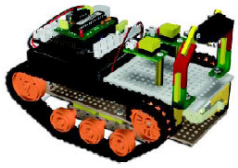
CONTENTS



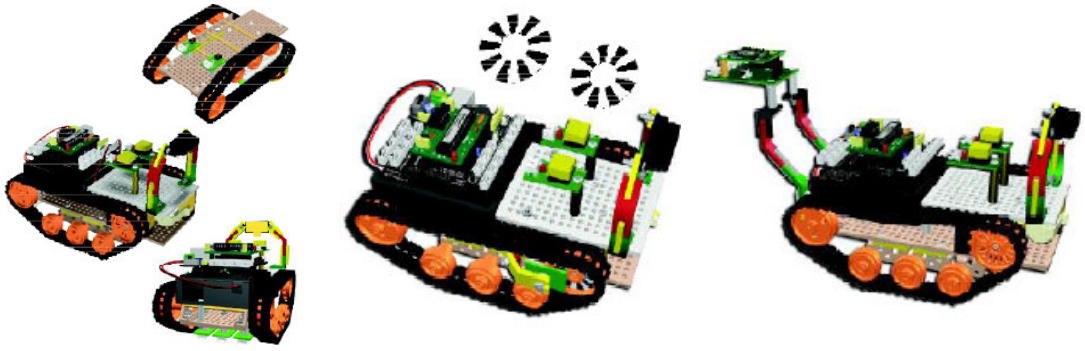
RoboTank

RoboTank: Detecting Objects with Infrared

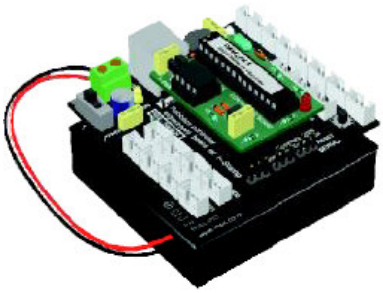
RoboTank: Moving along the line

RoboTank: Accurate Movement through Encoder Wheels

RoboTank: Controlling Movement with an Electronic Compass

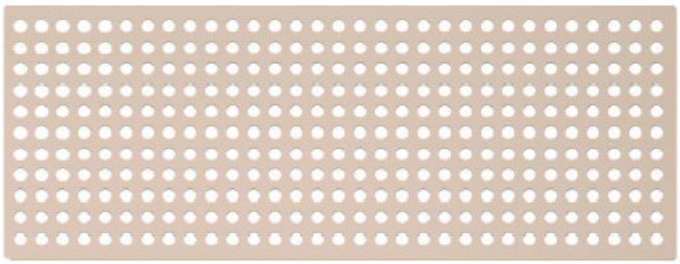


Components used in building a small automatic robot RoboTank

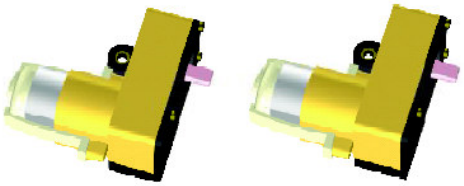


Stamp-BOX Robotic Controller board – This consists of the i-Stamp Basic Stamp2SX microcontroller board, RBX-Stamp24 board and a CX-4 cable to download programs. It has a 16KB memory (divided into 8 pages, each of 2 KB), 8 analog input channels, 7 digital input/output channels, and it drives two DC motors and three RC servo motors.

Box Holder – Used to support the Stamp-Box when attaching it to the robot.



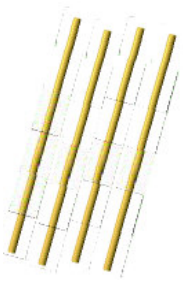
Universal plate – Plate size is 160 x 60 mm.. There is 341 3-mm. size holes with a distance of 5 mm. between each hole.



Motor Gearbox – Uses a 4.5 to 9V and 180 mA DC motor with a ratio of 48:1; torque 4kgF/cm; Two sets are provided.

Track – There are 3 sizes ; Four 8-Joint tracks, Four 10-Joint Tracks and Two 30-joint tracks.

Angled Shaft Base – Two pieces of each the long base and short base are provided.



Metal Axel – 4 mm. in diameter and 100 mm. in length. Four axels come in the set.

Wheels – There are four different types which are Main Sprocket Wheel (2 Pieces), Large Track Support Wheel (2 pieces), Small Track Support Wheel (10 pieces), and hubs (12 pieces).

Grid Plate – Used to install the various sensors

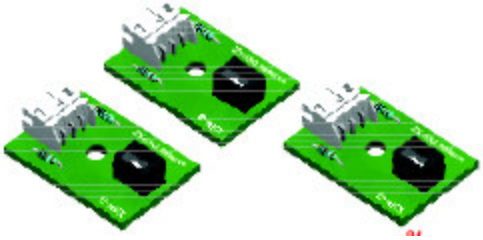
Encoder Wheel Sticker – Attached onto the Main Sprocket Wheel to determine movement.

Angled Joiner – 20 pieces of varied color joiners made from PVC plastic. They can be connected together or by using screws and 3 mm. nuts in installation.

Obtuse Joiner – 20 pieces of varied color 135 degree obtuse joiners made from PVC plastic. They can be connected together or by using screws and 3 mm. nuts in installation.

Strength Joiner – 20 pieces of varied color joiners made from PVC plastic. They can be connected together or by using screws and 3 mm. nuts in installation.

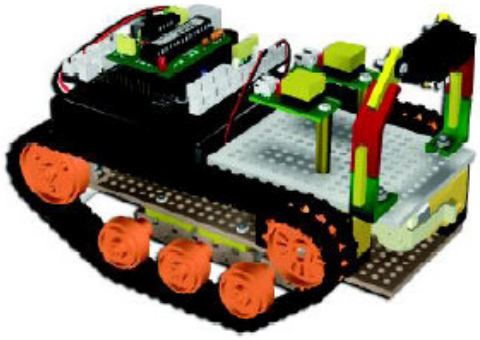
Nut, Screws & Spacers – There are two 2 mm. open-end screws, four 3 x 6 mm. screws, thirty 3x10 mm. screws, four 3 x 25 mm. screws, two 3 x 12 flat-head screws, thirty 3 mm. nuts, four 25 mm. alloy spacers, and four of 3 mm., 10 mm., 15 mm., and 25 mm. size plastic spacers each.



Switch Module – The switch sensor is used to detect collision at logic "0". Two sets along with the connecting cable are provided.

ZX-03Q IR Reflector Module – Used to detect the perimeter, lines, and the wheel code. The results are in voltage. Three sets along with the connecting cable are provided.

GP2D120 IR Ranger Module – Measures distance ranging from 4 to 30 cm using infrared light. The results are in voltage.



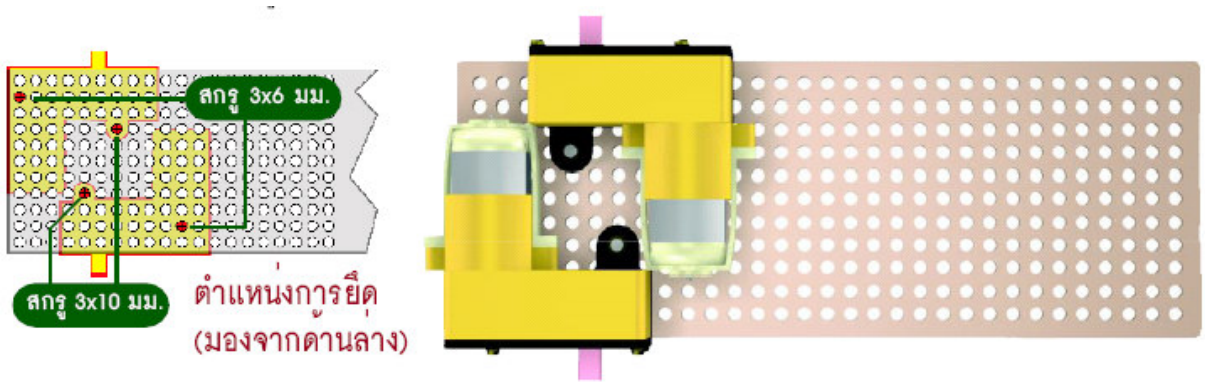
RoboTank Smart Track Wheel Robot

Robo-Stamp 2.0 Smart Robot kit

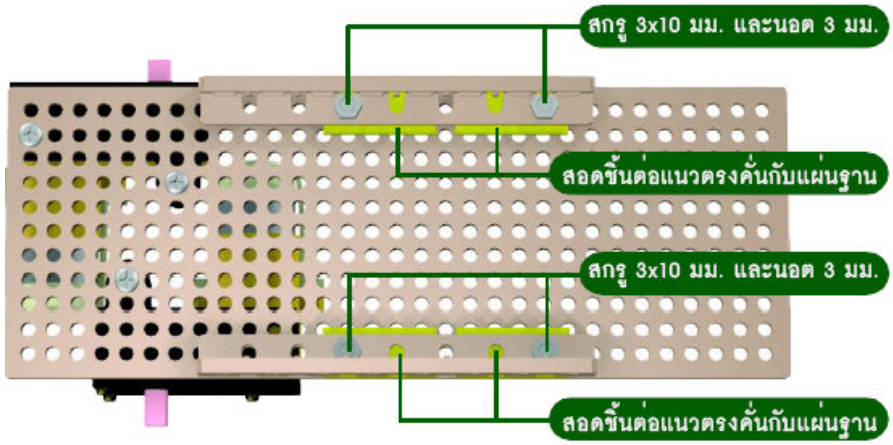
This automatic robot that uses track wheels similar to that of a tank car moves efficiently, accurately and effectively. It has space to install and operate a variety of different sensors which completely satisfies the needs of the user.

Activity #1: Building RoboTank

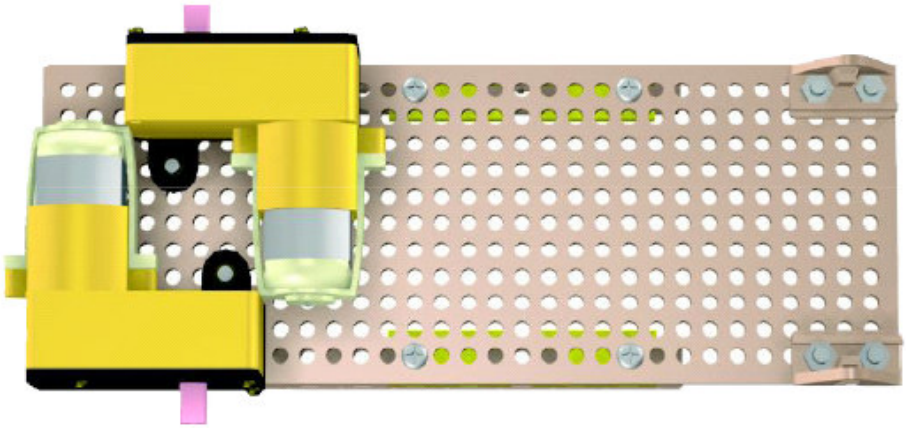
B1.1 Attach the motor gearbox sets to the universal plate using a 3 x 6 mm. and 3 x 10 mm. screw. Position the Motor Gearbox sets as shown in the picture below.



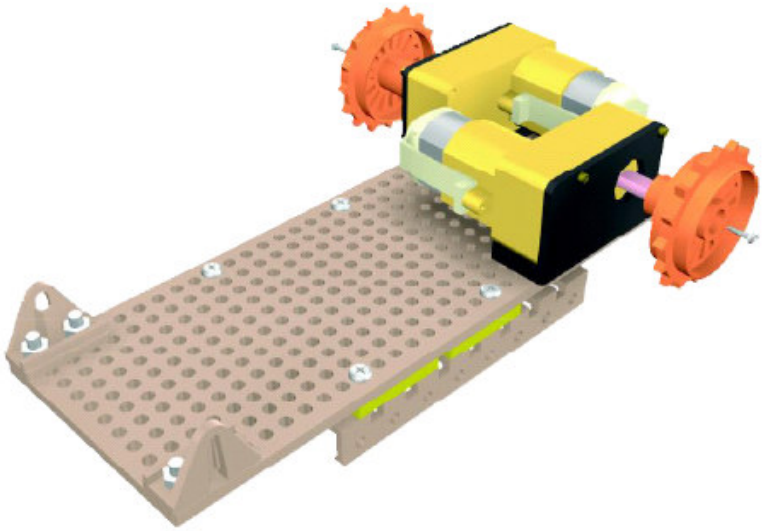
B1.2 Turn the plate over. Place the long angled shaft base underneath the plate. In between the plate and the long angled shaft, place the strength joiner. Screw them all in together using a 3 x 10 mm. screw and 3 mm. nut. Do this for both sides.



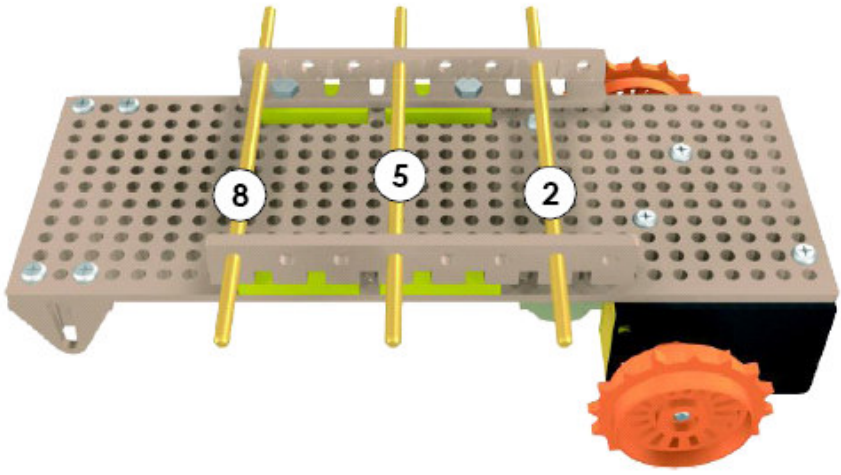
B1.3 Turn the plate back over. Attach the short angled shaft base at the end of the plate as shown in the picture.



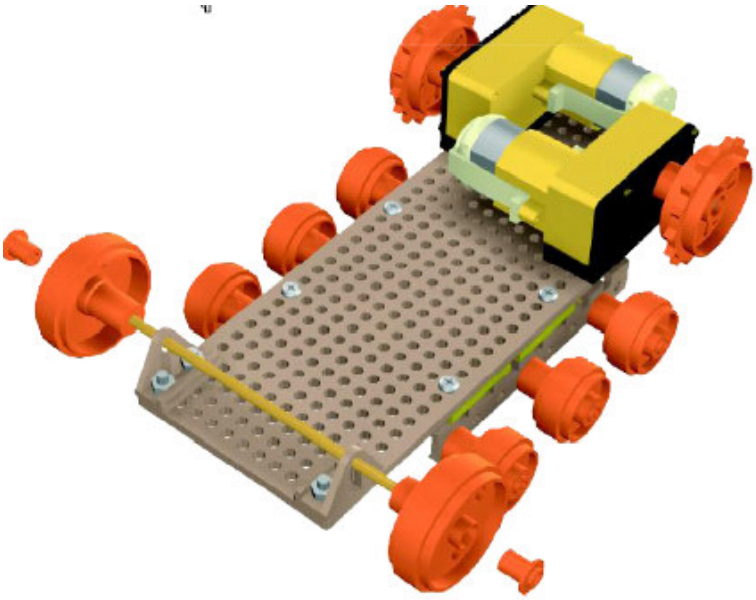
B1.4 Insert the Main Sprocket Wheel into the motor gearbox and screw it in tightly with a 2 mm. open-end screw.



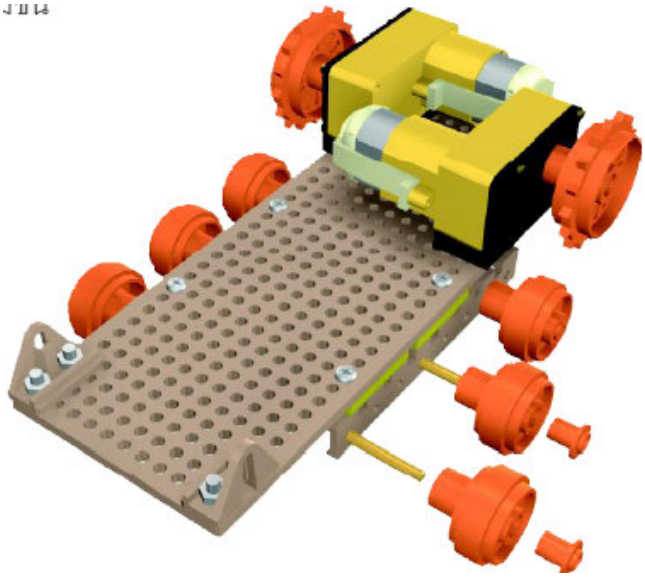
B1.5 Turn the plate bottom up and insert the metal axel into the holes of the long angled shaft in the hole positions of 2, 5, and 8 (Counting from the side with the installed motor).



B1.6 Place the Small Track support wheels over the metal axel as shown in the pictures. Insert the hubs over the wheels so that the wheels and the axels are connected tightly.

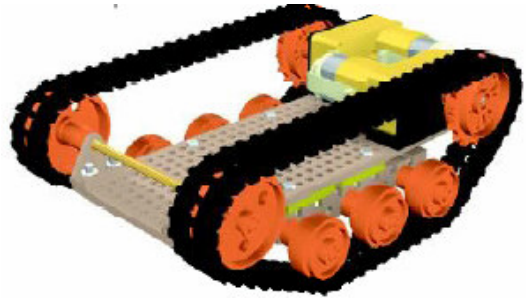


B1.7 Insert a metal axel into the two short angled shaft bases. Then insert the large track support wheels and hub to it as shown in the picture.



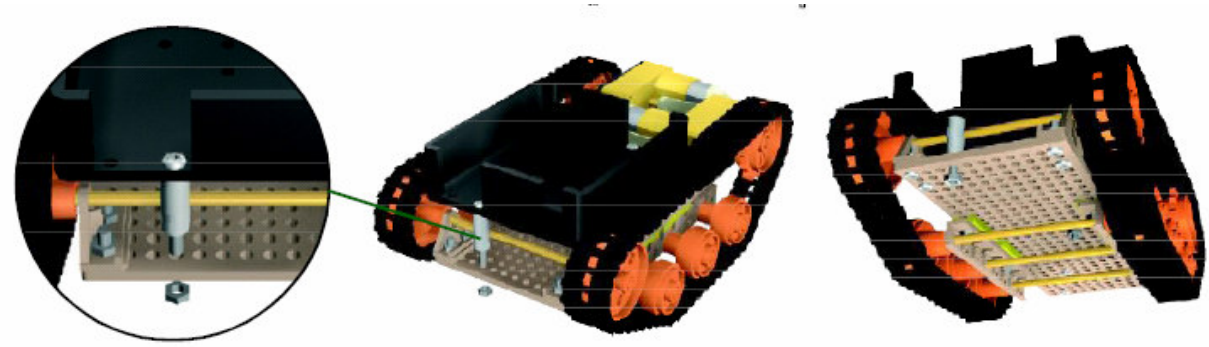
B1.8 Create two track belts by putting the different size tracks together. One track would consist of the following: One 30-joint track, one 10-joint track, and two 8-joint track. Connect the 10-joint track to the end of the 30-joint track. Next connect the two 8-joint tracks together. Take one end and connect it to the other end of the 10-joint track. Then take the other end of the 8-joint track and connect it to the remaining end of the 30-joint track to form one complete loop. Repeat the steps to make two track sets. If the track is too tight or loose, you can either adjust the length of the track or adjust the position of the short angled shaft base until the track has a good fit.

B1.9 Attach the tracks to the supporting wheels of the robot.



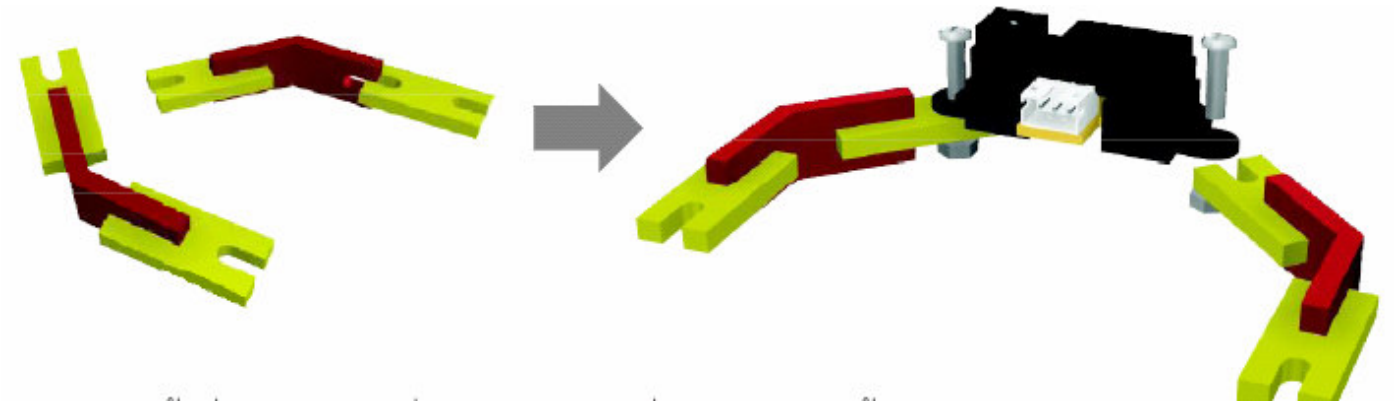
B1.10 Use a 3 x 20 mm. screw and insert it into the holes of the box holder. Place the 10 mm. plastic spacer over it. Do this for both sets, as shown in the picture.

B1.11 Place the completed box holder over the robot's body from B1.9 and use a 3 mm. nut to screw it in together tightly as shown in the picture.

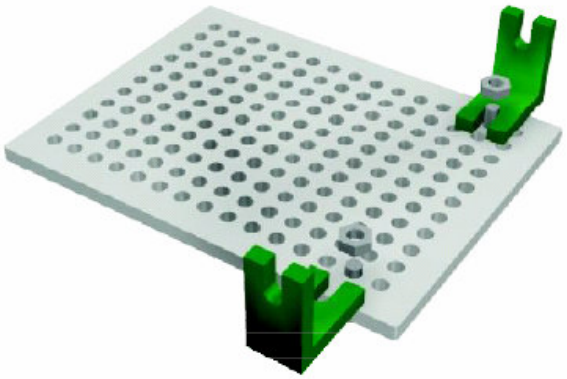


B1.12 Place the Stamp-BOX into the box holder and connect the left motor cable (using the Stamp-BOX as the reference point) to the connection point DIRECT of Motor A and connect the right motor cable to the connection point DIRECT of Motor B.

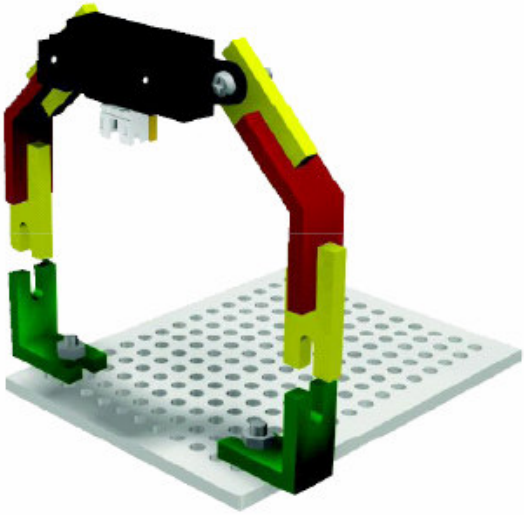
B1.13 The next step is connecting the IR Ranger Module GP2D120. We start by making the structure to connect the module. Take two strength joiners and attach it to an obtuse joiner at both ends. Make two sets. Then connect both of them to the GP2D120 module with a 3 x 10 mm. screw and 3 mm. nut.



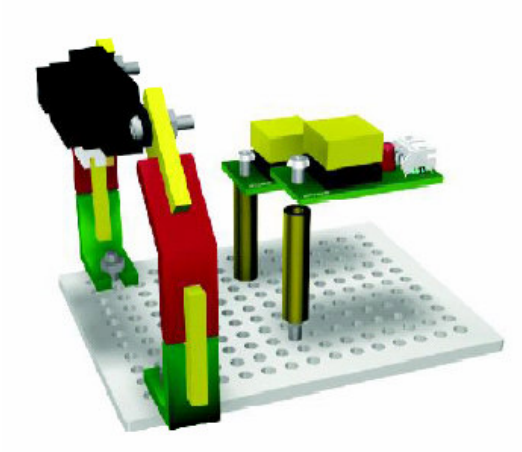
B1.14 Attach two angled joiners to the grid plate as shown in the picture with a 3 x 10 mm. screw and 3 mm. nut. This will be the base for the GP2D120 module.



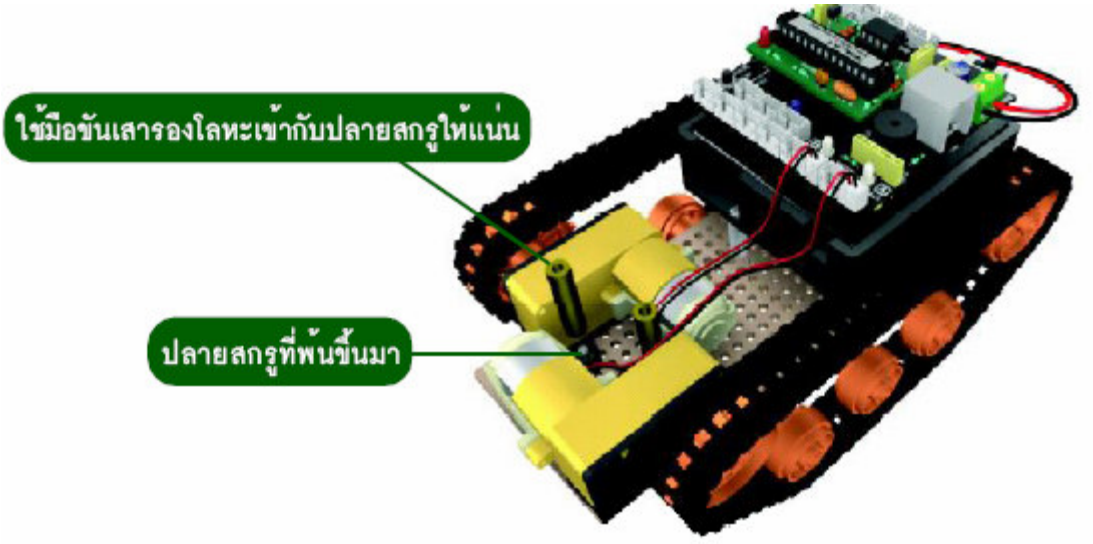
B1.15 Take the GP2D120 module structure in B1.13 and connect it to the grid plate from B1.14 as shown in the picture.



B1.16 Attach the 25 mm. alloy spacer to the Switch module with a 3 x 6 mm. screw and 3 mm. nut. Then attach the other end of the alloy spacer to the grid plate. Do the same for two sets as shown in the picture.



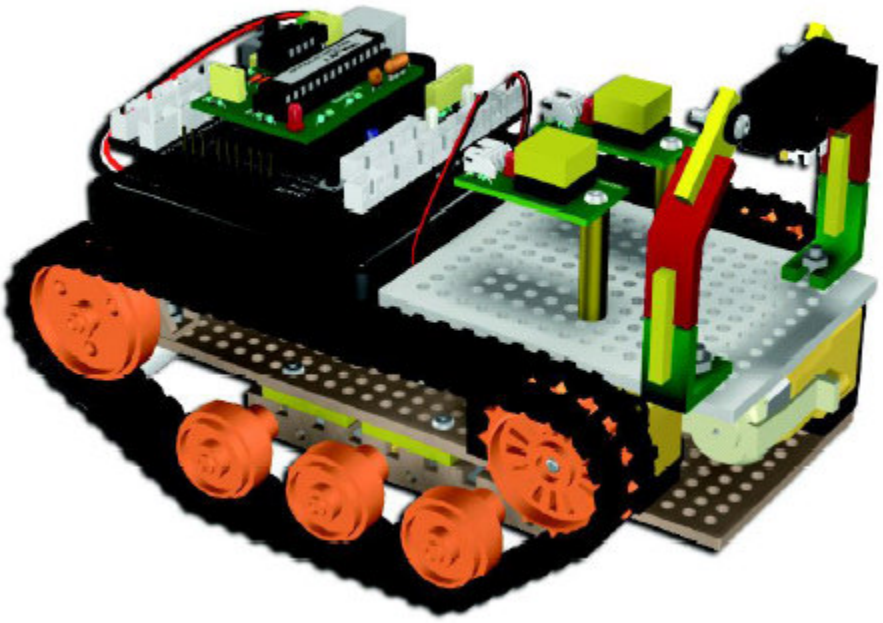
B1.17 Back to the robot body, use a 25 mm.. alloy spacer and use your hand to screw it tightly onto the two extruding screw on the motor gearbox set.



B1.18 Take the grid plate with the switch module and the GP2D120 module from step B1.16 and attach it to the robot where the two alloy spacers from B1.17 were placed. Use a 3 x 10 mm. screw to hold them together tightly at both points.

B1.19 Connect the signal cable of the switch module to P4 and P5. Then connect the cable from the IR Reflector Module GP2D120 to connection point Analog5

The RoboTank is now ready for the test program to be written.



Activity #2: Using the switch to turn the program on/off

Normally RoboTank would start to operate once power is supplied to it, which sometimes makes it work so fast that the user does not have time to prepare and position the robot first. Therefore in this activity, we will bring forward and use the switch module that is connected to the RoboTank robot. Switch 1 will be used to tell the robot to start operating whereas switch 2 will be used to tell the robot to stop all operations.

Test

B2.1 Open the Basic Stamp Editor Program and type in the following program. Download it into the RoboTank robot and turn off the POWER switch on the Stamp-BOX.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
Main: IF (IN4 = 1) THEN Main ' Wait key from P4
DO
FREQOUT 11,2500,2000 ' Sound On 0.5 Second : Frequency 5kHz
LOOP UNTIL (IN5 = 0) ' Wait key from P5 to stop
GOTO Main ' Again
```

How the Program Works

The program starts with the IF comm.and which checks the switch connected to P4 to see whether or not it has been pressed. If not, it will continue to loop at the same place (Label Main). When the switch is pressed, the condition is no longer true and the program moves to the next line which is a loop for the stereo to sound. At the same time, it will check the switch connected at P5. If the switch has been pressed, the program will come out of the loop to do the next comm.and which is GOTO Main. This means to go back to the beginning and start the program again.

B2.2 Remove the download cable (CX-4). Turn on the power switch and observe how the robot works.

When the switch connected to P4 is pressed, a noise will be heard. If you press the switch at P5, the noise will stop. However, it will not stop instantaneously.

This is because the program operates in a step by step order; therefore it must finish the FREQOUT comm.and first before it checks the switch.

Thus, if we change the time value of the FREQOUT comm.and to be shorter, the program would be able to respond to the switch input faster. A sample of the modified program is shown in Program B2-2

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
Main: IF (IN4 = 1) THEN Main ' Wait key from P4
DO
FREQOUT 11,250,2000 ' Sound On 0.1 Second : Frequency 5kHz
LOOP UNTIL (IN5 = 0) ' Wait key from P5 to stop
GOTO Main ' Again
```

Activity #3: Using a switch to test the function of the robot's track wheels

In this activity, we will write a program so that the RoboTank robot will check the switch before it starts to move.

Testing

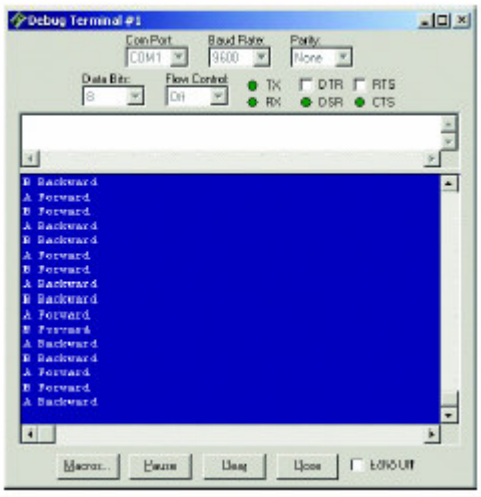
B3.1 Open the Basic Stamp Editor Program and type in the following program. Download it into the RoboTank robot.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
Main: IF (IN4 = 1) THEN Main ' Wait key From P4
GOSUB AB_OFF
GOSUB A_FWD ' Motor A go forward
DEBUG "A Forward"
Mode1: IF (IN4 = 1) THEN Mode1 ' Wait key from P4
GOSUB AB_OFF
GOSUB B_FWD ' Motor B go forward
DEBUG "B Forward"
Mode2: IF (IN4 = 1) THEN Mode2 ' Wait key from P4
GOSUB AB_OFF
GOSUB A_BWD ' Motor A go backward
DEBUG "A Backward"
Mode3: IF (IN4 = 1) THEN Mode3 ' Wait key from P4
GOSUB AB_OFF
GOSUB B_BWD 'Motor B go backward
DEBUG "B Backward"
Mode4: IF (IN4 = 1) THEN Mode4 ' Wait key from P4
GOTO Main
A_FWD: HIGH 12 : LOW 13 : RETURN
B_FWD: HIGH 14 : LOW 15 : RETURN
A_BWD: LOW 12 : HIGH 13: RETURN
B_BWD: LOW 14 : HIGH 15: RETURN
AB_OFF: LOW 12 : LOW 13 : LOW 14 : LOW 15 : RETURN
```

B3.2 With the download cable still connected, the Debug Terminal window will appear once the download is finished.

B3.3 Press and release the switch at P4 five times, each time observing how both the motors on the RoboTank robot operates. (During the test, the robot is not yet placed on the floor).

Press #1: Motor A spins and the Debug Terminal window will display the message "A Forward"
Press #2: Motor A stops and Motor B spins instead. The Debug Terminal window will display the message "B Forward"
Press #3: Motor B stops and Motor A will spin in the opposite direction. The Debug Terminal window will display the message "A Backward"
Press #4: Motor A stops and Motor B will spin in the opposite direction. The Debug Terminal window will display the message "B Backward"
Press #5: Both motors stop spinning.
From the test, it is found that when the switch is pressed the results are sometimes not as expected.



B3.4 The format of Program B3-1 is correct. However, when actually used, the program does not operate as it should. This is because the time used in pressing the switch is long, while the program runs too fast causing an unbalance in its operation. There are two ways to fix this problem as following:
Method 1: Insert a Time Pause by defining the Pause variable to be 250 ms after the switch is pressed. You will get the program as shown in Program B3-2

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
Main: IF (IN4 = 1) THEN Main ' Wait key From P4
PAUSE 250
GOSUB AB_OFF
GOSUB A_FWD ' Motor A go forward
DEBUG "A Forward"
Mode1: IF (IN4 = 1) THEN Mode1 ' Wait key from P4
PAUSE 250
GOSUB AB_OFF
GOSUB B_FWD ' Motor B go forward
DEBUG "B Forward"
Mode2: IF (IN4 = 1) THEN Mode2 ' Wait key from P4
PAUSE 250
GOSUB AB_OFF
GOSUB A_BWD ' Motor A go backward
DEBUG "A Backward"
Mode3: IF (IN4 = 1) THEN Mode3 ' Wait key from P4
PAUSE 250
GOSUB AB_OFF
GOSUB B_BWD 'Motor B go backward
DEBUG "B Backward"
Mode4: IF (IN4 = 1) THEN Mode4 ' Wait key from P4
PAUSE 250
GOSUB AB_OFF
GOTO Main
A_FWD: HIGH 12 : LOW 13 : RETURN
B_FWD: HIGH 14 : LOW 15 : RETURN
A_BWD: LOW 12 : HIGH 13 : RETURN
B_BWD: LOW 14 : HIGH 15 : RETURN
AB_OFF: LOW 12 : LOW 13 : LOW 14 : LOW 15 : RETURN
```

Method 2: Insert an additional comm.and to check when the switch is pressed. After the switch is pressed, let the program wait until the switch is released before it continues to the next comm.and. You will get the program as shown in Program B3-3.

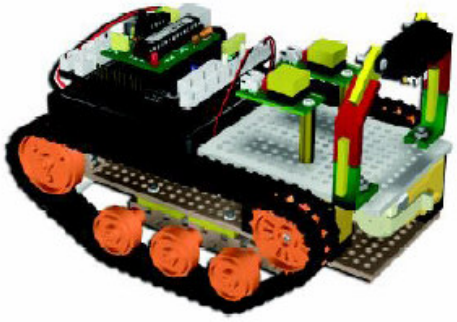
```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
Main: IF (IN4 = 1) THEN Main ' Wait key From P4
DO : LOOP UNTIL (IN4 =1) ' Loop until release key
GOSUB AB_OFF
GOSUB A_FWD ' Motor A go forward
DEBUG "A Forward"
Mode1: IF (IN4 = 1) THEN Mode1 ' Wait key from P4
DO : LOOP UNTIL (IN4 =1) ' Loop until release key
GOSUB AB_OFF
GOSUB B_FWD ' Motor B go forward
DEBUG "B Forward"
Mode2: IF (IN4 = 1) THEN Mode2 ' Wait key from P4
DO : LOOP UNTIL (IN4 =1) ' Loop until release key
GOSUB AB_OFF
GOSUB A_BWD ' Motor A go backward
DEBUG "A Backward"
Mode3: IF (IN4 = 1) THEN Mode3 ' Wait key from P4
DO : LOOP UNTIL (IN4 =1) ' Loop until release key
GOSUB AB_OFF
GOSUB B_BWD 'Motor B go backward
DEBUG "B Backward"
Mode4: IF (IN4 = 1) THEN Mode4 ' Wait key from P4
DO : LOOP UNTIL (IN4 =1) ' Loop until release key
GOSUB AB_OFF
GOTO Main
A_FWD: HIGH 12 : LOW 13 : RETURN
B_FWD: HIGH 14 : LOW 15 : RETURN
A_BWD: LOW 12 : HIGH 13: RETURN
B_BWD: LOW 14 : HIGH 15: RETURN
AB_OFF: LOW 12 : LOW 13 : LOW 14 : LOW 15 : RETURN
```

B3.5 Type in the program B3-2 and B3-3 and test how it works. Compare the results to that of Program B3-1.

B3.6 If you want to see the actual movement, modify the program by deleting all the comm.and lines with DEBUG (5 lines) and download the program again. Then turn off the switch, remove the download cable, and place the RoboTank robot on the floor to test its movement. Turn on the power switch and test the robot's operation by pressing the switch connected to P4 on the robot. Observe how it works.

RoboStamp Note

Program B3-2 and B3-3 can be used to test the motor and track wheels to see whether it has enough power to drive the RoboTank. Moreover, it can be used to test the tightness of the track wheels and whether it will fall off of the support wheels or not when the RoboTank robot moves.



RoboTank Detecting Objects Using Infrared

Robo-Stamp 2.0 Smart Robot kit

Installing the IR Ranger Module at the front of the RoboTank robot allows it to detect and calculate the range of objects that may block its path.

Activity #4: RoboTank Searches for Objects

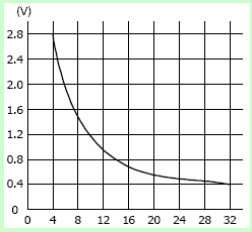
From iZEBOT, the IR Ranger Sensor Module GP2D120 was used to detect and avoid obstacles. However, in RoboTank, we will use the GP2D120 module in a different way by using it to move in search of objects instead. Thus, the program will be a bit more complex. Details of basic functionality of the GP2D120 module can be read from the activities of the iZEBOT robot.

In this activity, we will write a simple program to use the GP2D120 module and to observe how the robot approaches the object. If the object is within a specified distance, the robot will stop moving.

B4.1 Open the Basic Stamp Editor Program. Type Program B4-1 and then download it to the RoboTank robot. Turn off the POWER switch on the Stamp-BOX

```
' {$STAMP BS2sx}
' {$PBASIC 2.5}
' {$PORT COM1}
CH VAR Nib
ADC VAR Word
Init: DIRD = %1111
Main: GOSUB WAIT_Key ' Wait Switch From P1
GOSUB Forward ' Go Forward
DO
CH = 6 : GOSUB RD_ADC ' Read data From GP2D120
LOOP UNTIL ADC > 500 ' Wait until GP2D120 near object
GOSUB Motor_OFF ' Stop moving when GP2D120 near object
GOSUB Beep ' Generate sound
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 'Send Acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
```

```
'+++++ Generate Sound ++++++++++++++++++++++++++++++++
Beep: FREQOUT 11,300,800 ' Sound subroutine
RETURN
'+++++ Wait until keypress to start ++++++++++++++++++
Wait_KEY: DO : LOOP UNTIL (IN1 = 0) ' Check key from P1
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: OUTD = %1010 : RETURN ' Motor A --> and Motor B --> Go Forward
Backward: OUTD = %0101 : RETURN ' Motor A <-- and Motor B <-- Go Backward
S_Left: OUTD = %0110 : RETURN ' Robot Spin Left
S_Right: OUTD = %1001 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = %0000 : RETURN ' Stop All Motor
'++++++++++++++++++++++++++++++++++++++++++++++++++++++
```



How the Program Works:

In this program, the main program does not have many comm.ands. However, a complete set of comm.ands for various tasks can be found in different sub-programs so that it can be easily used in the development of other programs. The main program operates as following:

1) Determines the direction of terminals P12, P13, P14 and P15 which acts as the output terminal for Motor A and Motor B. The sub-program that drives the robot will use the comm.and OUTD to send a value out to the motor instead of using the comm.and HIGH or LOW. The advantage of using OUTD is that it uses less memory space, works faster, and can comm.and motor A and motor B to operate at the same time which makes it move straighter.
2) Calls the sub-program that waits for the switch to be pressed. The switch module has to be connected to P1. When the switch is pressed, the program will go to the next comm.and.
3) Comm.ands the robot to move forward before it reads the value from the GP2D120 module at ANALOG6. Therefore the GP2D120 Module has to be connected to connection point ANALOG6 too.
4) Check the value that was read from the GP2D120 module to see if it is more than 500 or not. For the A/D converter, the value 500 will have a voltage of approximately 2.44 V. From the formula:

Measured voltage = (value read/Full range value) x Full range voltage

When compared to the graph of the GP2D120 module, the measured range would be around 4 centimeters which is almost the lowest value that the GP2D120 module can measure.

5) If the value read is more than 500, the program will order the robot to stop moving as well as send out noise through its stereo by using the sub-program Beep.

B4.2 Remove the download cable from the RoboTank robot. Place it on the floor. Then place some objects at a distance that is not too far from the robot and turn on the power switch so that the RoboTank robot can start operating.

Activity #5 RoboTank Turns in Search of Objects

This is a modification from the previous activity to allow the RoboTank robot to turn around in search of objects. When the object is within a specified range, the robot will move forward until it is 4 centimeters from the object. It then stops and make a noise. This concept can be used in a robot sumo competition.

B5.1 Type in Program B5-1 and download it to the RoboTank robot. Turn off the POWER switch on the Stamp-BOX

B5.2 Remove the download cable from the RoboTank robot and place the robot on the floor. Then place objects on the floor at a distance that is farther than in activity #4 (but not more than 1 foot). Turn on the switch to let the robot operate.

```
' {$STAMP BS2sx}
' {$PBASIC 2.5}
' {$PORT COM1}
CH VAR Nib
ADC VAR Word
Init: DIRD = %1111
Main: GOSUB WAIT_Key ' Wait Switch From P1
DO
CH = 6 : GOSUB RD_ADC ' Read data From GP2D120
IF ADC > 61 THEN
GOSUB Forward : PAUSE 100 ' Check distace < 32 CM ?
' YES ! attack tt
ELSE
GOSUB S_Left : PAUSE 50 ' No ! rotate and find it
ENDIF
LOOP UNTIL ADC > 500 ' Wait until GP2D120 near object
GOSUB Motor_OFF ' Stop moving when GP2D120 near object
GOSUB Beep ' Generate sound
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Select Chip
SERIN 10,240,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
'+++++ Generate Sound ++++++++++++++++++++++++++++++++
Beep: FREQOUT 11,300,800 ' Sound subroutine
RETURN
```

```
'+++++ Wait until keypress to start ++++++++++++++++++
Wait_KEY: DO : LOOP UNTIL (IN1 = 0) ' Check key from P1
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: OUTD = %1010 : RETURN ' Motor A --> and Motor B --> Go Forward
Backward: OUTD = %0101 : RETURN ' Motor A <-- and Motor B <-- Go Backward
S_Left: OUTD = %0110 : RETURN ' Robot Spin Left
S_Right: OUTD = %1001 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = %0000 : RETURN ' Stop All Motor
```

How the Program Works

After the switch is pressed, the program will read the values from the GP2D120 module and check if the value received is more than 61 or if the distance is less than 32 centimeters or not. If less than, the robot will move forward towards the object. However, it will move for only 0.1 seconds and then it will recheck the distance of the object again. This is to verify that it is heading in the right direction.
If the distance is more than 32 centimeters the robot will turn left 0.05 seconds and will loop back to check for the object. If it does not find anything, it will continue to turn until it does.

Activity #6: RoboTank Searchs for Long-Range Objects

The problem that occurs in Activity #5 is that when the RoboTank robot is farther than a measurable distance from the object, the robot will continue to turn around in search of the object and will not find it. Therefore, the program must be modified so that it can find objects that are far from the robot.

B6.1 Type in Program B6-1 and download it into the robot. Turn off the switch.

B6.2 Remove the download cable from the RoboTank robot and place the robot on the floor. Then place objects on the floor at a distance from the robot that is farther than that in Activity #5. Turn on the switch to let the RoboTank robot operate.

```
' {$STAMP BS2sx}
' {$PBASIC 2.5}
' {$PORT COM1}
CH VAR Nib
CNT VAR Byte
ADC VAR Word
ADC1 VAR Word
Init: DIRD = %1111
Main: GOSUB WAIT_Key ' Wait Switch From P1
CNT = 0
ADC1 = 0
DO
CH = 6 : GOSUB RD_ADC ' Read data from GP2D120
IF ADC > 61 THEN ' Check distace < 32 CM ?
GOSUB Forward : PAUSE 100 ' YES ! attack it
ADC1 = ADC ' Save lastest object
GOSUB S_Left : PAUSE 100 ' Start seek from left
CH = 6 : GOSUB RD_ADC ' Read data from GP2D120 again
```

```
IF ADC < ADC1 THEN ' Compare old data with new
GOSUB S_Right : PAUSE 200 ' Seek from right too!
ENDIF
ELSE
GOSUB S_Left : PAUSE 50 ' Object not found. Rotate and find it
CNT = CNT +1
ENDIF
IF CNT > 20 THEN ' Check counter run over 20 ?
GOSUB Forward : PAUSE 400 ' YES ! go forward random direction
CNT = 0 ' and clear counter
ADC1 = 0 ' clear compare data too
ENDIF
LOOP UNTIL ADC > 500 ' Wait until GP2D120 near object
GOSUB Motor_OFF ' Stop moving when GP2D120 near object
GOSUB Beep ' Generate sound
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 'Send Acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
'+++++ Generate Sound ++++++++++++++++++++++++++++++++
Beep: FREQOUT 11,300,800 ' Sound subroutine
RETURN
'+++++ Wait until keypress to start ++++++++++++++++++
Wait_KEY: DO : LOOP UNTIL (IN1 = 0) ' Check key from P1
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: OUTD = %1010 : RETURN ' Motor A --> and Motor B --> Go Forward
Backward: OUTD = %0101 : RETURN ' Motor A <-- and Motor B <-- Go Backward
S_Left: OUTD = %0110 : RETURN ' Robot Spin Left
S_Right: OUTD = %1001 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = %0000 : RETURN ' Stop All Motor
```

How the Program Works:

In this program, two check functions are included. The first is to check the number of times it turns and the second it to check the distance to the object and compare it with the previous distance detected. The robot will then move towards the nearest object. The program operates in the following steps:

1) Define the variables CNT to check the number of turns and ADC1 to compare the data.
2) Wait for the start switch to be pressed. Start by reading the values from the GP2D120 module.
3) If the value is more than 61 it means that an object was found. Move forward for 0.1 seconds
4) Save the current value that was read in the variable ADC1 for comparison.
5) Turn the robot slightly left to test reading the values. Take the value and compare it to saved value in ADC1. If the value is less than (farther distance) it means that it should not head in that direction. It will then turn right twice the original distance it turned left.
6) If the object is still not found, it would turn slightly left again. And the count value will increase by 1.
7) Check the count value. If it is more than 20 it means that the robot has been searching for an object for some time now and did not find anything. Thus, the robot is ordered to move forward for 0.4 seconds and to clear the count and comparison value.
8) It will continue checking until it finds an object. It will then move towards the object until it reaches a pre-defined range which is less than 4 centimeters. It will then stop and send out a noise.

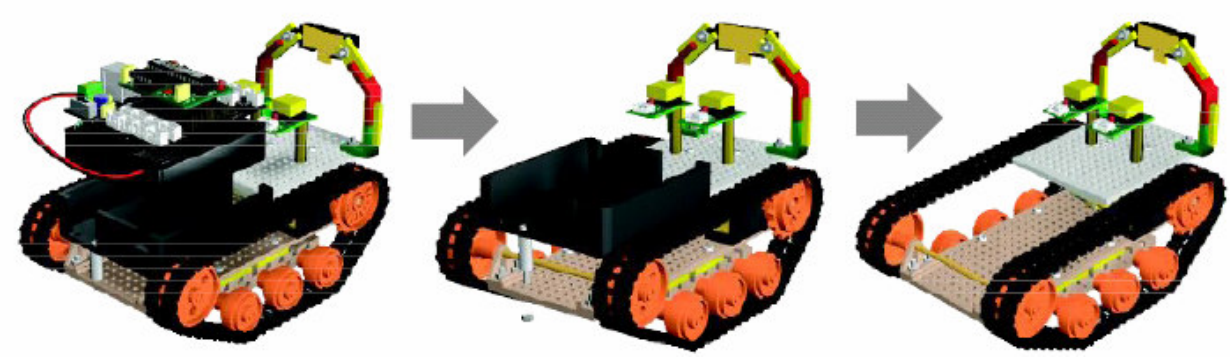
RoboTank Moving along the Line

Robo-Stamp 2.0 Smart Robot kit

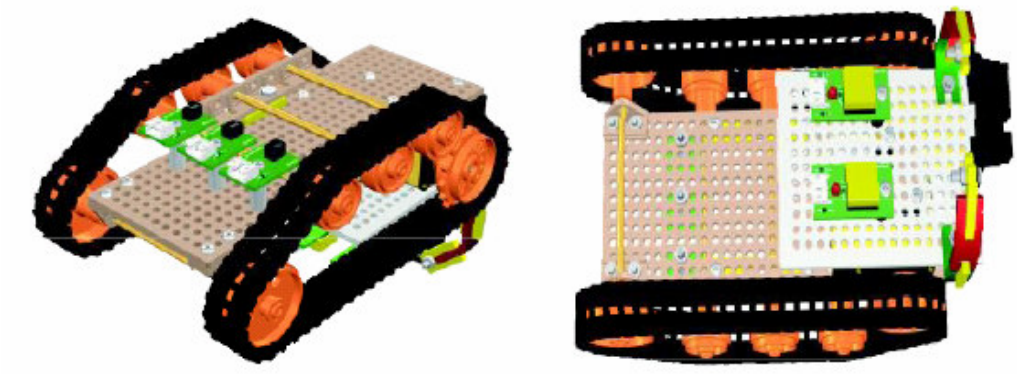
This is a classic act of learning and controlling the functionality of the small automatic robot. We have a complete set of samples which includes detecting the lines at 2 or 3 points so that 3 or 4 way intersections can be detected

Activity #7: Installing the IR Reflector Sensor to Detect lines

B7.1 Take the Stamp-BOX out of the box holder and remove the screws that hold the box holder together. Take the box holder out to prepare to install the IR Reflector Sensor ZX-03Q on the robot

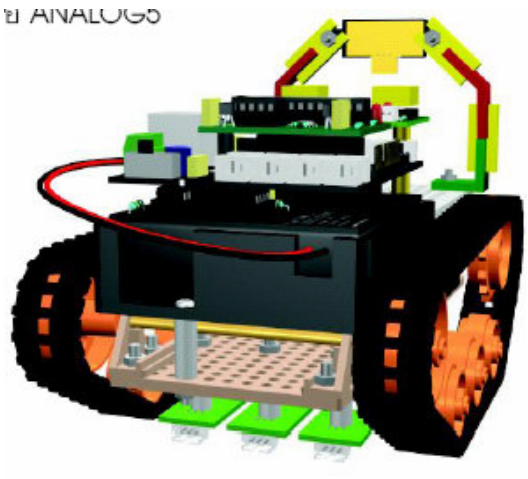


B7.2 Screw in a 3 x 25 mm. screw to the IR Reflector Sensor ZX-03Q and place a 15 mm. plastic spacer over it at the back. Do this for the 3 sets. Then turn the robot over and attach the ZX-03Q according to the positions shown in the picture (positions can change depending on the size of the lines to detect) and use a 3 mm. nut to screw them in together tightly.



B7.3 Take the box holder and attach it back to the same position. Then put the Stamp-BOX back in place. Connect the right signal cable of the ZX-03Q sensor to ANALOG1 on the Stamp-BOX, the middle cable to ANALOG3, and the left cable to ANALOG5.

Notes:
In installing the ZX-03Q sensor, the distance between the floor and sensor should be noted. The distance should not be more than 5 millimeters because if there is a lot of gap, the light might not reflect too well causing low distinction between white and black. This in turn would make it more susceptible to errors. Moreover, external light may also interfere with the operation of the ZX-03Q.



Activity #8: Testing the IR Reflector Sensor by Detect Black and White

After installing the IR Reflector Sensor, check to see how well it reflects light off the objects.

B8.1 Open the Basic Stamp Editor Program. Type in program B8-1 and download it to the RoboTank robot that already has the ZX-03Q sensor installed from activity #7. Do this to check the functionality of each ZX-03Q sensor and to display the value received on the Debug Terminal window. Do not remove the download cable after you have finished installing.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
DO
CH = 1 ' Select channel 1 for Analog1
GOSUB RD_ADC ' Read data from Left sensor
DEBUG 2,5,5, "LEFT SENSOR VALUE = ",DEC ADC
CH = 3 ' Select channel 3 for Analog3
GOSUB RD_ADC ' Read data from Mid sensor
DEBUG 2,5,6, "MID SENSOR VALUE = ",DEC ADC
CH = 5 ' Select Channel 5 for Analog5
GOSUB RD_ADC ' Read data from Right sensor
DEBUG 2,5,7, "RIGHT SENSOR VALUE = ",DEC ADC
PAUSE 200
LOOP
```

```
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
```

How the Program Works:

On the Stamp-BOX there are altogether 8 analog terminals. Comm.unicating with those terminals can be done through the sub-program RD_ADC. Define the terminal that you want to read and store it in the variable CH. The sub-program RD_ADC will then take the 10-bit data (with values betweens 0 to 1,023) that it reads from that terminal and store it in the variable ADC. The main program will then display this value on the Debug Terminal window.

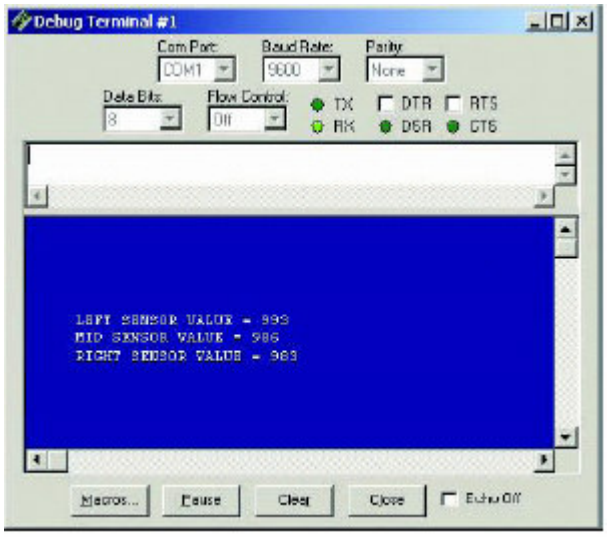
B8.2 Place the robot down on a white surface area. On the Debug Terminal window, observe the values that it reads from the IR Reflector sensor.

White areas will give a reflection value of approximately 700 to 1023

B8.3 Place the robot down on a black surface area. Observe the values it reads on the Debug Terminal window.

Black areas will give a reflection value of approximately 20 to 400.

If the value received does not fall within these specified ranges, check the installation of the IR Reflection Sensor ZX-03Q as well as the cable connection to see if everything was connected properly.



Activity #9: Testing the Lift Function of the RoboTank robot.

In the previous activity we tested the reflection of infrared light for areas that are white and black. It was seen that black areas would give lower values since it reflects less light. However, looking at it another way, we can also say that the RoboTank robot was lifted from the floor, resulting in lower reflection values. Therefore, this concept is used to write a program that will prevent the robot from being stolen.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
DO
CH = 1 ' Select channel 1 for Analog1
GOSUB RD_ADC ' Read data from Left sensor
DEBUG 2,5,5, "LEFT SENSOR VALUE = ",DEC ADC
IF (ADC < 400) THEN ' Robot is lifted from the white board ?
FREQOUT 11,500,800 ' Beep
ENDIF
LOOP
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
```

How the Program Works:

In this program, the robot will read values from the left IR Reflection Sensor ZX-03Q only, which is connected to ANALOG1. Ordinarily, the RoboTank robot will be placed on a white or light colored floor. The value read would then be more than 400 and the IF....THEN condition would be false. But if the robot is lifted from the floor, the value read from the ZX-03Q will be less than 400 and the IF...THEN condition will be true. The program then goes to the REQOUT comm.and and creates a noise at the frequency of 2 KHz. The noise will continue until the robot is placed back on the white area, which will cause the IF...THEN condition to be false again.

B9.1 Open the Basic Stamp Editor Program and type in Program B9-1. Download the program to the RoboTank robot which already has the ZX-03Q sensor installed from Activity #7. After the download is complete, turn off the switch and remove the download cable.

B9.2 Place the robot on a white surface area and turn on the power switch. Then wait a moment before lifting the RoboTank robot off from the floor. Observe how the robot works.

The RoboTank robot emits a noise continuously until it is placed back on the white floor.

Activity #10: Building a line field to test the robot.

To make this next activity test more efficiently, a line field should be built using black duct tape and a sheet of corrugated plastic.

Tools and Equipment Used

1) 1 Sheet of White corrugated plastic with the size of 60 x 100 cm
2) 1 Roll of Black duct tape with a width of 2.5 cm
3) 1 Cutter to cut the black duct tape.

Building It

The diagram of the robot field can be seen in Figure B4-1. Use a 60 x 100 cm white corrugated plastic as the area for the robot to move in, and use the black duct tape to make a path for the robot. The distance between the black line and the edge of the plastic should not be more than 7.5 cm so that the robot will not fall off the edge when it travels near that area.

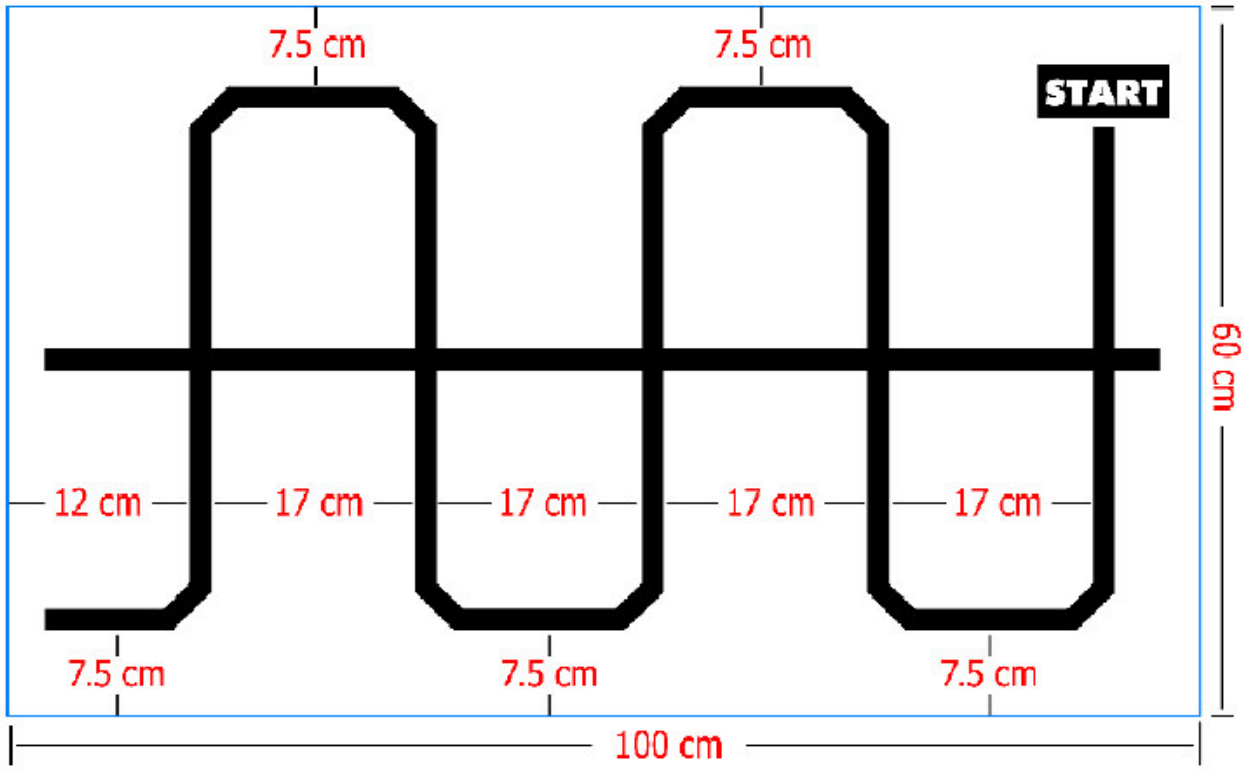


Figure B4-1 Displays the line field used to test the RoboTank robot. Black duct tape is placed on white corrugated plastic to make a line path for the robot to move on.

Activity #11: RoboTank Detects the Line

The objective of this activity is for the RoboTank robot to read the values from the IR Reflector Sensor ZX-03Q and check whether the area is black or not. If it is black the robot will stop moving.

B11.1 Open the Basic Stamp Editor Program. Type in Program B11-1 and download it to the RoboTank robot. Once the download is complete, turn off the switch and remove the download cable.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
FREQOUT 11,500,800 'Title sound
Main: DO : LOOP UNTIL (IN1=0 ) ' Press key to start program
GOSUB Forward 'Motor On
DO
CH = 1 ' Select channel 1 for Analog1
GOSUB RD_ADC ' Read data from Left sensor
IF (ADC < 400) THEN ' Is robot found the black line ?
EXIT ' Exit from DO..LOOP Command
ENDIF
LOOP
GOSUB Motor_OFF ' Stop motor after exit loop
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 'Send Acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: HIGH13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
Motor_OFF: LOW13 : LOW 12 : LOW 15 : LOW 14 : RETURN
'++++++++++++++++++++++++++++++++++++++++++++++++++++
```

How the Program Works:

The program starts by emitting noise out of the stereo. Then the program labeled Main will await the switch at P1 of the Stamp-BOX to be pressed. Once the switch is pressed, the RoboTank robot will move and read the value from the IR Reflector Sensor ZX-03Q that is connected to ANALOG1. If the value is more than 400, it will loop back to continue reading the values from the sensor. However, if the value is less than 400 it means that the black line has been found and the IF…THEN condition will be true. It will exit the loop to call upon the sub-program that stops the robot. Then it will go back and wait for the switch to be pressed again.

B11.2 Place the robot on the white floor area of the test field that was built in Activity #10. Turn on the switch and observe the robot.

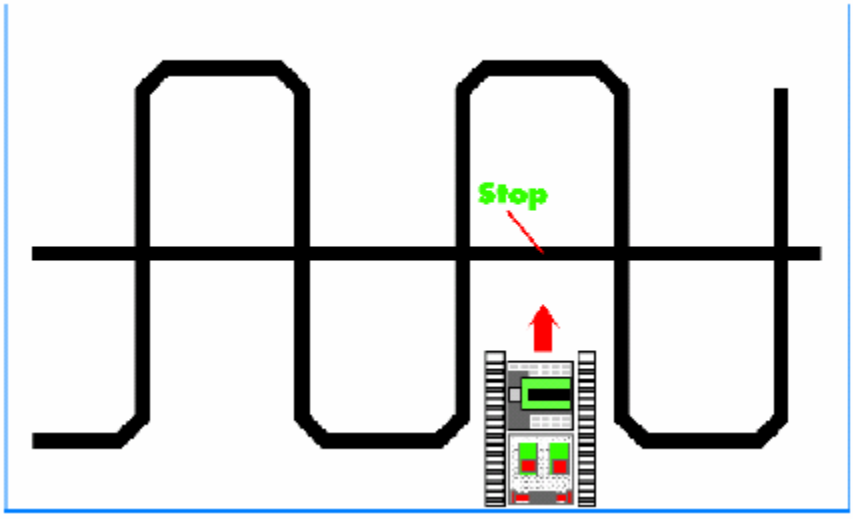


Figure B4-2 Displays the RoboTank robot on the field to test it stopping at the line

Activity #12: RoboTank Counts the Line Crossings

In this activity, the RoboTank robot will check and count the number of line crossings. It will store the value and move on until it reaches the specified number of line crossings (In this case, it is the 4th). Then the robot will stop moving.

B12.1 Type in Program B12-1 and download it to the RoboTank robot. Turn off the switch and remove the download cable. Place the robot on the white area of the test field and turn on the switch. Observe how it works.

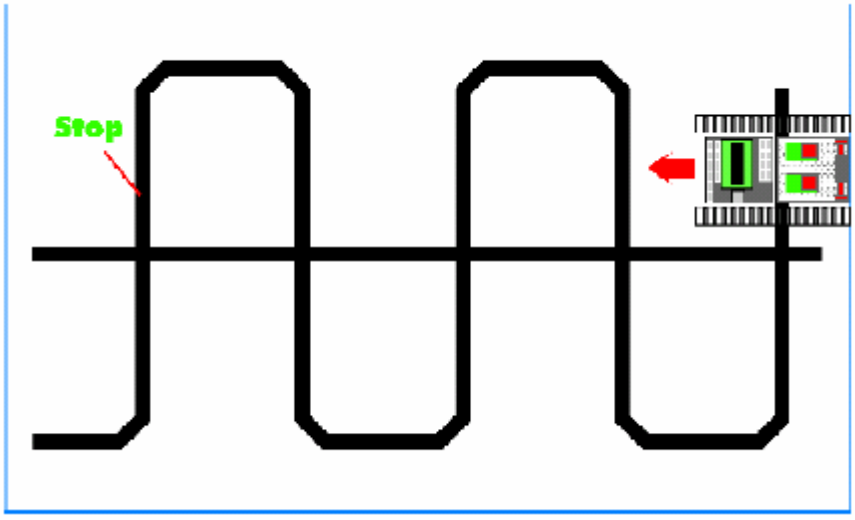


Figure B4-3 Displays the RoboTank robot on the field to test it stopping at the 4th crossing.

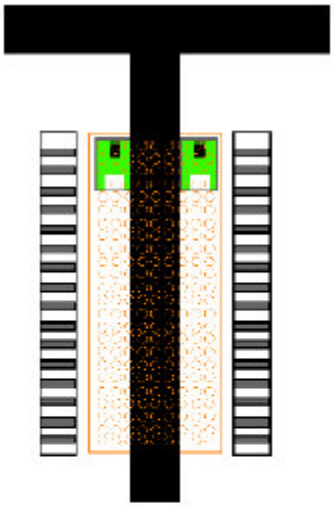
```
'{$STAMP BS2sx} -1
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
CNT VAR Nib
i VAR Nib
GOSUB Beep
Main: DO : LOOP UNTIL (IN1=0 ) ' Press key for start program
GOSUB Forward 'Motor on
CNT = 0
DO
CH = 1 : GOSUB RD_ADC ' Read data from Left sensor
IF (ADC < 400) THEN ' Check the black line
GOSUB Beep
CNT = CNT + 1 ' Count 1 time at 1 line
DO
CH = 1 : GOSUB RD_ADC ' Wait until Sensor out of line
LOOP UNTIL (ADC > 600)
ENDIF
LOOP UNTIL ( CNT = 4 ) ' Pass 4 lines ?
GOSUB Motor_OFF ' Stop motor after exit loop
FOR i = 1 TO 4 ' Beep 4 times to finish
GOSUB Beep
PAUSE 200
NEXT
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 'Send Acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
Beep: FREQOUT 11,200,800 ' Sound routine
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: HIGH13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
Motor_OFF: LOW13 : LOW 12 : LOW 15 : LOW 14 : RETURN
'++++++++++++++++++++++++++++++++++++++++++++++++++++++

DO
CH = 1 : GOSUB RD_ADC ' Read data from Left sensor
IF (ADC < 400) THEN ' Check the black line
GOSUB Beep
CNT = CNT + 1 ' Count 1 time at 1 line
DO
CH = 1 : GOSUB RD_ADC ' Wait until Sensor out of line
LOOP UNTIL (ADC > 600)
ENDIF
LOOP UNTIL ( CNT = 4 ) ' Pass 4 lines ?
```

Activity #13: RoboTank moves along the line using a 2-Point sensor

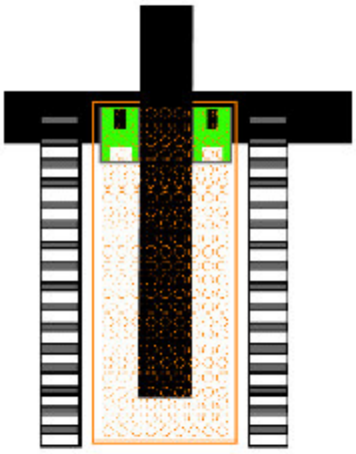
Concept

In order for the RoboTank robot to move along a line, it uses the IR Reflector Sensor ZX-03Q to determine the path by comparing the values it receives, thus using this as the Line Sensor.

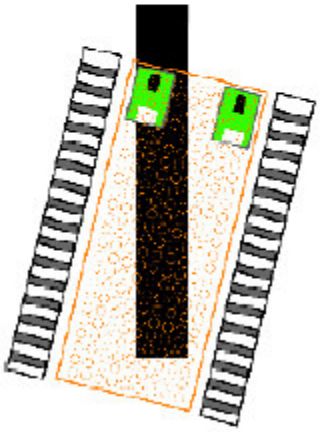
If we use 2 sensors, or at 2 points, 4 different scenarios can occur as following



1) Both Line sensors read values that are white: The robot will move forward. Thus, the program is written so that the robot moves forward normally.

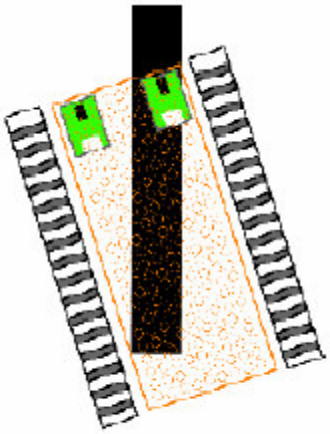


2) Both line sensors read values that are black: This means that the robot has reached an intersection (the line crossing has to be in front of both sensors. It would not be able to detect a 3-way intersection). You have a few options when writing the program. The robot can be ordered to move forward, turn 90 degrees left, or 90 degrees right.



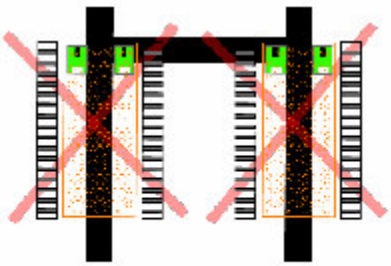
3) The left sensor reads black while the right sensor reads white: This occurs when the robot is slightly turned to the right. Thus, the program is written for the robot to move back left to resume its normal path.

4) The left sensor read white while the right sensor reads black: This occurs when the robot is slightly turned to the left. Thus, the program is written for the robot to move back to the right to resume its normal path.



Case there is a 3-way intersection on the right or left-hand side.

The robot cannot check for this condition using a 2-point sensor, because it will consider itself slanted either towards the left or the right.



B13.1 Type in Program B13-1 and download it to the RoboTank robot. Turn off the switch and then remove the download cable.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
ADC0 VAR Word
ADC1 VAR Word
Main: DO : LOOP UNTIL (IN1 = 0) ' Wait until key pressed
DO
CH = 1 : GOSUB RD_ADC : ADC0 = ADC
' Select chennel Analog1 , Read it and store to ADC0
CH = 5 : GOSUB RD_ADC : ADC1 = ADC
' Select chennel Analog1 , Read it and store to ADC1
IF ((ADC0 > 500) AND (ADC1 > 500)) THEN
' Step 1 : Check Sensor0 and Sensor1 all white
GOSUB FORWARD ' Yes ! Go direct
ENDIF
IF ((ADC0 < 500) AND (ADC1 < 500)) THEN
' Step 2 : Check Sensor0 and Sensor1 all black
GOSUB Forward ' Yes ! Found the cross way
PAUSE 200 ' Forward 0.2 second, Direct over crossway
ENDIF
IF (ADC0 < 500) THEN ' Step 3 : Check Sensor0 found black line
GOSUB S_Left ' Spin left <-- Return to line
ENDIF
IF (ADC1 < 500) THEN ' Step 4 : Check Sensor1 found black line
GOSUB S_Right ' Spin right --> Return to line
ENDIF
LOOP
```

```
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Send Select Chip
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++++
Forward: HIGH 13 : LOW 12 : HIGH 15 : LOW 14 :RETURN
' Motor A --> and Motor B --> Go Forward
Backward: HIGH 12 : LOW 13 : HIGH 14 : LOW 15 : RETURN
' Motor A <-- and Motor B <-- Go Backward
T_Left: HIGH 13 : LOW 12 : LOW 15 : LOW 14 : RETURN ' Robot Turn Left
T_Right: LOW 13 : LOW 12 : HIGH 15 : LOW 14 : RETURN ' Robot Turn Right
S_Left: HIGH 13 : LOW 12 : HIGH 14 : LOW 15 : RETURN ' Robot Spin Left
S_Right: HIGH 12 : LOW 13 : HIGH 15 : LOW 14 : RETURN ' Robot Spin Right
Motor_OFF: LOW 13 : LOW 12 : LOW 15 : LOW 14 : RETURN ' Stop All Motor
'++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

How the program Works:

This program will provide similar results as described in the "Concept" section. You can also study the functionality of this program by placing the RoboTank robot on the test field. The program will work as following:

1) The RoboTank robot waits for the switch at P1 of the Stamp-BOX to be pressed. It will then read the value from the left line sensor by calling the sub-program RD_ADC and store the value it reads in the variable ADC0.

2) The value from the right line sensor is read and stored in the variable ADC1.

3) Compare the values received from both sensors. If both ADC0 and ADC1 give values that are considered to be white, the sub-program Forward will be called to order the RoboTank robot to move forward.

4) If the value of ADC0 and ADC1 is considered to be black, the sub-program Forward will be called and it will wait 0.2 seconds so that the robot passes that intersection first before checking for other conditions.

5) If only the value of ADC0 reads as black, the sub-program S_Left will be called so that the robot will turn towards the left and back onto its original path.

6) If only the value of ADC1 reads as black, the sub-program S_Right will be called so that the robot will turn towards the right and back onto its original path.

B13-2 Place the RoboTank robot at the starting position of the line. Then press the switch so that the robot will start operating.

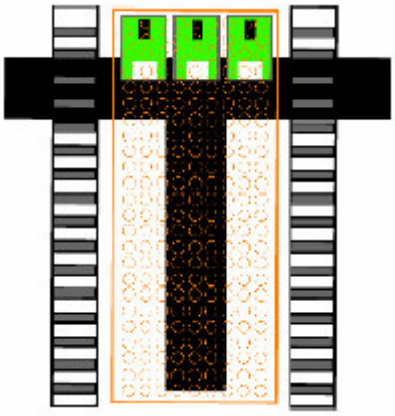
B13-3 Lift the RoboTank robot slightly off the floor and test moving the robot around. Observe the changes that occur with the motors by looking at the LED light that tells us how the motor is functioning..

B13-4 If there is a change in the RoboTank robot when it detects a black line, put the robot down and let it move along that line. Observe any errors that might occur.

B13-5 If there is no change when the RoboTank robot detects the line, it could mean that the reference value (500) is either too high or too low. Thus, use the previous activity to check the reference value from the IR Reflector Sensor again.

Activity #14: RoboTank moves along the line using a 3-Point Line Sensor.

The problem that occurred in Activity #13 is that it could not detect a 3-way intersection. Therefore, another line sensor needs to be added in between the first and second line sensor (Practically, all the line sensors have already been installed)

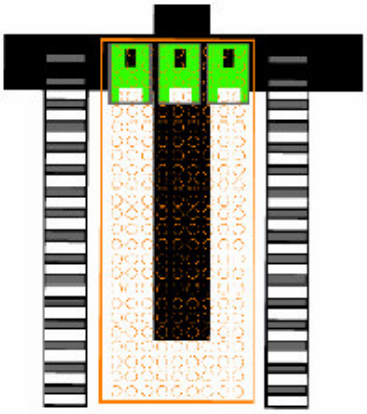


Concept

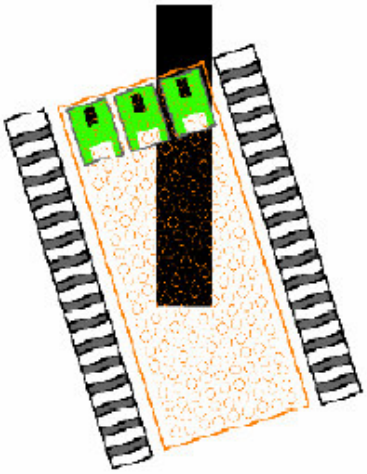
With another line sensor added, the conditions used in detecting the lines will also increase. Details of the conditions that can occur are as following:

1) All 3 line sensors read values that are white: This means that the robot is not on its path. In the program, the robot must be ordered to go back to its original path either by rotating 180 degrees or by moving backwards towards its original path.

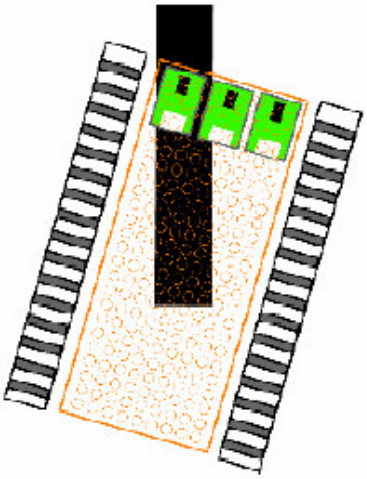
2) All 3 line sensors read values that are black: This means that the robot has reached an intersection in front of it. The intersection could be either a 3-way or 4-way intersection. In the program, the robot could turn either right or left 90 degrees or move straight forward, depending on the direction you want the robot to move.



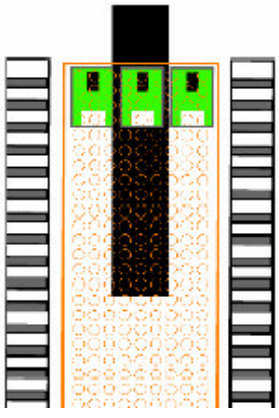
3) Both line sensors on the left read white values while the rightmost one reads a black value: This means that the robot is slanted towards the left. In the program, the robot must turn towards the right so that it is back at its original position which is at the middle of the line.



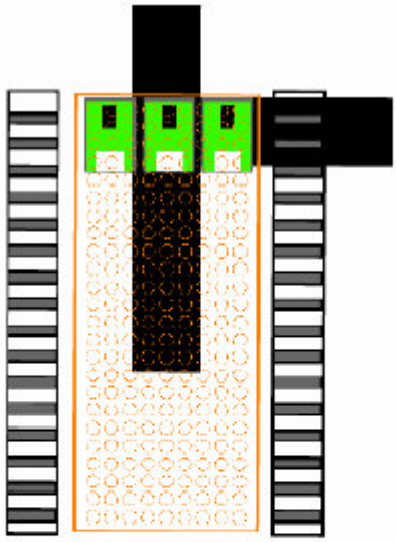
4) Both line sensors on the right read white values while the leftmost one reads a black value: This means that the robot is slanted towards the right. In the program, the robot must be ordered to turn back to the left towards its original position.



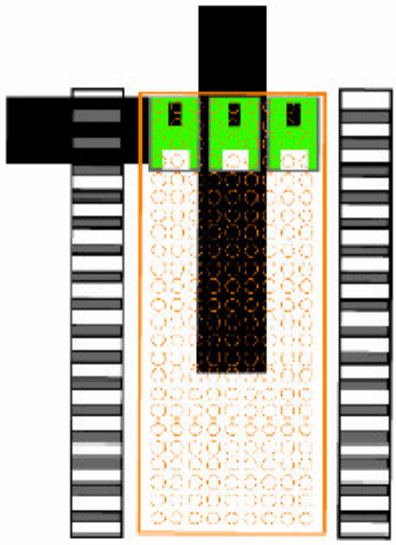
5) Both the sensors on the right and left read white values while the middle one reads a black value: This means that the robot is working normally. In the program, the robot should be ordered to move forward.



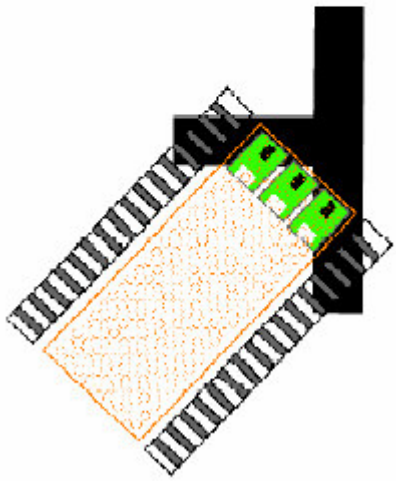
6) The line sensors at the middle and right read black values while the leftmost one reads a white value: This means that the robot has detected an intersection on its right side. In the program, the robot can be ordered to either move forward or to turn 90 degrees right. However, in some cases, this might occur if the robot's position is slanted on the line resulting in the right and middle line sensors being both black. The program must also take this case into consideration.



7) The left and middle line sensors read black values while the right sensor reads a white value: This means that the robot has detected an intersection on its left side. In the program, the robot can be ordered to either move forward or to turn 90 degrees left. However, in some cases, this might occur if the robot's position is slanted on the line resulting in the left and middle sensors being both black. The program must also take this case into consideration.



8) The left and right line sensors reads black values while the middle sensor reads a white value: In a normal operation of a robot, this case would hardly occur unless the robot moved out of its original path and towards another one as shown in the picture. The program can either consider or not consider this scenario.



B14-1 Type in Program B14-1 and download it into the RoboTank robot. Turn off the switch and remove the download cable.

```
' {$STAMP BS2sx}
' {$PBASIC 2.5}
' {$PORT COM1}
CH VAR Nib ' Select channel of analog to digital converter
ADC VAR Word ' save data from analog to digital converter
ADC0 VAR Word ' For save data left sensor
ADC1 VAR Word ' For save data middle sensor
ADC2 VAR Word ' For save data right sensor
Init: DIRD = %1111 ' P12 P13 P14 and P15 ==> output
Begin: GOSUB Wait_KEY ' Check key before start program
Main: GOSUB RD_ALL ' Read data from sensor
IF ((ADC0 > 500) AND (ADC2 > 500)) THEN
GOSUB Forward
' If Left and Right sensor found WHITE ==> forward
GOTO Main
ENDIF
IF ((ADC0 < 500) AND (ADC1 < 500) AND (ADC2 < 500)) THEN
GOSUB Cross 'If all BLACK, forward over cross
GOTO Main
ENDIF
IF ((ADC0 < 500) AND (ADC1 < 500)) THEN
GOTO Left ' If left sensor and mid sensor found BLACK
' goto Check Left Subroutine
ENDIF
IF ((ADC2 < 500) AND (ADC1 < 500)) THEN
GOSUB Right ' If mid sensor and right sensor found BLACK
' goto Check Right Subroutine
ENDIF
IF (ADC0 < 500) THEN
GOSUB S_Left 'If only Left sensor found BLACK
' Spin Left liitle time
ENDIF
IF (ADC2 < 500) THEN
GOSUB S_Right ' if only Right sensor found BLACK
' Spin Right little time
ENDIF
GOTO Main ' Check again
Left: GOSUB S_Left : PAUSE 60 ' Spin Left a little time
GOSUB Motor_Off : PAUSE 50 ' Stop motor before read data
GOSUB RD_ALL ' read data second time
IF ((ADC0 < 500) AND (ADC1 < 500) AND (ADC2 < 500)) THEN
GOSUB S_Right : PAUSE 50 ' Check all sensor are BLACK
' If yes, spin right a little time
GOSUB Motor_Off : PAUSE 50 ' Stop motor before read data
GOSUB RD_ALL ' Read data third time
IF (ADC2 > 500) THEN
GOSUB Left90 ' Check only Right sensor found WHITE ?
' Yes ! spin left 90 degree
GOTO Main
ENDIF
' if all sensor found BLACK after read data third time
IF ((ADC0 < 500) AND (ADC1 < 500) AND (ADC2 < 500)) THEN
GOSUB Cross ' Go forward over cross
GOTO Main
ENDIF
ENDIF
GOTO Main
Right: GOSUB S_Right :PAUSE 60 ' Spin right a little time
GOSUB Motor_Off : PAUSE 50 ' Stop motor before read data
GOSUB RD_ALL ' Read data second time
```

```
IF ((ADC0 < 500) AND (ADC1 < 500) AND (ADC2 < 500)) THEN
GOSUB S_Left : PAUSE 50 ' Check all sensor are BLACK
' YES ! spin left a little time
GOSUB Motor_Off : PAUSE 50 ' Stop motor before read data
GOSUB RD_ALL ' Read data third time
IF (ADC0 > 500) THEN ' Check only left sensor found WHITE ?
GOSUB Right90 ' Yes ! spin right 90 Degree
GOTO Main
ENDIF
' if all sensor found BLACK after read data third time
IF ((ADC0 < 500) AND (ADC1 < 500) AND (ADC2 < 500)) THEN
GOSUB Cross ' Go forward over cross
GOTO Main
ENDIF
ENDIF
GOTO Main
'+++++ Go forward over cross ++++++++++
Cross: GOSUB Forward
GOSUB Beep
PAUSE 200
RETURN
'+++++ Turn robot Right 90 Degree ++++++++++
Right90:GOSUB Forward
PAUSE 200
GOSUB S_Right : PAUSE 650
GOSUB Forward : PAUSE 200
RETURN
'+++++ Turn robot left 90 Degree ++++++++++
Left90: GOSUB Forward : PAUSE 200
GOSUB S_Left : PAUSE 650
GOSUB Forward : PAUSE 200
RETURN
'+++++ Save sensor 3 channel procedure ++++++++++
RD_ALL: CH = 1 : GOSUB RD_ADC : ADC0 = ADC
' Select ANALOG1 , Read and store to ADC0
CH = 3 : GOSUB RD_ADC : ADC1 = ADC
' Select ANALOG3 , Read and store to ADC1
CH = 5 : GOSUB RD_ADC : ADC2 = ADC
' Select ANALOG5 , Read and store to ADC2
RETURN
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Select channel
SERIN 10,240,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
'++++++++++++++++++++++++++++++++++++++++++++++++++
Beep: FREQOUT 11,300,800 ' Sound subroutine
RETURN
'+++++ Wait until keypress to start ++++++++++++++++++
Wait_KEY: DO : LOOP UNTIL (IN1 = 0) ' Check key from P1
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: OUTD = %1010 : RETURN ' Motor A --> and Motor B --> Go Forward
Backward: OUTD = %0101 : RETURN ' Motor A <-- and Motor B <-- Go Backward
S_Left: OUTD = %0110 : RETURN ' Robot Spin Left
S_Right: OUTD = %1001 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = %0000 : RETURN ' Stop All Motor
```

How the Program Works:

This program tells the robot to move along the line and to detect both 3-way and 4-way intersections. For 4-way intersections, the robot will send a "beep" once it has passed that intersection. For 3-way intersections, the robot will be ordered to make a left or right turn according to the direction of that intersection. Details of how the program works are as following:

1) When the program starts, ports P12, P13, P14, and P15 that are used to drive the motor are defined as output ports. Use the comm.and DIRD to define the directions.

2) Call the sub-program that awaits the switch at P1 to be pressed. When the switch is pressed it goes to the next comm.and line. If not, it continues to loop and wait.

3) Read the values from all the 3 line sensors and store it in the variables to check.

4) Check the most important condition first, that is, whether the robot is on a black line or not. If so, let it move forward.

5) Check whether it is a 4-way intersection or not. If so, the 3 sensors will all read a value of black. Call the sub-program that tells it to move forward and cross the intersection.

6) Check for the condition in which the left and middle sensor reads a black value, which may be caused by two reasons. First, there might be a left-turn 3-way intersection, or the robot may be slanted rather left on the line. Therefore, another sub-program Left is used to check again. Sub-program left works as following:

    6.1) Tells the robot to move slightly left and stop to read the value again

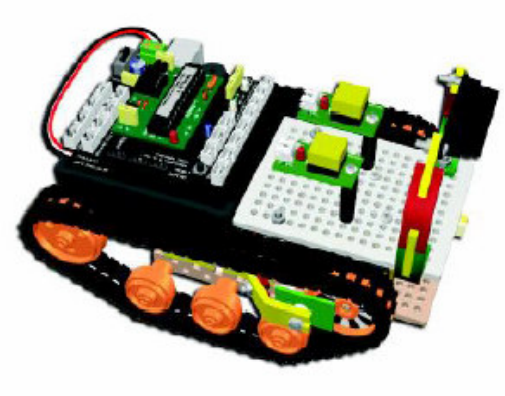
    6.2) If the value of all the sensors are black, let the robot turn right and check a third time. If the line sensor reads a value of white on the third time, it means that the robot has found a 3-way left-turn intersection. The sub-program Left90 should be called to tell the robot to make a 90 degree turn.

    6.3) If in the second reading, the sensors read a value of black it means that the robot has found an intersection and it should move forward.

7) Check for the condition in which the right and middle sensors read black values. The program would operate in the same way as in (6) but in the opposite direction. Also, it will call upon the Right subroutine instead.

8) If the readings from the sensor do not fall under any of the conditions, the program will loop back to start reading the sensors and checking the conditions once more.

B14.2 Place the RoboTank robot at the start line and press the switch to start the robot.



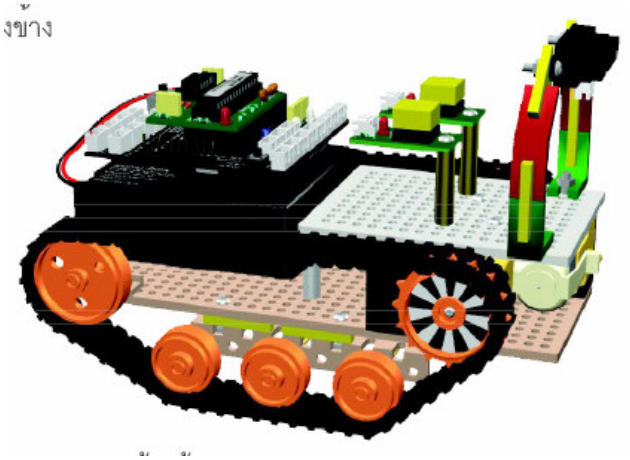
RoboTank Moving Accurately using the Encoder Wheel

Robo-Stamp 2.0 Smart Robot kit

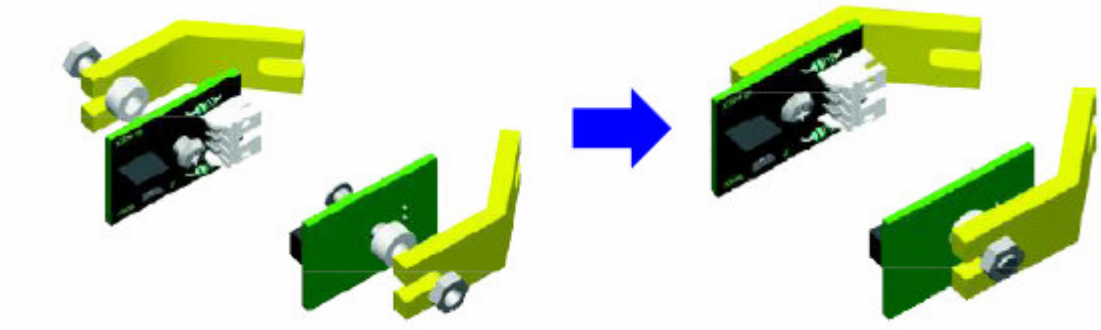
To help the RoboTank robot move along the specified distance accurately, an encoder wheel as well as a ZX-21 Encoder Wheel Sensor is installed to the driving wheels of the robot.

Activity #15: Installing the IR Reflector Sensor to detect the Encoder Wheel

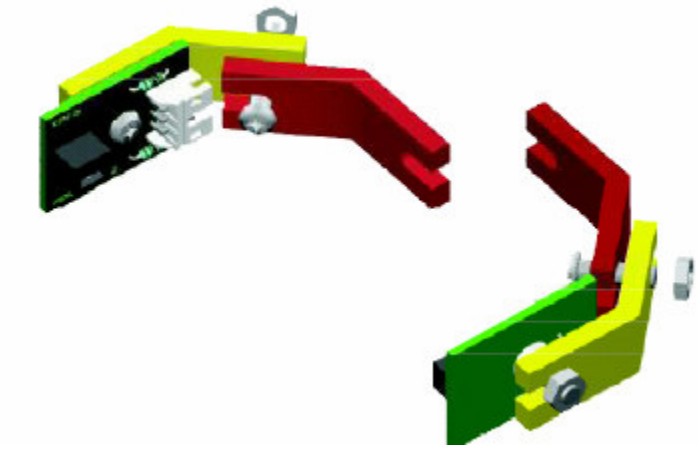
B15.1 Take the black and white encoder wheel sticker and stick it to the outside of the main drive wheel of the RoboTank motor.



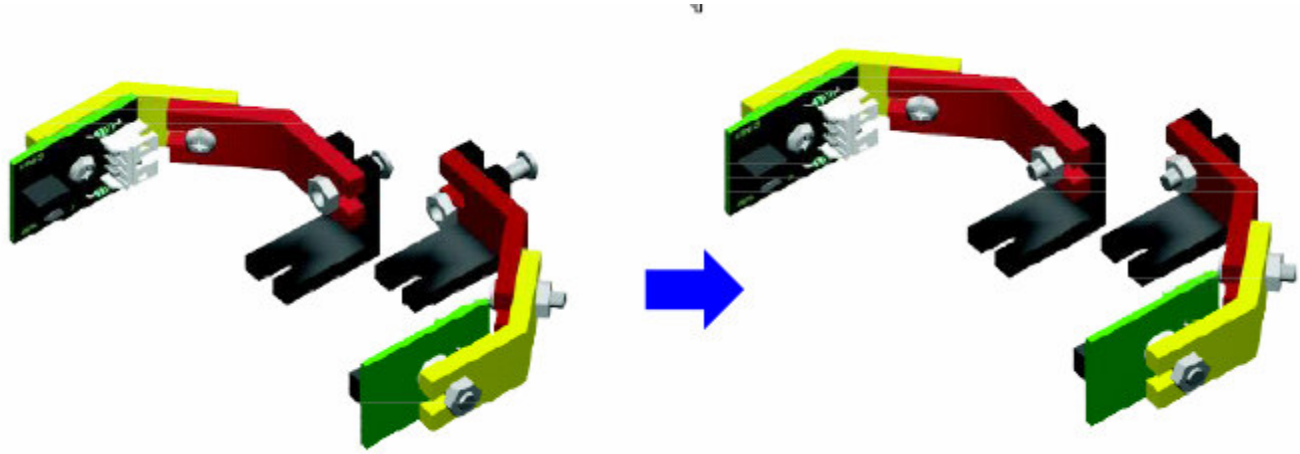
B15.2 Take the ZX-03Q IR Reflector Sensor and place a 3 x 10 mm. screw through it, followed by a 3 mm. plastic spacer. Then attach it to an obtuse joiner and screw it together tightly with a 3 mm. nut. Since the ZX-03Q was used in Activity #14 as a line sensor, you would either have to find one more or remove the ZX-03Q from the body of the robot so that it can be used with the encoder wheel.



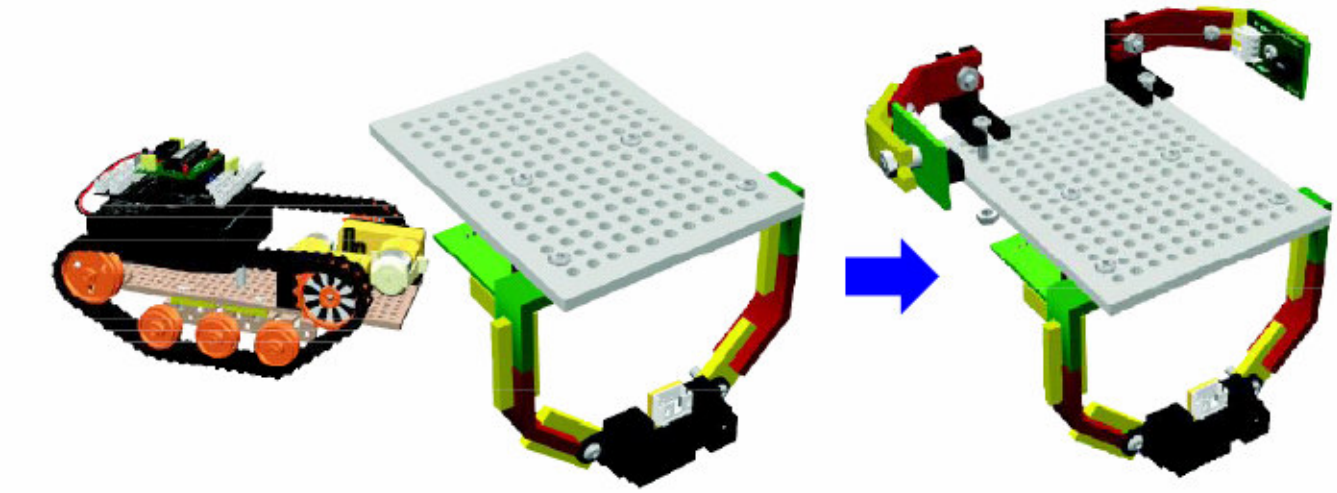
B15.3 Connect the end of the obtuse joiner to another obtuse joiner with a 3 x 10 mm. screw and 3 mm. nut.



B15.4 Use a 3 x 10 mm. screw and 3 mm.. nut to attach an angled joiner to the obtuse joiner from B15.3. Do this for both sets, but for opposite directions since it must be connected to the left and right side of the wheel as shown in the picture.



B15.5. Remove the white plate on the RoboTank robot and turn it over. Attach the ZX-03Q IR Reflector sensor with the connected joiners from step B15.2 to B15.4 onto the third hole of the white plate, as shown in the figure. Use a 3 x 10 mm. screw and 3 mm. nut to screw them in together tightly.



B15.6 Attach the white plate back onto the RoboTank robot, making sure that the ZX-03Q module is approximately 3mm. away from the encoder wheel. If not, loosen the screw to reposition the plate and screw it back in tightly again.

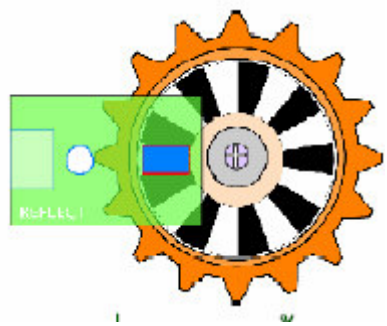
Connect the signal cable of the left encoder wheel sensor to ANALOG0 and the right encoder wheel sensor to ANALOG2 of the Stamp-BOX. You will then get a RoboTank robot that can specify its range of movement.

Activity #16: Reading the Encoder wheel using the IR Reflector Sensor.

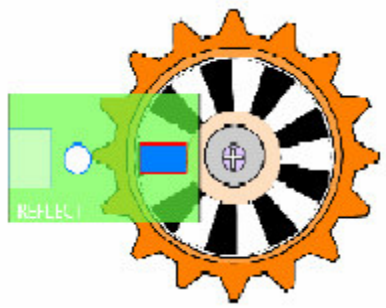
B16.1 Type in Program B16-1 and download it to the RoboTank robot. After the download is complete, the Basic Stamp Editor Program will open the Debug Terminal window to display the value it read from the ZX-03Q IR Reflector Sensor used with the encoder wheel.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
DO
CH = 0 : GOSUB RD_ADC ' Read ADC Position Analog0
DEBUG 2,2,2, DEC4 ADC, CR ' Show Data On debug Terminal
PAUSE 300 ' Delay For Display
LOOP
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Send Select Chip
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
```

B16.2 Use your hand, turn the track of the left wheel so that the ZX-03Q IR Reflector Sensor would be reading the white area of the encoder wheel as shown in the figure. Observe the value that is displayed on the Debug Terminal window. The value should be high, ranging between 500 and1000, depending on the distance between the ZX-03Q and the encoder wheel.



B16.3 Use your hand, turn the track of the left wheel so that the ZX-03Q IR Reflector Sensor would be reading the black area of the encoder wheel as shown in the figure. Observe the value that is displayed on the Debug Terminal window. The value should be low, ranging approximately between 50 and 250.



B16.4 Use your hand to continue turning the track wheels. Observe the changes for each area color to see if any irregular readings occur.

B16.5 Modify Program 16-1 by changing the signal terminal from ANALOG0 to ANALOG2 by editing the program :

CH = 0 to CH = 2

Download this program to the robot again to check the operation of the right track wheel.

Conclusion

The values were read in this activity in order to find a reference value that will be used for specifying the conditions used in counting the number of rounds in the next activity. From the test, it was found that a value of not more than 250 will be used in detecting black areas while white areas will give a reading of more than 500. However, the results may not be the same for each robot. Therefore, the user must complete his or her own test in order to get the most accurate reference value.

Activity #17: Counting the Position of the wheel's Turn

B17.1 Type in Program B17-1 and download it to the RoboTank robot

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
CNT VAR Byte
Main:DO : LOOP UNTIL (IN1 = 0) 'Wait Until Key Press
CNT = 0 'Clear Counter
DO
CH = 0 : GOSUB RD_ADC 'Read ADC Analog0
DEBUG 2,2,1,"ADC Value ",DEC4 ADC 'Show  On Debug Terminal
IF ADC > 500 THEN ' Check White ?
DO 'Loop Here
CH = 0 : GOSUB RD_ADC 'Read Again
DEBUG 2,2,1,"ADC Value ",DEC4 ADC 'Show  On Debug Terminal
LOOP UNTIL ADC < 250 'Loop Until Found Black
CNT =CNT+1 'When White and Black Increment Counter
ENDIF
DEBUG 2,2,2, "Counter Value ",DEC3 CNT
'Show  Counter On Debug Terminal
LOOP
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 'Send Acknowledge
SEROUT 10,240,[CH] 'Send Select Chip
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR
RETURN
```

How the Program Works:

Define the variable CNT to count the values. Loop and wait until the switch connected to P1 is pressed in order to start counting. Clear the variable used in the count to 0 so that the count starts from 0.

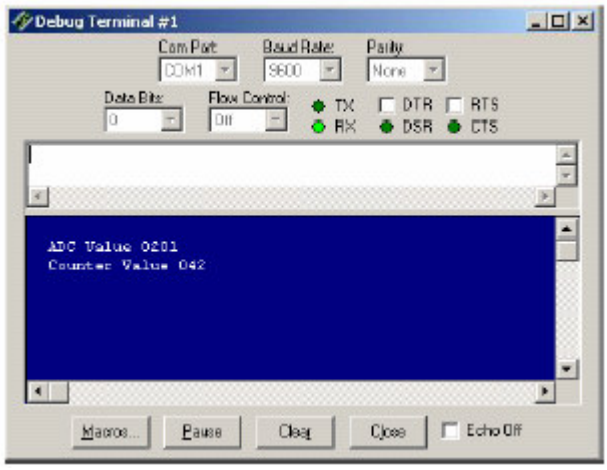
Then read the analog signals to check whether the sensor is on a white or black area.

If the sensor detects a white area, it will continue to loop and read values until the black area is found.

When the value is a black area, it means that 1 complete step is read; therefore the count value is increased by 1. The results of the count will be displayed on the Debug Terminal window before it continues to check and count.

B17.2 Run the program. Then press the switch at P1 to start reading the values from the ZX-03Q module.

B17.3 Use your hand to turn the left track continuously. Observe the count value that increases on the Debug Terminal window to see if any error has occurred in the count. If there is an error, adjust the reference values until it reads the value correctly.



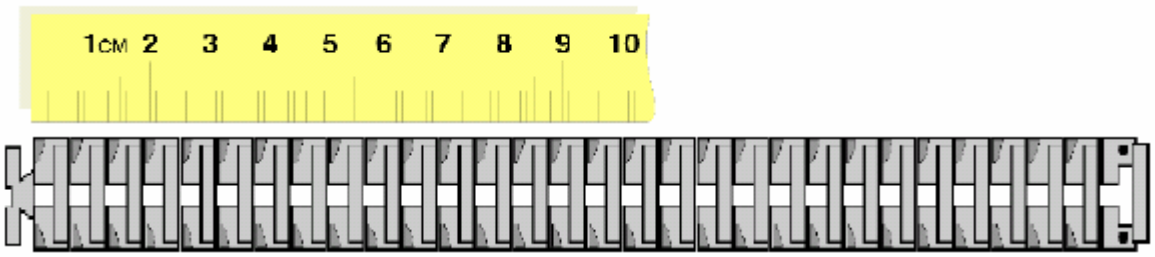
B17.4 Modify Program B17-1 by changing the signal terminal that is used to ANALOG2. Thus, in the command

CH=0 to CH =2

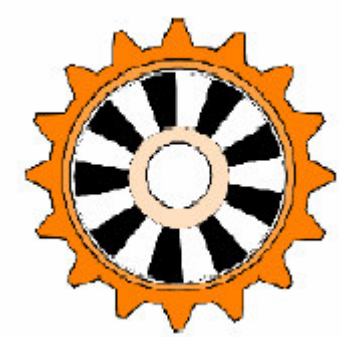
In order to check the track wheel on the other side.

Activity #18: Calculating the Movement Range of the track wheel.

The track wheel has 16 joint pieces altogether. Therefore, in one complete round of the track wheel, all 16 joints of the track will move. If you measure the length of all 16 joints of the track wheel you will get exactly 10 cm. This means that when the track wheel moves one complete round, it will have traveled a distance of 10 cm.



As for counting the distance through the slots on the encoder wheel, there are 9 black slots and 9 white slots. Therefore, in one round, the variable used to store the counts should count until 9. In this next activity, we will test measuring the movement of the RoboTank robot that has the encoder wheel sensor attached



B18.1 Type in Program B18-1 and download it to the RoboTank robot.

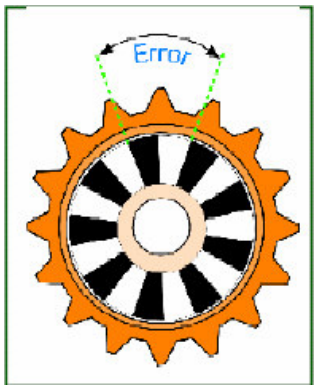
```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
CNT VAR Byte
DIRD = $F ' Motor control pin set to output
Main:DO : LOOP UNTIL (IN1 = 0) ' Wait until keypress to start
CNT = 0 ' Clear COUNTER
DO
GOSUB Forward ' Start moving robot
CH = 2 : GOSUB RD_ADC ' Check status from ENCODER
IF ADC > 300 THEN ' Check WHITE first
DO
CH = 2 : GOSUB RD_ADC
LOOP UNTIL ADC < 200 ' Loop check until goto BLACK
CNT =CNT+1 ' Increment COUNTER
ENDIF
LOOP UNTIL CNT = 8 ' Check counter exit loop when COUNTER = 8
GOSUB Motor_OFF ' Stop moving robot
GOTO Main ' Again
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Send Select Chip
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR ' Show warning when can't read from A to D
RETURN
'+++++ Movement Procedure ++++++++++++++++++++++++++++
Forward: HIGH 13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
' Motor A --> and Motor B --> Go Forward
Backward: HIGH 12 : LOW 13 : HIGH 14 : LOW 15 : RETURN
' Motor A <-- and Motor B <-- Go Backward
T_Left: HIGH 13 : LOW 12 : LOW 15 : LOW 14 : RETURN ' Robot Turn Left
T_Right: LOW 13 : LOW 12 : HIGH 15 : LOW 14 : RETURN ' Robot Turn Right
S_Left: HIGH 13 : LOW 12 : HIGH 14 : LOW 15 : RETURN ' Robot Spin Left
S_Right: HIGH 12 : LOW 13 : HIGH 15 : LOW 14 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = 0 : RETURN ' Stop All Motor
```

How the Program Works:

This program uses the count function from Program B17-1 and adds to it a program to drive the robot. It determines the distance by defining 1 complete round of the track wheel to equal a movement distance of 10 cm. The count value starts at 0 and completes one round when the value equals 8. From this type of movement, if the distance is known, the count value can be calculated using the following equation:

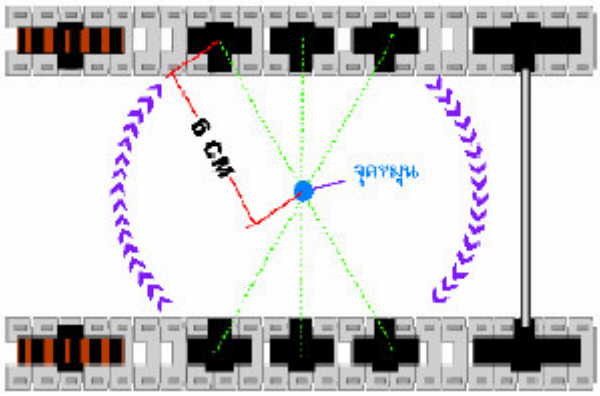
The +-1 value is the error value that occurs from searching for the white area in the first count, since at that time it was unknown whether it started detecting from a white or black slot.

B18.2 Turn off the switch and remove the download cable from the RoboTank robot. Place the robot on a flat surface and turn on the power switch. Observe the robot's movements and measure the distance from the starting point of the RoboTank robot to the point it stopped. Compare the results with the one that you got through calculation.



Calculating to find the value of a 90 degree turn.

The distance that occurs when the RoboTank robot makes a 90 degree turn can be calculated from the circumference of the circle caused when turning.
The circumference can be found by finding the radius of the turn first. Here, the radius was found to be 6 cm. Therefore, the circumference of the turn is …..



When you need to find the value of a 90 degree turn, you must divide it by 4 again, which makes the distance of a 90 degree turn equal to 9.42 cm. If used to calculate the count value, you will get a value of 7.

Conclusion: To find the distance value of a 90 degree turn, the count value must be 7.

Activity #19: The RoboTank moves in a square.

In this activity, we present a program to make the RoboTank robot move square-like, similar to the previous activity. The difference is that we will not be specifying the time it takes to turn, but will use a value to control the turn distance and its forward movement.

B19.1 Type in Program B19-1 and download it to the RoboTank robot.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
CH VAR Nib
ADC VAR Word
CNT VAR Byte
Distance VAR Byte
DIRD = $F ' Motor control pin set to output
GOSUB Wait_KEY
Main:GOSUB Forward
Distance = 9 : GOSUB Encoder
GOSUB S_Left ' Start moving robot
Distance = 7 : GOSUB Encoder
GOSUB Motor_OFF ' Stop moving robot
GOTO Main ' Again
Encoder: CNT = 0 ' Clear COUNTER
DO
CH = 2 : GOSUB RD_ADC ' Check status from ENCODER
IF ADC > 300 THEN ' Check WHITE first
DO
CH = 2 : GOSUB RD_ADC
LOOP UNTIL ADC < 200 ' Loop check until goto BLACK
CNT =CNT+1 ' Increment COUNTER
ENDIF
LOOP UNTIL CNT = Distance ' Check counter exit loop when COUNTER = 8
RETURN
Wait_KEY: DO : LOOP UNTIL (IN1 = 0) : RETURN ' Wait until keypress to start
'+++++ Analog to Digital Converter Procedure ++++++++++
RD_ADC: LOW 10: PAUSE 2: HIGH 10 ' Send Acknowledge
SEROUT 10,240,[CH] ' Send Select Chip
SERIN 10,240,250,Error,[ADC.BYTE0,ADC.BYTE1] ' Read ADC
RETURN
Error: DEBUG "Error Reading",CR ' Show warning when can't read from A to D
RETURN
```

```
'+++++ Movement Procedure ++++++++++++++++++++++++++++++
Forward: OUTD = %1010 : RETURN ' Motor A --> and Motor B --> Go Forward
Backward: OUTD = %0101 : RETURN ' Motor A <-- and Motor B <-- Go Backward
S_Left: OUTD = %0110 : RETURN ' Robot Spin Left
S_Right: OUTD = %1001 : RETURN ' Robot Spin Right
Motor_OFF: OUTD = 0 : RETURN ' Stop All Motor
'++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Additional Program Explanation:

The most important part of this program is the sub-program Encoder which is used to read the values from the black and white encoders on the wheel. When the value in the variable CNT equals the value that is stored in the variable Distance, the sub-program will be exited. The programm.er can determine the distance of movement of the robot by defining the value to the variable Distance.

Observe the changes in this program, especially the sub-program used to drive the robot. Instead of the comm.ands HIGH and LOW, this program uses the comm.and OUTD which sends data to P12, P13, P14, and P15 of the i-Stamp, the main micro-controller used to control the RoboTank robot. The values are defined in binary numbers.

In using the OUTD comm.and, the output ports must first be defined. In the first line you see the comm.and DIRD - $F in which $F is a hexadecimal number which is equal to %1111 when changed to binary format. Let all 4 terminals P12, P13, P14, and P15 be output terminals.

B19-2 Turn off the switch and remove the download cable from the RoboTank robot. Place the robot on a flat surface and turn on the power switch. Observe the movements of the robot.

RoboTank Electronic Compass

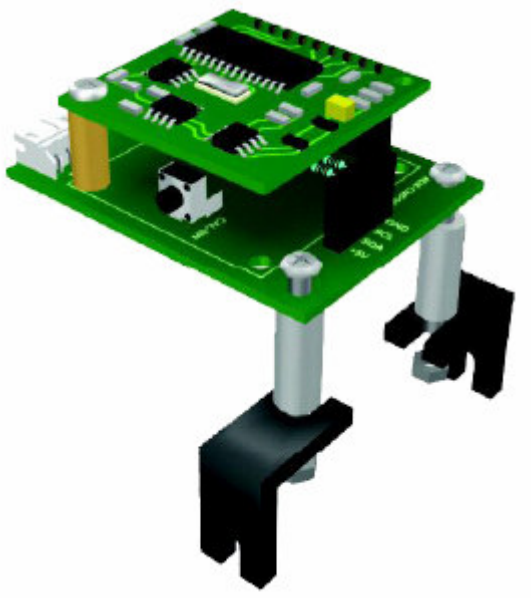
Robo-Stamp 2.0 Smart Robot kit

Since the RoboTank is not able to check its own position, using an Electronic Compass module will help the RoboTank robot move in the desired direction.

The electronic compass module CMPS03 is an additional component that needs to be obtained separately.

Activity #20: Installing the CMPS03 Electronic Compass Module

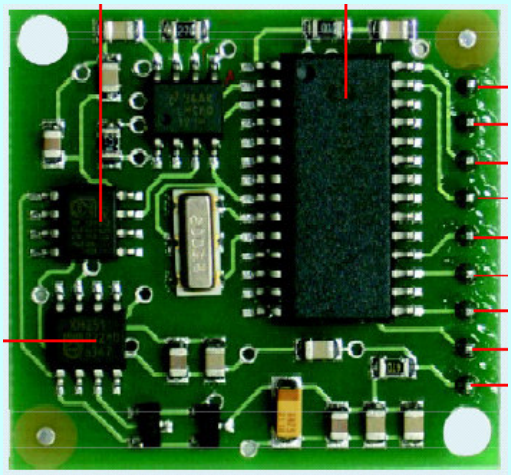
Since the CMPS03 electronic compass module uses the earth's magnetic field to determine the reference direction, the CMPS03 should not be installed near the motor on the RoboTank robot. It should be lifted away from the floor.



B20.1 Attach the CMPS03 Electronic Compass to a connection module ADX-CMPS03. Then place a 3 x 25 mm. screw in the two corners of the ADX-CMPS03, followed by a 10 mm. plastic spacer and screw it in together loosely with a 3 mm. nut. Then place an angled joiner between the nut and spacer before screwing it tightly. Repeat the same for both sides, positioning the angled joiner as shown in the figure.

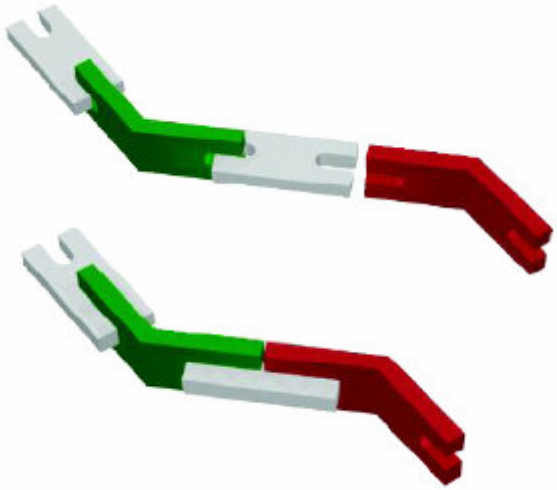
CMPS03 Electronic Compass Module

- Uses +5V power supply and current of 20 mA
- Uses 2 Phillips KMZ51 magnetic field detector
- 0.1 degree angle accuracy and approximate 3-4 degree error
- The width of the output pulse is 1 to 36.99 ms, with an increase rate of 0.1 ms/pulse
- Two forms of data output through the Bus I²C system: 0 to 255 and 0 to 3599.

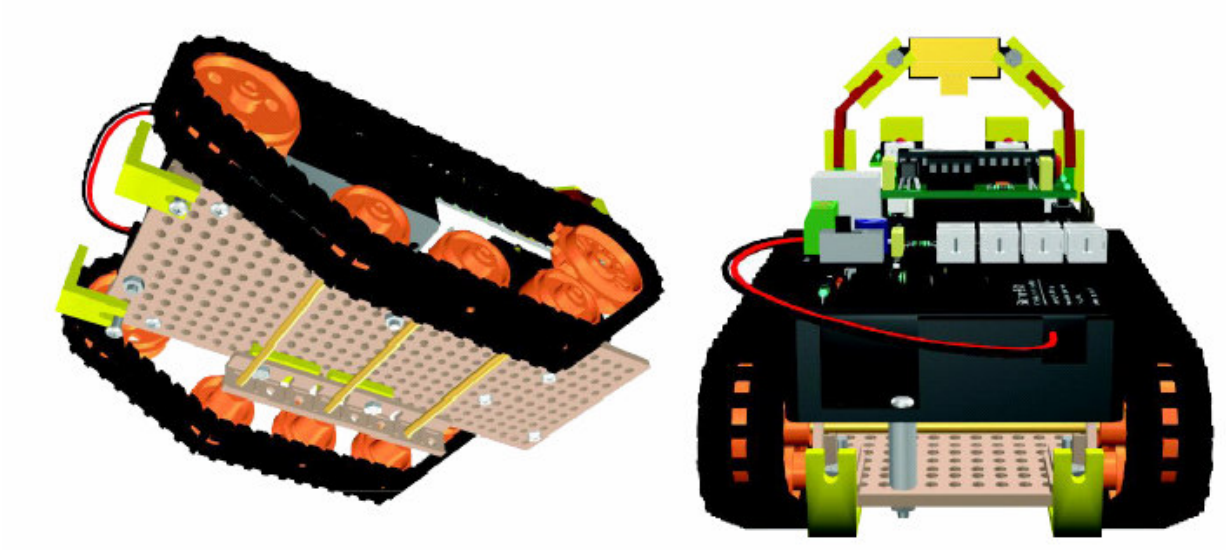


ADX-CMPS03 แผงวงจรเชื่อมต่อโมดูล CMPS03

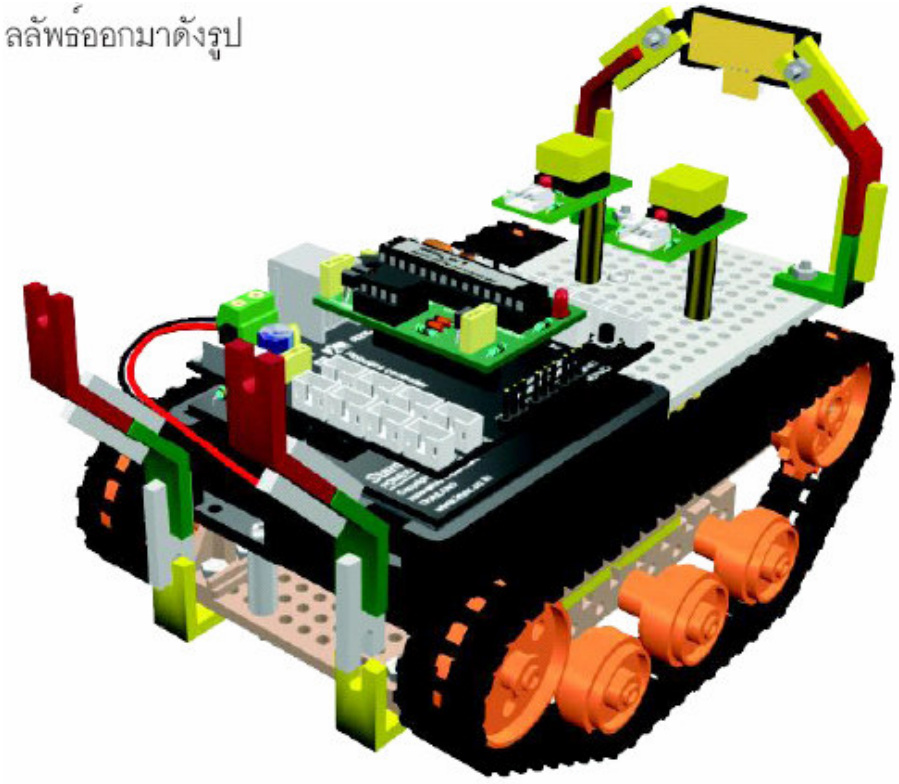
B20.2 Create the robot arms by connecting an obtuse joiner to a strength joiner. Connect another obtuse joiner to the other end of the angled joiner. Then connect a strength joiner to the other end of the obtuse joiner. Make two sets and you will get arm sets as shown in the figure.



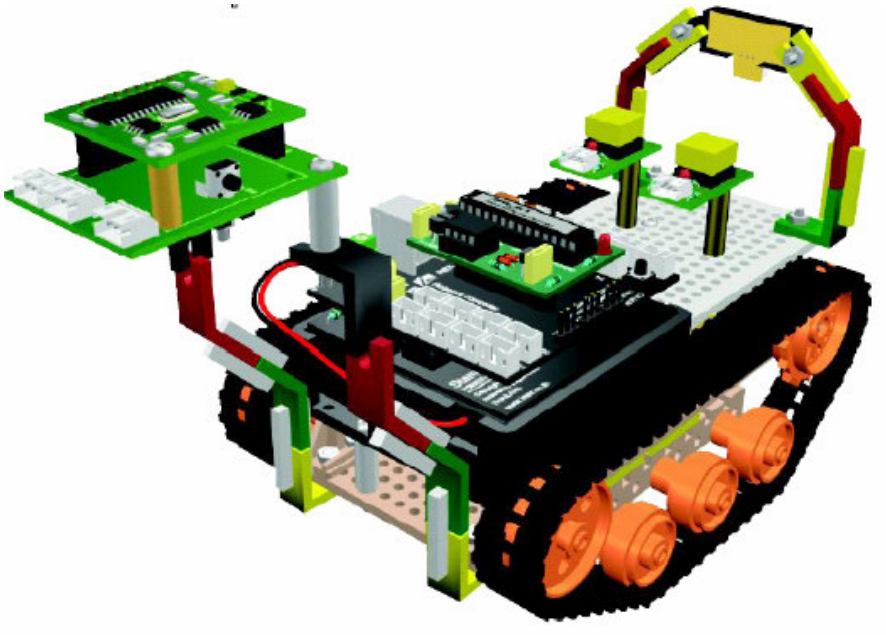
B20.3 On the RoboTank robot, loosen the screws that are used to hold in the short angled shaft base. Then place an angled joiner between it and retighten the screw again with a screwdriver. Do this for both sides of the short angled shaft base.



B20.4 Take the arms that was built in B20.2 and attach it to the other end of the angled joiner that is now connected to the RoboTank robot from B20.3. Make sure that the side with the strength joiner is used to connect with the angled joiner of the robot. Do the same for both the left and right sets which will result in the figure below.



B20.5 Bring the CMPS03 Digital Compass Module and place it on the arm set, turning the cables towards the robot as shown in the figure.



B20.6 Connect the signal cables from the ADX-CMPS03 to the Stamp-BOX on the RoboTank robot as following

- Connect terminal PWM from the ADX-CMPS03 board to connection point P3.
- Connect terminal SCL from the ADX-CMPS03 board to connection point P0
- Connect terminal SDA from the ADX-CMPS03 board to connection point P1.

The RoboTank robot is now ready to read the values from the CMPS03 Electronic Compass Module. It can be seen that the CMPS03 module is installed high from the robot, and even higher from the motor. This is to prevent interference from the magnetic field of the motor's magnet.

Activity #21: Reading Directions from the CMPS03 Using PWM mode

Reading the values from the CMPS03 module using this mode means that the width of the pulse signal will be used to determine the position of the angles from 0 to 359.9 degree. The width of the pulse signal is between 1 to 36.99 milliseconds, or 0.1 milliseconds per degree. For each pulse signal, there would be a logic pulse "0" which equals 65 ms.

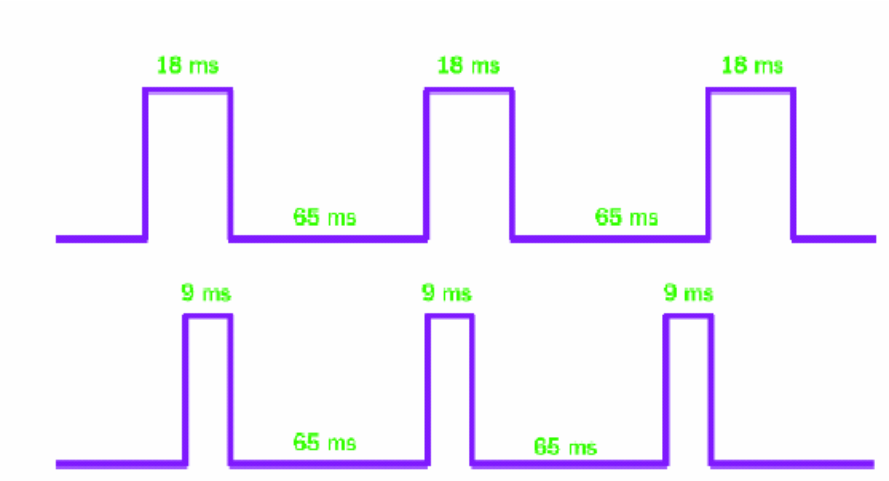


Figure B20-1 Displays the PWM signal received from the CMPS03 Electronic Compass module.

Writing a control program with i-Stamp

The i-Stamp, which is the main micro-controller used to control the RoboTank robot, uses the PULSIN command to read the PWM pulse values from the CMPS03 module. This comm.and will count the width of the positive pulses, with one count being 0.8 ms. The program will display the direction value on the screen, as shown in Program B21-1.

B21.1 Type in Program B21-1 and download it to the RoboTank robot that has the CMPS03 Electronic Compass module already installed from Activity #20. After the download is complete, do not remove the download cable.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
ANGLE VAR Word
Main: PULSIN 3, 1, ANGLE ' Get reading
ANGLE = (ANGLE-1250)/125 ' Calculate bearing in degrees
DEBUG "Position = ", DEC ANGLE ,CR ' Display Compass Bearing
PAUSE 300
GOTO Main
```

Additional Program Explanation:

The lowest value of the pulse width that can be read from the CMPS03 module is 1 ms at 0 degrees, which has a data value equaling 1250. Since we want 0 to be displayed on the Debug Terminal window, we must minus the value by 1250 first. Likewise, the width of the highest pulse is 36.99 ms. The value from the PULSIN comm.and would be equal to 46237. The value must be subtracted by 1250 and divided by 125 to get a value of 259, the highest degree. The results will be displayed on the Debug Terminal screen in the decimal format.

Robo-Stamp Note

In using the CMPS03 module, sometimes the highest value read from the PULSIN comm.and of the i-Stamp may not reach 46,237. This is because the CMPS03 module sent a pulse whose width does not reach an angle of 359.9 degrees. The user may modify the calculation to determine a new value in order to get a 359.9 degree angle.

B21.2 Run the program. The Debug Terminal window will appear. Test turning the RoboTank robot in different directions and observe the value of the direction angle that the CMPS03 reads. Compare it to the actual angle degree.

North would display an angle value of 0 degrees
East would display an angle value of 90 degrees
South would display an angle value of 180 degrees.
West would display an angle value of 270 degrees

If the results are not as above, the hardware might have to be modified, which will be talked about in the next topic.

Modifying the reference direction for the CMPS03 module.

In using the CMPS03 digital compass module, the north reference value may not be correct at first. Therefore, a modification is needed by pressing the switch on the ADX-CMPS03 module. The steps for the modification are as following:

1) Place the robot so that the compass module CMPS03 is parallel to the floor and that the front of the module is facing north. Then press the switch once.

2) Place the robot so that the compass module CMPS03 is parallel to the floor and that the front of the module is facing east. Then press the switch once.

3) Place the robot so that the compass module CMPS03 is parallel to the floor and that the front of the module is facing south. Then press the switch once.

4) Place the robot so that the compass module CMPS03 is parallel to the floor and that the front of the module is facing west. Then press the switch once.

This concludes the modifications of the reference directions. The CMPS03 module will store the value of these reference directions in the EE-prompt memory. The values need not be adjusted again when the power is turned on, unless there is a need to set the values of the angles in a different way.

Activity #22: RoboTank searches for the Direction

In this activity, we use the Electronic Compass Module CMPS03 to help the RoboTank robot search for the direction it wants.

B22.1 Type in program B22-1 and download it to the RoboTank robot which has the CMPS02 module installed already from Activity 20. After the download is complete, do not remove the download cable yet.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
bearing VAR Word
Main:DO : LOOP UNTIL (IN1 = 0)
GOSUB S_Left
DO
PULSIN 3, 1, bearing ' Get reading
bearing = (bearing-1250)/125 ' BS2sx - Calculate Bearing in degrees
DEBUG "Compass Bearing ", DEC bearing ,CR ' Display Compass Bearing
LOOP UNTIL ((bearing < 220 ) AND (bearing > 195))
GOSUB Motor_OFF
GOTO Main
'+++++ Movement Procedure ++++++++++++++++++++++++++++++
Forward: HIGH13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
Backward: HIGH12 : LOW 13 : HIGH 14 : LOW 15 : RETURN
T_Left: HIGH13 : LOW 12 : LOW 15 : LOW 14 : RETURN
T_Right: LOW 13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
S_Left: HIGH13 : LOW 12 : HIGH 14 : LOW 15 : RETURN
S_Right: HIGH12 : LOW 13 : HIGH 15 : LOW 14 : RETURN
Motor_OFF: LOW13 : LOW 12 : LOW 15 : LOW 14 : RETURN
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Additional program Explanation:

At the label Main, the program will wait for the switch connected to P1 to be pressed. When the switch is pressed, the RoboTank robot will rotate left and then the CMPS03 module will read the direction, convert it to angle degrees, and display it on the Debug Terminal. window.

Then it will check if the direction is within the specified range. If not, it will tell the robot to continue rotating left and read the value from the CMPS03 again. It will continue to operate like this until the value of the direction is within the specified range, which is between 196 and 219. The RoboTank robot will then stop and the program will jump to MAIN to wait for the switch to be pressed again and to start working once more.

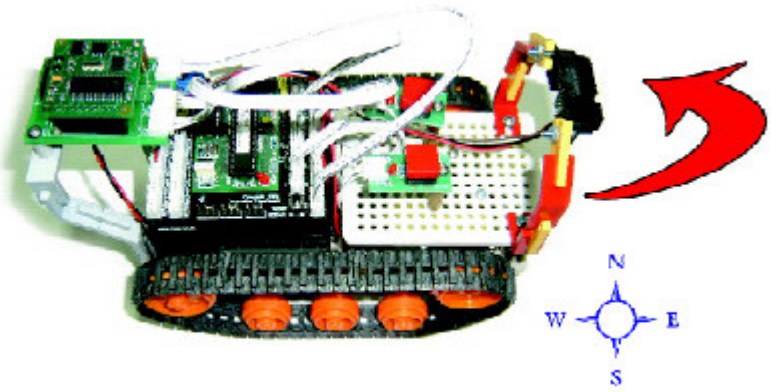
B22.2 Place the RoboTank robot on a flat surface, with the download cable still attached. Run the program and the Debug Terminal window will appear. Press the switch connected to P1 on the Stamp-BOX of the RoboTank robot. Observe how the robot works.

The RoboTank robot will read the values from the CMPS03 module first to check whether the position of the robot is within the specified direction or angle value or not. If not, the RoboTank robot will turn left and read the value from the CMPS03 module to check the position again. It will continue this until it gets the angle value within the range it wants. The RoboTank robot will then stop moving.

Modifications

From Program B22-1 the RoboTank robot only rotates in one direction. If the specified direction of the robot is towards the right of the robot, it would have to waste time rotating 1 round before it reaches the specified position. Also, the calculation used to display the values on the Debug Terminal may cause the Electronic compass module to not be able to read the values fast enough and the RoboTank robot may rotate past the specified direction. The above problems can be solved by editing the software. The program is modified to Program B22-2.

B22.3 Type in Program B22-2 and download it to the RoboTank robot. That has the CMPS03 electronic compass installed from Activity #20. After the download is complete, do not remove the download cable.



```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
ANGLE VAR Word
Main: PULSIN 3, 1, ANGLE ' Get reading
DEBUG "Position = ", DEC ANGLE ,CR ' Display Compass Bearing
IF ((ANGLE > 30000) AND (ANGLE < 33000)) THEN' Check Range to stop
GOSUB Motor_OFF
ELSEIF (ANGLE < 31500) THEN
GOSUB S_RIGHT 'Turn right when angle lower
ELSE
GOSUB S_LEFT 'Turn left when angle higher
ENDIF
GOTO Main
'+++++ Movement Procedure ++++++++++++++++++++++++++++++
Forward: HIGH13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
Backward: HIGH12 : LOW 13 : HIGH 14 : LOW 15 : RETURN
T_Left: HIGH13 : LOW 12 : LOW 15 : LOW 14 : RETURN
T_Right: LOW 13 : LOW 12 : HIGH 15 : LOW 14 : RETURN
S_Left: HIGH13 : LOW 12 : HIGH 14 : LOW 15 : RETURN
S_Right: HIGH12 : LOW 13 : HIGH 15 : LOW 14 : RETURN
Motor_OFF: LOW13 : LOW 12 : LOW 15 : LOW 14 : RETURN
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

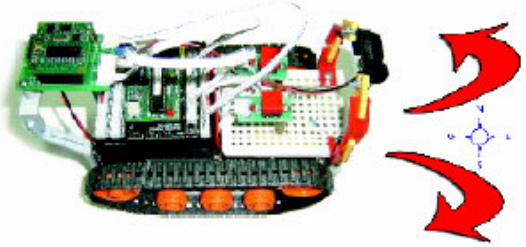
Addition Program Explanations

In this program the i-Stamp will read the pulse signal from the PWM terminal of the CMPS03 module and display the value on the Debug Terminal window instantaneously without converting anything. Then the value will be compared with the desired angle value which in this place is specified as between 30,000 to 33,000. This is the approximate value of the southwestern direction, which is (30000-1250)/125 = 230 degrees.

If the direction is within the specified range, the program will order the motor to stop. If not, it must compare again to see if the value is more or less than 31,500. If the value is less, the robot will rotate right to reach the desired position with the shortest route. But if the value read is more, it should rotate left for the shortest route.

B22.4 Place the RoboTank on a flat surface, with the download cable still attached. Run the program and the Debug Terminal window will appear. Press the switch connected to P1 of the Stamp-BOX on the RoboTank robot. Observe how the robot works.

The RoboTank robot will read the values from the CMPS03 module first to check whether the position of the robot is at the specified direction or angle value or not. The RoboTank robot will move left or right depending on the value that it reads. The RoboTank robot will stop moving when it arrives at the desired angle.



Activity #23: Reading directions from the CMPS03 module Using the I²C Bus Mode

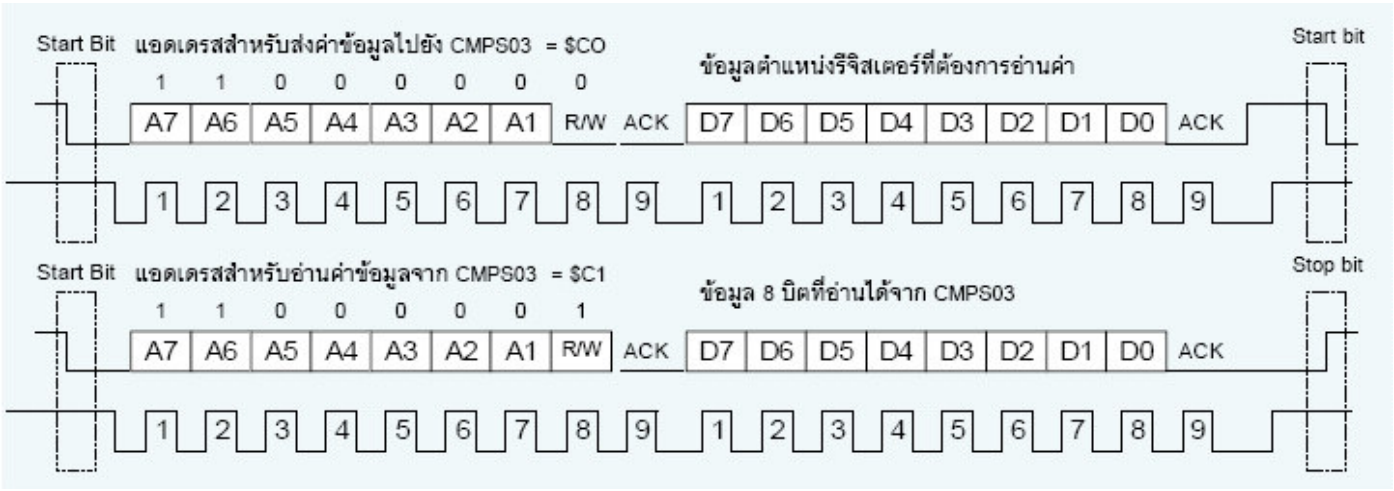
To get a high accuracy reading through the CMPS03 module, the I²C Bus system should be used. It could send position data up to 0.1 degrees without have to do any calculations or conversions. However, comm.unicating with the I²C Bus system may be a bit complicated in the beginning.

I²C Bus System

The I²C Bus will connect to the micro-controller by using 2 signal cables at terminal SDA (send and receive data) and terminal SCL (clock signal). Both terminals need to have pull-up resistors to define the logic "1" status for the bus system (pull-up resistors mean that one leg of the resistor is connected to the power supply which is +5V in this case). These components are already included in the ADX-CMPS03 module.

Steps of Communication

Many components can be connected to the I²C Bus. You can choose to comm.unicate which each individual component by defining an address value. The address of the CMPS03 module has 2 locations, $CO for sending and receiving data, and $C1 for reading the data. The steps for comm.unicating with the CMPS03 Module to read the data values are as following:



1) Send the start bit to let the Bus I2C system prepare to receive data.

2) Send the address $C0 to specify that you want to comm.unicate with the CMPS03 module in order to write data.

3) Send the register location of the CMPS03 module that you want to read. Details of the different registers can be found in Table B23-1.

4) Send the address $C0 to specify that you want to read data from the CMPS03 module.

5) Read the data from the CMPS03 module and store it in memory.

6) Send a stop bit to stop communicating and to let the bus system be in an empty bus status.

Details of the Register	
0	Version number of the CMPS03 module
1	Send a rough location value
4,5	Check the internal values by displaying the difference in Sensor 1 in the form of a 16-bit sign-sensitive number.
6,7	Check the internal values by displaying the difference in Sensor 2 in the form of a 16-bit sign-sensitive number
8,,9	16-bit sign-sensitive number
10,11	16-bit sign-sensitive number
12,13	Not in use; Reads a value of 0
14	Not in use; Undefined value
15	Command to modify the value. To modify the value, 255 must be entered into this register location.

From the steps in communication, the i-Stamp in the Stamp-BOX of the RoboTank robot can be used to write a sample program to read the data from the CMPS03 module, as shown in Program B23-1

B23-1 Type in Program B23-1 and then download it to the RoboTank robot that already has the Electronic Compass Module CMPS03 installed from Activity #20. After finished, do not remove the download cable yet.

```
'{$STAMP BS2sx}
'{$PBASIC 2.5}
SDA CON 1 ' I2C serial data line
SCL CON 0 ' I2C serial clock line
WrCMPS03 CON $C0 'write to compass
RdCMPS03 CON $C1 ' read from compass
Ack CON 0 ' acknowledge bit
Nak CON 1 ' no ack bit
i2cSDA VAR Nib ' I2C serial data pin
i2cData VAR Word ' data to/from device
REGISTER VAR Byte ' register address
i2cWork VAR Byte ' work byte for TX routine
i2cAck VAR Bit ' Ack bit from device
temp VAR Word ' for rj printing
digits VAR Nib
width VAR Nib
Init:
PAUSE 250
i2cSDA = SDA ' define SDA pin
REGISTER = 0 ' compass revision number
GOSUB Read_Byte
DEBUG 2,1,1, "Revision Number = ",DEC2 i2cData
Main:
REGISTER = 1 ' Show Data 0-255 for 0-360 degree
GOSUB Read_Byte ' Read Byte From I2C
DEBUG 2,1,3, "The Coarse Data(0-255) = ",DEC i2cData
REGISTER = 2 ' get Data in degrees, 0.0 - 359.9 Degree
GOSUB Read_Word
DEBUG 2,1,5, "Position = ",DEC i2cData/10,".",DEC1 i2cData," Degree"
PAUSE 250
GOTO Main
' Compass Access Subroutines
' Writes low byte of i2cData to REGISTER
Write_Byte:
GOSUB I2C_Start
i2cWork = WrCMPS03
GOSUB I2C_TX_Byte ' send device address
i2cWork = REGISTER
GOSUB I2C_TX_Byte ' send register number
i2cWork = i2cData.LOWBYTE
GOSUB I2C_TX_Byte ' send the data
GOSUB I2C_Stop
RETURN
```

```
' Writes i2cData to REGISTER
Write_Word:
GOSUB I2C_Start
i2cWork = WrCMPS03
GOSUB I2C_TX_Byte ' send device address
i2cWork = REGISTER
GOSUB I2C_TX_Byte ' send register number
i2cWork = i2cData.HIGHBYTE
GOSUB I2C_TX_Byte ' send the data - high byte
i2cWork = i2cData.LOWBYTE
GOSUB I2C_TX_Byte ' send the data - low byte
GOSUB I2C_Stop
RETURN
' Read i2cData (8 bits) from REGISTER
Read_Byte:
GOSUB I2C_Start
i2cWork = WrCMPS03
GOSUB I2C_TX_Byte ' send compass address
i2cWork = REGISTER
GOSUB I2C_TX_Byte ' send register number
GOSUB I2C_Start ' repeat start (sets register)
i2cWork = RdCMPS03
GOSUB I2C_TX_Byte ' send read command
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData = i2cWork ' return the data
RETURN
' Read i2cData (16 bits) from REGISTER
Read_Word:
GOSUB I2C_Start
i2cWork = WrCMPS03
GOSUB I2C_TX_Byte ' send compass address
i2cWork = REGISTER
GOSUB I2C_TX_Byte ' send register number
GOSUB I2C_Start ' repeat start (sets register)
i2cWork = RdCMPS03
GOSUB I2C_TX_Byte ' send read command
GOSUB I2C_RX_Byte
i2cData.HIGHBYTE = i2cWork ' read high byte of data
GOSUB I2C_RX_Byte_Nak
GOSUB I2C_Stop
i2cData.LOWBYTE = i2cWork ' read low byte of data
RETURN
' Low Level I2C Subroutines
'Start
I2C_Start: ' I2C start bit sequence
INPUT i2cSDA
INPUT SCL
LOW i2cSDA ' SDA -> low while SCL high
Clock_Hold:
IF (INS.LOWBIT(SCL) = 0) THEN Clock_Hold ' device ready?
RETURN
'Transmit
I2C_TX_Byte:
SHIFTOUT i2cSDA,SCL,MSBFIRST,[i2cWork\8] ' send byte to device
SHIFTIN i2cSDA,SCL,MSBPRE,[i2cAck\1] ' get acknowledge bit
RETURN
```

```
'Receive
I2C_RX_Byte_Nak:
i2cAck = Nak ' no Ack = high
GOTO I2C_RX
I2C_RX_Byte:
i2cAck = Ack ' Ack = low
I2C_RX:
SHIFTIN i2cSDA,SCL,MSBPRE,[i2cWork\8] ' get byte from device
SHIFTOUT i2cSDA,SCL,LSBFIRST,[i2cAck\1] ' send ack or nak
RETURN
'Stop
I2C_Stop: 'I2C stop bit sequence
LOW i2cSDA
INPUT SCL
INPUT i2cSDA ' SDA --> high while SCL high
RETURN
```

Additional Program Explanation:

This program is written from the comm.unication steps with the equipment using the I²C Bus system; therefore the program consists mainly of sub-programs that comm.unicate with the Bus I2C system. However, the main program has the following steps of operation

1) Define the register as 0 to read the version of the CMPS03. Then display the value on the Debug Terminal window.

2) Let the program loop at the main program. Then send a register value of 1 so that it reads the rough data and displays it on the Debug Terminal window. The value you get will be between 0-255

3) Send the register value of 2 to read the complete data. Then read the data from the Bus I2C in the word format (16-bit)

4) Take the data value received and divide it by 10 first in order to convert it into degrees. Display it on the Debug Terminal window. Then display the last digit which is the decimal place.

5) Loop in order to read and display the values continuously.

From this program example, the sub-programs that are used to comm.unicate with the I²C bus system components here can be used with any other component that uses the I²C Bus system to comm.unicate.

B23.2 Run the program and the Debug Terminal window will appear. Test rotating the RoboTank robot in different directions and observe the values of the direction angle that you read from the CMPS03 module. Compare the results to the actual angle.