Project report Implementing, Testing and Debugging an A4WD1 Rover

Team Members:

Rohit Bindal, Prashant Vakte, Prateek Jadhwani, Mohit Patel, Santosh Paudel, Labin Tamang, Mohan Paudel, Nischal Buddhathoki, Sheikh Ali Tasfin (Team leader), AKM Saiful Islam, Saju Shreshtha, Sarita Medhekar

Coordinator: Dr. Stefan Andrei

Lamar University
Department of Computer Science
Beaumont, Texas

Abstract

Embedded systems can perform special defined tasks with real-time computing constraints. The program instructions written for embedded systems are referred to as firmware, and are stored in Read Only Memory (ROM). By integrating software and hardware components it is easy to solve a programming logic problem with a programmable electronic device. This paper describes how we implemented and examined the development of an embedded system, namely the A4WD1 Rover. It is a robot manufactured by Lynxmotion, Inc., a U.S. company specialized in designing a great variety of programmable embedded systems, especially robots. A powerful Bot Board/SSC-32/BA28 and a PS2 Controller are attached to the rover to simulate the problem. Its mission is to detect the obstacles, avoid them and make the right path for further works. The very same design is used for robots which can also works on dessert area, on moon and any rough surface. While working on this project, our team experienced certain challenges during implementation phase. Our project has a high complexity, similar to the earlier projects such as the Mini-hexapod, the Johnny-5 in terms of movements and functionality.

Acknowledgement

We would like to give our gratitude and express our utmost and sincere appreciation to our course instructor Dr. Stefan Andrei for his guidance and support for his instructions and continuous support for the different phases. This project has definitely helped us to develop our programming skills as well as real time software development with the embedded system and robotics.

Table of contents

Contents

1. Introduction

An Embedded System is a specialized computer system that is part of a larger system or a machine. An embedded system is implementing on a single microprocessor board with the help of some programs, stored in ROM. Many appliances likes watches, microwaves, Video Camera Recording (VCR), cars are utilizing embedded systems. Some embedded systems also include an operating system, but many are so specialized that the entire logic can be implemented as a single program.

The Lynxmotion Aluminum 4WD1 Robot Kit is a robust, modifiable, and expandable for an autonomous robot experimentation. The robot may use Radio Control (RC) truck tires and wheels, so the robot has excellent traction. There are also small NiMH battery packs and the Sabertooth 2x10 R/C motor controller. It has an additional deck that can be added on the top for future expansion. On the deck it is easy to implement project requirements. The robot is capable of carrying up to a 5lb load.

The robot is made from heavy-duty anodized aluminum structural brackets and ultra-tough laser-cut Lexan panels. It includes four 12.0vdc 30:1 gear head motors and four 4.75" tires and wheels. This version of the robot uses Bot Board, Basic Atom Pro and three GP2D12 distance sensors for obstacle detection and avoidance.

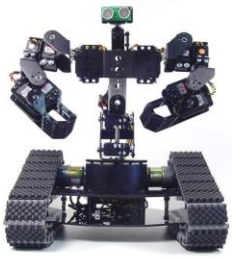
2. Related Work

There exist many robots useful in our modern society. Many companies provide implementation of robots with specific goals. Lynxmotion, Inc. is one example of a company that provides small and medium variety of robots, which can be manipulated using a remote control or are autonomous.

The A4WD1 v2 Rover

Our project implemented one of the robots from Lynxmotion **A4WD1 v2 Rover**. **A4WD1 v2 Rover** is a robust, modifiable, and expandable chassis for our Remote Control or autonomous rover experimentation. The **A4WD1 v2 Rover** utilizes popular RC truck tires and wheels the robot helping it to achieve excellent traction. The Rover uses small NiMH battery packs and the Saber tooth 2x10 R/C motor controllers hence leaving plenty of room inside for additional electronics. Rover is capable of carrying up to a 5lb load.
Rover chassis is made from heavy-duty anodized aluminum structural brackets and ultra tough laser-cut Lexan panels. It includes four 12.0vdc 30:1 gear head motors and our 4.75" tires and wheels. Rover has sensors attached to both front and back parts in top panel. Current program allows rover to detect collision in both front and back ends by help of the attached sensors.

The Johnny 5 Robot



Another complex robot provided by Lynxmotion Inc. is Johnny 5. The **Johnny 5 Robot** is made from Servo Erector Set aluminum brackets, custom injection molded components, and ultra-tough laser-cut Lexan structural components. The torso is fully articulated utilizing 8 x HS-645MG, 3 x HS-475HB / HS-485HB, and 3 x HS-422 servos, and our SSC-32 servo controller. By utilizing heavy duty polypropylene and rubber tracks with durable ABS molded sprockets the robot has excellent traction. It includes two 12vdc 50:1 gear head motors and the Saber tooth 2 x 5 motor controller. The robot is designed for indoor or outdoor use and performs well on many different surfaces.

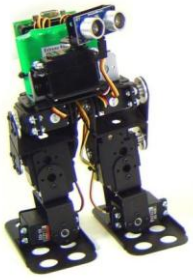
The BH3 robot

Another complex robot provided by Lynxmotion Inc. is BH3 robot**.** The BH3 robot offers the most advanced leg design available today. The three Degree Of Freedom (DOF) leg design makes this robot to walk in any direction. The robot uses 18 Hitec HS-475 / HS-485 servos for the legs. The robot is made from ultra-tough laser-cut Lexan structural components, custom injection molded components, and high-quality aluminum Servo Erector Set brackets.

The Robotic Arm

Another complex robot provided by Lynxmotion Inc. is Robotic Arm. The robotic arm delivers fast, accurate, and repeatable movement. The robot features: base rotation, single plane shoulder, elbow, wrist motion, a functional gripper, and optional wrist rotate. The AL5A robotic arm is an affordable system with a time tested rock solid design that will last and last.
The aluminum robotic arm is made from our Servo Erector Set components for the ultimate in flexibility and expandability. The arm uses 1 x HS-422 in the base, 1 x HS-755HB in the shoulder, 1 x HS-645MG in the elbow, 1 x HS-422 in the wrist, and 1 x HS-422 in the gripper.

The Biped BRAT Robot



BRAT stands for Bipedal Robotic Articulating Transport. The robot is a 6 servo biped walker featuring three DOF per leg. The robot can walk forward or backwards and turn in place left or right with variable speed. It can even do lots of Robo One style acrobatic moves. The robot is made exclusively from brushed, or black anodized aluminum servo brackets from our Servo Erector Set. It also includes an electronics carrier made from ultra-tough laser-cut Lexan. We are providing the walker with Hitec HS-422 servos. Due to the robot's light weight, these servos work well. As with any walking robot, weight is a major concern. The best approach is to keep the weight to an absolute minimum**.**
The robot is available in two basic configurations, Animatronic (tethered connection to a PC) or Autonomous (controlled by the onboard Basic Atom Pro processor). The autonomous version includes the Bot Board II and the Basic Atom Pro processor. The PC version uses our SSC-32 and the Visual Sequencer to control the robot motion. It is a tethered configuration but can be made wireless with a wireless serial device such as the Spark fun BlueSMiRF modem.

The ASIMO

Apart from Lynxmotion. Inc, Honda Company Introduced a humanoid Robot ASIMO in 2000. ASIMO, which is an acronym for Advanced Step in Innovative Mobility, was created to be a helper to people. With aspirations of helping people who lack full mobility, ASIMO is used to encourage young people to study science and mathematics. At 130 cm (4 feet, 3 inches) tall and 54 kg (119 lbs), ASIMO was designed to operate in real-world environments, with the ability to walk or run on two feet at speeds up to 6 kilometers per hour (3.7 mph).

3.1 Software Specification

Basic Micro Studio is the main piece of software that is used to write code for any of the Basic Micro products such as BasicATOMs or BasicATOM Pro modules. It is commonly referred to as an Integrated Development Environment or IDE for short. The IDE contains three main parts.

1. A text editor for writing programs
2. A compiler to translate the program into something the microcontroller will understand
3. A loader to download the program to the microcontroller.

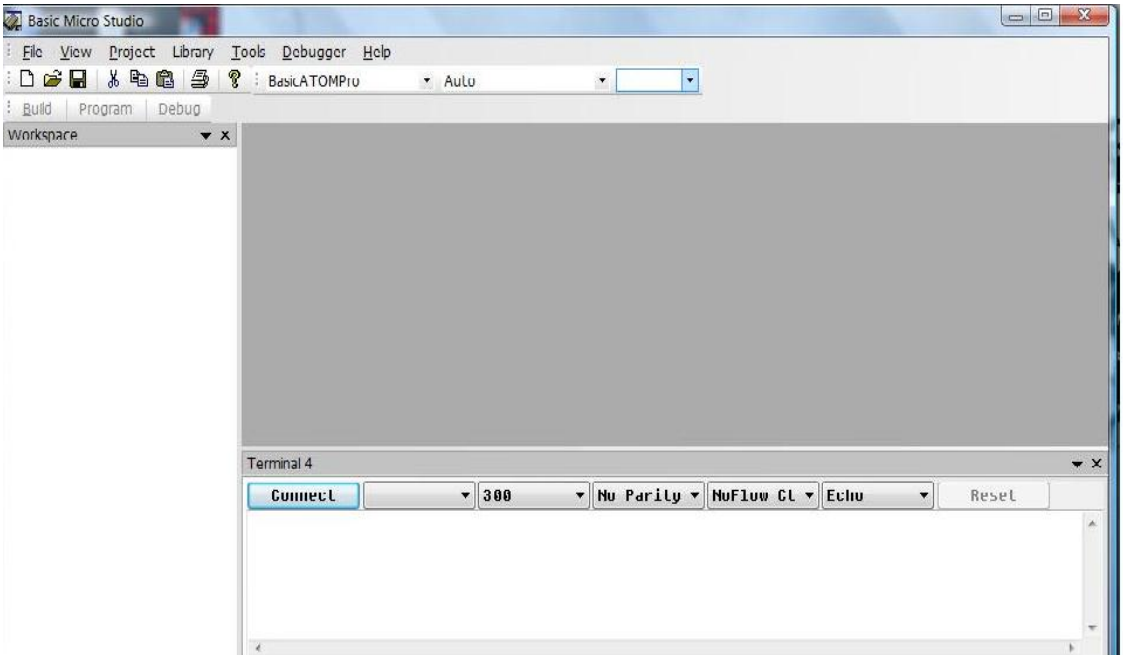


Figure 3.1.1: The Basic Micro Studio V2.0.0.15

The main Steps on uploading a program are as follows

1. First of all, we need to place the robot in stable position, with all the switches off.
2. We have to find program (.bas extension) which are wither programmer written or downloadable from any of robot vendor from www.lynxmotion.com
3. Now, we need to find a male to male 9-pin serial connector, one end of which goes to DB9 terminal of the robot, and other end goes to any of COM port in computer.
4. We now open the software interface, Basic Micro Studio and first of all try to locate the CPU found in IC chip on robot, as shown below:

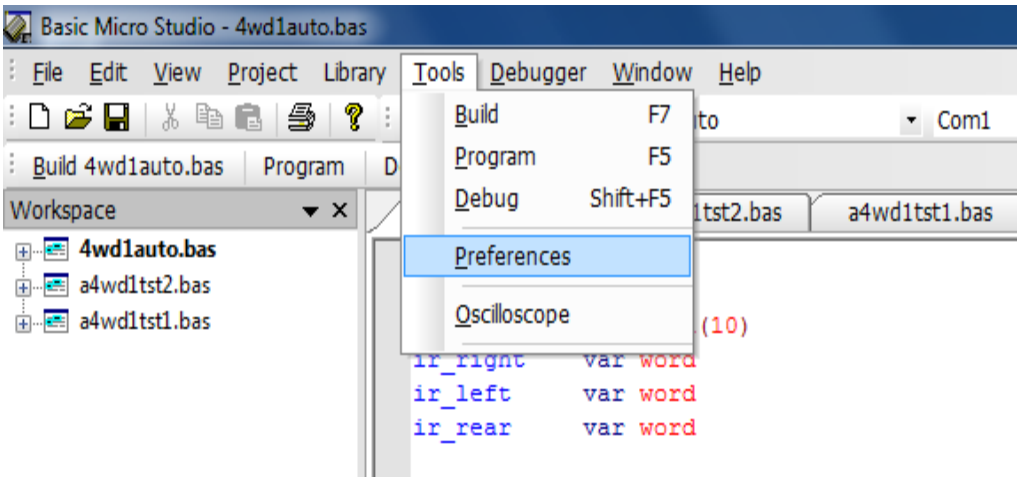


Figure: 3.1.2 Step for finding CPU

To check the preferences, we click on 'Tools' then select from the pop-down menu 'Preferences' and then we click 'Find Processor' and then we found the processor which is shown in the below figures.

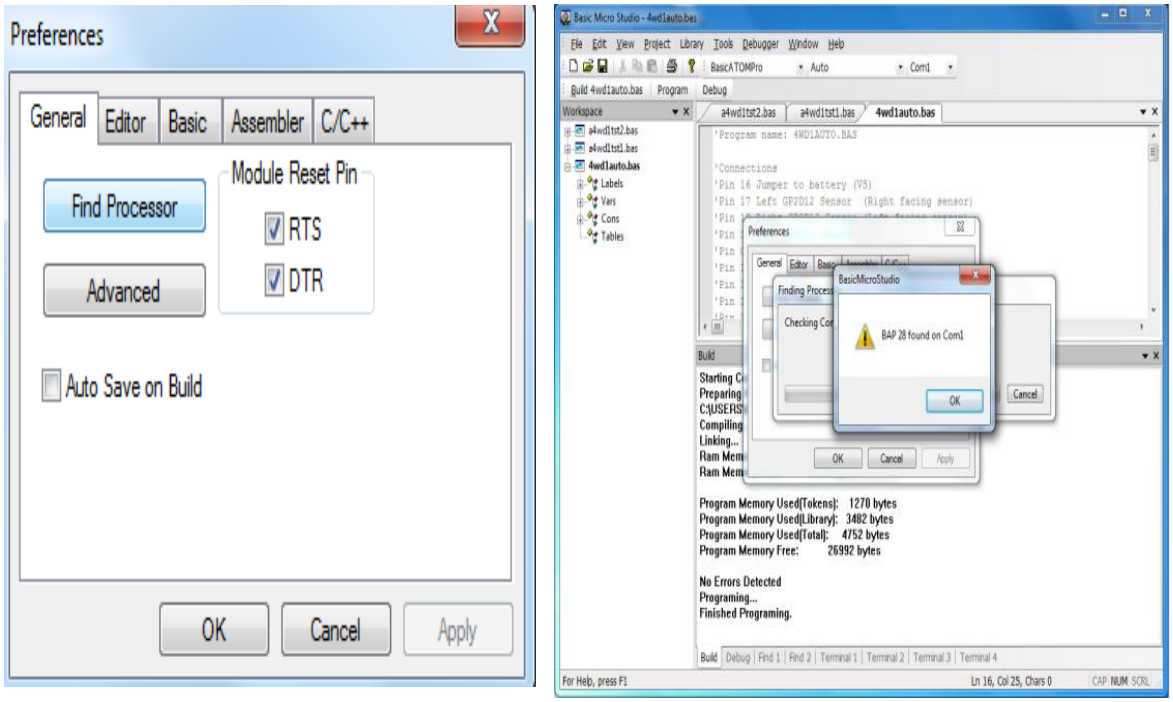


Figure: 3.1.3 configuring preferences and finding processor

5. Now we have to Configure Terminal as advanced option

   Configuring Terminal is a software component that makes it easy to quickly set up all functionality of the four Servo Controllers of the robot, unlike other terminal as different program; it is combined inside the same software that is Basic Micro Studio. All values are to be set as the figure illustrated below:

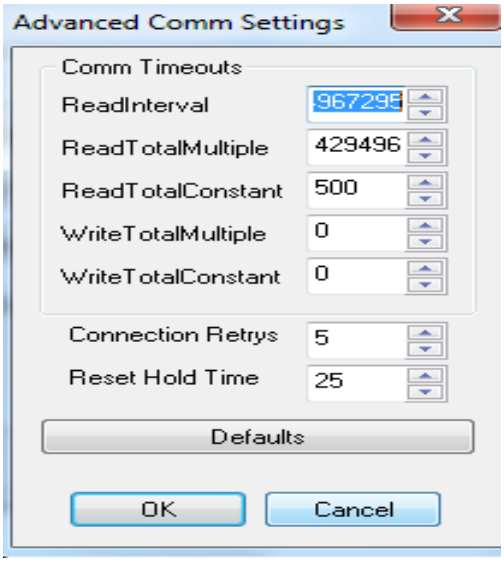


Figure: 3.1.4 configuring Advanced Settings for terminal CPU

6. Click on the "Program" button located in the toolbar as shown. This will compile the program and load it on window.

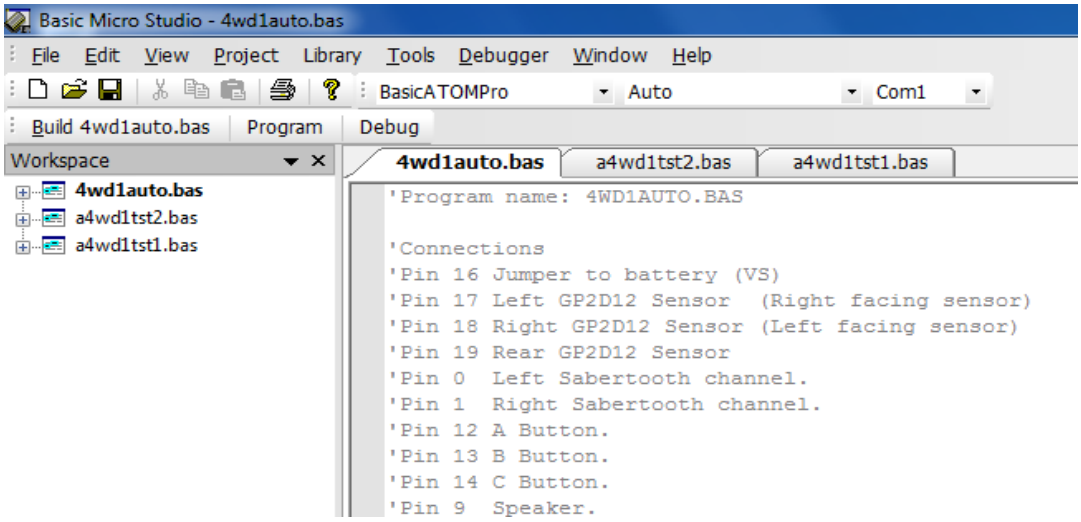


Figure: 3.1.5 Workspace window after loading programs

7. Now, we have to click on Terminal 1 on output window, and have to choose the setting as below figure

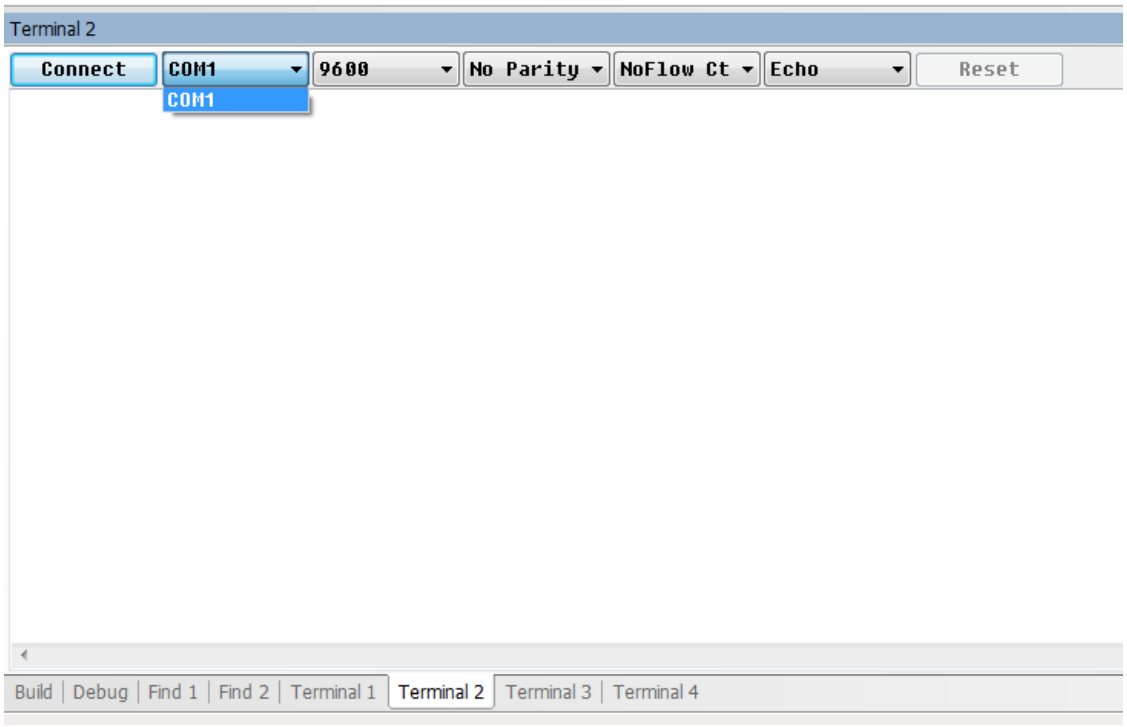


Figure: 3.1.6 Configuring Terminal and its settings

Used Connectors and devices

The following connectors are intended to be used. However, in doing our test, we used a DB9 serial port cable because of our pc limitations.

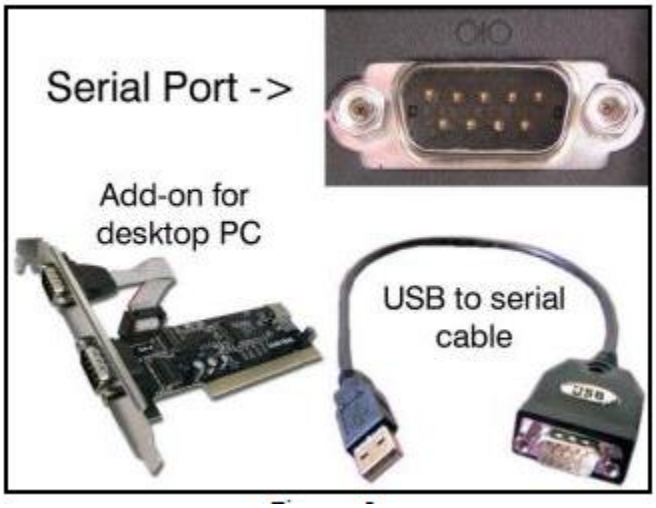


Fig 3.1.7: Used Connectors and devices to load program

7. 3.2. Hardware Specification

The overall specification of the A4WD1 Rover is as follows.

- Overall Length: 12.00"
- Overall Width: 13.50"
- Tire Height: 4.75"
- Chassis Length: 9.75"
- Chassis Width: 8.00"
- Chassis Height: 4.00"
- Ground Clearance: 1.63"
- Weight: 4lbs 6 oz.
- Speed: 36" per second.

The A4WD1 Rover consists of following hardware.

a) A4WD1 Rover Body Kit

The A4WD1 Rover chassis is made from heavy-duty anodized aluminum structural brackets and ultra-tough laser-cut Lexan panels. The chassis of the robot is expandable. The body is rectangular shape in which we can mount the four servo motors on its four corners, servo controller on bottom and Bot Board II on top side of the Body Kit.

Figure 3.2.1 The A4WD1 Rover Body Kit

b) Bot Board II

The Bot Board II is the best carrier board for the Basic Atom, or any other 24 or 28 pin microcontrollers. It has an onboard speaker, three buttons and three LEDs, a Sony PS2 controller port, a reset button, logic and servo power inputs, an I/O bus with power and ground, and a 5vdc 250mA regulator. Up to 20 servos can be plugged in directly.

Figure 3.2.2 The Bot Board II

c) The BASIC Atom 28 Pin

The powerful Basic Atom is faster and has more memory than a BS2. This chip is easy to program and reliable to use. It can be plugged into the Bot Board for complete access to all of the I/O pins. It is BS2 Pin Compatible and includes O'Scope and In Circuit Debugger (ICD).

Figure 3.2.3 The BASIC Atom 28 Pin

d) Saber tooth V1.03(SSC-32 Servo Controller)

Saber tooth V1.03 has 32 channels of 1uS resolution servo control. The features of Saber tooth V1.03 supports bidirectional communication with Query commands, Synchronized or "Group" moves and 12 built in Servo Hexapod Gait Sequencer, MiniSSC-II emulation.

Figure 3.2.4 The Saber tooth V1.03 (SSC-32 Servo Controller)

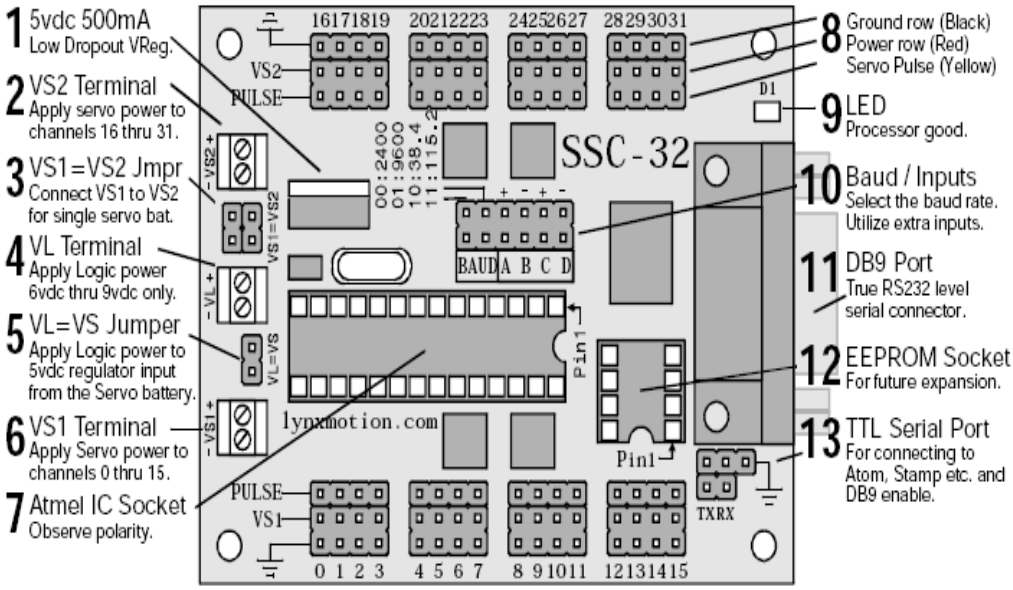


Figure 3.2.5 Detail diagram of the Saber tooth V1.03 (SSC-32 Servo Controller)

e) The DB9 Serial Data Cable - 6'

The DB9 Serial Data Cable is used to connect to a Bot Board II or SSC-32 servo controller with the computer in order to configure the A4WD1 Rover.

Figure 3.2.6 The DB9 Serial Data Cable - 6'

f) Servo Motors

The motors are efficient and reliable. They have neutral time for robotics use.

Voltage = 12vdc
RPM = 200
Reduction = 30:1
Stall Torque = 63.89 oz-in (4.6 kg-cm)
Outside Diameter = 37mm
Weight = 5.44 oz.

Figure 3.2.7 The Servo Motor

g) Off Road Robot Tire - 4.75"D x 2.375"W (pair)

These are the tires for the A4WD1 Rover.

Traxxas Stampede Tera (Firm)
Diameter = 4.75"
Width = 2.375"
Hubs = 6mm (HUB-12)
Motor = Any 6mm output shaft
Model Number: TRC-01
Weight: 0.75 oz.

Figure 3.2.8 The Wheels

h) 12mm Hex Mounting Hub - 6mm shaft (pair)

They are high quality RC truck (12mm hex pattern) tire hub and work with any motor with a 6mm shaft. We used them for mounting Off Road (TRC-01) tires.

Model Number: HUB-12
Weight: 0.07 oz.

Figure 3.3.9 The Mounting Hubs

i) 12.0 Volt Ni-MH 2800mAh Battery Pack

This 12.0vdc Ni-MH 2800mAh battery pack is perfect for small robots. The weight is half the weight of an equal capacity Ni-Cad pack.

Model Number: BAT-06
Weight: 1.20 oz.

Figure 3.2.10 12.0 Volt Battery

j) 6.0 - 12vdc Ni-CD & Ni-MH Universal Smart Charger

Peak charges NiCad and NiMH 5 - 10 cell battery packs with either 900mA or 1.8amp charging rates. The charger has automatic V (Delta Peak) cutoff and automatic trickle charge. This charger plugs into the wall for 110VAC operation.

Model Number: USC-02
Weight: 0.85 oz.

Figure: 3.2.11 The Battery Charger

k) Wiring Harness - Battery Connector

These heavy duty wiring harnesses are perfect for our small robot. The quick connect end mates to our NiMH battery packs, and the other end is stripped, but not tinned for screw terminal use. The toggle switch attached with this connector is used to turn on and off the robot.

Model Number: WH-01
Weight: 0.04 oz.

Figure 3.2.12 The Battery Connector

4. Design and Implementation

This project has both hardware and software components. We did not actually design the hardware. It mainly consisted of assembly of the Rover.

There are three main part of assembly of the Rover.

1. Mounting the tires, connecting motor controller and battery
2. Installing board and the SSC-32
3. Connecting SSC 32 to the motor controller.

4.1 Mounting the tires

Pull one side of the tire and insert one side of the rim into the opening at an angle. Rotate the rim so that we end up with the tire and rim looking like in Figure 4.1

Figure 4.1 Inserting the tire in the rim

The tricky part is to get bead of the tires over the flange in order to provide a perfect fit and the grip for the tires. Pressing the rim into the tire from both sides, we must be able to put one side of the bead over the flange.

Now just flip it over and with almost half the bead in the flange then, hold the side in and pull the rest of the bead away from the center. We should be able to get a perfect fit.

4.2 Connecting the motor controller and the battery

- For connecting the motor make sure the red wire is connected to (+) and the yellow wire is connected to the (-).
- Now mounting the motor to the chassis is a simple. Using two 3X6 mm screws for a good fit. Make sure the motor is not loose.

- To attach the motor controller use a double sided tape and refer the table for the connection. Make sure that the right side motor wire goes in the same terminal and left hand motor wire goes in the same terminal. Refer to the Figure 4.2 for reference.

Terminals	Connection
M1A	Right side motor yellow
M1B	Right side motor red
B+	Battery red (+)
B-	Battery black (-)
M2A	Left side motor yellow
M2B	Left side motor red

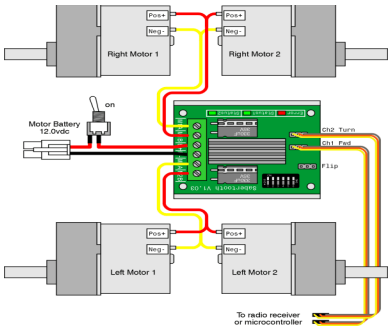


Figure 4.2 Connections between the motor and the controller

- We are using a single 12V battery which is can be easily plugged into the wiring harness and attached to the chassis using a two sided tape.

4.3 Assembling the Bot board II and the SSC -32 processor

Now this is the most important part of the assembly connecting the board and the chip

- For installing the board for rover we use four .250" 4-40 screws. And then install the Atom Pro chip as shown.

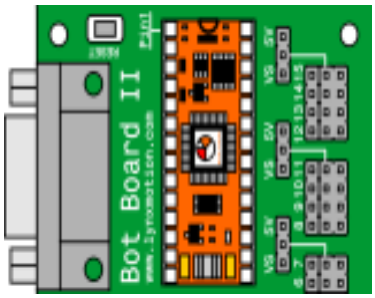


Figure 4.3.1 The Bot board II

- To connect the motor controller and the Bot board II use the Figure 4.3.2 given below

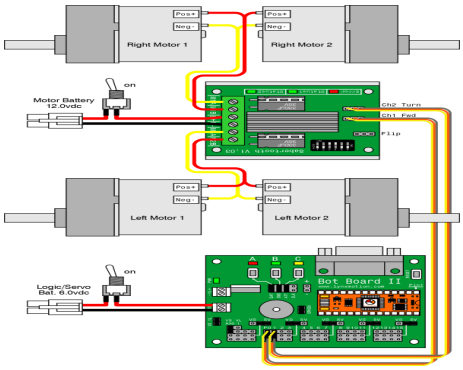


Figure 4.3.2 Connection between Bo board II and motor controller

4.4 Pictures of assembling 4 wheeler rover in our project

Here are some pictures taken while our team was assembling the rover. The team was divided into three groups each, first team responsible for performing mechanical and electrical work, second for performing integration of the rover and third for programming the rover.

Figure 4.4.1 Mounting the tires.

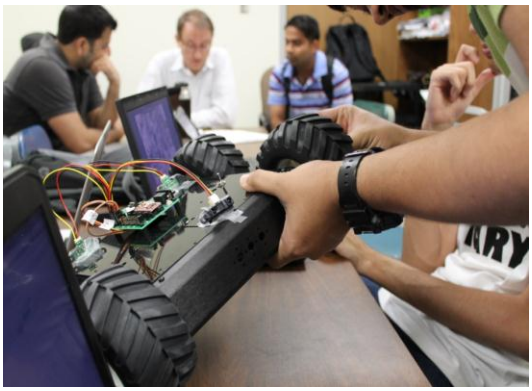


Figure 4.4.2 Attaching the tires to the rover

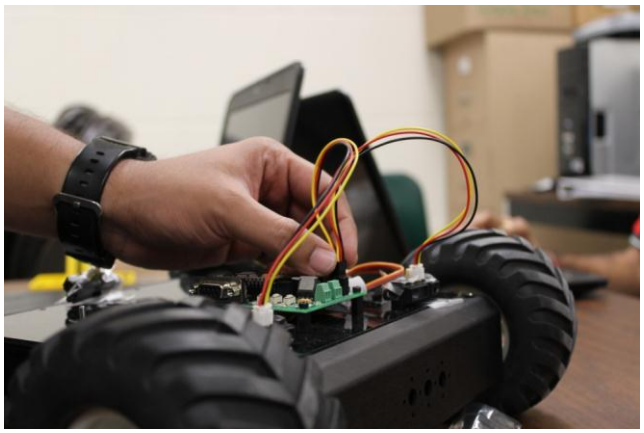


Figure 4.4.3 Connecting Bot Board with the sensors.

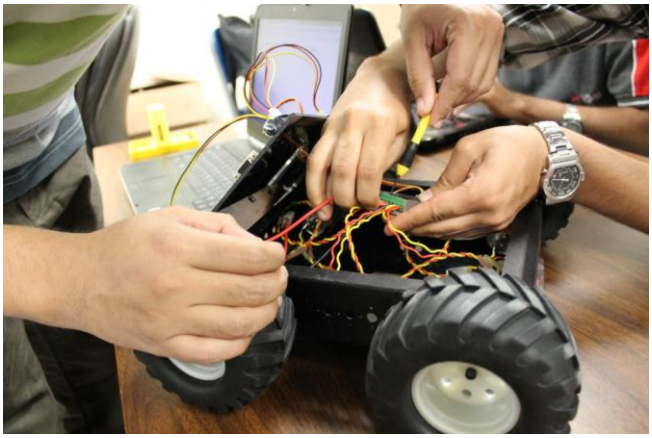


Figure 4.4.4 Reassembling the legs with proper servo positions

One of the challenges that we faced in our project was to properly place the bead of the tires into the rim's flange in order to provide proper friction.

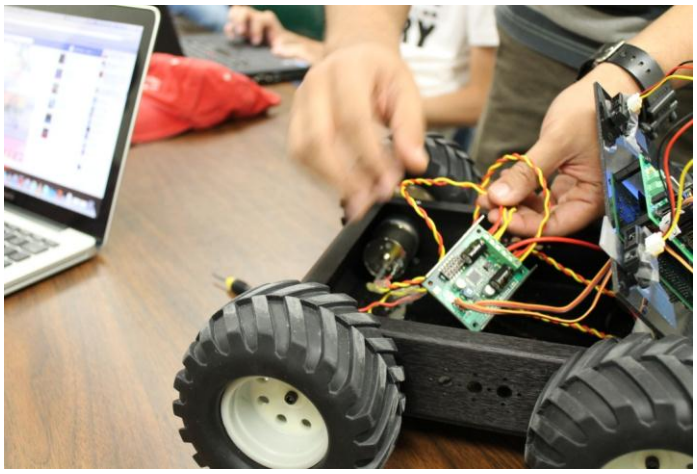


Figure 4.4.5. Connecting the tires to the appropriate I/O channels

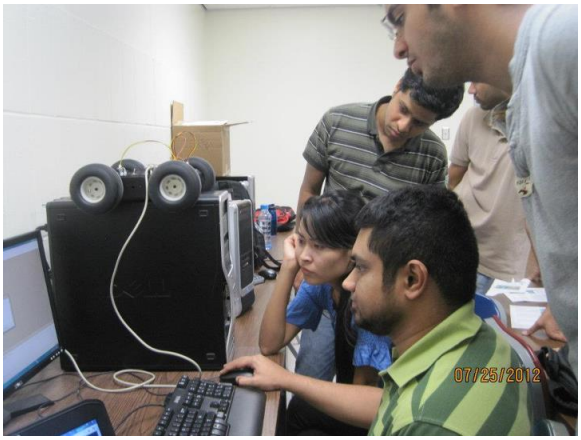


Figure 4.4.6. Installing and running the BasicMicro Studio Software

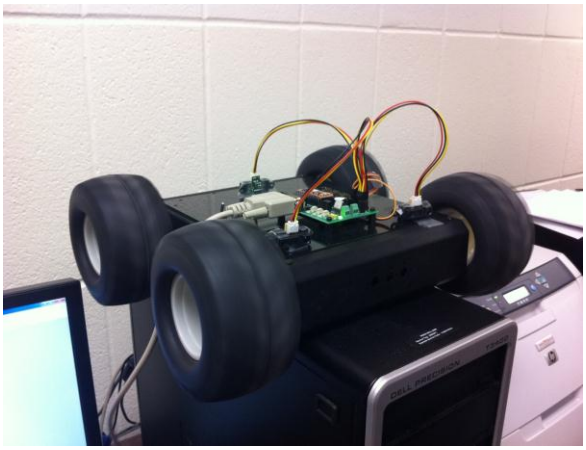


Figure 4.4.7 Configuring the 4 wheeler rover using the BasicMicro Studio software.

After configuring all the parts of the rover, we made a rover which detects the obstacles through the sensors and changes its direction accordingly. The speed of the rover can also be configured by making changes in the code.

5.Testing and Outputs

Integration testing was done on the rover with four different application programs.

The first program named "aprotut1.bas" showed that floating point registers in the microcontroller were working properly. The output of the program is shown in the snapshot below.

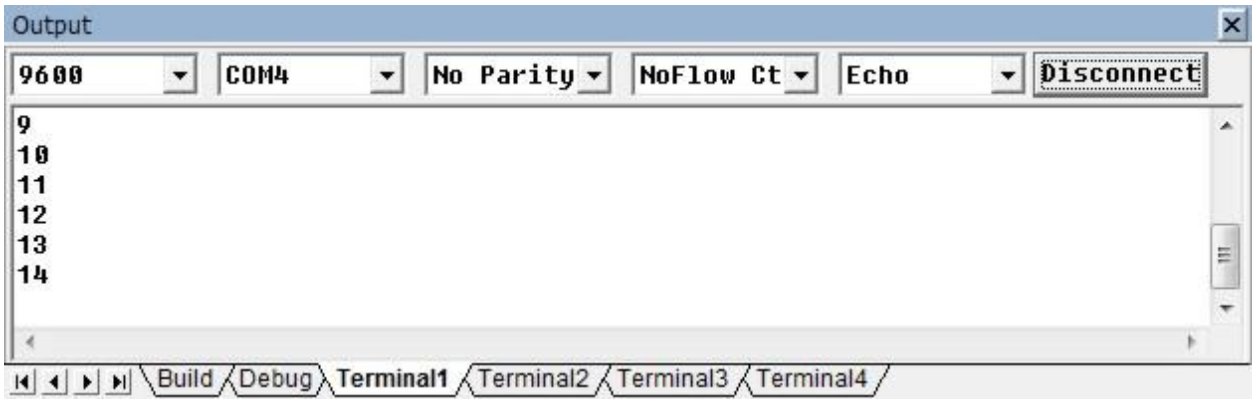


fig 6.1:snapshot showing output of the first program

In the second program named "a4wdtstl1.bas", when 'A' button is pressed, then the left side tires accelerate and when 'B' button is pressed, then the right side tires accelerate too. As we keep on pressing 'B' the acceleration of the right side tires increase. Now when 'C' button is pressed, then deceleration occurs in both sides. If we keep on pressing 'C' the movement occurs in reverse direction. It is demonstrated in the video provided below.

MVI_1304.avi

The third program named "a4wdtstl2.bas" is for braking purpose. When 'A' button is pressed, then the left side tires accelerate and increases its acceleration as we keep on pressing 'A'. But when 'B' button is pressed, then the left side tires stop immediately. Now when 'C' button is pressed , then the right side tires accelerate. It is demonstrated in the video provided below.

MVI_1303.avi

The last program "4wd1auto.bas" is the main program in this project. It has implementation for detecting sensors too. If any of the two frontal sensors detects any obstacle it reverses backward. Likewise, if the back sensor detects any obstacle it moves forward. It is demonstrated in the video provided below.

MVI_1307.avi

6. Source Code used:

The language used is BASIC, and the software package itself has compiler and linker associated with it. We just need to load the program written in the same software on clicking Tools and select the Build button.
In optional cases, we can use debugger program to look around the erratic behavior of robot, and these can be corrected by changing the source code in same program.
We have used 4 different programs to play with different modes of robot listed as herewith:

Source code 1: 4WD1AUTO.BAS

This code is used for,

1. Setting up all the parameters for servos, setting up min, max speed, direction of movement when faced with obstacle, servo pulse out, and basic robot movement loops

```
'Program name: 4WD1AUTO.BAS

'Connections
'Pin 16 Jumper to battery (VS)
'Pin 17 Left GP2D12 Sensor   (Right facing sensor)
'Pin 18 Right GP2D12 Sensor(Left facing sensor)
'Pin 19 Rear GP2D12 Sensor
'Pin 0   Left Sabertooth channel.
'Pin 1   Right Sabertooth channel.
'Pin 12 A Button.
'Pin 13 B Button.
'Pin 14 C Button.
'Pin 9   Speaker.

temp            var byte
filter          var word(10)
ir_rightvar word
ir_left         var word  ‘ declaration of data variables
ir_rear         var word  ‘ declaration of data variables

LSpeed                  var word
RSpeed                  var word

minspeed        con 1750        ‘ min speed set
maxspeed        con 1250        ‘ min speed set

LSpeed = 1500
RSpeed = 1500
```

```
low p0
low p1

sound 9, [100\880, 100\988, 100\1046, 100\1175]

main
gosub sensor_check

; Numbers lower than 1500 result in forward direction.
; Numbers higher than 1500 result in reverse direction.

LSpeed = (LSpeed - 10) min maxspeed        ‘accelerating the motors
RSpeed = (RSpeed - 10) min maxspeed        ‘setting up acceleration

LSpeed = (LSpeed + ir_left) max minspeed
        ;when something is detected, this decelerates the opposite side
RSpeed = (RSpeed + ir_right) max minspeed

if (ir_rear > 15) then
LSpeed = (LSpeed - ir_rear) min maxspeed ‘if something is detected behind the robot, accelerates both sides
RSpeed = (RSpeed - ir_rear) min maxspeed
endif

; Send out the servo pulses
        pulsout 0,(LSpeed*2) ‘ Left Sabertooth channel.
        pulsout 1,(RSpeed*2) ‘ Right Sabertooth channel.
        pause 20

goto main


sensor_check   ‘ main checking and looping of the movement

for temp = 0 to 9
  adin 17, filter(temp)
next
ir_right = 0
for temp = 0 to 9
  ir_right = ir_right + filter(temp)
next
ir_right = ir_right / 85

for temp = 0 to 9
  adin 18, filter(temp)
next
```

```
ir_left = 0
for temp = 0 to 9
  ir_left = ir_left + filter(temp)
next
ir_left = ir_left / 85

for temp = 0 to 9
  adin 19, filter(temp)
next
ir_rear = 0
for temp = 0 to 9
  ir_rear = ir_rear + filter(temp)
next
ir_rear = ir_rear / 85


 serout s_out,i38400,["ir_right - ", dec ir_right, " ir_left - ", dec ir_left, " ir_rear - ", dec ir_rear, "LSpeed - ", dec LSpeed, " RSpeed - ", dec RSpeed, 13]

return ‘ ending of the program
```

Source code 2: 4WD1TST1.BAS

The Throttle and Steering Mode Test
Make sure the Sabertooth's Switch 1 is flipped back into the "On" or "Enable Mixed Mode" position!

Here, in Bot Board II, the "B" button is throttle and the "A" and "C" buttons are steering.

Upon powering up the robot, we should hear four ascending notes. Pressing B once results in a beep and slow forward motion (10%). Pressing nine more times results in 100% power. After the motor is at 100% power, pressing B will reduce the speed in 10% increments until it stops. Continue to press B to make the robot move as above, only in reverse.

Press Reset, then B twice. Now press C a few times to see the robot make a gradual left turn. Pressing A a few times will return to forward motion, and continuing to press A will result in gradual right turn.

Experiment with these buttons to understand how throttle and steering can be used to control the vehicle's motion.

Note: The Sabertooth's red Error LED will light to indicate overheating or current limit. The green Status1 LED will glow dimly when power is applied, and brightly when it's receiving pulses from the microcontroller. The green Status2 LED will flash out the detected number of lithium cells when lithium mode is enabled.

```
' This program tests the motion of the robot using the A, B, and C buttons on
' the Bot Board. Pressing the A button increments steering variable by 10%
' each press. Pressing the C button decrements steering variable by 10% each
' press. The B button increments to full forward, then decrements to full
' reverse speed by 10% each press.

' The Scorpion has two modes of operation. The default is the mixer mode. This
' is where you have a Throttle for forward and reverse speed control, and a
' Steering control for turning. In this mode the left input is throttle and
' the right input is steering. The other mode is a differential style control.
' This is where the left and right channels are controled indepentantly to
' steer like a tank.

'Bot Board Jumpers
' Speaker enable
' VS to VL
' A, B, C, button enable
' AX 0-3 power bus to VL

'Connections
' Pin 0  Left Scorpion channel. (Throttle)
' Pin 1  Right Scorpion channel. (Steering)
' Pin 4  A Button.
' Pin 5  B Button.
' Pin 6  C Button.
' Pin 9  Speaker.

temp        var     byte    ,Variable definetions.
throttle var    byte
steering    var     byte
direction   var     bit

throttle = 150          ' Do not move.
steering = 150          ' Do not turn.
direction = 1

low     p0                      ' Ensure pulsout commands are positive going.
low     p1

sound 9, [100\880, 100\988, 100\1046, 100\1175]

main:
if in12 = 0 then right_turn         'A button increments steering variable by 10% each press.
if in13 = 0 then throttle_up        'B button increments then decrements speed variable by
10% each press.
if in14 = 0 then left_turn          'C button decrements steering variable by 10% each press.
```

```
' Send out the servo pulses
        pulsout 0,(throttle*20)' Left Scorpion channel.
        pulsout 1,(steering*20)         ' Right Scorpion channel.
        pause 20
'       serout S_OUT,i57600,["T ", dec throttle, "   S ", dec steering, "   D ", dec direction, 13] '
Remove the rem at the beginning to see in term1.
goto main

throttle_up:
sound 9, [100\880]
if in13 = 0 then throttle_up
  if direction = 0 then throttle_down
    throttle = (throttle -5) min 100
  if throttle = 100 then t_up
goto main

t_up:
  direction = 0
goto main

throttle_down:
  if in13 = 0 then throttle_down
    throttle = (throttle +5) max 200
  if throttle = 200 then t_down
goto main

t_down:
  direction = 1
goto main

right_turn:
sound 9, [100\988]
 if in12 = 0 then right_turn
  steering = (steering -5) min 100
goto main

left_turn:
sound 9, [100\1046]
 if in14 = 0 then left_turn
  steering = (steering +5) max 200
goto main
```

Source code 3: 4WD1TST2.BAS

The Differential or Tank Mode Test
Here, on the Bot Board II, the "A" button is left channel throttle, "C" button is right channel throttle, and "B" is a speed and direction reset.

This program requires the Sabertooth's Switch 1 to be flipped to the "Off" or "Independent Control" position.

Upon powering up the robot, we should hear four ascending notes. Pressing A once results in a beep and slow forward motion (10%) on the left channel only. Pressing nine more times results in 100% power. Continuing to press A will make the motor act as above, except only for the right channel. The C button will control the left motor in a similar manner. Pressing the B button will reset the speed and direction of both left and right.

Note: The Sabertooth's red Error LED will light to indicate overheating or current limit. The green Status1 LED will glow dimly when power is applied, and brightly when it's receiving pulses from the microcontroller. The green Status2 LED will flash out the detected number of lithium cells when lithium mode is enabled.

```
' This program tests the motion of the robot using the A, B, and C buttons on
' the Bot Board. Pressing the C button increments the Left channel to full
' forward, then decrements to full reverse in 10% increments. Pressing the A
' button increments the right channel to full forward, then decrements to full
' reverse in 10% increments. Pressing the B button resets to stopped values.

' The Scorpion has two modes of operation. The default is the mixer mode. This
' is where you have a Throttle for forward and reverse speed control, and a
' Steering control for turning. In this mode the left input is throttle and
' the right input is steering. The other mode is a differential style control.
' This is where the left and right channels are controled indepentantly to
' steer like a tank. This program uses differential control. You will need to
' install a jumper to put the Scorpion into defferential mode.

'Bot Board Jumpers
' Speaker enable
' VS to VL
' A, B, C, button enable
' AX 0-3 power bus to VL

'Connections
'Pin 0   Left Scorpion channel. (Throttle)
'Pin 1   Right Scorpion channel. (Steering)
'Pin 4   A Button.
'Pin 5   B Button.
'Pin 6   C Button.
'Pin 9   Speaker.
```

```
temp                    var     byte    ' Variable definetions.
left_speed              var     byte
right_speed             var     byte
l_dir                   var bit
r_dir                   var bit

left_speed = 150                ' Left Scorpion stop value.
right_speed = 150               ' Right Scorpion stop value.

low     p0                              ' Ensure pulsout commands are positive going.
low     p1

sound 9, [100\880, 100\988, 100\1046, 100\1175]

l_dir = 0
r_dir = 0

main:    ‘ main program starts here
if in12 = 0 then right_adjust   'A button increments then decrements right_speed variable by 10% each press.
if in13 = 0 then reset_both             'B button resets both speed variables to stopped.
if in14 = 0 then left_adjust            'C button increments then decrements left_speed variable by 10% each press.
' Send out the servo pulses
        pulsout 0,(left_speed*20)       ' Left Scorpion channel.
        pulsout 1,(right_speed*20)      ' Right Scorpion channel.
        pause 20
        serout S_OUT,i57600,["L ", dec left_speed, "    R ", dec right_speed, 13] ' Remove the rem at the beginning to see in term1.
goto main

right_adjust:
sound 9, [100\880]
if in12 = 0 then right_adjust
  if r_dir = 0 then r_down
    right_speed = (right_speed +5) max 200
  if right_speed = 200 then r_toggle_up
goto main

r_toggle_up:
  r_dir = 0
goto main

r_down:
  if in12 = 0 then r_toggle_down
    right_speed = (right_speed -5) min 100
```

```
  if right_speed = 100 then r_toggle_down
goto main

r_toggle_down:
  r_dir = 1
goto main

left_adjust:
sound 9, [100\880]
if in14 = 0 then left_adjust
  if l_dir = 0 then l_down
    left_speed = (left_speed +5) max 200
  if left_speed = 200 then l_toggle_up
goto main

l_toggle_up:
  l_dir = 0
goto main

l_down:
  if in14 = 0 then l_toggle_down
    left_speed = (left_speed -5) min 100
  if left_speed = 100 then l_toggle_down
goto main

l_toggle_down:
  l_dir = 1
goto main

reset_both:
right_speed = 150
left_speed = 150
r_dir = 0
l_dir = 0
goto main  'ending of the main program
```

Source code 4 aprotut1.bas

Here, we will want to set the robot on something so that the wheels aren't touching the ground.

Before running the program, turn on the PS2 controller. When the program is run it will turn most PS2 game controllers to analog mode (required) automatically. If controller does not automatically go into analog mode, it will need to do so manually.

This program lets you control the movement of the bot and several add-on components. Use the left joystick to move the bot forward and backward, and make left and right turns. The turns can be gradual or on-the-spot, depending on how far you push the joystick.

```
;The compiler will ignore any commands
;or text after a ; or '

x var word

start

;this will sound 3 ascending beeps
sound 9,[150\2500, 150\3000, 150\3500]
;pause for one second
pause 1000
;add one to the count
x = x + 1
;sends the x back to the computer
serout s_out,i9600,[DEC x, 13]

;repeat
goto start
```

The next step goes to testing, deploying and correction of robotic movements written as herewith.

7.Conclusion and Future Works

In this project, we assembled, programmed and successfully implemented an autonomous **A4WD1 v2 Robot.** Depending on the direction in which the rover finds obstacle, it moves forward, backward, left or right. While implementing the robot, our classmates worked effectively to achieve the desired goal, that is, to test the 4WD rover with two fronts and a rear sensor. Later we can use the robot for various purposes such as object detection, image capturing, spying etc.

However, it has certain limitations such as the sensor of the 4WD autonomous rover cannot detect some obstacles that are on its way but not within the range of the sensor. In case the rover has a high speed, it continues its run and results a certain collision. Also the autonomous rover has no sensor at the bottom part of its chassis. Therefore, it cannot detect any obstacle that comes underneath the rover. Also it has no sensor or no program to prevent falling from the brink.

As future work, we would like to add more sensors to detect each and every obstacle that comes on its way. We would also like to attach a wireless camera to get the actual view from a distance. Thus analyzing the camera and sensor we would like to program it in such a way that it prevents falling from the brink. In future we would like to put a GPS over the rover, make a ground base control and control the rover from the base. Setting some additional hardware to the existing 4WD autonomous rover for advanced application, such as the robotic arm to grab something, is another plan.

8. References

1. Dr. Stefan Andrei: Lectures notes for 'Embedded Systems' Class (COSC-4301-01/COSC-5340-01), Summer of 2012: Embedded Systems. *Lamar University*, Department of Computer Science, Beaumont, Texas

2. http://galaxy.lamar.edu/~sandrei/work.html#teaching

3. http://www.lynxmotion.com/driver.aspx?Topic=assem05

4. http://galaxy.lamar.edu/~sandrei/Johnny5/

5. http://galaxy.lamar.edu/~sandrei/MinihexRobot/

6. http://en.wikipedia.org/wiki/Rover_%28space_exploration%29

7. http://www.superdroidrobots.com/ATR_std2.aspx

8. http://aaqilkhan.blogspot.com/2007/08/autonomous-robotic-rover.html

9. http://www.superdroidrobots.com/ATR.aspx