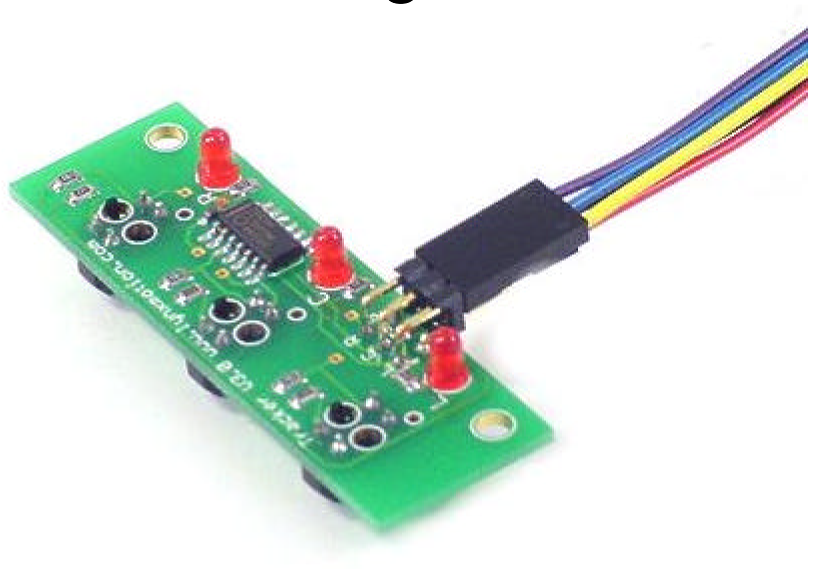
Tracker Ver 3.0

Line Tracking Sensor



Lynxmotion, Inc.
PO Box 818
Pekin, IL 61555-0818
Tel: 309-382-1816 (Sales)
Tel: 309-382-2760 (Support)
Fax: 309-382-1254
E-m: sales@lynxmotion.com
E-m: tech@lynxmotion.com
Web: http://www.lynxmotion.com

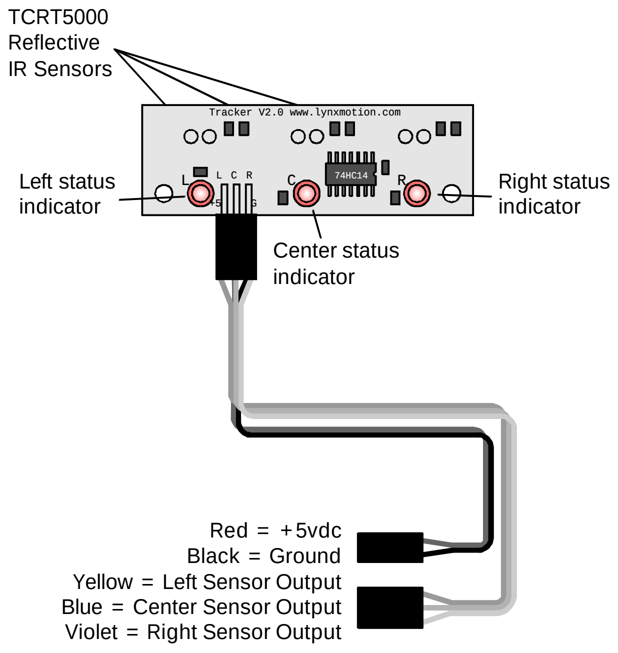
Users Manual TRA-01 Ver 6.0

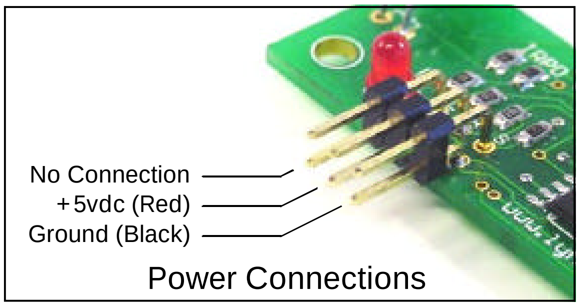
Tracker Ver 3.0

Caution! Read the manual completely before wiring and applying power to the board! Errors in wiring can damage the Tracker or the host microcontroller.

Caution! The wires for microcontroller I/O and power can break off if wires are bent back and forth repetitively. Mount the board and terminate the wires as soon as possible.

Caution! Do not reverse the power plug as damage to the sensor board will occur. These sensors are fully tested before they leave the factory. If the sensor is damaged by reverse power the warranty is void.





Using the Tracker

Mounting the Tracker Sensor

The sensor can be mounted to the underside of a robot chassis toward the front of the vehicle. Position the sensor close to the floor with the red LEDs facing up. The sensor will operate in an extremely wide range from about 0.5" from the floor to almost touching the surface. The sensor appears to be immune to normal ambient lighting, although it may be necessary to shield the sensor from extremes.

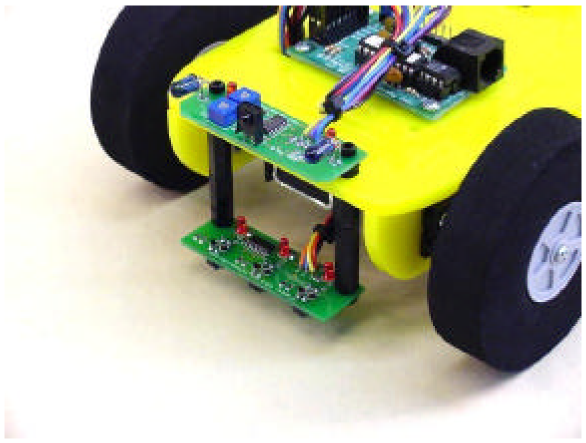


Illustration showing the Tracker mounted to a CR1 Robot.

Testing the Tracker Sensor

The sensor can easily be tested by following this simple step by step procedure.

1) Prepare a test bed using a sheet of white paper and a short section of black electrical tape.

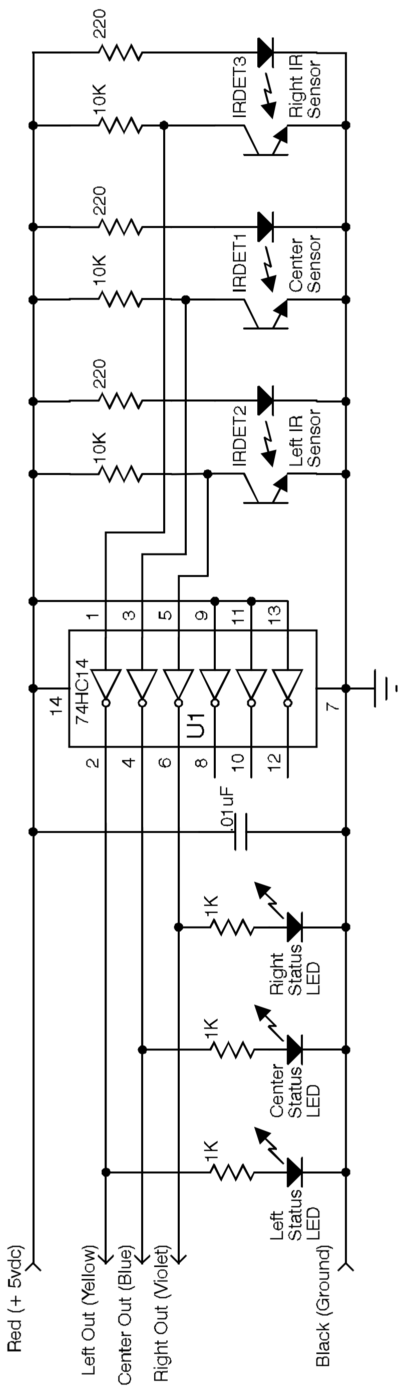
2) Apply a regulated 5vdc to the red wire and ground to the black wire.

3) Hold the sensor about 0.25" above the white paper and notice the red LEDs turn on. If not quickly remove power and double check your wiring!

4) Move the sensor over the electrical tape positioning each of the three IR sensor pairs over the tape one at a time. You will notice the LED that's positioned over the tape go out and come back on as it's moved back to the white section of the paper.

5) You can connect the outputs to a microcontroller to test them, or just connect the outputs to a volt mater. Each output will go high when positioned over white, and low when positioned over black.

Schematic Diagram



Using the Tracker

The Theory

The theory behind line tracking is actually pretty simple. An infrared LED is paired with an infrared detector. The LED is illuminated and directed to the surface where the line is to be detected. The detector is biased on and fed into a comparator to clean up the signal.

In order to keep the electronics as simple as possible a 74HC14 Schmidt-trigger hex inverter will replace the comparator circuitry. The very high input impedance, built in hysteresis and low parts count makes the CMOS version an excellent alternative.

Making a Course

Here is a list of the verified materials that can be used to make a line tracking course. I am sure with experimentation other materials can be used with equally successful results.

1) Black electrical tape on white or light colored floor tile.

2) Black electrical tape on white 3mm Sintra PVC. A 4x8 sheet can be cut into 32, 12" x 12" panels. Each panel can have a straight or turning pattern, so the course can be changed. A radius of 6" can easily be made. This combination makes a great, durable and portable line tracking course.

3) Black Sharpie marker on end rolls of newsprint or butcher paper. With this material the course can be made as long as needed.Use this for line tracking dragsters.

Additional Information

The programming is very simple. The outputs go low when the LED/IR Detector pair is positioned over a black surface and high when positioned over a white surface. Check for detection of a line. If the center sensor sees the line, go forward. If the left sensor sees the line, turn left. If the right sensor sees the line, turn right. You will want the robot to remember which sensor saw the line last and continue to make corrections in order to negotiate turns where the line will be out of the sensors view for a limited amount of time. I have included some sample files for the Basic Stamp. There is even some code that can handle a search for the line if it loses it completely. Experiment with the code and have fun.

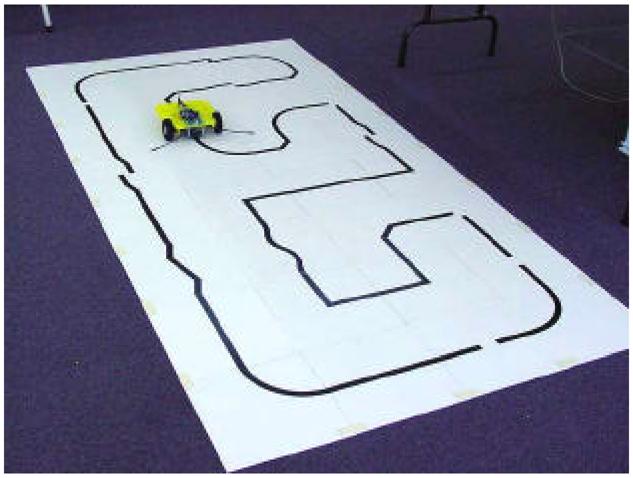


Illustration showing a CR2 following a line.

Programming Examples

```
'program: ltrack1.bas
'This program does simple line following.

symbol error_level = b0
symbol left_sensor = pin5
symbol center_sensor = pin6
symbol right_sensor = pin7
low 0    'Left Servo
low 1    'Right Servo

start:   'Line seen, correct direction
   if left_sensor = 0 then a
   if right_sensor = 0 then b
   if center_sensor = 0 then c
         'Line not seen, do last action
   if error_level = 1 then a
   if error_level = 2 then b
   if error_level = 3 then c
goto start

a: error_level = 1
   pulsout 0,150 'Left wheel stop
   pulsout 1,200 'Right wheel forward
   pause 10
goto start

b: error_level = 2
   pulsout 0,200 'Left wheel forward
   pulsout 1,150 'Right wheel stop
   pause 10
goto start

c: error_level = 3
   pulsout 0,200 'Left wheel forward
   pulsout 1,200 'Right wheel forward
   pause 10
goto start
```

```
'program: ltrack2.bas
'This program does advanced line
'following. If it loses the line it will
'do a search for it.

symbol x = b0
symbol y = b1
symbol correct = b2
symbol lost = b3
symbol left = pin5
symbol center = pin6
symbol right = pin7
low 0    'Left Servo
low 1    'Right Servo
lost = 1
correct = 1

'Line seen, correct position
start:
   if right = 0 then on_trac_right
start1:
   if left = 0 then on_trac_left
start2:
   if center = 0 then on_trac_center

'Line not seen, do last action
   lost = lost + 1
   if lost = 40 then find_line
   if correct = 1 then off_trac_center
   if correct = 2 then off_trac_left
   if correct = 3 then off_trac_right
goto start

on_trac_center:
   correct = 1
   lost = 0
   gosub forward
goto start
on_trac_left:
   correct = 2
   lost = 0
   gosub left_turn
goto start1
on_trac_right:
   correct = 3
   lost = 0
   gosub right_turn
goto start2

off_trac_center:
   correct = 1
   gosub forward
goto start
off_trac_left:
   correct = 2
   gosub left_turn
goto start
off_trac_right:
   correct = 3
   gosub right_turn
goto start

find_line:
  for x=1 to 50
    gosub look
    gosub left_spin
  next
  for x=1 to 100
    gosub look
    gosub right_spin
  next
  for x=1 to 50
    gosub look
    gosub left_spin
  next
  for x=1 to 50
    gosub look
    gosub forward
  next
goto find_line

look:
   if right = 0 then start
   if left = 0 then start
   if center = 0 then start
return

forward:
   pulsout 0,200 'Left wheel f orward
   pulsout 1,200 'Right wheel forward
   pause 10
return

left_turn:
   pulsout 0,150 'Left wheel stop
   pulsout 1,200 'Right wheel forward
   pause 10
return

right_turn:
   pulsout 0,200 'Left wheel forward
   pulsout 1,150 'Right wheel stop
   pause 10
return

left_spin:
   pulsout 0,100 'Left wheel reverse
   pulsout 1,200 'Right wheel forward
   pause 10
return

right_spin:
   pulsout 0,200 'Left wheel forward
   pulsout 1,100 'Right wheel reverse
   pause 10
Return
```