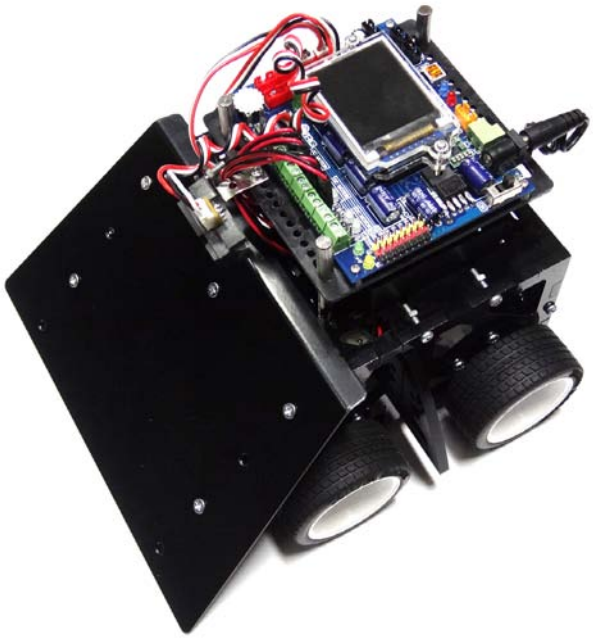
Robo-Creator 2
creative robotics kit

Sumo-BOT



The autonomous Sumo robot programmed with C/C++ language

Activity book

inex
INNOVATIVE EXPERIMENT

Robo-Creator 2
creative robotics kit

Sumo-BOT activity

Who should use this handbook?

1. Students and other people who are interested in applying to microcontrollers for testing working process of automatic system or people who are fascinated in learning and examining the microcontrollers in new approaches such as using an autonomous robot as a form of an interactive media.

2. Academic institutes such as schools, colleges and universities, where provide electronic subjects or electronic and computing engineering departments.

3. Lecturers and teachers who would like to study and prepare lesson plans for microcontroller courses, including applied science which focuses on integrating electronics, microcontrollers, computer programming and scientific examination in high school education, vocational education, and bachelor's degree.

Published and distributed by

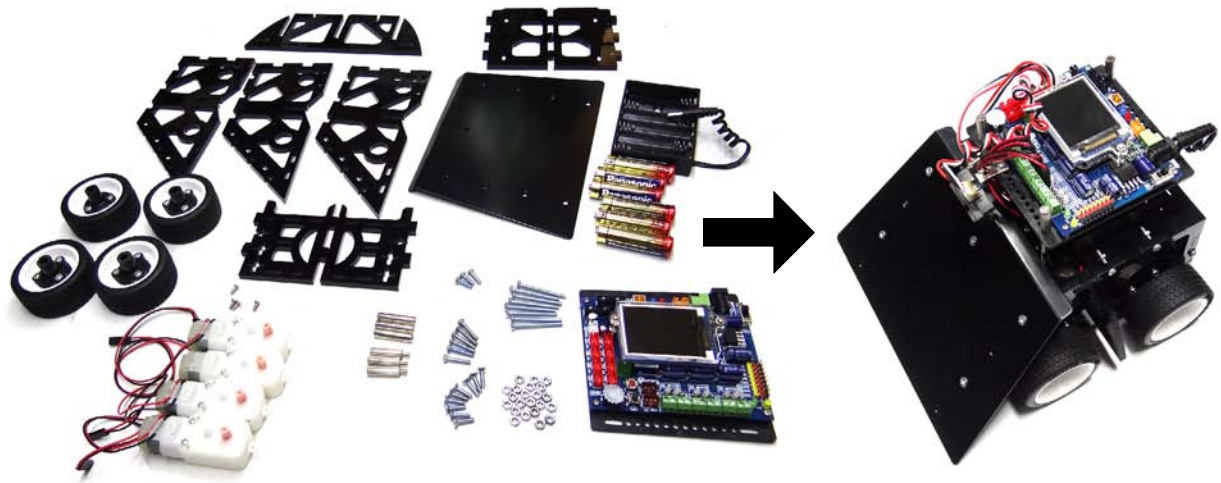
Innovative Experiment Co.,Ltd.

108 Soi Sukumvit 101/2 Sukumvit Road, Bangna, Bangkok 10260 THAILAND

Chapter 1

Sumo-BOT part description

Sumo-BOT is an autonomous robot performed by DC motors with the set of 4 DC motor gearboxes come with medium wheels and real tires in order to aid passing through flat surface more efficiently. The main purpose is Sumo robot competitions. The possibility is that Sumo-BOT is capable of supporting any mission in either a smooth competition court with lines appeared for determining directions of motion or rough court with barriers or participating with the World RoboCup Junior-Rescue and Fire fighting if install the extinguish part.



The Sumo-BOT also is one of the robotic activities of Robo-Creator2, the robotic kit for creative education.

1.1 Sumo-BOT features

- 4-Wheel driving autonomous robot. Powered by INEX's ATX2 controller board
- 2-Infrared Reflector sensor for line detection
- One of distance sensor to detect the competitor
- Size 20x20cm. Weight 1kg. approximation. Suitable for World Robot Game in Sumo 1kg. game (junior and family)
- Body is made from laser-cut acrylic sheet 5mm. thickness. Assembly by using screws and nuts. The front scoop is made from metal sheet with black coating.
- Programming in C/C++ language with Arduino 1.0.7 customized version.
- Power resource is 6 of AA battery. Alkaline or Rechargeable type is recommended . Also support Lithuim Polymer (Li-PO) 7.4V 1000mAH or higher.

1.2 Sumo-BOT part list

1. ATX2 controller board

2. Infrared reflector (ZX-03) 2 sets

3. Touch sensor (Switch input boards) 2 sets

4. GP2Y0A41 distance sensor with cable

5. BO-1 DC motor gearbox 87:1 ratio with wires 4 sets

6. Sport wheel sets (diameter 65mm., width 26mm. also include the hub for BO-1 gearbox) 4 sets

7. Plasitc joiner set

8. Strip joiner set

9. Plastic spacer pack

10. Metal bracket set

11. Nuts and screws set

12. Acrylic chasis part set

13. Metal front scoop

14. USB-miniB cable

15. 6AA battery holder with wire

16. 6 pieces of AA battery (not included inthe kit)

Assembly Tool : Screw driver and Nut driver

1.3 ATX2 controller board

1.3.1 Features

In the figure 1-1, it is shown components of the ATX2 controller board and there are significant technical features as follows:

● Main microcontroller is Atmel's ATmeg644. It features 8-ch 10-bit Analog to Digial Converter, 128-KByte Flash memory , 4-KByte EEPROM, 4-KByte RAM. Operated with 16MHz clock from external crystal

●Define all ports compatible with Arduino I/O standard hardware. The number of port are 0 to 31. All ports available with 3-pin of 2.00mm. header. (+5V, Signal and GND)

● 13 programmable port in JST connector type. Includes Digital I/O port (2, 3, 8, 9, 18, 24 to 31), A/D port (8 : port 24/A0 to 31/A7), Two-wire interface or TWI (8/SCL and 9/SDA) and UART serial port communication (2/RxD1 and 3/TxD1). Both TWI and UART ports can config to digital input/output port for more I/O applications.

● Extension analog input (A8 to A12) and One variable resistor is labeled KNOB and OK button for ADC experiments. Only these inputs are analog input only.

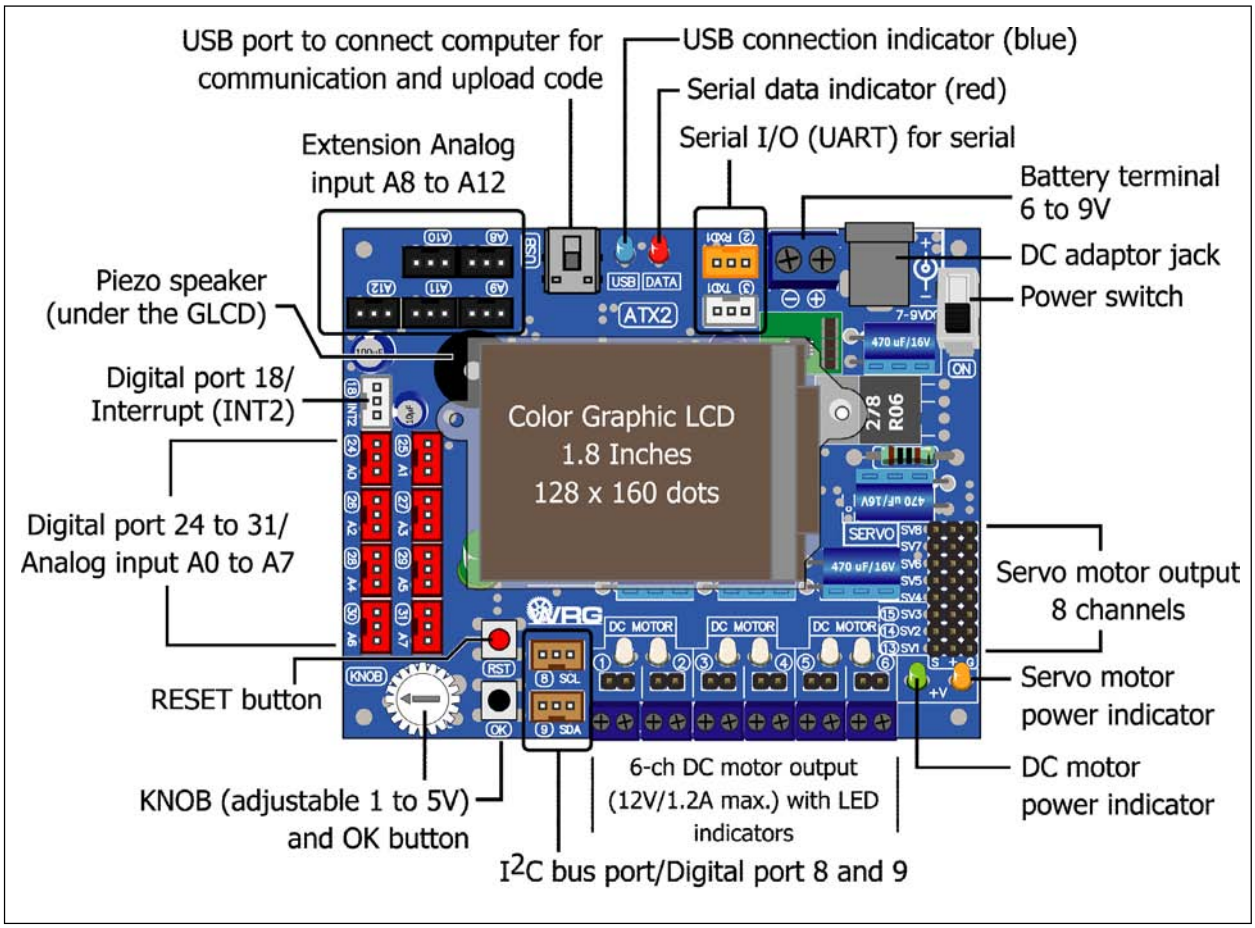


Figure 1-1 : ATX2 controller board layout

● All analog supports 0 to +5Vdc input. The converter resolution is 10-bit. The result value is 0 to 1,023 range.

● One of button switch "OK" for simple digital input experiment.

● One piezo speaker

● UART port (RxD1 and TxD1) for interfacing serial module device such as Pixy camera module, servo controller board (Parallax servo controller, ZX-SERVO16i), Real-time clock (ZX-17 : serial real-time clock module)

● 128x160 dots color GLCD. It only supports line-art , color background and text with 21 characters 16 lines (no support the image file).

● 6-ch of DC motor driver with LED indicator. Support 4.5 to 9V motor 1.2A (max)

● 8-ch of Servo motor outputs

● Upload with computer via USB port

● 2 of power inputs ; DC adaptor jack and 2-pin terminal block for battery

● Power switch and RESET switch available

● Requires +6 to 9V 500mA supply voltage in normal operation (no motor driving) and/or at least 1500mA for robotics application.

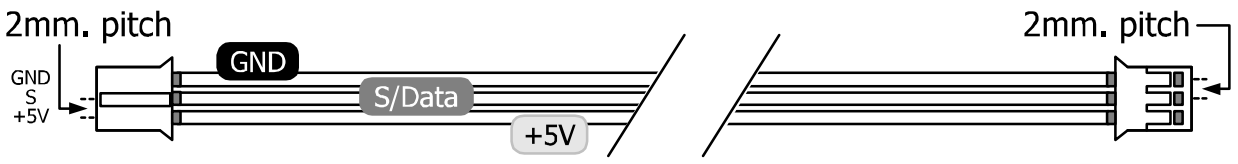
● +5V switching regualator on-board with polarity protection circuit

1.3.2 ATX2 cable information

For using the ATX2 controller board requires 2 kinds of cable. One is miniB-USB cable for interfacing with computer and JST3AA-8 cable for connecting with any sensor board and input/output devices.

1.3.2.1 JST3AA-8 cable

This is an INEX standard cable, 3-wires combined with 2mm. The JST connector is at each end. 8 inches (20cm.) in length. The wire assignment is shown in the diagram below.



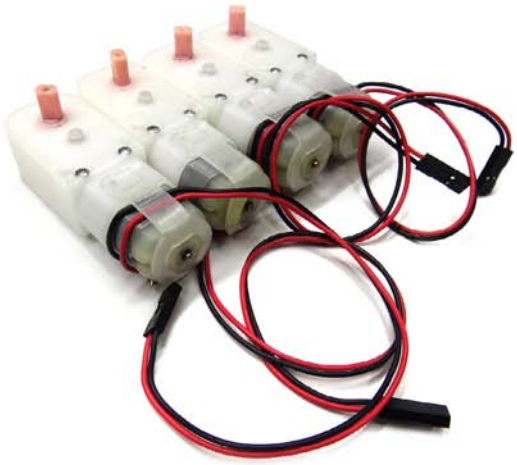
1.3.2.2 Standard USB-miniB cable

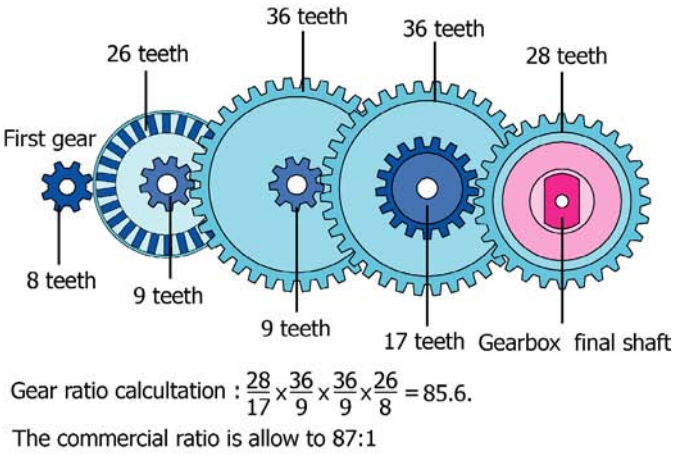
This is used to connect between the computer's USB port and the ATX2 controller board. Its length is 1.5 metres approximation.

1.4 DC motor gearbox

DC motor gearbox of Sumo-BOT is the BO1 model. The technical features are as follows :

- Requires the supply voltage +4.8 to +9Vdc 130mA @6V and no load
- Gear ratio 87:1
- Speed 100 rounds per minute @6V and no load
- Weight 30 gram
- Torque 0.8kg.-cm.





Gear ratio calcultation : $\frac{28}{17} \times \frac{36}{9} \times \frac{36}{9} \times \frac{26}{8} = 85.6.$

The commercial ratio is allow to 87:1

1.5 Sensor features

1.5.1 Touch sensor/Switch input board

The circuit is shown in the figure 1-2 including a switch with a LED and considered output as logic '0' when switch is pressed.

If the switch is pressed : the logic '0' will be sent and the red LED is on.

If no press : LED is off and the logic is '1'.

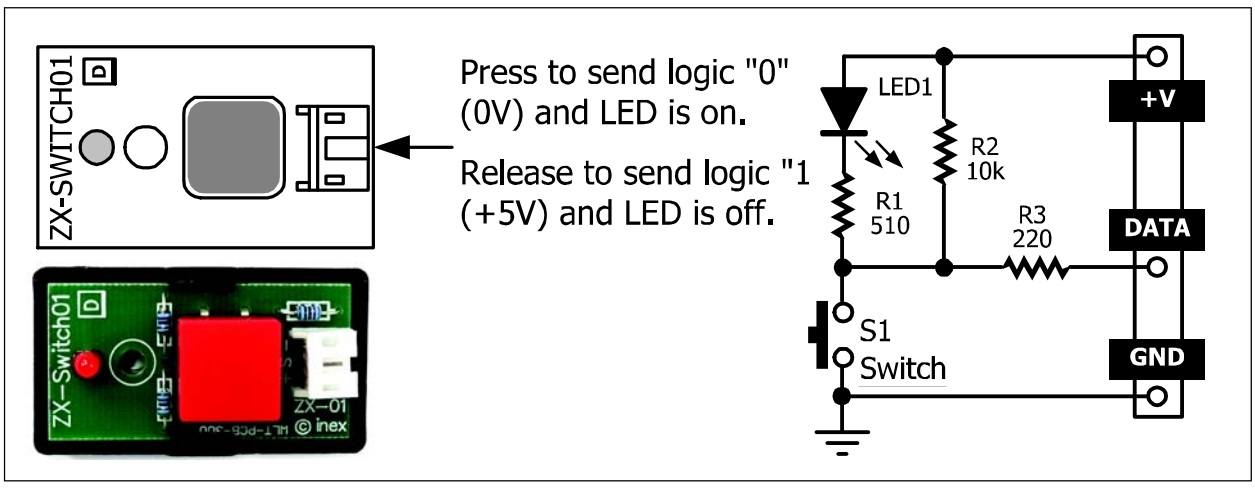


Figure 1-2 : The touch sensor or Switch input board

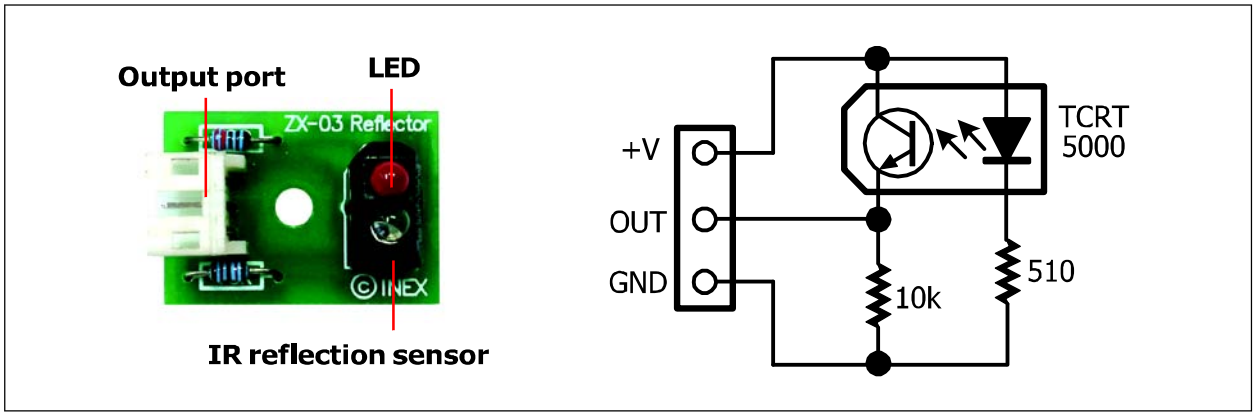


Figure 1-3 : IR reflector sensor layout and schematic diagram

1.5.2 The infrared reflection sensor : ZX-03

The circuit and layout of this sensor are displayed in the figure 1-3. The circuit is used to detect the reflected lights from surface or lines.

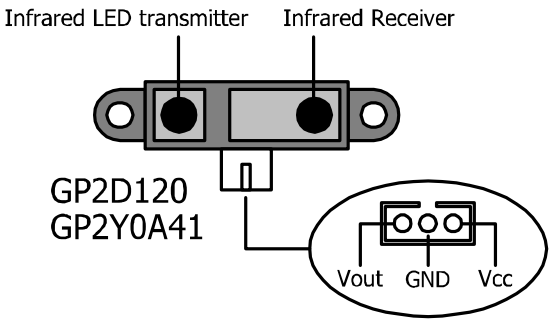
During apply the power supply, the Infrared LED is on at all the time. Meanwhile, the photo-transistor will get IR lights from the reflection of objects or surface. The amount of the reflected light will be more or less depending whether there is an obstacle or not and how well the object can reflect infrared. The reflection of IR is based on the surface texture and colour of objects. It is said that the white smooth objects are able to reflect light well so the infrared receptor gets a lot of reflected light and the output voltage will be high. As black objects reflect less light, the light receptor sends low voltage. With such features, the sensor circuit board is often used to trap the reflected light on the surface and lines. It is necessary to install the circuit at the lower part of a robot.

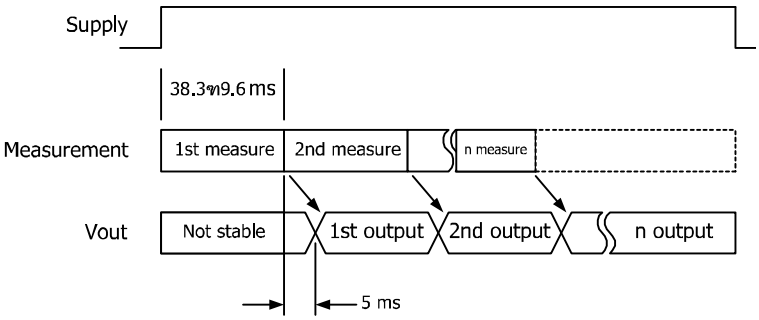
Due to ZX-03 IR reflector gives the result as DC voltage, applying on ATX2 controller board of Sumo-BOT. Connect the signal to 12 channels of analog input on the ATX2 controller board, from A0 to A12. After reading the analog signal value, use this value to check the value on reflected light detection circuit and then apply to detect lines.

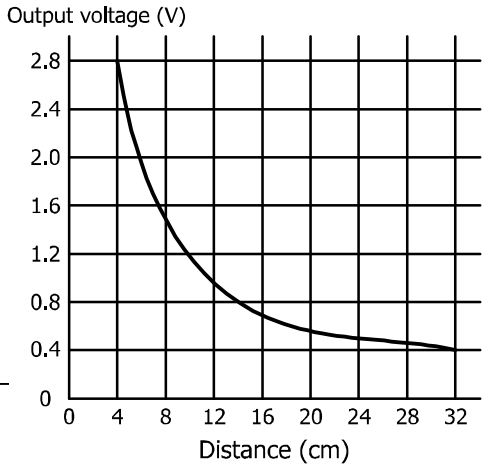
1.5.3 GP2Y0A41 module detecting distance with Infrared

GP2Y0A41 is a module that detects distance with Infrared and with the packet of 3 extension parts, including 3 pins; +Vcc, GND and Vout. Reading voltage value from GP2Y0A41 must do after the preparation period of the module, which takes 32.7 to 52.9 millisecond (1 millisecond equal 0.001 second). So reading the value should wait for the suitable time as mentioned above and shown the basic data in the figure 1-4.

Output voltage of GP2Y0A41 at the distance of 30 centimetres, the power supply at +5V as in the range of 0.25 to 0.55V has the mean as 0.4V and the range of output voltage change at the distance of 4 centimetres is 2.25V±0.3V.



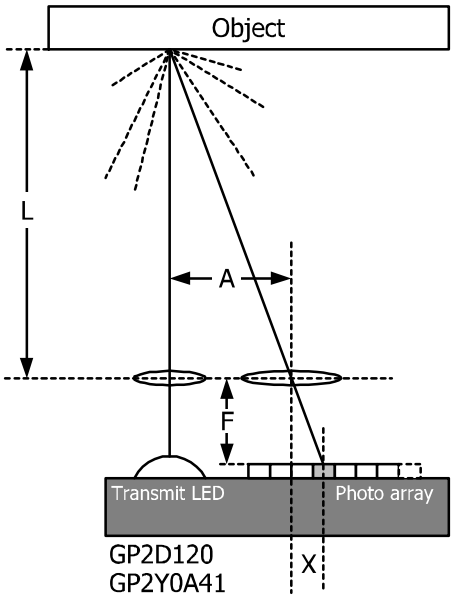




* Use Kodak R-27 gray-white paper. The white side has a 90% reflection rate, made from a material that reflects light for range measurement.

How to measure distance

The infrared light is sent out from a transmitter to the object in front, by passing through a condense lens so that the light intensity is focused on a certain point. Refraction occurs once the light hits the surface of the object. Part of the refracted light will be sent back to the receiver end, in which another lens will combine these lights and determine the point of impact. The light will then be passed on to an array of photo-transistors. The position in which the light falls can be used to calculate the distance (L) from the transmitter to the obstacle using the following formula:



$$\frac{L}{A} = \frac{F}{X}$$

Therefore, L equals

$$L = \frac{F \times A}{X}$$

Thus, the distance value from the phototransistors will be sent to the Signal Evaluation Module before it is changed to voltage, resulting in a change of voltage according to the measured distance.

Figure 1-4 : shown shape, arrangement of pins, diagram of time of operation, and graph shown the operation of GP2D120/GP2Y0A41

1.6 Mechanical compoments

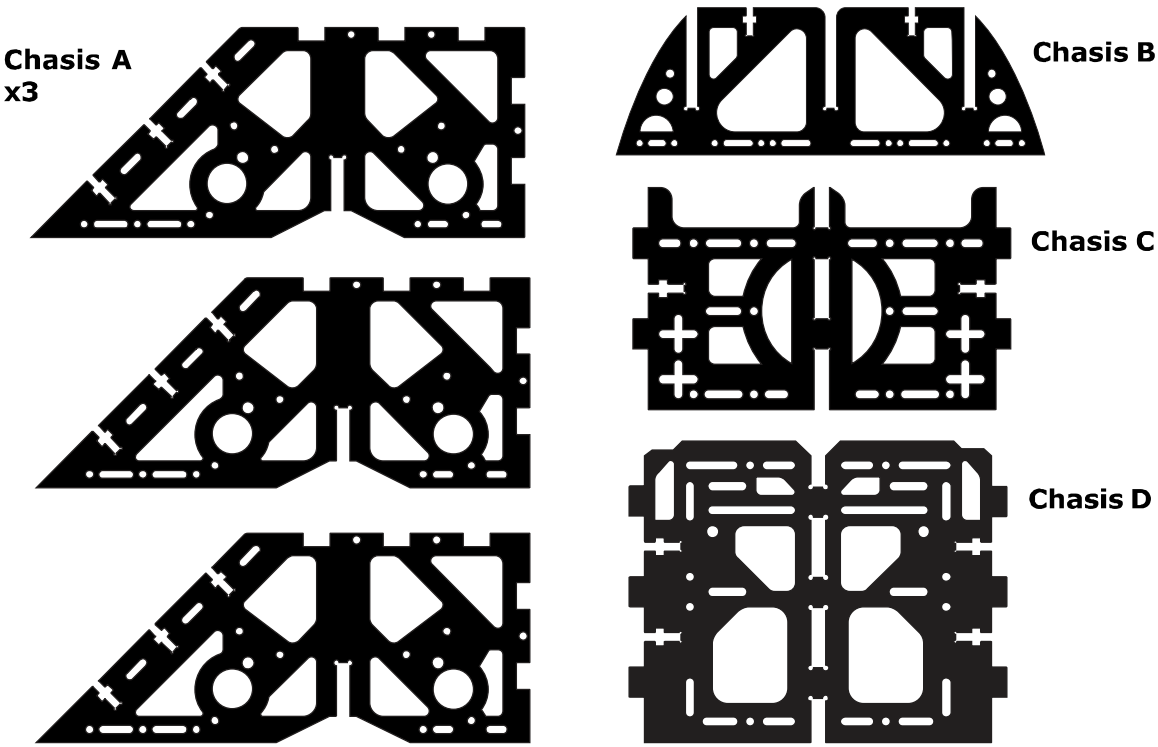
1.6.1 Sport wheel, Tire and Hub

The wheels are unique. The wheel surface is real rubber.

To install this kind of plastic wheels with the axis of the gear motor of model BO-1 will be required to use the wheel hub to help and attach with a screw and 3 mm nuts.

1.6.2 Sumo Chasis part set

They are laser-cut acrylic parts with 5mm. thickness and black color. In one set includes 4 kinds of part : Chasis A (x3), Chasis B, Chasis C and Chasis D.



1.6.3 Grid plates

Plates are plastic manufactured from ABS materials in the size of 80 x 60 mm and 80 x 80 mm each. Each hole has the size of 3 mm and the distance between each hole is 5 mm.

1.6.4 Plastic joiners

They are stiffed plastic components and there are three designs, including, straight joiners, right angle joiners, and obtuse angle joiners. Each piece can be inserted together and they are used to construct a decorated structure (or decoration). A set contains all three types, available 5 colors and 30 pieces in total.

1.6.5 Plastic strip joiners

They are rigid and tough and there are holes in the size of 3 mm for each piece for installing or connecting with other structure components by a screw. At the end of each rod can insert with plastic joiners. There are three different sizes including 3, 5 and 12 holes and each size has 4 pieces in a set.

1.6.6 Metal bracket

Metal angle bars are metal components with the width of 7.5 mm and they are cut into the right angle shape. There are holes with the size of 3 mm for using a screw to install or construct with other structure parts. Three different sizes are provided in a set, including 1x2 holes, 1x2 holes, and 2x5 holes and each size contains 4 pieces.

1.6.7 Screws and nuts

Screws and nuts are equipment for fastening many components together. They compose of 2 mm-tapping screws (4 pieces), 3x10 mm. screw (10 pieces), 3x15 mm. screw (8 pieces), 3x30 mm. screw (8 pieces), 3x12 mm. flat head screws (20 pieces), 3x20 mm. thumb screws (3 pieces) and 3mm. nut (40 pieces).

1.6.8 Metal standoff

This kind of materials helps to hold different components together and support boards, grid plates and base plates. They are made of rustproof nickle-plated metal and have a characteristic of cylinder with the length of 25 mm. Inside of a cylinder, there is a spiral hole along its body for a 3 mm. screw to fasten.

1.6.9 Plastic stands-off

Plastic stands-off help fasten different parts and prop up boards, grid plates, and base plates. They are made from cohesive ABS plastic but able to be cut. The shape of the rod is cylindrical and there is a hole throughout the rod to insert 3mm. screws. You can get different sizes and number of support poles, including, 3mm. (4 pieces), 10mm. (4 pieces), 15mm. (4 pieces), and 25mm. (4 pieces) in the set.

1.6.10 6AA Battery holder

It is used to carry 6 of AA batteries. It comes with the barrel plug (DC adaptor plug) at the end of connection wires.

1.6.11 Battery (not included in the kit)

The recommended battery for this Sumo-BOT kit is Alkaline or Rechargeable battery such as Ni-MH in AA size. The quantities are 6 pieces. The Lithuim Polymer (Li-Po) 2 cells 7.4V 1000mAH also is recommended for power user.

Chapter 2

Building the Sumo-BOT

2.1 Preparing parts

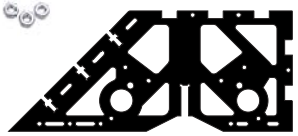
ATX2 controller board

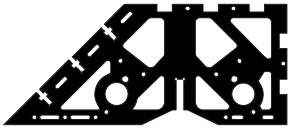
BO-1 DC motor gearbox 87:1 ratio with wires (x4)

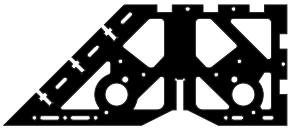
ZX-03 IR Reflector (x2)

GP2Y0A41 distance sensor

Sport wheel set with hub (x4)







Chassis A x3

6AA battery holder with barrel plug

Metal front scoop

Chassis B

Chassis C

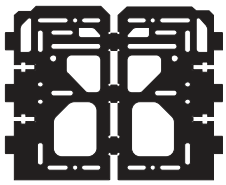
Metal bracket 2x1 (x2)

Metal bracket 2x2

25mm. standoff (x3)

20mm. thumbscrew (x3)

2mm. tap-screw (x4)



Chassis D

3 x 12mm. flat head screw (x8)

3 x 10mm. screw (x8)

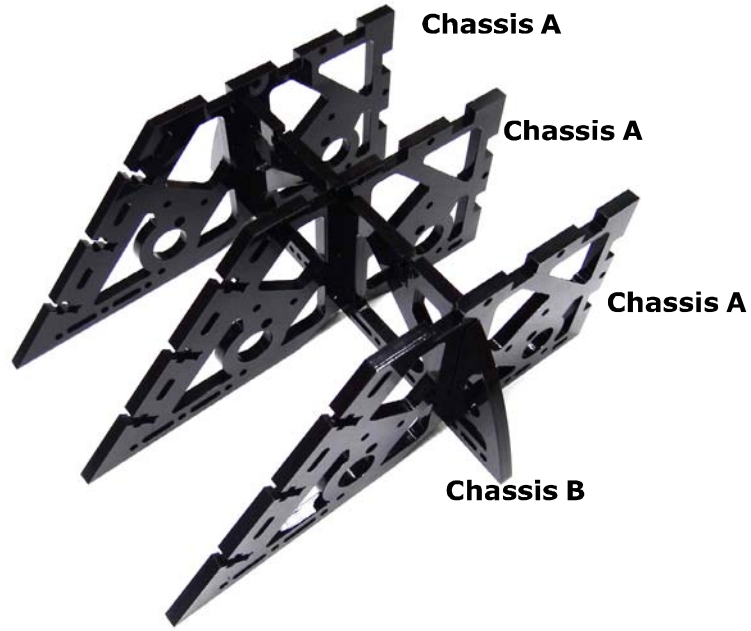
3 x 15mm. screw (x8)

3 x 30mm. screw (x8)

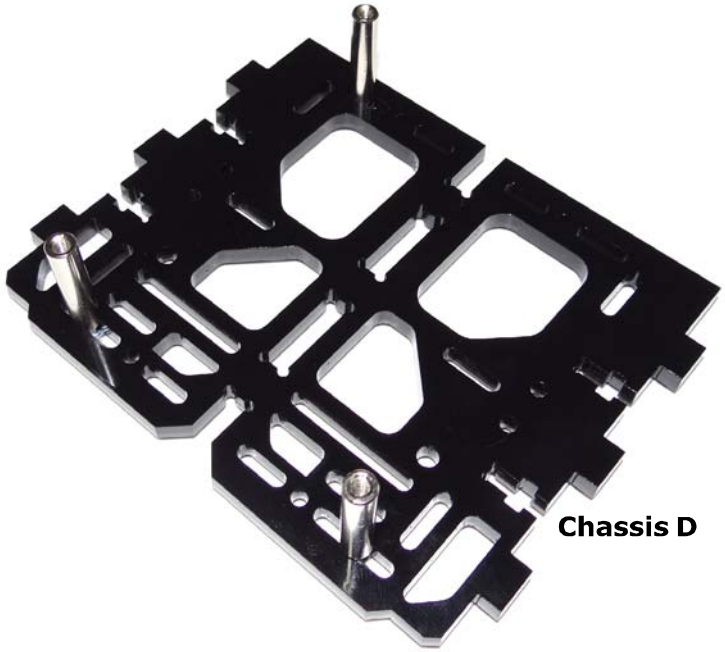
3 mm. nut (x40)

2.2 Assembly steps

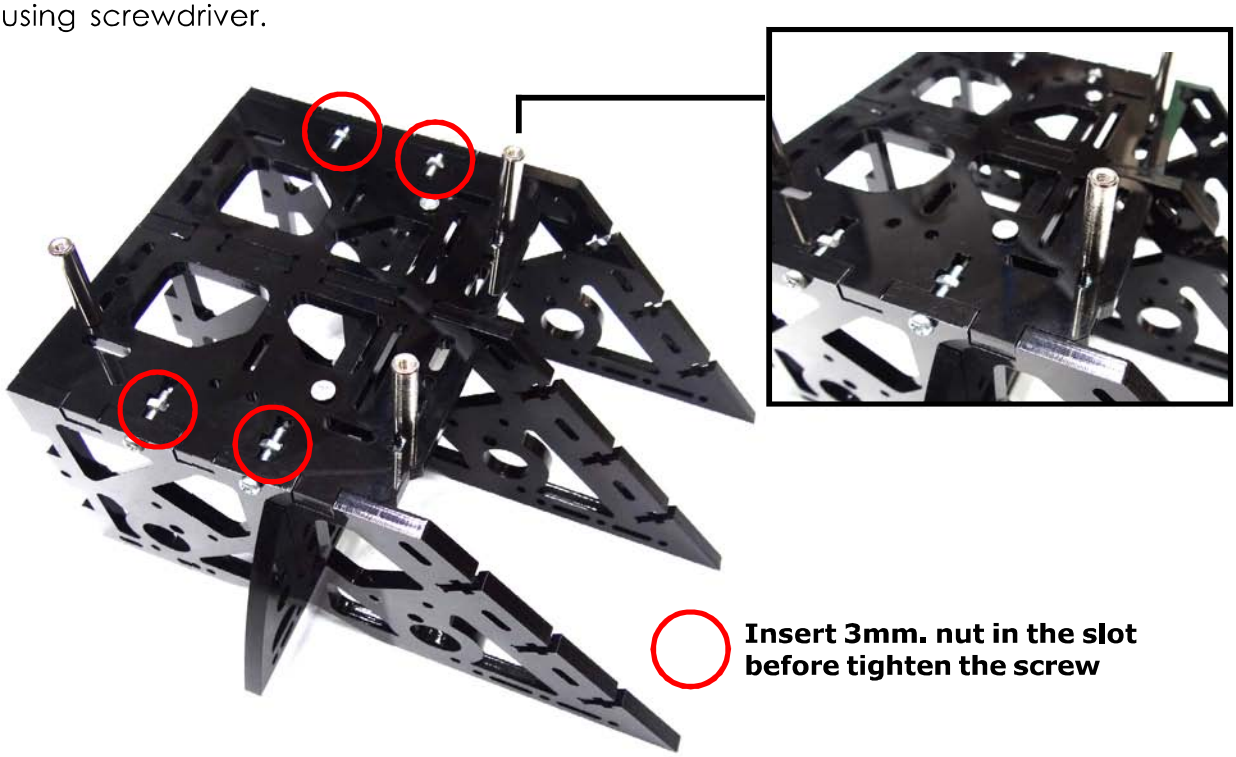
(1) Insert all the chassis part A 3 pieces with the chassis part B following the photo below.



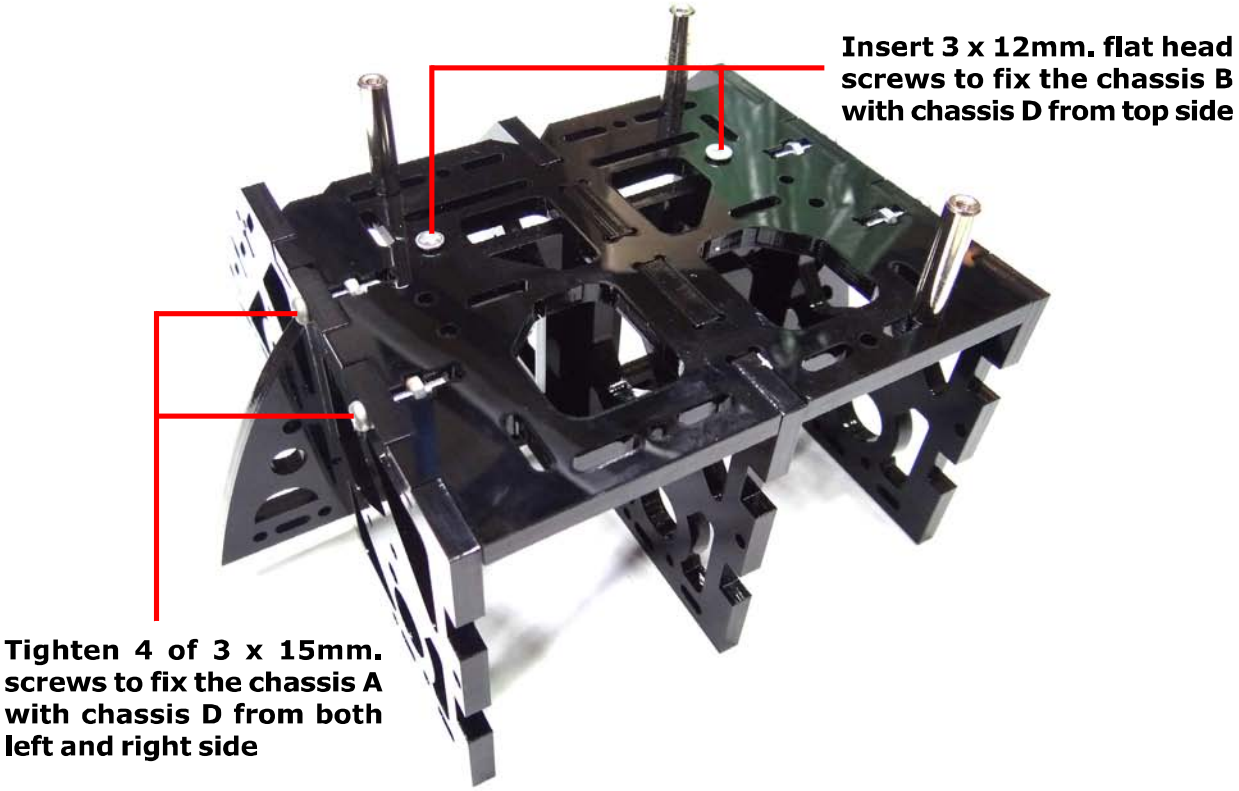
(2) Attach 3 pieces of 25mm. metal standoff at the chassis part D at the position are shown below. Tighten with 3 x 10mm. screws.



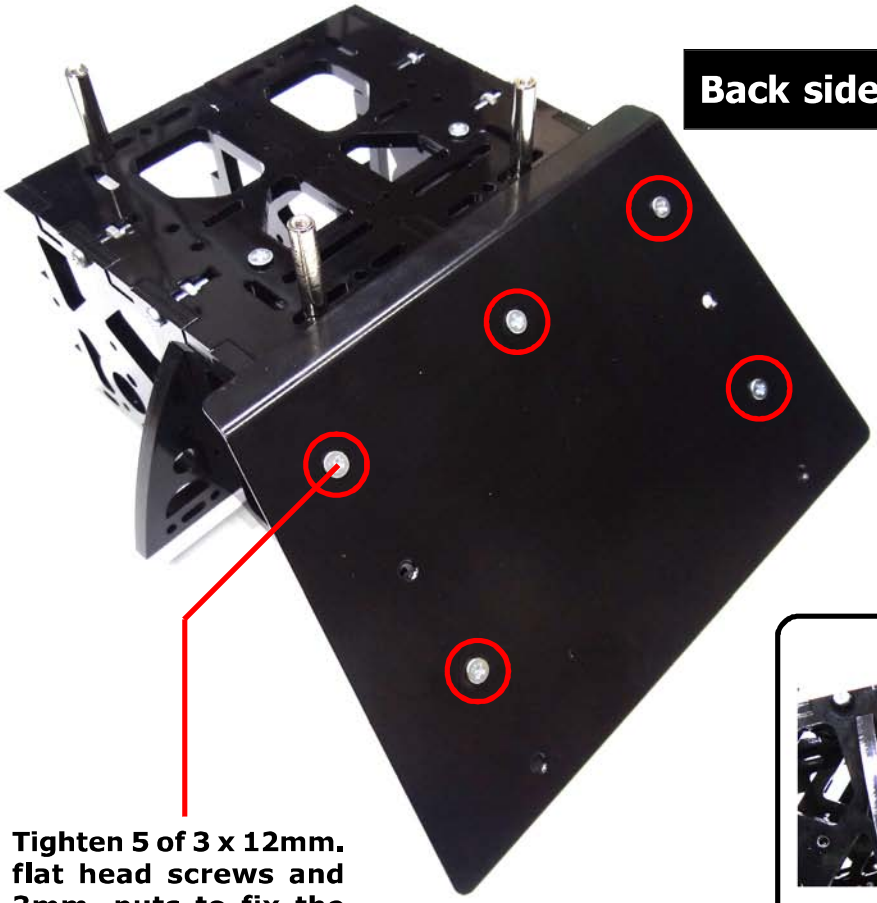
(3) Place the chassis part D from step (2) on the structure from step (1). Tighten together by using 3 x 15mm. screws and 3mm. nuts. For tightening, insert 3mm. nut into the slot first. Then insert 3 x 15mm. screw from beside and tighten via nuts and holes by using screwdriver.



(4) Insert 2 of 3 x 12mm. flat head screws via holes on top of chassis that shown position in the photo and insert 3mm. nuts to tighten.

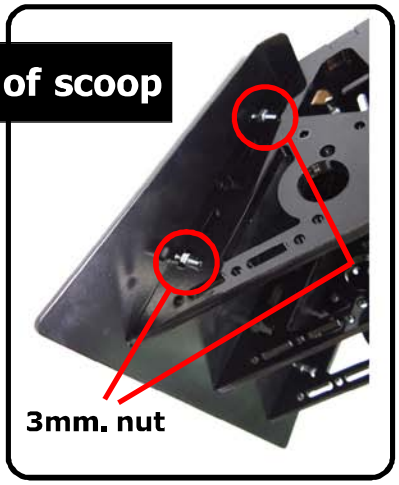


(5) Fix the front scoop with the main chassis by using 5 of 3 x 15mm. flat head screws and 3mm. nuts.

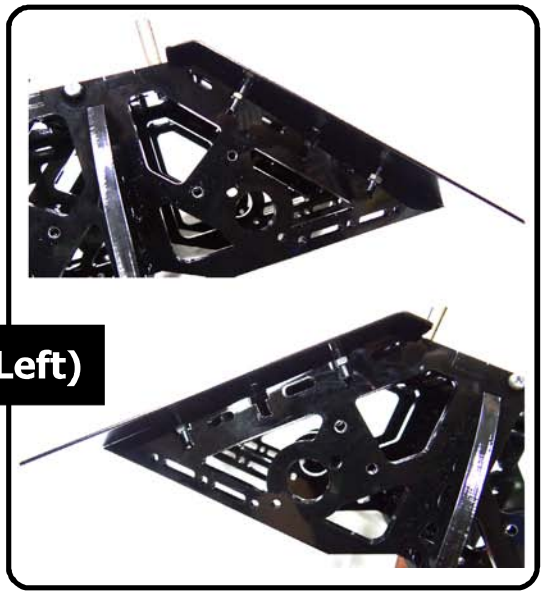


Tighten 5 of 3 x 12mm. flat head screws and 3mm. nuts to fix the front scoop with the robot chassis

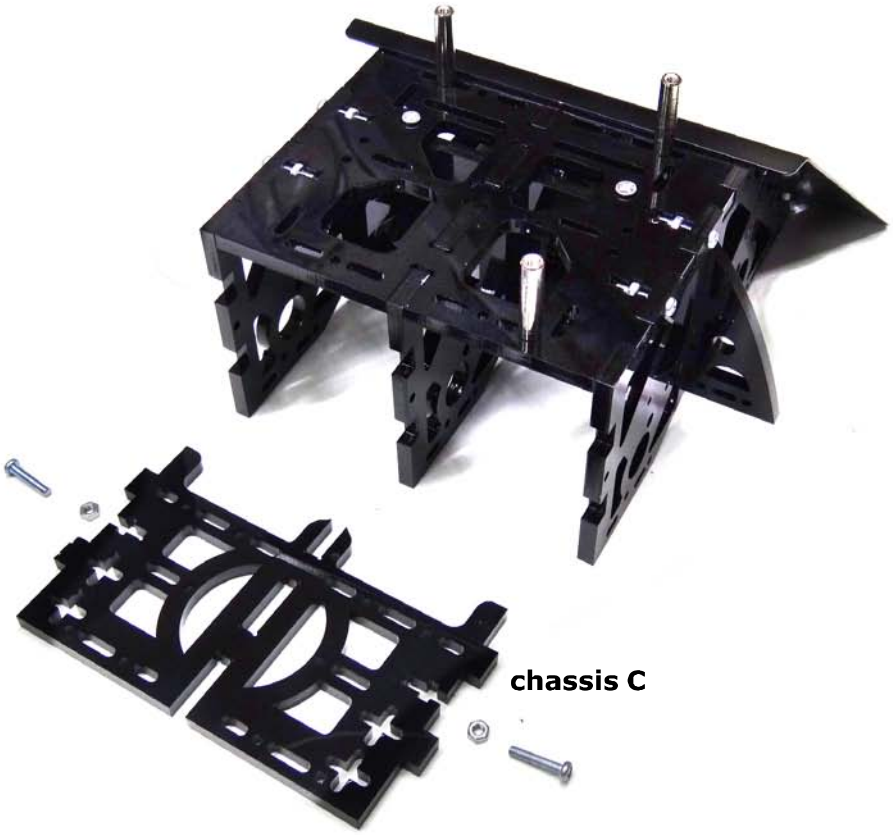
Back side of scoop

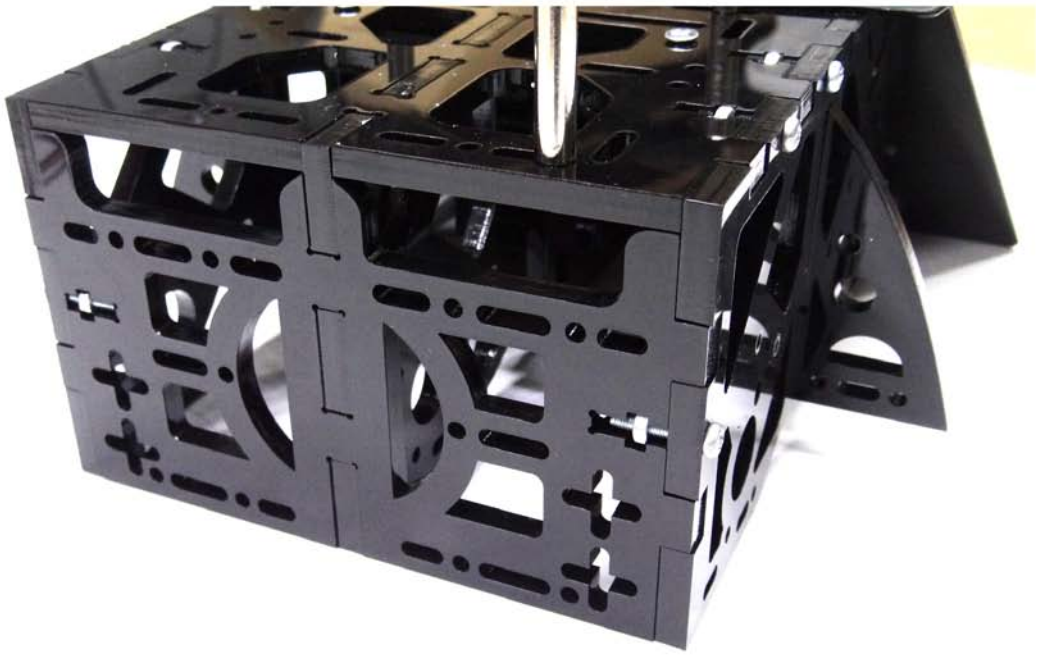


Side view (Right / Left)



(6) Attach the chassis part C with the main chassis following the photo below. Tighten by using 3 x 15mm. screws and 3mm. nuts.

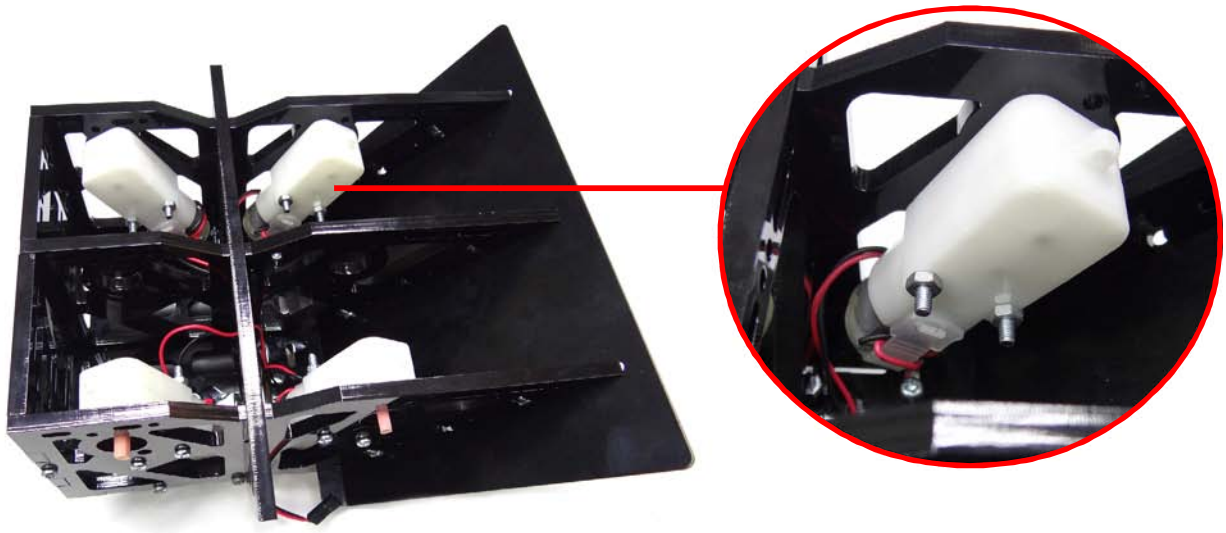




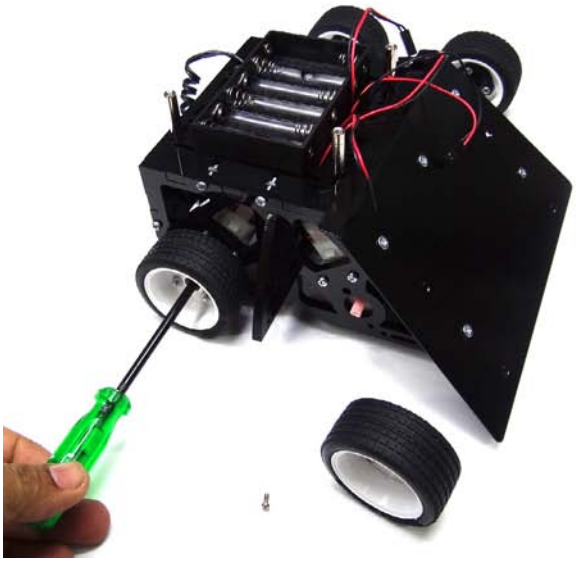
(7) Insert tire with wheel. Fix the wheel hub with wheel by using 3 x 10mm. screw. Do same 4 sets.

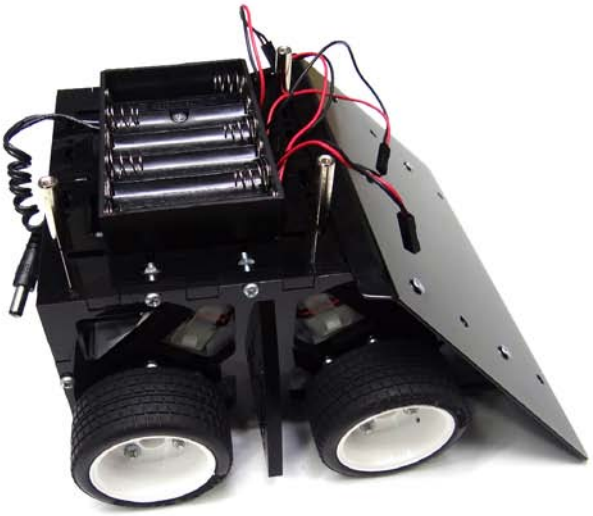
(8) Place the 6AA battery holder on the top of main chassis. Tighten with 3 x 12mm. flat head screws and 3mm. nuts.

(9) Attach the DC motor gearbox within the chassis following the photo below. Tighten by using 3 x 30mm. screws and 3mm. nuts.

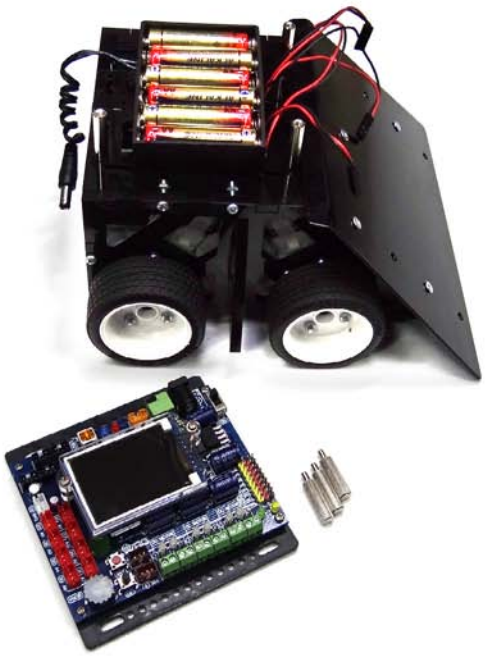


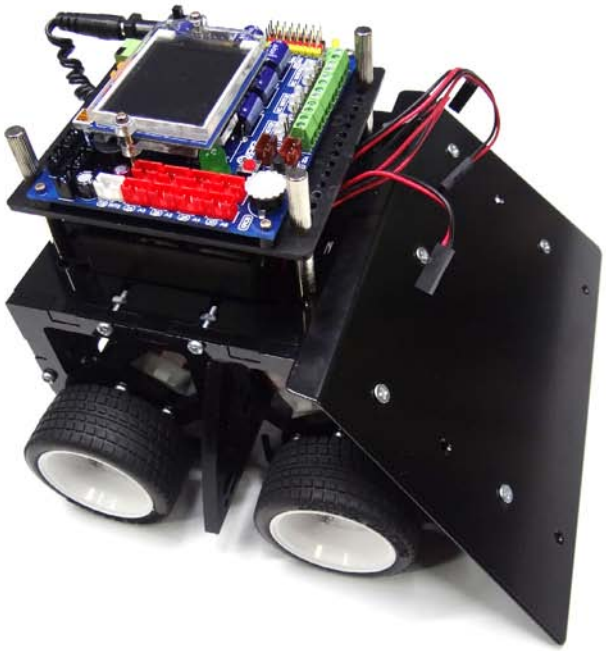
(10) Insert all motor wires via hole to top of chassis. Next, insert wheels from step (7) to shaft of DC motor gearboxes. Tighten with 2mm. tapping screw for all 4 wheels.



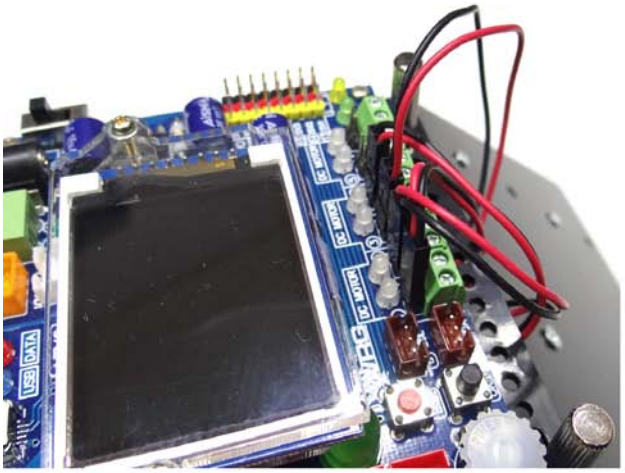


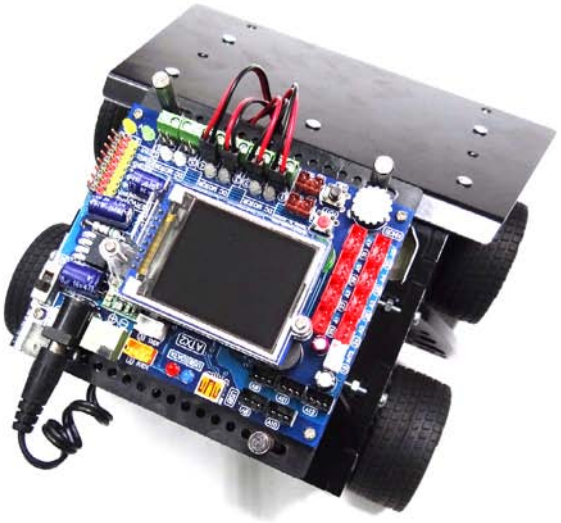
(11) Put 6 of AA batteries into battery holder. Place the ATX2 controller board on the metal standoff over the battery holder. Tighten by using 3 of thumbscrews.



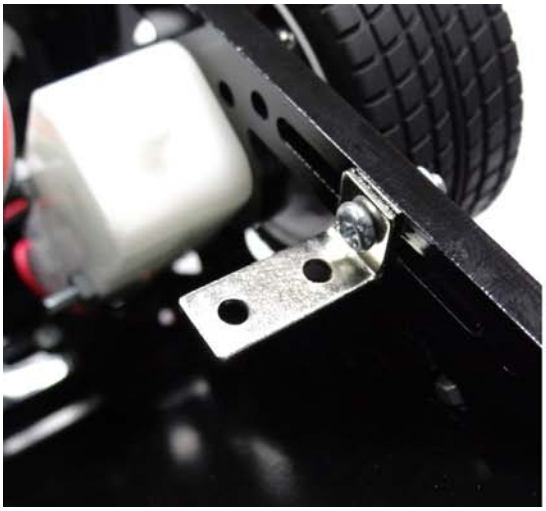


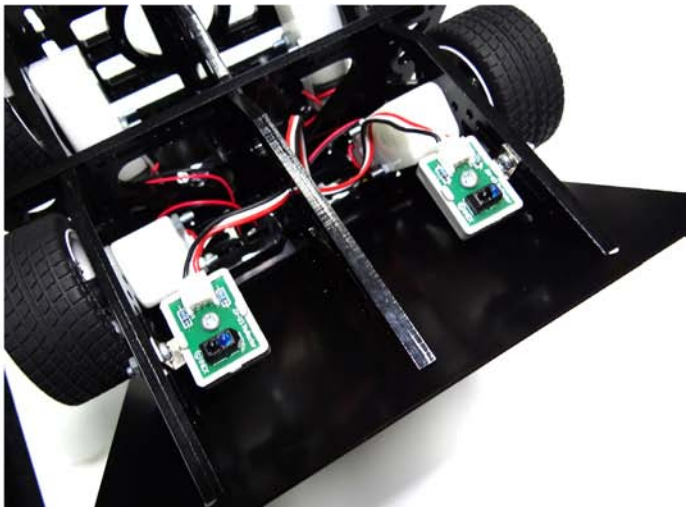
(12) Connect all motor wires to ATX2 board respectively that shown in the photo. Check the motor polarity by turning the wheel of each motor. If turn to forward, LED of motor driver will green. If turn backward, LED will be red. If incorrect, swap the motor wire connections. Have to do all motors.





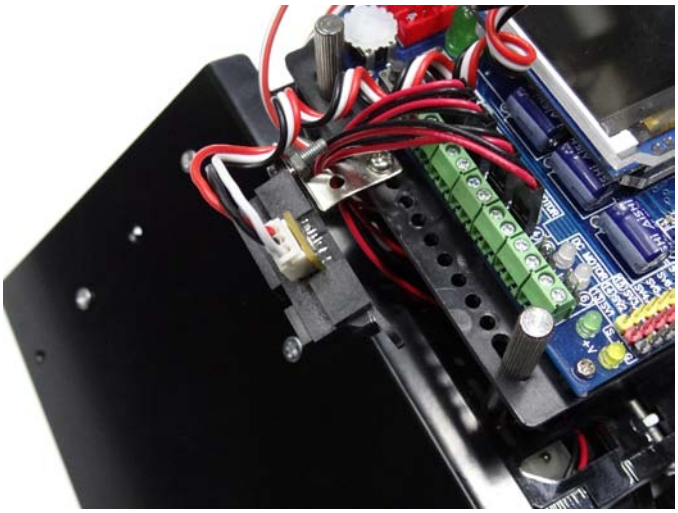
(13) Attach the metal bracket 2x1 with the base of the robot chassis inside by using 3 x 10mm. screw and 3mm. nut. Attach the ZX-03 sensor with this metal bracket from step (13) by using 3 x 10mm. screw and 3mm. nut. Do same 2 sets for both sides.





(14) Attach the metal bracket 2x2 with GP2Y0A41 module by using 3 x 10mm. screw and 3mm. nut.

(15) Next, attach the GP2Y0A41 structure from step (15) with the base of ATX2 board by using 3 x 10mm. screw and 3mm. nut.



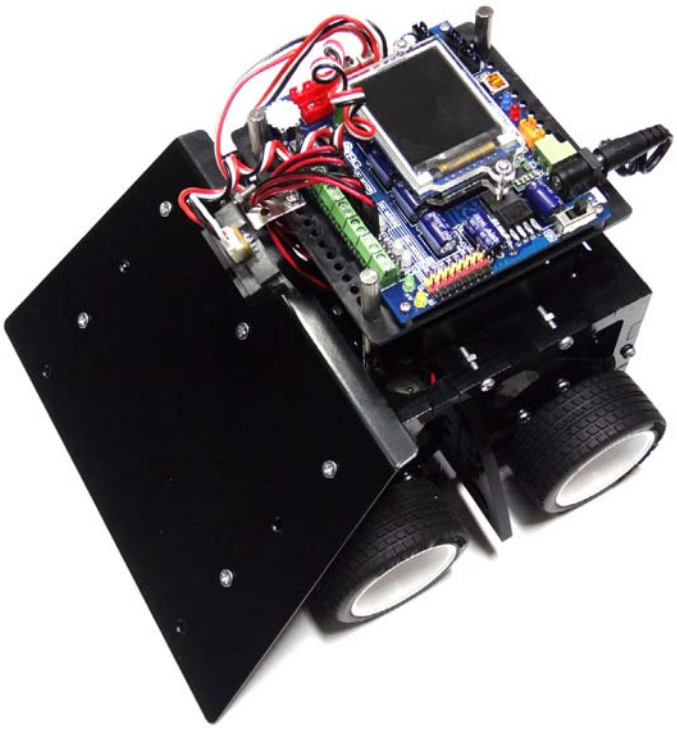
(16) Connect the GP2Y0A41 and ZX-03 sensor cable to ATX2 controller board as follows :

Left ZX-03 sensor : A0 pin of the ATX2 board

Right ZX-03 sensor : A1 pi of the ATX2 board

Gp2Y0A41 sensor : A2 pin of the ATX2 board

Now the Sumo-BOT ready for programming, test, run and FUN !!!



Chapter 3

Sumo-BOT simple movement

After assembly the Sumo-BOT already, it will be the part of understanding to the mechanical structure of Sumo-BOT robots and how to move it. The movement of Sumo-BOT robots is different from familiar two wheeled mobile robots because a Sumo-BOT has 4 sets of DC motor gearbox and wheels. Therefore, programming to drive robots also needs to be considered the functions of all 4 motors. In this chapter, the content will be presented with examples of programming under C/C++language of Arduino to manage Sumo-BOT to move in the basic patterns, either moving straight, backward and turning or turning around in various ways.

3.1 Mechanical structure of AT-BOT

A Sumo-BOT robot has 4 sets of driving wheels and they are arranged according to the positions as in the figure 3-1. Generally, driving Sumo-BOT robots with all 4 sets of driving wheels can be efficient based on one factor you need to consider. The factor is the friction between touching surface and wheels of a robot. In Sumo-BOT, we will use the 87:1 DC motor gearboxes and wheels. This will make motion good and fast on virtually all surfaces. Torque obtained from the motor sets will be a lot and enough to drive a robot woth more torque for Sumo competitions.

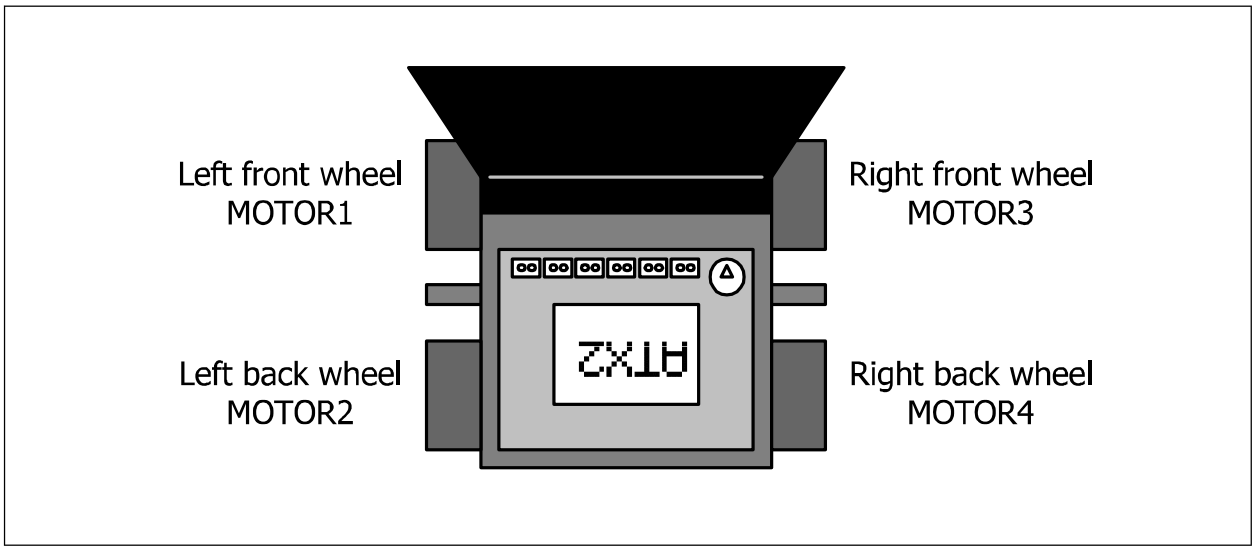


Figure 6-1 : Illustration of determination of motors and wheels positions of the Sumo-BOT

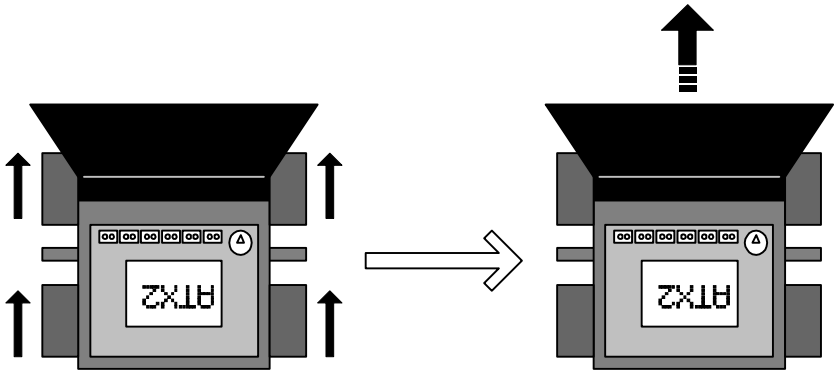
Connection of each motor with a DC motor driving circuit on the ATX2 board of a Sumo-BOT robot detailed as follows

- **The left front wheel is connected with the Motor1output.**
- **The left back wheel is connected with the Motor2output**
- **The right back wheel is connected with the Motor3output**
- **The right front wheel is connected with the Motor4 output**

3.2 Principles of the movement of AT-BOT

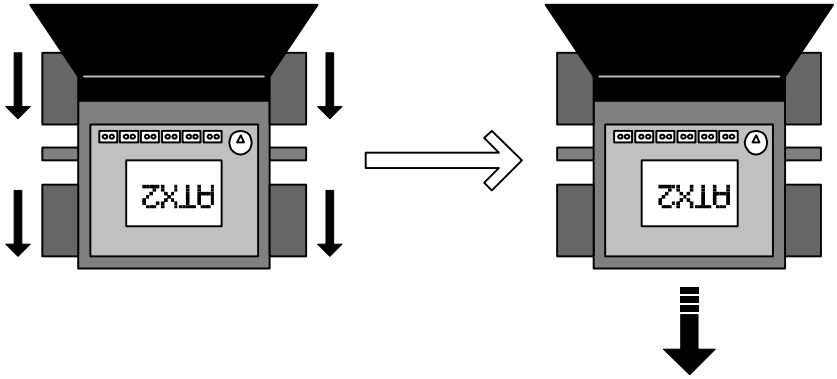
3.2.1 Moving forward

To drive a Sumo-BOT to move forward can be done by controlling the DC motor gearboxes of all 4 wheels to turn in the direction that forces the robot to move forward with a stable driving power, such as driving with the power of 70% or 100%, etc. to not cause turning around.



3.2.2 Moving backwards

To drive a Sumo-BOT to move backwards can do by handling the DC motor gearbox of all 4 wheels to turn in the direction that forces the robot to move backwards with a stable driving power.

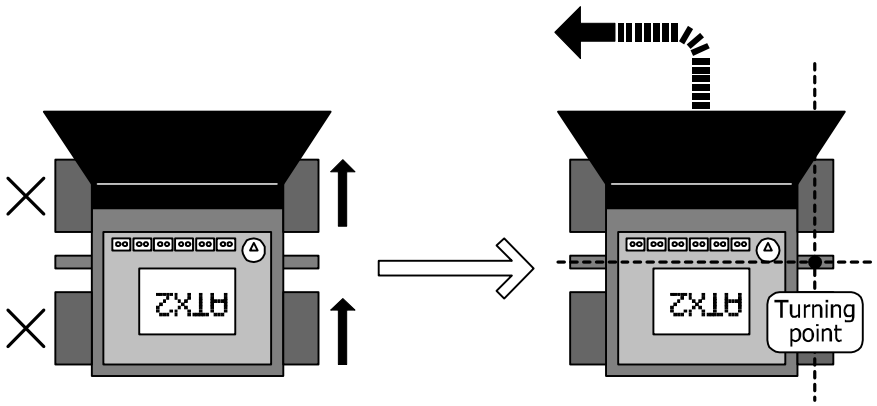


3.2.3 Turning to the left

There are two formats of the movement, including turning with two wheels and turning around.

3.2.3.1 Turning to the left with two wheels

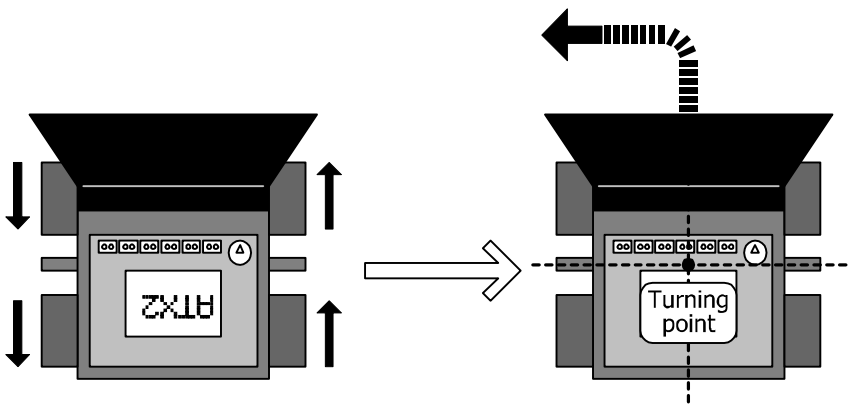
This movement uses 2 sets of motors and wheels in driving a Sumo-BOT to turn left by controlling the right front wheel and the right back wheel to rotate forward. Meanwhile, the left front wheel and the left back wheel are stopped, so the robot will be able to turn left and the rotation point is between the left front wheel and the left back wheel as the diagram below.



3.2.3.2 Turning to the left by spining

To drive a Sumo-BOT robot to turn left with another pattern is available by managing the right front wheel and the right back wheel turning forward but the left front wheel and the left back wheel will turn backwards or in the opposite direction. Therefore, Sumo-BOT will turn left with spining and the turning point is at the middle of the robot's body.

Turning itself to the left by this method will give a high power of the movement but the robot needs more energy as well.

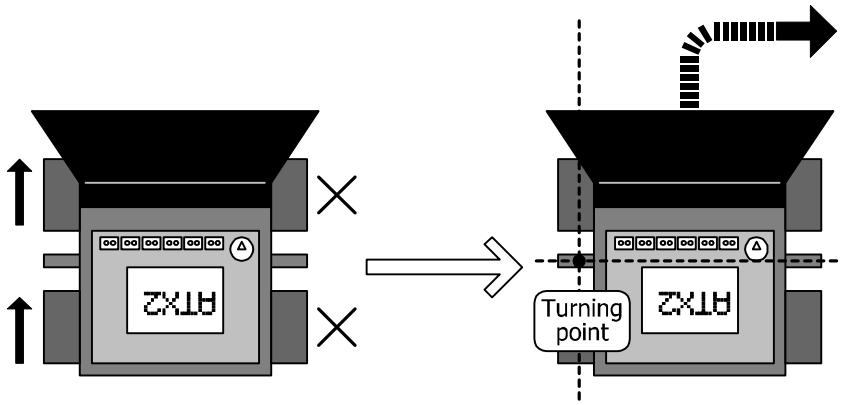


3.2.4 Turning to the right

There are 2 formats of the movement, including turning by 2 wheels and spining.

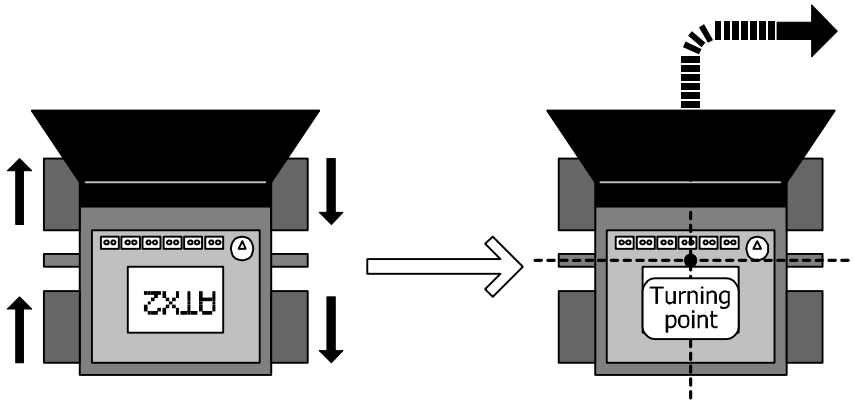
3.2.4.1 Turning to the right by two wheels

To make Sumo-BOT robot move in this pattern of turning to the right will be done differently from turning left. It is said that controlling the left front wheel and the left back wheel to turn forward but the right front wheel and the right back wheel to stop. Therefore, the robot will be able to turn left and has the rotation point between the right front wheel and the right back wheel.



3.2.4.2 Turning to the right by spining

This driving will be done in the opposite way with turning to the left of the pattern 2 (topic 3.2.4). By controlling the left front wheel and the left back wheel to turn forward but the right front wheel and the right back wheel to be turn backward. The Sumo-BOT will turn right with spining and the turning point is at the middle of the robot's body. With this method will give a high power in the movement but need to use more energy as well.



3.3 DC motor function for Sumo-BOT programming

The programming language C / C + + to drive the motors of the Sumo-BOT robot has been made easier with DC motor function in the ATX2.h library file. Summary of all functions are as follows :

`motor(_channel,_power);` Select the motor output and set speed.

`motor_stop(_channel);` Stop motor operation

`fd(speed);` Move forward

`bk(speed);` Move backward

`fd2(speed);` Move forward

`bk2(speed);` Move backward

`tl(speed);` Turn left

`tr(speed);` Turn right

`sl(speed);` Spin left

`sr(speed);` Spin right

`ao();` Stop all motors

`FD(speed);` Move forward for 4 wheels robot (such as Sumo-BOT)

`BK(speed);` Move backward for 4 wheels robot (such as Sumo-BOT)

`FD2(speed);` Move forward for 4 wheels robot (such as Sumo-BOT)

`BK2(speed);` Move backward for 4 wheels robot (such as Sumo-BOT)

`TL(speed);` Turn left for 4 wheels robot (such as Sumo-BOT)

`TR(speed);` Turn right for 4 wheels robot (such as Sumo-BOT)

`SL(speed);` Spin left for 4 wheels robot (such as Sumo-BOT)

`SR(speed);` Spin right for 4 wheels robot (such as Sumo-BOT)

`AO();` Stop all 4 motors for 4 wheels robot (such as Sumo-BOT)

3.3.1 General motor function

There are 2 functions. They are concist of **motor()** and **motor_stop()**. These function are used to control all motor outputs of the ATX2 controller board. Programmer can control each motor output independent.

3.3.1.1 motor

Drive the DC motor function for any output of ATX2 controller board.

Syntax

```
void motor(char _channel,int _power)
```

Parameter

_channel - DC motor output. Includes :

1 to 6 for each channel 1, 2, 3, 4, 5 or 6

12 for channel 1 and 2

34 for channel 3 and 4

56 for channel 5 and 6

100 or ALL or ALL4 for channel 1 to 4

106 or ALL6 for all channels (1 to 6)

_power - Power value. It is -100 to 100

If set **_power** as positive value (1 to 100), motor moves forward

If set **_power** as negative value (-1 to -100), motor moves backward

If set as 0, motor stop but not recommended. Please choose the **motor_stop** function better.

Example 3-1

```
motor(1,60);        // Drive motor A with 80% power
motor(1,-60);       // Drive motor A backward with 80% power
```

Example 3-2

```
motor(2,100);    // Drive motor B with 100% power
```

3.3.1.2 motor_stop

Stop motor driving function

Syntax

```
void motor_stop(char _channel)
```

Parameter

_channel - DC motor output. Includes :

1 to 6 for each channel 1, 2, 3, 4, 5 or 6

12 for channel 1 and 2

34 for channel 3 and 4

56 for channel 5 and 6

100 or ALL or ALL4 for channel 1 to 4

106 or ALL6 for all channels (1 to 6)

Example 3-3

```
motor_stop(1);          // Stop motor1
motor_stop(2);          // Stop motor2
```

Example 3-4

```
motor_stop(ALL);        // Stop motor1 to motor4
motor_stop(ALL6);       // Stop motor1 to motor4
```

3.3.2 Motor function for 2-wheel robot

For helping the beginner to create the sketch to control the autonomoius robot, the ATX2.h library provides many functions for supporting about driving the 2-wheel robot. User can construct any 2-wheel robots and use these functions to drives the robot movement.

In these functions determine the motor output connection as follows :

1. **Left wheel motor is conneted with Motor1 output of the ATX2 board**
2. **Right wheel motor is connected with Motor2 output of the ATX2 board.**

Information of all function that support the 2-wheel robot that created from ATX2 controller board or Robo-Creatror2 robot kit is described as follows.

3.3.2.1 fd

Move forward function. It is to move the robot forward.

Syntax

```
fd(unsigned int speed)
```

Parameter

speed - percentage of motor power (0 to 100%)

Example 3-5

```
fd(60); // Robot moves forward with 60% power
```

3.3.2.2 fd2

This full name of this function is forward2. It is a dependent function of the motor control command for forward.

Syntax

```
fd2(unsigned int speed1, unsigned int speed2)
```

Parameter

speed1 - Speed of motor1 (0 to 100%)

speed2 - Speed of motor2 (0 to 100%)

Example 3-6

```
fd2(30,80);   // Move the robot in circle shape.
              // Because the speed of motor2 is greater than motor1
```

3.3.2.3 bk

Move backward function. It is to move the robot backward.

Syntax

bk(unsigned int speed)

Parameter

speed - percentage of motor power (0 to 100%)

Example 3-7

```
bk(90);        // Robot moves backward with 90% power
```

3.3.2.4 bk2

This full name of this function is backward2. It is a dependant function of the motor control command for backward.

Syntax

bk2(unsigned int speed1 ,unsigned int speed2)

Parameter

speed1 - Speed of motor 1 (0 to 100%)

speed2 - Speed of motor 2 (0 to 100%)

Example 3-8

```
bk2(80,80);      // Move backward both motor with same power
```

Both **fd2()** and **bk2()** functions help and adjust the motor speed for improving the robot moving. Normally each motor speed is not equal 100%. It is some different. With these function, user possible to adjust the different speed for each motor to adjust the moving direction to more straight.

3.3.2.5 tl and tr

They are from turn left (tl) and turn right (tr). The turning method is stop one motor and move another one. The tunring point is the stop wheel.

Syntax

tl(unsigned int speed) / tr(unsigned int speed)

Parameter

speed - Speed of motor (0 to 100%)

Example 3-9

```
tl(60);          // Turn left with 80% power
tr(100);         // Turn right with full speed
```

3.3.2.6 sl and sr

They are from spin left (sl) and spin right (sr). This function control each motor turn in opposit direction. The turning point is the center of robot

Syntax

sl(unsigned int speed) / sr(unsigned int speed)

Parameter

speed - Speed of motor (0 to 100%)

Example 3-10

```
sl(70);          // Root spin left with 70% power
sr(100);         // Robot spin right with full speed
```

3.3.2.7 ao

It is to off all motor function. Just off only Motor1 and Motor2 ouputs.

Syntax

ao()

Example 3-11

```
void setup()
{
   fd(100);      // Robot moves forward with full speed
   sleep(2000);  // in 2 seconds
   ao();         // Stop all motors (only motor1 and 2)
}
```

3.3.3 Motor function for 4-wheel robot

For helping the beginner to create the sketch to control the autonomoius robot, the ATX2.h library provides many functions for supporting about driving the 4-wheel robot. The Sumo-BOT programmer also can use these functions to control the Sumo-BOT easier.

Again; all these functions determine the motor output connection as follows :

- **The left front wheel is connected with the Motor1output.**
- **The left back wheel is connected with the Motor2output**
- **The right back wheel is connected with the Motor3output**
- **The right front wheel is connected with the Motor4 output**

Information of all function that support the 4-wheel robot that created from ATX2 controller board or Robo-Creatror2 robot kit is described as follows.

3.3.3.1 FD

Move forward function. It is to move the robot forward.

Syntax

```
fd(unsigned int speed)
```

Parameter

speed - percentage of motor power (0 to 100%)

Example 3-5

```
FD(60); // Robot moves forward with 60% power
```

3.3.3.2 FD2

This full name of this function is FORWARD2 for 4-wheel robot. It is a dependent function of the motor control command for forward.

Syntax

```
fd2(unsigned int speed1, unsigned int speed2)
```

Parameter

speed1 - Speed of motor1 and motor2 (0 to 100%)

speed2 - Speed of motor3 and motor4 (0 to 100%)

Example 3-6

```
FD2(30,80);   // Move the robot in circle shape.
              // Because the speed of pair of motor1 and motor2 is
              // greater than a pair of motor3 and motor4
```

3.3.3.3 BK

Move backward function. It is to move the robot backward.

Syntax

`BK(unsigned int speed)`

Parameter

`speed` - percentage of motor power (0 to 100%)

Example 3-7

```
BK(90);       // Robot moves backward with 90% power
```

3.3.3.4 BK2

This full name of this function is BACKWARD2 for 4-wheel robot. It is a dependent function of the motor control command for backward.

Syntax

`BK2(unsigned int speed1 ,unsigned int speed2)`

Parameter

`speed1` - Speed of motor1 and motor2 (0 to 100%)

`speed2` - Speed of motor3 and motor4 (0 to 100%)

Example 3-8

```
BK2(80,80);      // Move backward both motor with same power
```

Both **`FD2()`** and **`BK2()`** functions help and adjust the motor speed for improving the robot moving. Normally each motor speed is not equal 100%. It is some different. With these function, user possible to adjust the different speed for each motor to adjust the moving direction to more straight.

3.3.3.5 TL and TR

They are from Turn Left (TL) and Turn Right (TR). The turning method is stop a pair of motor and move another pair. The tunring point is the stop wheels.

Syntax

`TL(unsigned int speed) /     TR(unsigned int speed)`

Parameter

`speed` - Speed of motor (0 to 100%)

Example 3-9

```
TL(60);          // Turn left with 60% power
TR(100);         // Turn right with full speed
```

3.3.3.6 SL and SR

They are from Spin Left (SL) and spin right (sr). This function control each motor turn in opposit direction. The turning point is the center of robot.

Syntax

SL(unsigned int speed) / SR(unsigned int speed)

Parameter

speed - Speed of motor (0 to 100%)

Example 3-10

```
SL(70);          // Root spin left with 70% power
SR(100);         // Robot spin right with full speed
```

3.3.3.7 AO

It is to off all motor function. Just off only Motor1 to Motor4 ouputs.

Syntax

AO()

Example 3-11

```
void setup()
{
   FD(100);      // Robot moves forward with full speed
   sleep(2000);  // in 2 seconds
   AO();         // Stop all 4 motors (only motor1 to motor4)
}
```

Robot activity 1 : Sumo-BOT simple movement

(R1.1) Open the Arduino 1.0.7 software. Create the new sketch file. Type the code following the Listing R1-1. then save as the sketch.

(R1.2) Apply the supply voltage to the Sumo-BOT and turn on power. Connect the USB cable between Sumo-BOT and computer.

(R1.3) Compile and upload the sketch to the Sumo-BOT.

The robot displays the welcome message and waits for OK button pressing. For the first pressing the OK button, Sumo-BOT moves forward for 1 second then stop. It waits for the OK button pressing for running next function.

Sumo-BOT will move forward, backward, spin left, spin right, turn left And turn right each for 1 second and stop to wait for next activation.

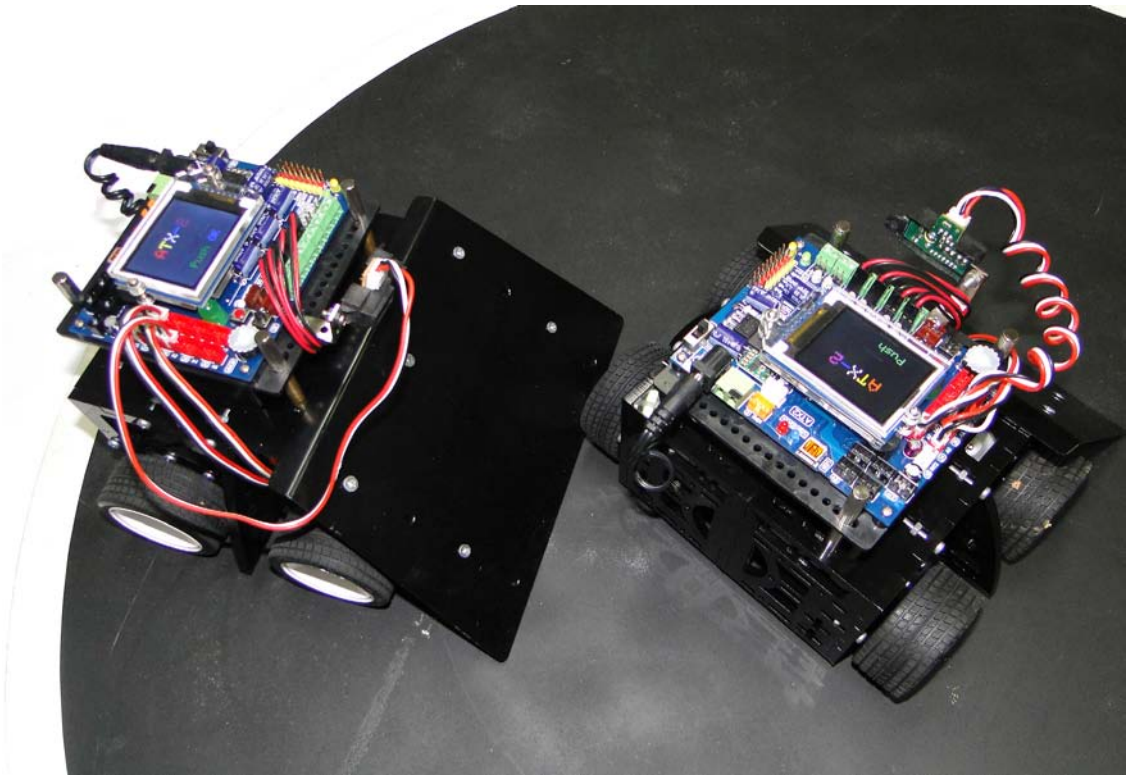
This activity demonstrates the movement function of Sumo-BOT. User can see each movement after pressing the OK button.

```
#include <ATX2.h>
void setup()
{
   OK();                          // Wait for OK
   glcdClear();
}
void loop()
{
   glcd(2,0,"FORWARD 50%   ");   // Shows operation name
   FD(50);
   delay(1000);
   AO();                          // Sumo-BOT moves forward 1 second then stop
   sw_OK_press();                 // Wait for next activated
   glcd(2,0,"BACKWORD 50%  ");   // Shows operation name
   BK(50);
   delay(1000);
   AO();                          // Sumo-BOT moves backward 1 second then stop
   sw_OK_press();                 // Wait for next activated
   glcd(2,0,"SPIN LEFT 50% ");   // Shows operation name
   SL(50);
   delay(1000);
   AO();                          // Sumo-BOT spins left 1 second then stop
   sw_OK_press();                 // Wait for next activated
   glcd(2,0,"SPIN RIGHT 50%");   // Shows operation name
   SR(50);
   delay(1000);
   AO();                         // Sumo-BOT spins right 1 second then stop
   sw_OK_press();                 // Wait for next activated
   glcd(2,0,"TURN LEFT 50% ");   // Shows operation name
   TL(50);
   delay(1000);
   AO();                         // Sumo-BOT turns left 1 second then stop
   sw_OK_press();                 // Wait for next activated
   glcd(2,0,"TURN RIGHT 50%");   // Shows operation name
   TR(50);
   delay(1000);
   AO();                          // Sumo-BOT turns right 1 second then stop
   sw_OK_press();                 // Wait for next activated
}
```

Listing R1-1 : SumoBOT_SimpleMove.ino; the sketch file for demonstration the movement of Sumo-BOT

Robot activity 2 : Sumo robot games

This activity demonstrates the robot operation for Sumo competition. The Sumo-BOT use the distance sensor to find the competitor. When found, robot will move faster to push the competitor. Moreover, Sumo-BOT have to aware about the white line of Sumo ring. The robot does not move out of the ring. When the robot detects the white line, robot must move to get away from the edge of sumo ring.



(R2.1) Open the Arduino 1.0.7 software. Create the new sketch file. Type the code following the Listing R2-1. then save as the sketch.

(R2.2) Apply the supply voltage to the Sumo-BOT and turn on power. Connect the USB cable between Sumo-BOT and computer.

(R2.3) Compile and upload the sketch to the Sumo-BOT.

(R2.4) Place the Sumo-BOT on the Sumo ring.

```
#include <ATX2.h>          // Include main library
#include <ATX2_gp2d120.h>  // Include distance sensor library
int CL=500,CR=450;         // Set criteria of white line
void setup()
{
  OK();
  delay(5000);             // Delay 5 seconds after start
}
void loop()
{
  SL(60);
  if(analog(0)>CL)         // Left sensor detects the white line
  {
    BK(60);                // Move backward
    delay(300);
    SR(60);                // and spin right
    delay(800);
  }
  if(analog(1)>CR)        // Right sensor detects the white line
  {
    BK(60);                // Move backward
    delay(500);
    SL(60);                // and spin left
    delay(900);
  }
  if(analog(2)>50)        // Detect the competitor
  {
    FD(100);              // Move forward with maximum speed
    delay(100);
  }
}
```

Listing R2-1 : SumoBOT_Sumoe.ino; the sketch file for demonstration the Sumo-BOT play the Sumo robot game

(R2.5) Turn-on power and press OK button.

The robot displays the welcome message and waits for OK button pressing. After pressing the OK button, Sumo-BOT will wait 5 seconds (following the Sumo robot competiton rule). After that, it moves to find the competitor.

For demonstration, use the box or doll for competitor. Please the object on the Sumo ring. When robot found the object, it moves fastest to push the object until it out of the ring. In case robot moves to found the white line at the circumference of sumo ring. Sumo-BOT will move back and spin to find the object or competitor continue.