to the camera.



***[Final Robotic Vehicle]***

All operations of the robotic vehicle are done through the wireless controller. The setup consists of two joystick controllers. Joystick one provides direction of the vehicle. It is similar to that of standard joysticks, and provides forward, reverse, left and right direction. Joystick two serves two purposes. It provides the lifting and lowering action of the robotic arm, and also controls the opening and closing of the claw. The controller also includes a LCD interface. This provides information about the direction of the vehicle, how close it is to objects, and also an alarm indication when the infrared sensors are tripped. The controller also has a pushbutton to turn on the LED front lights. Data input and output is achieved through a transceiver, and allows communication to the vehicle, and brings back information that displays on the LCD screen. Additionally, there are two LED status indicators. The Red LED comes on when the power switch is turned on. The Green LED will flash to indicate that wireless transmission is active.



***[Final Wireless Controller]***

The robotic arm serves the function of grasping and lifting objects. It will grasp objects in the claw, and can elevate the object off of the ground to allow clearance when moving it to another location. The movement of the arm and claw is supplied by two servo motors. One servo lifts and lowers the arm, while the other servo controls the opening and closing of the claw.



***[Final Robotic Arm]***

In order to complete the project, the build process was broken down into three stages for manageability.

* Stage1 the robotic vehicle
* Stage2 the Wireless Controller
* Stage3 the Robotic Arm

The required components [see Appendix (A), page 12] and parts that were needed [see Appendix (E), page 21] were gathered and stage 1 was started.

**STAGE 1: ROBOT**

The chassis of the robot is based on another robot, the “Boe-Bot”. The aluminum frame seemed to meet the needs of the project, and was implemented as the building block.



***[Boe-Bot Frame]***

The build process was started by mounting the microprocessor, servos, and LED indicator lights onto the Boe-Bot frame. Wires were measured and cut to length and attached to the LEDs and infrared components. The wires for each component were run from the basic stamp input board to the approximate location of mounting.



***[Stage 1 Building]***

The wires were left physically unattached until the positive placement and correct functioning could be determined. The buzzer was plugged in directly to the basic stamp. A basic program was written and ran to test the lights and infrared. The lights functioned as intended, but additional code was implemented on the infrared detectors. This was done in order to get the results that were needed. Instead of just measuring the infrared distance, a program code was implemented to break it into 5 zones. Zone 5 through zone 2 would not have any objective. However, if zone 1 was triggered, the buzzer would sound and shut the servo motors off. After some trial and error on the code, this was achieved. The tank tracks (kit form) were put together, and mounted onto the Boe-Bot Frame.

To attach the ping sensor, a mounting bracket from the lab was implemented. The ping sensor can be directly plugged into the basic stamp, so it made hookup fairly simple. Again, a basic program code was written to allow testing to ensure that the correct distance was being displayed. From previous code experience through a Microprocessor class, this was achieved relatively easy.

The final components were implemented onto the robot. This consisted of the transceiver and indicator light to show that transmission was taking place. The transceiver was plugged into the basic stamp board, but not tested at this time as the team waited until the wireless controller box could be active to allow two way communications.

**STAGE 2: WIRELESS CONTROLLER**

To house the components for the controller, some type of box was needed. Initially, templates were made and Plexiglas was cut. It was decided not to use this idea, due to the amount of time it would require. Instead, a premade project box was purchased from Radio Shack that would be suitable to hold all of the needed components for the wireless controller.



***[Wireless Controller Box Housing]***

For the first step, a circuit was bread boarded for the joysticks. This consisted of the joysticks and a RC time constant to allow measurements based upon the joystick X and Y axis position. Hookup of the joysticks was a little tricky, and documentation from the Parallax website (<http://www.parallax.com/>) was obtained for help. A program code was written that measured the time constant values, which were directly proportional to the X/Y axis position. Our initial approach was to compare the X-axis and Y-axis position. Once these readings were obtained, the RC time constant value could then be sent across wirelessly to the robot. This type of scenario would not work, and caused the robot to behave erratically. With help from our instructor, Vincent Palagi, it was learned that comparison of both the X and Y axis is not necessary to gain directional movement. To achieve joystick position, a simple means was incorporated that based the position of each axis on the center point. For example, if the joystick was pushed forward, and the Y axis value is less than 10, a variable value of “1” was assigned. This was repeated for all directions except neutral. Neutral contains an X/Y axis comparison that gives the joystick a center reference. Once this type of program code was written, the issue was resolved.

Our second step was to get the robot into a drivable state. Since a value for each direction was assigned, the data needed to be sent wirelessly from the controller to the robot. A basic program was written based on the sample program code that was provided with the wireless transceivers. The intended function was to send the variables that represented our direction wirelessly over to the robot. After some modifications to the sample code, this was achieved and the robot was able to move forward, reverse, left and right.

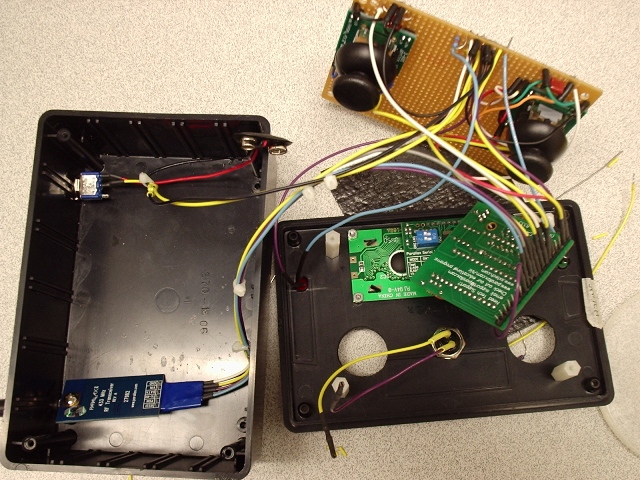
The LCD was added to the bread board controller circuit. The display data needed to show what direction the robot was traveling, data from the Ping sensor and when the infrared detectors were activated. The initial display for the direction was fairly simple. It was approached on the controller side, so this data did not need to be transferred via wireless. Problems arose when trying to send data from the robot to the controller. The current wireless program only provided one way communication. This was due mainly to the order of operation in the program code. Also, the transmitter and receiver code was placed throughout the entire program, which in turn kept the wireless transmission in sync. With the circuit now supplying two way communications, testing of incoming and outgoing data from the Ping and infrared sensors were performed. The infrared sensors were pretty straightforward, as this was sent across wireless as a variable. The Ping sensor showed some issues when displayed on the LCD screen. The wireless data was in ASCII code. In order to get it to display correctly, the BS2 formatter command “DEC” was used.

The physical part of putting together the controller box ensued. A hole was cut, and the LCD was flush mounted on the controller box lid. Holes for the joysticks, pushbutton, and on/off switch were all done with a drill press. The controller circuit was built on a PCB board and soldered together. Holes were drilled at each corner of the board to allow mounting of the board to the underside of the lid.

***[LCD Mounting] [Wireless and On/Off Switch] [Pushbutton and Pwr/Tx Lights]***

The wireless transceiver was mounted on the top of the controller box. The On/Off switch was mounted in place on the top of the controller box. The BS2 Microcontroller was tied in with all of the components. The wires were hooked up based upon our bread boarded circuit. The final assembly was completed, and the box was closed up. Testing of the wireless controller box was completed and it performed as it did on the bread boarded prototype.



***[Final Assembly of Components into Controller Box]***

**STAGE 3: ROBOTIC ARM**

In order to incorporate a robotic arm onto the robot, a Lynx5 Gripper Kit version 1.0 was used. To achieve the arm and gripper motion, the original kit used 2 servo motors and an actuator cable. One servo motor controlled the up and down movement, while the other servo motor (along with the actuator cable) controlled the opening and closing movement of the gripper.



***[Original Gripper Kit Design]***

After the kit was put together, a simple circuit using 4 pushbuttons was built to test the arm and gripper. Two pushbuttons controlled the up and down movement, while the other two pushbuttons controlled the opening and closing of the claw. With the circuit built, the arm motion was tested and found to be operational. While testing the gripper, a problem was found on the gripper part of the assembly. The cable, that was attached to the gripper unit and then to the servo, would not provide reliable operation. The cable would pull the gripper closed, but had trouble pushing the cable to open the gripper back up. To help implement a solution to this problem, customer support from the Lynx website was contacted. Support stated that they were aware of the unreliable operation, and suggested placing some type of spring that would hold the gripper closed. This would help with the operation of closing the gripper. Lynx support also informed me that they incorporated a modified version of the gripper kit, which used a micro servo that performed this function better.

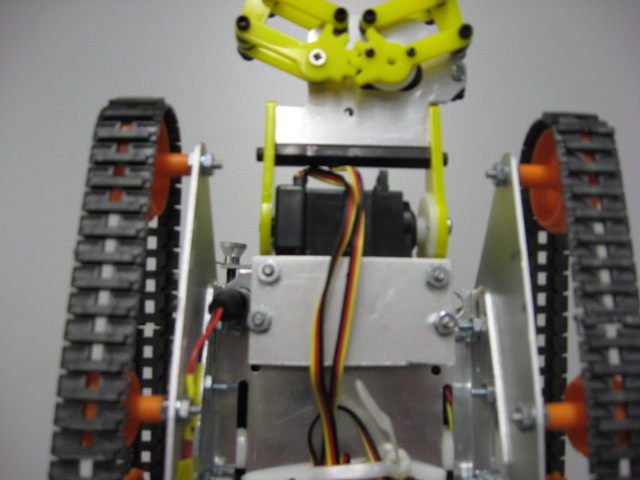
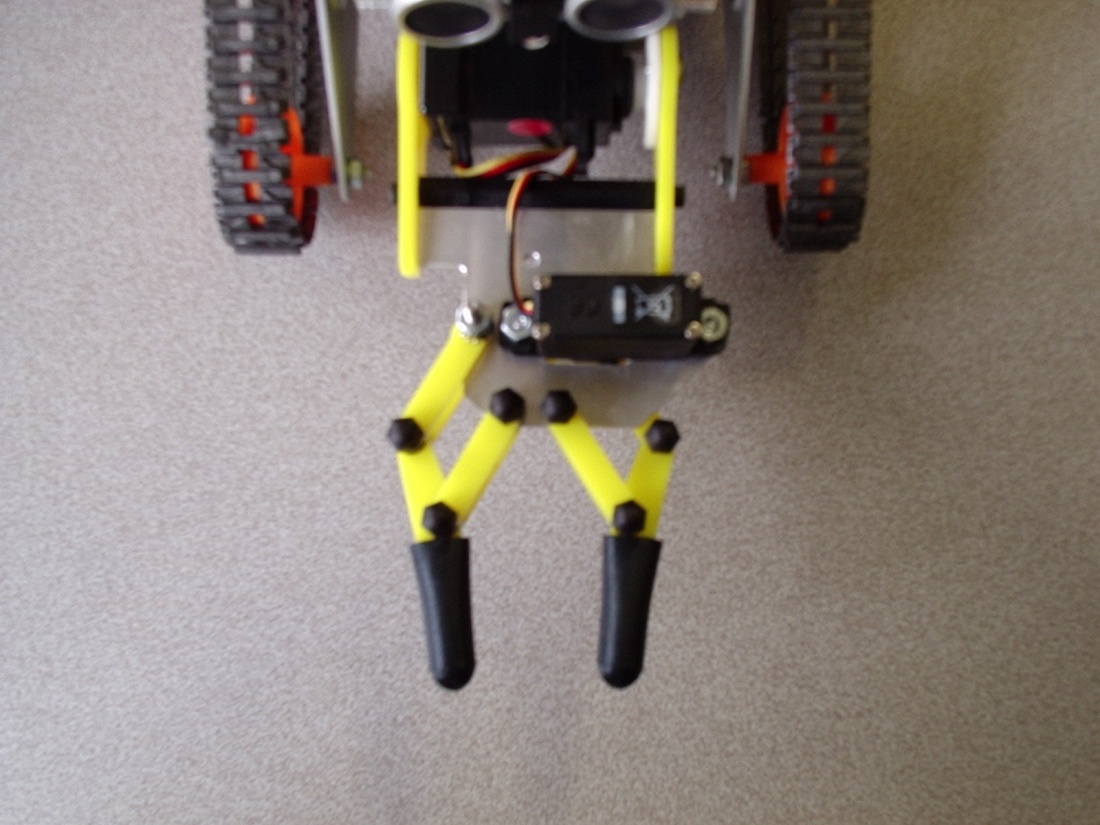
In order to remedy the unreliable operation of the robotic arm, team Rover-Bot wanted to incorporate the design that included the micro servo. In the CWI electronics lab room, there was a full version of the robotic arm that incorporated the micro servo design. Comparing the two designs (cable versus micro servo), it appeared that the only major difference was the base plate that the other parts of the gripper components were attached to.

***[Original Kit, Cable Actuated] [Lab Room Kit, Micro Servo] [Machine Shop Design]***

After analysis, it appeared that we could modify our kit if a similar base plate could be obtained. Our existing components could then be transferred to the new plate and incorporate the micro servo design. The newer version gripper kit in the lab room was disassembled to gain access to the base plate. The newer version base plate was then taken to the CWI Machine Shop where a suitable plate was machined out of aluminum. Using this aluminum plate, along with our original gripper kit components, the modified version was put together and tested. Our modified version provided reliable operation for the gripper movement.

In order to mount the arm onto the robot, team Rover-Bot decided to place the servo that controlled the up and down movement between two pieces of aluminum. Then mount the aluminum plate onto the front underside of the robot.



***[Modified Robotic Arm] [Underside Mounted Arm]***

Programming code was implemented into the main program for the robotic arm. It was tested, but did not perform as expected. The opening and closing of the gripper performed fine, but a problem with the up and down movement was present. The arm would move, but it would not hold any weight. In order to perform as intended, the arm needed to have a continuous pulse present on the servo. This part of the circuit was hooked up to an oscilloscope to look at the pulse. There were large gaps in the pulse. The team first tried to remedy this problem with additional code to pulse out more often to hold the arm in position. This helped to some extent, but the problem of keeping the pulse on the servo was still present. The arm would bounce up and down if any weight was present. After further investigation, the true cause of the problem was found. When using the serial in or serial out command, it disables the pin as an output, and places it into an input state. This drops the pulse to the servo for a brief moment, which caused the loss of pulse.

There were two possibilities that could fix this pulse issue. The first consisted of implementing a servo motor controller that has the capability of holding the pulse independently. The second would be to add another microprocessor to control the arm movement. There was no access to a servo motor controller. Since team Rover-Bot had another microprocessor that was available for use it was decided to incorporate the microprocessor to remedy the issue of pulse sustainability.

The microprocessor that was used for the arm is not the same as the ones that were programmed for the robot and wireless controller. Instead of using BS2 program code an ANSI C code had to be implemented. The code was written alongside the BS2 microprocessor program, and ran separately. Initial testing showed that the pulse problem was fixed, but stepping the servomotor proved to be another problem. Due to the limitations of the microprocessor that was incorporated, it could not be stepped incrementally. Movement of the arm could only be achieved with position control. Therefore, a position control was setup that would allow for five positions. After implementing this code, the arm has a center position and two positions above and below center. Testing of this design showed that the additional microprocessor and code fixed the issue of keeping a pulse on the servo.

**ADDITIONAL FEATURES**

Since a camera was available for use, it was decided to incorporate this to the project design. The camera was mounted on the same bracket that the Ping sensor was attached to. For ease of functionality, a separate 9 Volt battery was implemented onto the back of the Robot to power the camera.

A transport/storage box was purchased to house all of the build components which include:

* Robotic Vehicle
* Wireless Controller
* Robot Charger and 9 Volt batteries
* 7 inch LCD Display Screen
* Video Tuner
* Required Cables for Camera hookup

**DESIGN EVALUATION**

The team’s objective to build a wireless remote control vehicle that has the capability of moving objects to different locations in the room was achieved. While performing the physical part of the testing, here is what the team found:

The Robot

* It was noticed that it was a little difficult to get the robot lined up to our wooden block so that the arm could pick it up. But with practice, this could be achieved.

Wireless Controller

* Performed well in the testing. Although, the distance that the robot is able to travel is limited due to the amount of current needed by the wireless controller.

Robotic Arm

* The arm and gripper performed well. Although, on initial power up, it was a little bit jumpy the first time it was used. It is also limited on how much weight it can pick up.

Camera

* The camera worked as intended. Although the placement could be improved. The placement of the camera allows the user to either look ahead of the vehicle or watch the gripper in action. In order to perform both actions simultaneously, the camera would need to be mounted farther back on the robot or on one of the sides to allow for a wider field of view.

Improvements that could be made:

* Implement DC motors for two reasons-

1. Faster speed

2. Better control when lining up to an object (when the arm is being used)

* Use a rechargeable battery for the wireless controller with a high “mAH” to allow further distance between the robot and wireless controller
* Heavier frame to allow the arm to pick up more weight
* Place the camera either to one side or farther back on the frame to allow for a wider view

The project met the objective and implemented all of the required components along with two additional. The physical part of moving an object to another location was achieved, therefore, the Mechatronics Project by team Rover-Bot has been deemed a success.

**APPENDIX:**

**(A)** REQUIRED COMONENTS USED

1. Display Device

* Implemented: LCD Display Screen, Led’s
* Additional implementations: Camera and 7inch LCD screen

1. Audio Output Device

* Implemented: Buzzer

1. Manual Data Input used

* Implemented: Joystick, Pushbutton, Switches

1. Automatic Sensor// Data Input

* Implemented: Ping sensor, Infrared Sensors, RC Circuit

1. Actuators used

* Implemented: Servo motors

1. Logic, Counting, and Control

* Implemented: Programmable Logic (BS2 code)
* Additional implementation: ANSI C code

1. vii. Controller

* Implemented: PIC (Basic Stamp Microprocessor)
* Additional implementation: ATMega 8 Microprocessor

**(B)** WIRELESS CONTROLLER PROGRAM CODE

' {$STAMP BS2}

' {$PBASIC 2.5}

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'\* Program: RC-Control.BS2 Author: Team Rover-Bot

'\* Date: 5/08/2010 Revision: 13

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'This program is the control program for Rover-Bot Wireless Control Box.

' --------[ I/O Definitions ]------------------------------------------------------------------------

Joystick\_LR0 PIN 0 'Joystick, Left/Right, Left

Joystick\_UD0 PIN 1 'Joystick, Up/Down, Left

Joystick\_LR1 PIN 2 'Joystick, Left/Right, Right

Joystick\_UD1 PIN 3 'Joystick, Up/Down, Right

LCD PIN 4 '16x2 Serial LCD Pin.

Tx PIN 5 'Transmitter(27982)DATA pin

TxEnable PIN 6 'Transmitter(27982)TR pin

PowerDown PIN 7 'Wireless Sleep = 0

WiFi PIN 8 'Wireless Data TX=1 RX=0

PB0 PIN 9 '0=None, 1=Increment

' --------[ Variables ]------------------------------------------------------------------------------

PB1 VAR Bit 'Lights, Pushbutton, Normally Low

Counter VAR Nib 'Enables Light Control

distanceLeft VAR Nib 'Right Distance, Value

distanceRight VAR Nib 'Left Distance, Value

Left\_X VAR Byte 'Stores RC Value, Left X Joystick

Left\_Y VAR Byte 'Stores RC Value, Left Y Joystick

Right\_X VAR Byte 'Stores RC Value, Right X Joystick

Right\_Y VAR Byte 'Stores RC Value, Right Y Joystick

Pos VAR Byte 'Position of Joystick

Distance VAR Word 'Ping Measurement

' --------[ Constants ]------------------------------------------------------------------------------

-----

T2400 CON 396 'Baud 2400, N

T9600 CON 84 'Baud 9600, N

T19K2 CON 32 'Baud 190K, N

LcdBaud CON T19K2 'Baud LCD

LcdBkSpc CON $08 'move cursor left

LcdRt CON $09 'move cursor right

LcdLF CON $0A ‘move cursor down 1 line

LcdCls CON $0C 'clear LCD (use PAUSE 5 after)

LcdCR CON $0D 'move pos 0 of next line

LcdBLon CON $11 'backlight on

LcdBLoff CON $12 'backlight off

LcdOff CON $15 'LCD off

LcdOn1 CON $16 'LCD on; cursor off, blink off

LcdOn2 CON $17 'LCD on; cursor off, blink on

LcdOn3 CON $18 'LCD ON; cursor ON, blink off

LcdOn4 CON $19 'LCD on; cursor on, blink on

LcdLine1 CON $80 'move to line 1, column 0

LcdLine2 CON $94 'move to line 2, column 0

' --------[ Initialization ]-------------------------------------------------------------------------

HIGH PowerDown 'Wakeup Wireless

PAUSE 200 'Wait, Before Sending Data

' --------[Set Values ]------------------------------------------------------------------------------

-----

Left\_X = 0 'Set Left X = 0

Left\_Y = 0 'Set Left Y = 0

Right\_X = 0 'Set Right X = 0

Right\_Y = 0 'Set Right Y = 0

Counter = 0 'Light Control = 0

' --------[ Main Program ]---------------------------------------------------------------------------

Main:

GOSUB DataRx 'Receive Data

GOSUB Joystick 'RC, Joysticks

IF Left\_X < 30 OR Left\_X > 20 OR Left\_Y < 40 OR Left\_Y > 25 THEN GOSUB Neutral ' Hold: Neutral

IF Left\_Y < 10 THEN GOSUB Backward 'Hold: Backward

IF Left\_Y > 35 THEN GOSUB Forward 'Hold: Forward

IF Left\_X < 15 THEN GOSUB Left 'Hold: Left

IF Left\_X > 40 THEN GOSUB Right 'Hold: Right

IF PB0 = 1 THEN PB1 = PB1 + 1 'Pushbuttons

IF Right\_Y > 40 THEN GOSUB Up 'Move: Arm Up

IF Right\_Y < 20 THEN GOSUB Down 'Move: Arm Down

IF Right\_X > 40 THEN GOSUB Open 'Claw: Open

IF Right\_X < 20 THEN GOSUB Close 'Claw: Close

GOSUB DataTx 'Do Not Run Code Below TX

GOTO Main

' --------[Subroutines ]-----------------------------------------------------------------------------

DataRx:

LOW WiFi 'LED RX=0

LOW TxEnable 'Enable Receiver

SERIN Tx, 16572, [WAIT("DataTx"), DEC3 Distance, DEC1 distanceLeft, DEC1 distanceRight ] 'ReceivesData

HIGH WiFi 'LED RX=1

PAUSE 10 'Sync Pulse

RETURN

DataTx:

LOW WiFi 'LED TX=0

HIGH TxEnable 'Enable Transmitter

PULSOUT Tx,1200 'Send Sync Pulse to Radio

SEROUT Tx, 16572, ["DataRx",DEC Pos, DEC PB1] 'Sends Data

HIGH WiFi 'LED TX=1

PAUSE 10 'Sync Pulse

RETURN

Joystick:

HIGH Joystick\_UD0 'Set PIN High

RCTIME Joystick\_UD0, 1, Left\_X 'Set RC Timeconstant into Left-X

PAUSE 2 'Wait

HIGH Joystick\_LR0 'Set PIN High

RCTIME Joystick\_LR0, 1, Left\_Y 'Set RC Timeconstant into Left-Y

PAUSE 2 'Wait

HIGH Joystick\_UD1 'Set PIN High

RCTIME Joystick\_UD1, 1, Right\_X 'Set RC Timeconstant into Right-X

PAUSE 2 'Wait

HIGH Joystick\_LR1 'Set PIN High

RCTIME Joystick\_LR1, 1, Right\_Y 'Set RC Timeconstant into Right-Y

PAUSE 2 'Wait

RETURN

Neutral:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls, LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLon, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 0 'Position Value

RETURN

Forward:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Forward"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 1 'Position Value

RETURN

Backward:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Backward"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

IF distanceLeft < 2 THEN GOSUB Error 'IR tripped

IF distanceRight < 2 THEN GOSUB Error 'IR tripped

Pos = 2 'Position Value

RETURN

Left:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Left"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 3 'Position Value

RETURN

Right:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Right"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 4 'Position Value

RETURN

Up:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral,Arm Up"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 5 'Position Value

RETURN

Down:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral,Arm Down"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 6 'Position Value

RETURN

Open:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral,A Open"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 7 'Position Value

RETURN

Close:

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral,A Close"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Ping:",DEC3 Distance ] 'Sends text to LCD

Pos = 8 'Position Value

RETURN

Error:

PAUSE 5

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls,LcdOn1,LcdLine1] 'Initializes LCD

SEROUT lcd, LcdBaud, ["Neutral"] 'Sends text to LCD

SEROUT lcd, LcdBaud, [LcdBLoff,LcdCls, LcdOn1,LcdLine2] 'Initializes LCD

SEROUT lcd, LcdBaud, ["IR Detected!"] 'Sends text to LCD

RETURN

GOTO Main

**(C)** ROBOT PROGRAM CODE

' {$STAMP BS2}

' {$PBASIC 2.5}

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'\* Program: RC-BOT.BS2 Author: Team Rover-Bot \*

'\* Date: 5/08/2010 Revision: 13 \*

'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

'--------[ Program Description ]--------------------------------------------------------

'This program is the control program for the Rover-Bot Tracked Vehicle.

' --------[ I/O Definitions ]-----------------------------------------------------------

LfIrOut PIN 0 'Left IR LED, Output

LfIrIn PIN 1 'Left IR Sensor, Input

RtIrOut PIN 2 'Right IR LED, Output

RtIrIn PIN 3 'Right IR Sensor, Input

TxEnable PIN 4 'Transmitter(27982)TR PIN

Tx PIN 5 Transmitter(27982)DATA pin

Ping PIN 6 'Ping Sensor

ALeft PIN 7 'Activate Left Anode

CRight PIN 8 'Activate Right/ Activate Left Cathode

ARight PIN 9 'Activate Right Anode

Lights PIN 10 'Lights, Front

Speaker PIN 11 'Buzzer, Output

Servo\_AB PIN 12 'Left + Right Servos

Servo\_C PIN 14 'Up/Down Servo

Servo\_D PIN 15 'Left/Right Claw Servo

' --------[ Constants ]-----------------------------------------------------------------

Zone CON 4000 'Buzzer Frequency

T2400 CON 396 'Baud Rate 2400

T9600 CON 84 'Baud Rate 9600

T19K2 CON 32 'Baud Rate 12000

Trigger CON 5 'trigger pulse = 10 uS

Scale CON $200 'raw x 2.00 = uS

LcdBaud CON T19K2 'baud rate for LCD

LcdBkSpc CON $08 'move cursor left

LcdRt CON $09 'move cursor right

LcdLF CON $0A 'move cursor down 1 line

LcdCls CON $0C 'clear LCD (use PAUSE 5 after)

LcdCR CON $0D 'move pos 0 of next line

LcdOff CON $15 'LCD off

LcdLine1 CON $80 'move to line 1, column 0

LcdLine2 CON $94 'move to line 2, column 0

inscale CON 872 'scale factor for inches at 2000 feet

' --------[ Variables ]-----------------------------------------------------------------

PB1 VAR Bit 'Enables Light Control

irDetectLeft VAR Bit 'Temporary Storage for IR Left

irDetectRight VAR Bit 'Temporary Storage for IR Right

distanceLeft VAR Nib 'Right Distance, Value

distanceRight VAR Nib 'Left Distance, Value

freqSelect VAR Nib 'Value of Frequency

Pos VAR Byte 'Position of Joystick Value

irFrequency VAR Word 'Temporary Storage for Frequency

DispNote VAR Word 'Count for FREQOUT

Distance VAR Word 'Ping Distance Measurment

ServoCountA VAR Word 'Servo Counter for Servo C

ServoCountB VAR Word 'Servo Counter for Servo D

' --------[ Initialization for Servos and Ping ]----------------------------------------

FREQOUT Speaker, 500, Zone 'Send a Test Note on Powerup

INPUT Servo\_AB 'Make Sure Servos are Not Moving.

DO : LOOP UNTIL Servo\_AB 'Wait for ServoPAL to Power Up.

LOW Servo\_AB 'Set Pin, and Hold

PAUSE 100 '100mS Reset, ServoPAL.

HIGH Servo\_AB 'Raise PIN

PAUSE 200 'Wait for PIN, Active.

ServoCountA = 670 'Set Value to Servo C

ServoCountB = 700 'Set Value to Servo D

' --------[ Main Program ]--------------------------------------------------------------

Main:

SEROUT Servo\_C, 84, ["C"] 'Send "C" to Hold Servo

GOSUB IR ‘Read IR Leds, Store

GOSUB DataTx 'Tranmit Data

GOSUB DataRx 'Receive Data

GOSUB PingOut 'Read Ping Value, Store

GOSUB Move 'Move Robot

IF PB1 = 1 THEN GOSUB Light 'Activate Lights PB1 = 1

IF PB1 = 0 THEN GOSUB NoLight 'Deactivate Lights PB1 = 0

GOTO Main

' --------[Subroutines ]----------------------------------------------------------------

DataTx:

HIGH TxEnable 'Enable Transmitter

PULSOUT Tx,1200 'Send Sync Pulse to Radio

SEROUT Tx, 16572, ["DataTx",DEC3 Distance,' Sends Ping

DEC1 distanceLeft,DEC1 distanceRight] 'Sends Left + Right IR

PAUSE 10 'Wait

RETURN

DataRx:

LOW TxEnable 'Enable Receiver

SERIN Tx, 16572, [WAIT("DataRx"),DEC1 Pos,DEC PB1] ' Receives Data

PAUSE 10 'Wait

RETURN

IR:

distanceLeft = 0 'Set IR Zone 5

distanceRight = 0 'Set IR Zone 5

FOR freqSelect = 0 TO 4 'Select Frequency

LOOKUP freqSelect,[37500,38250,39500,40500,41500], irFrequency 'Table Lookup

FREQOUT LfIrOut,1,irFrequency 'IR Left, Transmit

irDetectLeft = IN1 'Store Distance

FREQOUT RtIrOut,1,irFrequency 'IR Right, Transmit

irDetectRight = IN3 'Store Distance

distanceLeft = distanceLeft + irDetectLeft 'Sum Values, Left

distanceRight = distanceRight + irDetectRight 'Sum Values, Right

NEXT

RETURN

PingOut:

PULSOUT Ping, 5 'Set PIN 15 for 10uS

PULSIN Ping, 5, Distance 'Wait for Pin to Low

Distance = Distance \*\* inscale 'Scale Inches Down

PAUSE 100 'Wait

RETURN

Move:

IF Pos = 0 THEN GOSUB Neutral 'Robot: Neutral

IF Pos = 1 THEN GOSUB Forward 'Robot: Forward

IF Pos = 2 THEN GOSUB Backward 'Robot: Backward

IF Pos = 3 THEN GOSUB Right 'Robot: Right

IF Pos = 4 THEN GOSUB Left 'Robot: Left

IF Pos = 5 THEN GOSUB Up 'Robot: Claw Up

IF Pos = 6 THEN GOSUB Down 'Robot: Claw Down

IF Pos = 7 THEN GOSUB Open 'Robot: Claw Open

IF Pos = 8 THEN GOSUB Close 'Robot: Claw Close

RETURN

Neutral:

PULSOUT Servo\_AB, 2000 'Turn Right Servo Off.

PULSOUT Servo\_AB, 2000 'Turn Left Servo Off.

RETURN

Forward:

DO

PULSOUT Servo\_AB, 1000 'Right Servo: Forward

PULSOUT Servo\_AB, 500 'Left Servo: Forward

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 1 'Loop Position = 1

RETURN

Backward:

DO

IF distanceRight = 0 THEN GOSUB Buzz 'IR Right Zone 0, Buzz

IF distanceRight = 1 THEN GOSUB Buzz 'IR Right Zone 1, Buzz

IF distanceLeft = 0 THEN GOSUB Buzz 'IR Left Zone 0, Buzz

IF distanceLeft = 1 THEN GOSUB Buzz 'IR Left Zone 1, Buzz

PULSOUT Servo\_AB, 500 'Right Servo: Reverse

PULSOUT Servo\_AB, 1000 'Left Servo: Reverse

GOSUB IR 'Read IR Leds, Store

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 2 'Loop Position = 2

RETURN

Right:

DO

PULSOUT Servo\_AB, 500 'Right Servo: Reverse

PULSOUT Servo\_AB, 500 'Left Servo: Forward

LOW ALeft 'Turn off Left Blinker

HIGH ARight 'Turn on Right Blinker

LOW CRight 'Common Ground

PAUSE 10 'Wait

LOW ALeft 'Turn off Left Blinker

LOW ARight 'Turn off Right Blinker

LOW CRight 'Common Ground

PAUSE 10 'Wait

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 3 'Loop Position = 3

RETURN

Left:

DO

PULSOUT Servo\_AB, 1000 'Right Servo: Forward

PULSOUT Servo\_AB, 1000 'Left Servo: Reverse

HIGH ALeft 'Turn on Left Blinker

LOW ARight 'Turn off Right Blinker

LOW CRight 'Common Ground

PAUSE 10 'Wait

LOW ALeft 'Turn off Left Blinker

LOW ARight 'Turn off Right Blinker

LOW CRight 'Common Ground

PAUSE 10 'Wait

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 4 'Loop Position = 4

RETURN

Up:

DO

SEROUT Servo\_C, 84, ["0"] 'Add Count, MAX 4

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 5 'Loop Position = 5

RETURN

Down:

DO

SEROUT Servo\_C, 84, ["5"] 'Subtract Count, MIN 0

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 6 'Loop Position = 6

RETURN

Open:

DO

IF POS = 7 THEN ServoCountB = ServoCountB + 10 'Increments servo

ServoCountB = ServoCountB MAX 830 'Sets MAX 830

PULSOUT Servo\_D , ServoCountB 'Moves Servo

PAUSE 5 'Pause

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 7 'Loop Position = 7

RETURN

Close:

DO

IF POS = 8 THEN ServoCountB = ServoCountB - 10 'De-Increments servo

ServoCountB = ServoCountB MIN 600 'Sets MIN 600

PULSOUT Servo\_D , ServoCountB 'Moves Servo

PAUSE 5 'Pause

GOSUB DataTx 'Sends DATA

GOSUB DataRx 'Receives DATA

LOOP WHILE Pos = 8 'Loop Position = 8

RETURN

Light:

HIGH Lights 'Turn on Front Lights

RETURN

NoLight:

LOW Lights 'Turn off Front Lights

RETURN

Buzz:

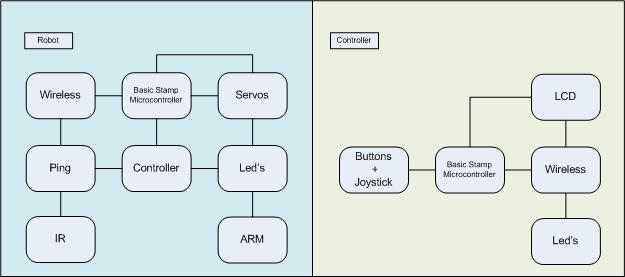
FREQOUT Speaker, 500, Zone 'Send Alarm Signal

PULSOUT Servo\_AB, 2000 'Turn Right Servo Off.

PULSOUT Servo\_AB, 2000 'Turn Left Servo Off.

GOTO Main 'Return to Top

**(D)** ROBOT AND CONTROLLER BLOCK DIAGRAM



**(E)** PARTS LIST

Qty Value            Device           Parts

4   0.1uF            C-US025-060X050  C1, C2, C3, C4

3   1K               R-US\_0207/2V     R6, R7, R8

2   10M              TAC\_SWITCHPTH    S1, S2

1   16x2 LCD         16X2\_SER         U3

7   220              R-US\_0207/2V     R1, R2, R3, R4, R5, R10, R11

3   330              R-US\_0207/2V     R9, R12, R13

1   BS2              M12PTH           U5

1   BS2              M18\_             U4

1   Green            LED5MM           D1

2   IR               LED5MM           D5, D6

1   IR-L             M03PTH           U13

1   IR-R             M03PTH           U12

2   JOYSTICK-PSP1000 JOYSTICK-PSP1000 U1, U2

1   KSS1201          KSS1201          SP1

1   Ping             M03PTH           U8

3   Red              LED5MM           D4, D7, D8

1   Servo AB         M03PTH           U9

1   Servo C          M03PTH           U10

1   Servo D          M03PTH           U11

1   WiFi             M04PTH           U7

1   WiFi             M05PTH           U6

2   Yellow           LED5MM           D2, D3

1   BOT              Tank Tread Kit   X

1   BOT              Bot-Body         X

3   BOT              Standard Servo   X

1   BOT              7.2V Battery     X

1   BOT              AVR-ATMEGA 8     X

35  BOT              Wires-20G        X

1   BOT              Servo Pal        X

1   BOT              Plastic Hous.    X

1   BOT              Ping Brkt.       X

1   BOT              Camera           X

1   BOT              ARM-DS22         X

1   BOT              Mount Plate      X

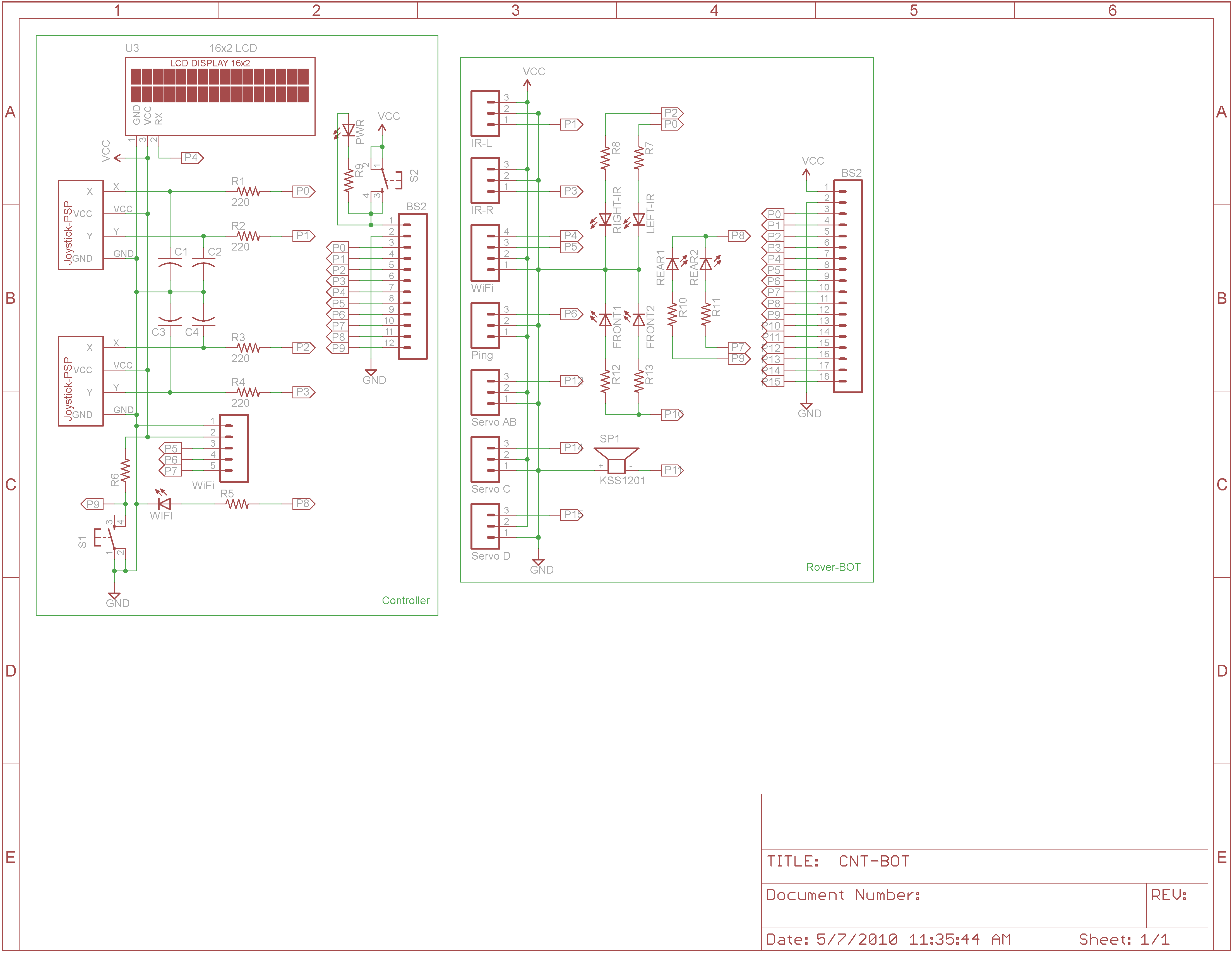
1   BOT              Micro Servo      X

20  BOT              Screws           X

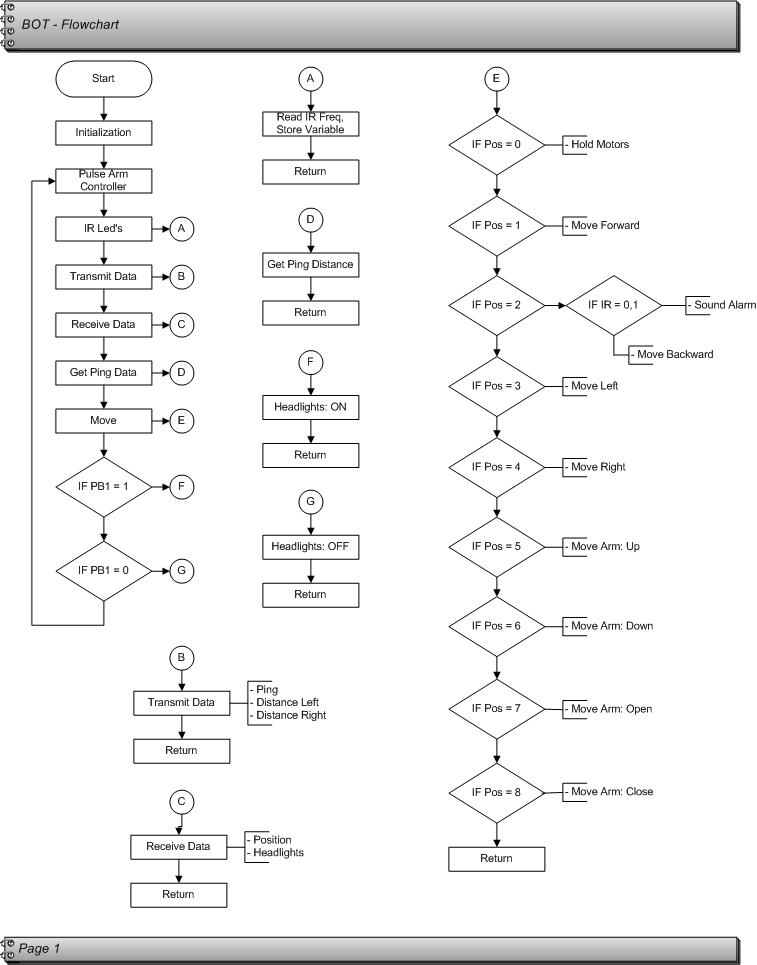
1   CONTRL           Housing          X

1   CONTRL           16x2 LCD         X

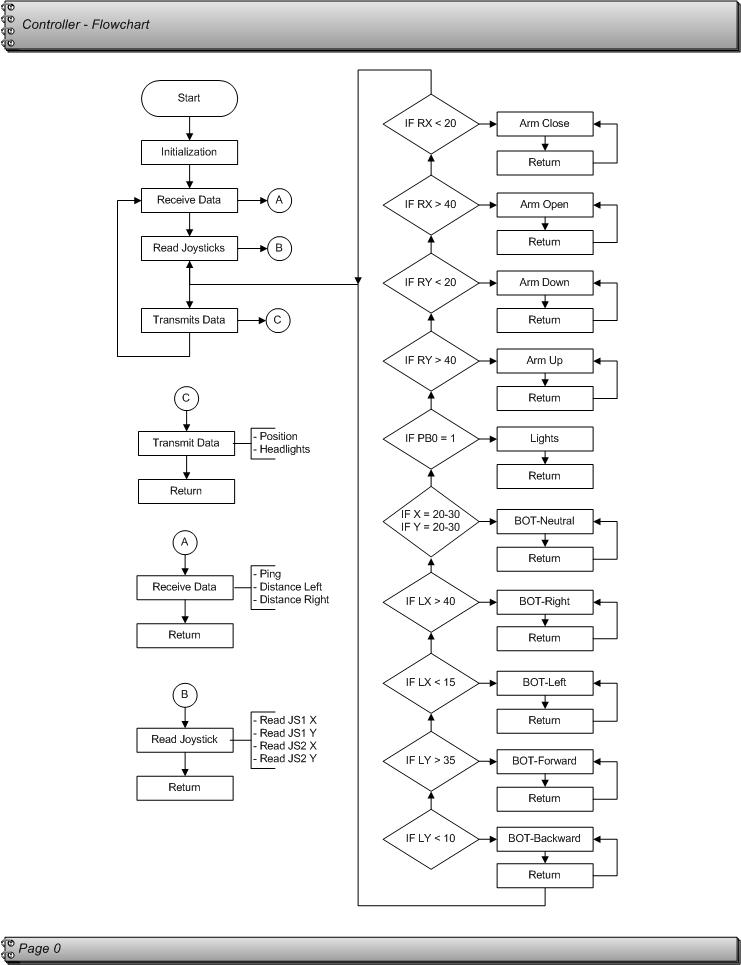
**(F)** SCHEMATICS

****

**(G)**FLOW CHART - BOT

****

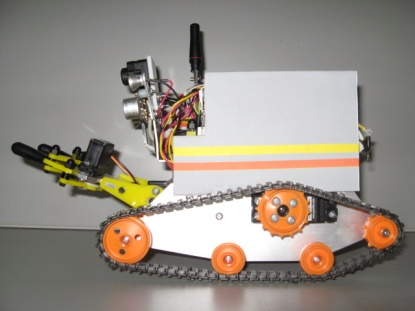
**(H)**FLOW CHART - CONTROLLER



**INSTRUCTION GUIDE**

INTRODUCTION-Before getting started, let’s take a brief tutorial on the robot control and functions, as well as what is needed to prepare the robotic vehicle for operation.

Contents and Parts Identification:



[ROBOT] [CONTROLLER] [WALL TRANSFORMER CHARGER]

Control of the robot is done through the wireless controller (refer to figure 1 below to familiarize yourself with the controls).



[FIGURE 1]

Joystick 1(left side) controls the movement of the robot. It has the capability of moving forward, reverse, left and right. Joystick 2(right side) controls the movement of the robotic arm. It has the capability of grasping and lifting objects. Arm movement includes up, down, open and close. The LCD display interface on the wireless controller brings data over from the robot and will output it on the screen. The LCD has two lines available for visual data. The following data will show up on the LCD screen:

-***Direction*** of the vehicle based on the joystick 1 position. The actual data displayed will be “Neutral, Forward, Reverse, Left and Right”. This will be displayed on line 1, and is dependent upon joystick 1 position.

-***Ping sensor distance*** (measures distance from objects). The actual data displayed will be

“Ping: xxx”, where x=distance in inches. This will be displayed on line 2.

-***Infrared sensor warning*** (only displays when the robot is too close to an object when reverse is engaged, and will shut off the servo motors). The actual data displayed will be “IR Detected!”. This will be displayed on line 2.

-***Positioning of the robotic arm***. The actual data displayed will be “Arm Up” or “Arm Down” and is dependent upon joystick 2 position. This will be displayed on line 1 in conjunction with the direction data.

-***Positioning of the claw.*** The actual data displayed will be “A Open” or “A Close” and is dependent upon joystick 2 position. This will be displayed on line 1 in conjunction with the direction.

The red pushbutton (located between the two joysticks) will operate the headlights that are located on the front of the robot. Pushing the button once, turns the lights on and pushing the button a second time will turn the lights off. The two LED’s that are located on the rear of the robot act as turn signals, and will light up independently when the joystick 1 left or right direction is engaged.

Additionally, there are two LED status indicators on the wireless controller, and two for the robot.

Controller: The first LED is green and is labeled “Tx”. This will flash to show that wireless communication is being achieved. The second Led is red and is labeled “Pwr”. When the power is switched on, it will stay on continuously.

Robot: The robot vehicle also has LED Tx and Pwr indicators, which are both green. The “Tx” is located on the top of the plastic housing, and will flash when the wireless is active. The “Pwr” is located on the microprocessor that is attached to the tracked vehicle frame (under the plastic housing) and will stay on continuously when the vehicle is powered on.

The camera will need additional hardware to operate. See **Section 2** for additional instructions.

The robotic arm is limited to picking up items that are less than 8 ounces. Trying to pick up more weight than this is not recommended as damage to the robot may occur.

**To prepare your robot for operation, see SECTION 1: Battery installation/removal and *charging* and SECTION 2: Preparing the Robot and Controller for use. Additional troubleshooting tips are included in Sections 3, 4, and 5.**

**SECTION 1: Battery installation/removal and charging**

* **STEP 1.1** Controller Battery (2- 9V Battery)

**a.** Remove 4 outer screws.



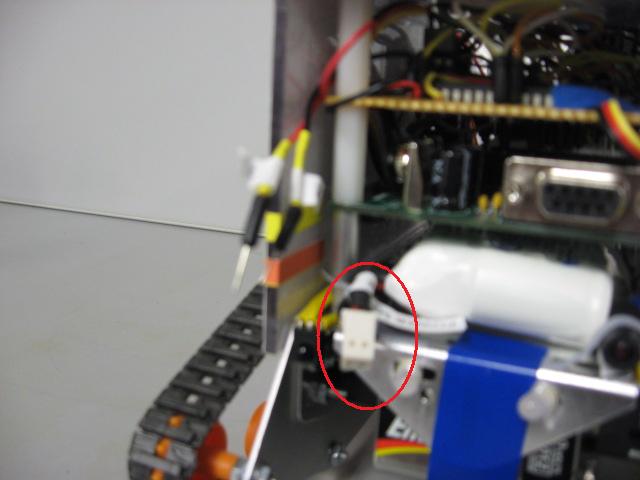
**b.** Carefully remove top cover.

**c.** Locate battery plug-ins, install batteries.

**d.** Replace top cover and tighten screws.

* **STEP 1.2**  Robot Battery (1-rechargeable 7.2V)

**a.** Locate the charging receptacle located in the rear of the Robot. It is labeled Power Source/Charging



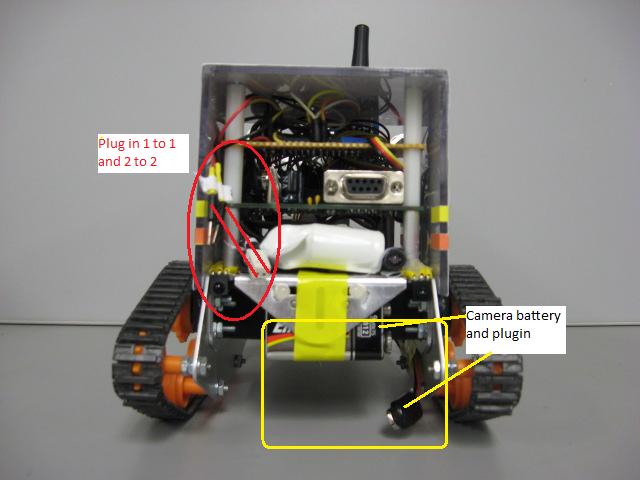
**b.** Using the included wall transformer, plug it into an AC outlet and charge the rechargeable battery by plugging it into the robot.

\*Note- This usually takes about 2 hours to fully charge.

**c.** Unplug the wall transformer from the robot and wall receptacle and put it away.

**d.** Locate the two wires near the charging receptacle of the robot and plug these two wires into the charging receptacle of the robot.

\*Note-The two wires and the charging receptacle are labeled “1” and “2”. Plug these in with the corresponding number (“1” into “1”, and “2” into “2”).



* **STEP 1.3** Camera Battery (1-9V Battery)-Optional

a. Locate and install a battery for the camera. (See above photo) See Section 2 for additional instructions.

\*Note-This battery is optional and is for the camera only. The robot and controller can be operated without this battery.

**SECTION 2: Preparing the Robot and Controller for use**

\*Note: The Controller switch must be turned on first to activate the wireless correctly.

* **STEP 2.1** Locate On/Off switch on Controller Box and turn to the on position.



* **STEP 2.2** Locate On/Off switch on Robot and turn to the on position.

\*Note-switch must be in position 2 for the robot to function.



* **STEP 2.3** Ensure that the wireless LED indicators (TX) on both the controller and robot are flashing.

\*Note-see troubleshooting sections 3 and 4 if LEDs are not flashing.

* **STEP 2.4 Step 2.4 is optional and can be skipped if the camera on the robot will not be implemented.** For camera operation, locate the power cords, 7 inch LCD, tuner, and cables. Make the connections for the cables, numbers 3 through 9. Match numbers to identical numbers. (i.e. Plug in 3 to 3). When these connections are complete, plug in the LCD power cord and the tuner power cord. Ensure that a 9V battery for the camera is plugged in on the robot (see step 1.3). Raise the “Tuners” antenna into a vertical position (does not pull out, just turn up). Turn on the LCD and adjust “Tune” knob until the camera is working. Proceed to step 2.5
* **STEP 2.5** Robot is ready to operate.

\*Note-to familiarize yourself with the controls; please refer to figure 1 in the introduction section.

**SECTION 3:Troubleshooting Controller**

A. Red Pwr LED light on controller box not on:

-No power is present to the controller. Check the battery for loose connections and/or replace the battery if necessary.

B. Green Tx LED light on wireless controller box not flashing:

-Wireless transmission has been lost or dropped, and/or robot has lost power. Ensure robot and wireless batteries are good. If batteries are ok, locate the reset switch on the robotic vehicle *through the window* (next to on/off switch) and push to reset the robot. Refer to the following Figure 2 for example reset switch.



[Figure 2] Example reset switch location on microprocessor board

**SECTION 4: Troubleshooting Robot**

A. Green PWR LED light on robot not on (located under the plastic housing on the microprocessor):

-No power is present to the robot. Check battery and charge if necessary.

B. Green TX LED light on robot not flashing (located on the top of the plastic housing near the rear of the robot):

-Wireless transmission has been lost or dropped, and/or robot has lost power. Ensure robot and wireless battery are good. If batteries are ok, locate reset switch on the robotic vehicle through the window and reset the robot.

**SECTION 5: Troubleshooting Camera**

A. No picture on 7 inch LCD:

-Ensure camera batter is fully charged.

-Verify all connections on cables, LCD, and tuner.

-When adjusting “tuner” knob, increment slowly to dial in.

-Check to ensure antennae has been raised into a vertical position.

**ENDNOTES:**

To reprint this manual or for other robot documentation, please see the CD located in the robot storage box.

