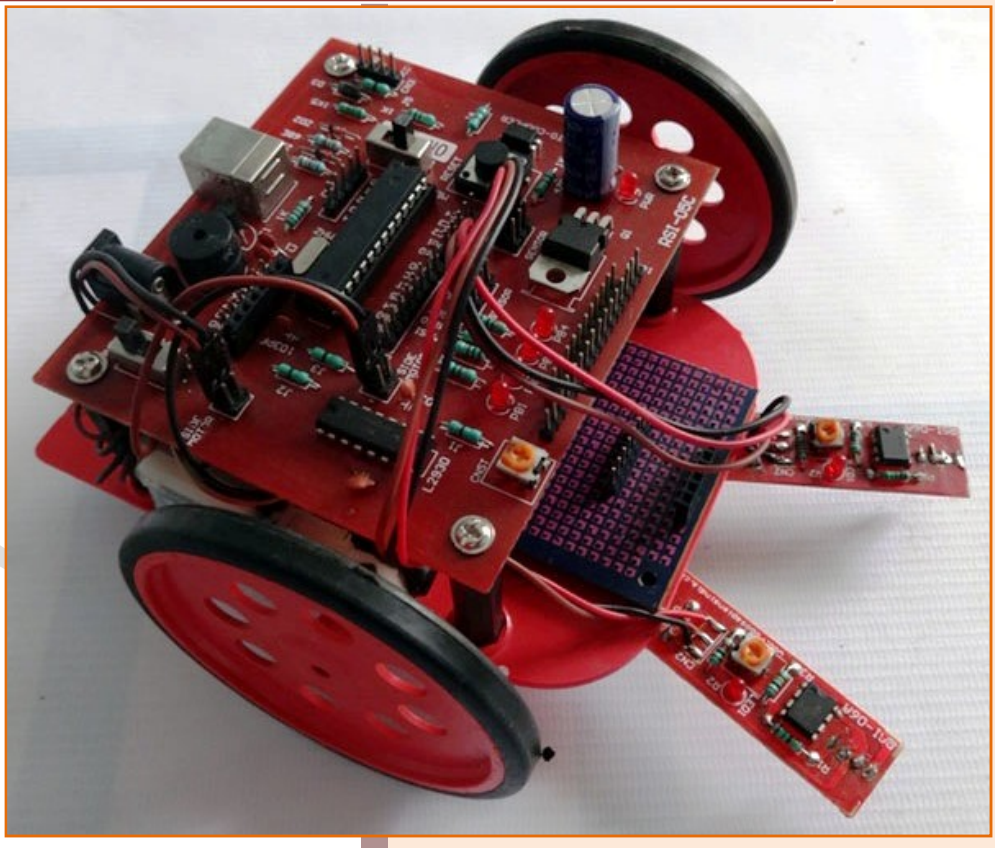
Learn basics of robotics in a week!

Complete Guide

Simple Robotic Programs



Dattaraj Vidyasagar

Vidyasagar Academy, Akola

www.vsagar.org

NOTES SPACE

Revised Edition

iBook Version @Amazon.com

Simple Robotic Programs

Complete guide on how to learn and enjoy robotic programming

(With more than 30 programs for your single robotic kit of ATMega8/16/128)

Dattaraj Vidyasagar

(Expert faculty in robotics and applied/hobby electronics since last 30 years)

Vidyasagar Academy, Akola

Phone: 099-60-991-991

Simple robotic programs

Useful notes for the students of robotics
For fundamental & advanced course in robotics
(The notes includes do-it-yourself complete practical guide of learning robotics)

First Edition: November 2015

Published By:
Vidyasagar Academy,
Mrs. C.D. Vidyasagar
42/3A, 772, Renuka, Ranpise Nagar, Akola
☎99-60-06-45-64

Designed, Typed & Edited by:
Dattaraj Vidyasagar (Author)
☎ 99 60 991 991

Printed at:
Milind Traders,
Ranpise Nagar, Akola
☎98-902-131-37

Acknowledgements:
The author extends his thanks and profound appreciation for all those who helped him directly or indirectly in bringing this notes in present stature.
The author welcomes any suggestions, *both from the teachers and the students,* for further improvement of this notes, at dsvakola@gmail.com. Visit his profile on www.vidyasagarsir.com to know more about him.

Price: Rs. 100/- *(In India only)*

Overseas Prices: ***$14.99 or £ 9.77, € 23.35,*** **¥1480**
When ordering this title online, use: VSA-SRP-DIY-2015

Contents

INTRODUCTION

Learning robotics is a real fun and gaining logical knowledge. In this guide, you will understand the step by step process of designing infinite combinations of different programs in robotics.

Each program is carefully written with lots of comments within the program. This will help you understand the command lines, different syntax and logical structure of the programs.

I suggest you to start from the very first program. It will give you the basic idea of controlling the primary hardware of your robot – the LEDs. Then the next program will be more fun to see that the LEDs produce running effect.

In running effect of LEDs, lots of command lines are needed to control the sequential on/off of the LEDs. So in the next program using the `'for loop'`, you understand that how we can reduce the length of the program in just two to three steps.

Then start with the basics of using IR sensors. A line following robot consists of an infrared light sensor and an infrared LED. It works by illuminating a surface with infrared light; the sensor then picks up the reflected infrared radiation and, based on its intensity, determines the reflectivity of the surface in question.

So in this next program, you will understand the use of sensor to control the LEDs first. Then step by step you can go through to understand the basics of black line and white line following robot's logic behind its simple working.

After that you can move to understand the use of different types of sensors as given in the following programs.

If you come across any difficulty or want to ask questions regarding the programs, please feel free to contact me at: www.vsagar.org/contact-us. I will reply you, ASAP.

So friends! Happy learning...!

Robotic kit used for all robotic programs

BASIC PROGRAMS USING LEDS

Program of Blinking 4 LEDs

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      PB4-PB1     : output pins of PORTB, connected to 4 LEDs

*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8

#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder


#include <util/delay.h> // including the delay file
                        // this file is inside the
                        // utilities (util) folder

      int main() // starting the main function of program

      { // main function brace opened

      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins

      while(1) // starting the infinite loop to repeat the action infinitely

      { // while loop brace opened

      PORTB=0b00000000; // PB4-PB1 producing '0' output
                                // so the 4 LEDs are OFF
      _delay_ms(1000); // the LEDs remain OFF for 1000ms=1sec

      PORTB=0b00011110; // PB4-PB1 producing '1' output
                        // so the 4 LEDs become ON
      _delay_ms(2000); // the LEDs remain ON for 1000ms=1sec

      } // while loop brace closed

      } // main function brace closed
```

HOW TO USE AND RUN THE PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. All the LEDs connected to PB4-PB1 will glow for some time
5. After some time all the LEDs will be OFF.
6. This on/off will repeat continuously till the battery is connected.
7. Is it working? Nice! You did it.
8. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

Running effect of 4 LEDs

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      PB4-PB1     : output pins of PORTB, connected to 4 LEDs
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                              // to define the input output ports and pins
                              // this file is inside the AVR folder

#include <util/delay.h> // including the delay file
                         // this file is inside the
                         // utilities (util) folder

      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      PORTB=0b00000010; // 1st LED at PB1 becomes ON
      _delay_ms(300); // it remains ON for 300ms=0.3sec
      PORTB=0b00000100; // 2nd LED at PB2 becomes ON
      _delay_ms(300); // it remains ON for 300ms=0.3sec
      PORTB=0b00001000; // 3rd LED at PB3 becomes ON
      _delay_ms(300); // it remains ON for 300ms=0.3sec
      PORTB=0b00010000; // 4th LED at PB4 becomes ON
      _delay_ms(300); // it remains ON for 300ms=0.3sec

      } // while loop brace closed

      } // main function brace closed
```

IMPORTANT COMMENT

After this last step, the program will jump to first step and will repeat infinitely, since it is within **`while(1)`** loop called as infinite loop.

You can also change the delay in milliseconds as required.

Running effect of LEDs using FOR LOOP

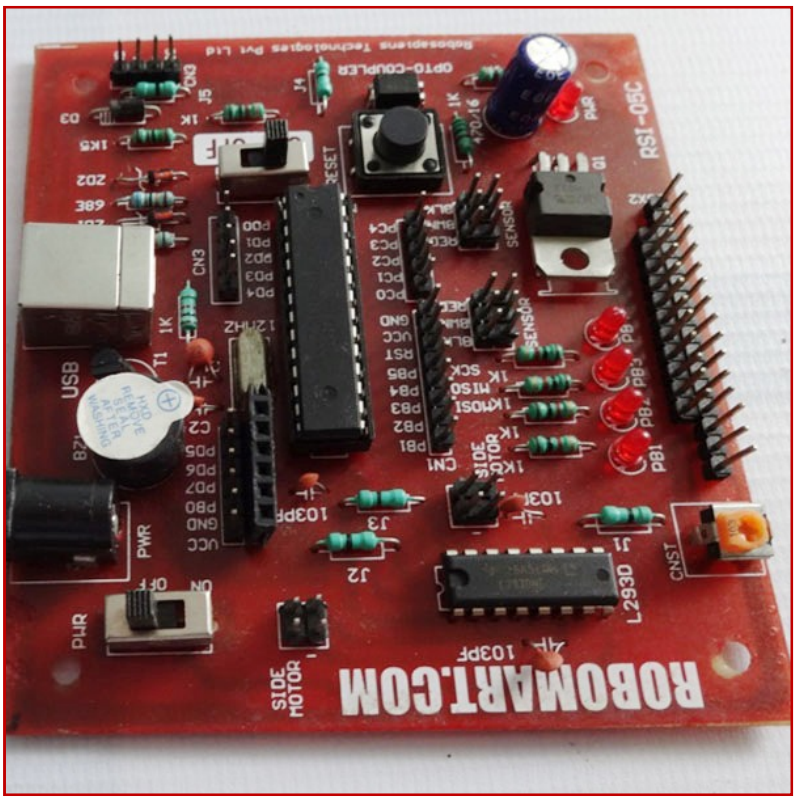
```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      PB4-PB1     : output pins of PORTB, connected to 4 LEDs
*/
#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                  // to define the input output ports and pins
                  // this file is inside the AVR folder
#include <util/delay.h> // including the delay file
                         // this file is inside the
                         // utilities (util) folder

      int main() // starting the main function of program
      { // main function brace opened
            int i=0; // variable to store the particular value
            DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
            for (i=0; i<4; i++)     // loop to count value of 'i'
            {
            PORTB=(1<<i);           // left shift operator
            _delay_ms (500);  // delay of 1 second
            }
      } // while loop brace closed
      } // main function brace closed
```


HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Now connect battery and switch on the kit.
5. The 4 LEDs will glow one-by-one in a particular direction.
6. Is it working? Nice! You did it.
7. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

IMPORTANT NOTE

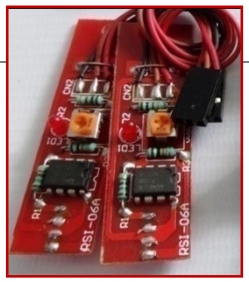
1. The **'for loop'** will start from '0'. At this condition, **PB1=1** only.
2. When **i=1, PB2=1** only.
3. When **i=2, PB3=1** only.
4. When **i=3, PB2=4** only.
5. In this way, using **'for loop'** you can get the same effect of running LEDs which we saw in previous project of running LEDs.



Development board used in the robotic kit

You can get it through our distance learning programme in robotics at: www.vsagar.org

UNDERSTANDING THE IR SENSORS



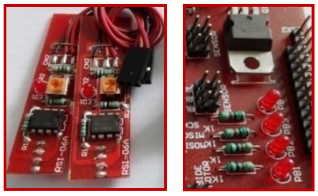
Basic program of IR sensor

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 4 LEDs in your kit, are internally connected to PB4-PB1
      2) Connect one IR sensor to PC0 in your kit.
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder
      int main() // starting the main function of program
      { // main function brace opened
            DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
            DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
            int s=0; // 's' is the variable to store sensor status
                     // when we write 'int s',
                    // it creates a location in memory of microcontroller.
                    // initially '0' is stored into 's' memory location
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      s=PINC&0b0000001; // assigning the variable 's' to PC0 of PORTC
                        // so that the output status of sensor will be
                        // stored into the variable 's'
                       // since one sensor in our kit is connected
                      // to PC0 and other to PC3
                    // Note: PC3 sensor is not used in this program
                    // only PC0 sensor is used
            if(s==0b0000001) // white surface below the sensor
            {
            PORTB=0b00011110; // all LEDs will be ON
            }
            else // black surface below the sensor
            {
            PORTB=0b00000000; // all LEDs will be OFF
            }
      } // while loop brace closed

      } // main function brace closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Connect one IR sensor, to PC0 in your kit.
5. Now connect battery and switch on the kit.
6. Keep a white paper below the IR sensor.
7. All the LEDs connected to PB4-PB1 will glow.
8. Remove the white paper, so that all the LEDs will be OFF.
9. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

Controlling 4 LEDs with 2 sensors



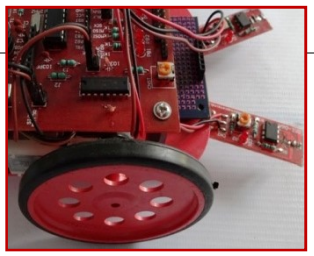
```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 4 LEDs in your kit, are internally connected to PB4-PB1
      2) Connect two IR sensors: LS to PC3 and RS to PC0.
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder
      int main() // starting the main function of program
      { // main function brace opened
            DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
            DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
            int s=0; // 's' is the variable to store sensor status
                     // when we write 'int s', it creates
                     // a location in memory of microcontroller.
                     // initially '0' is stored into 's' memory location
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      s=PINC&0b0001001; // masking the variable 's' to PC0 & PC3 of PORTC
                        // so that the output status of sensors will be
                        // recorded in variable 's'
      // Note: PC3 is left sensor and PC0 is right sensor
      // When it is black surface below a sensor, its output=0
      // When it is white surface below it, its output=1
            if(s==0b0001001) // white surface below both sensors
            {
            PORTB=0b00011110; // all LEDs ON
            }
```

```
        if(s==0b0000001) // white surface below right sensor
        {
        PORTB=0b00000110; // LEDs connected to PB2&PB1 are ON only
        }

        if(s==0b0001000) // white surface below left sensor
        {
        PORTB=0b00011000; // LEDs connected to PB4&PB3 are ON only
        }

        else // black surface below both sensors
        {
        PORTB=0b00000000; // all LEDs will be OFF
        }

} // while loop closed

} // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Connect both the IR sensors, to PC3&PC0 in your kit.
5. Now connect battery and switch on the kit.
6. Keep a white paper below one IR sensor.
7. First 2 LEDs will glow.
8. Now keep white paper below the next sensor, the other 2 LEDs will glow.
9. When you keep white paper below both sensors, all LEDs will glow.
10. Is it working? Nice! You did it.
11. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

HOW TO CONTROL MOTORS?



Moving the robot FORWARD & BACKWARD

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are internally connected, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                  // to define the input output ports and pins
                  // this file is inside the AVR folder of AVR Studio folder

#include <util/delay.h> // including the delay file
                         // this file is inside the
                         // utilities (util) folder of AVR Studio folder
      int main() // starting the main function of program
      { // main function brace opened
            DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
            PORTB=0b00000000; // both motors OFF
            _delay_ms(2000); // delay of 2000ms=2sec
            PORTB=0b00010010; // both motors rotate FORWARD
                              // so your robot will move forward for 1.5sec.
            _delay_ms(1500); // delay of 1500ms=1.5sec.
            PORTB=0b00000000; // both motors OFF, so robot STOPS
            _delay_ms(1000); // delay of 1000ms=1sec
            PORTB=0b00001100; // both motors rotate BACKWARD
                        // so your robot will move backward for 1.5sec.
            _delay_ms(1500); // delay of 1500ms=1.5sec.

            // after this step the program will jump to 1st line
            // and your robot will move to-and-fro

      } // while loop closed

      } // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Now keep your robot on a plane surface and switch it on.
5. Wait for 2sec. Now your robot will move forward for some time.
6. Then it will stop.
7. Now it will move in backward direction for some time.
8. In this way, it will continue to move to-and-fro for infinite times.
9. When you finish testing the program switch it OFF.
10. Is it working? Nice! You did it.
11. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

TURNING THE ROBOT LEFT & RIGHT

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      The 2 motors in your kit, are connected to PB4-PB1, as follows:
      Left motor:  PB4 -> (+) and PB3 -> (-)
      Right motor: PB1 -> (+) and PB1 -> (-)
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8

#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder
      int main() // starting the main function of program
      { // main function brace opened
            DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
            PORTB=0b00000000; // both motors OFF
            _delay_ms(1000); // delay of 1000ms=1sec
            PORTB=0b00010010; // both motors rotate FORWARD
                              // so your robot will move forward for 3sec.
            _delay_ms(2000); // delay of 2000ms=2sec.
            PORTB=0b00010000; // turning right
            _delay_ms(300); // for 0.3sec
            PORTB=0b00010010; // moving forward
            _delay_ms(1000); // for 1sec
            PORTB=0b00000010; // turning left
            _delay_ms(300); // for 0.3sec
```

```
            PORTB=0b00010010; // moving forward
            _delay_ms(1000); // for 1sec

      } // while loop closed

      } // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Connect both the IR sensors, to PC3 & PC0 in your kit.
5. Now connect battery and switch on the kit.
6. First your robot will move forward.
7. Then it will turn right.
8. Then again it will move forward, the turn left.
9. Finally it will go forward and then stop. And repeat endlessly.
10. *Is it working? Nice! You did it.*
11. Now don't forget to visit www.vsagar.org/contact-us and give your feedback.

U-TURNING ROBOT

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      The 2 motors in your kit, are connected to PB4-PB1, as follows:
      Left motor:  PB4 -> (+) and PB3 -> (-)
      Right motor: PB1 -> (+) and PB1 -> (-)
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8

#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder
#include <util/delay.h> // including the delay file
                        // this file is inside the
                        // utilities (util) folder
      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
```

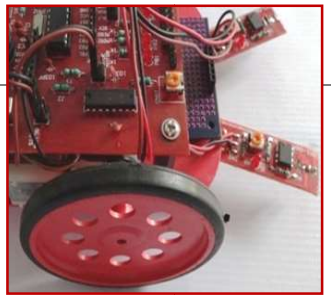
```
while(1) // starting the infinite loop to repeat the action infinitely
{ // while loop brace opened
      PORTB=0b00000000; // both motors OFF
      _delay_ms(1000); // delay of 1000ms=1sec
      PORTB=0b00010010; // both motors rotate FORWARD
                        // so your robot will move forward for 3sec.
      _delay_ms(2000); // delay of 2000ms=2sec.
      PORTB=0b00010000; // turning right
      _delay_ms(500); // for 0.5sec so that the robot will take U-turn
      PORTB=0b00010010; // moving forward
      _delay_ms(1000); // for 1sec
      PORTB=0b00000010; // turning left
      _delay_ms(500); // for 0.5sec so that the robot will take U-turn
                  // in opposite direction
      PORTB=0b00010010; // moving forward
      _delay_ms(1000); // for 1sec

} // while loop closed

} // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Connect both the IR sensors, to PC3&PC0 in your kit.
5. Now connect battery and switch on the kit.
6. First your robot will move forward.
7. Then it will take U-turn. Adjust the delay if required.
8. Then again it will move forward, then left U-turn.
9. Finally it will go forward and then stop. And repeat endlessly.
10. Is it working? Nice! You did it.
11. Now don't forget to give your feedback.

Note: You can adjust the delay of 500ms for U-turn by trial and error. Try changing it to 700ms to 800ms.

MOTORS WITH SENSORS



Controlling motors with sensors

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect one IR sensor to PC0 in your kit.
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8

#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder

#include <util/delay.h> // including the delay file
                        // this file is inside the
                        // utilities (util) folder
      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
      int s=0; // 's' is the variable to store the status value of sensor
            // when we write 'int s', it creates a location in memory of
            // microcontroller. Initially '0' is stored.

      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      s=PINC&0b0000001; // assigning the variable 's' to PC0 of PORTC
                        // so that the output status of sensor will be
                        // stored into the variable 's'
                        // since one sensor in our kit is connected
                        // to PC0 and other to PC3
Note: PC3 sensor is not used in this program only PC0 sensor is used

            if(s==0b0000001) // white surface below the sensor
            {
            PORTB=0b00010010; // both motors rotate in forward direction
            }
```

```
            else // black surface below the sensor
            {
            PORTB=0b00000000; // both motors are OFF
            }

      } // while loop closed

      } // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port. Burn the 'hex' file into your kit.
3. Connect both the IR sensors, to PC3 & PC0 in your kit. Now connect battery and switch on the kit.
4. First keep white paper below the sensor. The robot will move in forward direction.
5. Now keep black surface below it. Both motors will be OFF and the robot stops.
6. *Is it working? Nice! You did it.*
7. *Now don't forget to give your feedback.*

Controlling 2 motors with 2 sensors

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the two IR sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/

#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder
#include <util/delay.h> // including the delay file
                         // this file is inside the
                         // utilities (util) folder
      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
```

```
int s=0; // 's' is the variable to store the status value of sensor
      // when we write 'int s', it creates
      // a location in memory of microcontroller.
      // initially '0' is stored into 's' memory location
while(1) // starting the infinite loop to repeat the action infinitely
{ // while loop brace opened
s=PINC&0b0001001; // assigning the variable 's' to PC0 of PORTC
                  // so that the output status of sensor will be
                  // stored into the variable 's'
                  // since one sensor in our kit is connected
                  // to PC0 and other to PC3
```

Note: Left sensor is connected to PC3 and right sensor to PC0

```
      if(s==0b0001001) // white surface below both sensors
      {
      PORTB=0b00010010; // both motors rotate in forward direction
                        // so robot moves forward
      }

      else // black surface below the sensor
      {
      PORTB=0b00000000; // both motors are OFF
      }

} // while loop closed

} // main function closed
```

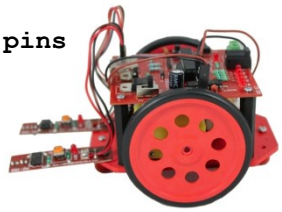

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port.
3. Burn the 'hex' file into your kit.
4. Connect both the IR sensors, to PC3&PC0 in your kit.
5. Now connect battery and switch on the kit.
6. First keep white paper below both the sensors.
7. The robot will move in forward direction.
8. Now keep black surface below the sensors.
9. Both motors will be OFF and the robot stops.
10. Is it working? Nice! You did it.
11. Now don't forget to give your feedback on : http://www.vsagar.org/contact-us

LINE FOLLOWING ROBOTS

Black line following robot (BLFR)

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the two IR sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/
#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8

#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder

      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
      int s=0; // 's' is the variable to store the status value of sensor
               // when we write 'int s', it creates
               // a location in memory of microcontroller.
              // initially '0' is stored into 's' memory location
      while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      s=PINC&0b0001001; // assigning the variable 's' to PC0 of PORTC
                        // so that the output status of sensor will be
                        // stored into the variable 's'
                        // since one sensor in our kit is connected
                        // to PC0 and other to PC3
```



Note: Left sensor is connected to PC3 and right sensor to PC0

```
            if(s==0b0001001) // white surface below both sensors
            {
            PORTB=0b00010010; // both motors rotate in forward direction
                              // so robot moves forward
            }
```

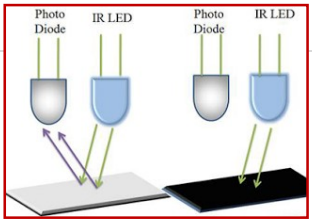
```
        if(s==0b0001000) // white below LS and black below RS
        {
        PORTB=0b00000010; // only right motor rotates forward
                          // so robot turns left
        }

        if(s==0b0000001) // white below RS and black below LS
        {
        PORTB=0b00010000; // only left motor rotates forward
                          // so robot turns right
        }

        else // black surface below both the sensors
        {
        PORTB=0b00000000; // both motors are OFF and robot stops
        }

} // while loop closed

} // main function closed
```


HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port. Burn the 'hex' file into your kit.
3. Connect both the IR sensors, to PC3&PC0 in your kit.
4. Now make a black track of oval shape using black tape. Make this track particularly on smooth surface.
5. Now according to the width of track adjust the distance between two sensors.
6. Keep them apart from each other as required. Place your robot on the track such that the two sensors will be on white. Now connect battery and switch on the kit. Your robot will follow the track.
7. Is it working? Nice! You did it.

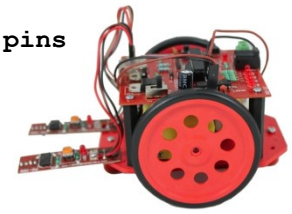
IMPORTANT NOTE

1. If the robot is not following the track correctly, then adjust the sensitivity of the IR sensor by turning adj. screw ANTICLOCKWISE to decrease sensitivity of the sensors.
2. For this, first keep black surface below both sensors. The indicators of the sensors must remain OFF.
3. So by trial and error, adjust the sensitivity and then check it on black track.
4. Remember, when you are done correctly with this adjustment, your robot must follow the track correctly.



White line following robot (WLFR)

```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the two IR sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/
#define F_CPU 12000000UL // defining the crystal frequency 12MHz
                         // given on your dev. board of ATMega8
#include <avr/io.h> // including the input-output
                    // to define the input output ports and pins
                    // this file is inside the AVR folder

      int main() // starting the main function of program
      { // main function brace opened
      DDRB=0b00011110; // PB4-PB1 of PORTB are defined as output pins
      DDRC=0b0000000;  // PC6-PC0 of PORTC are defined as input pins
      int s=0; // 's' is the variable to store the status value of sensor
               // when we write 'int s', it creates
               // a location in memory of microcontroller.
              // initially '0' is stored into 's' memory location

while(1) // starting the infinite loop to repeat the action infinitely
      { // while loop brace opened
      s=PINC&0b0001001; // assigning the variable 's' to PC0 of PORTC
                        // so that the output status of sensor will be
                        // stored into the variable 's'
                        // since one sensor in our kit is connected
                        // to PC0 and other to PC3
```



Note: Left sensor is connected to PC3 and right sensor to PC0

```
            if(s==0b0000000) // black surface below both sensors
            {
            PORTB=0b00010010; // both motors rotate in forward direction
                              // so robot moves forward
            }
            if(s==0b0001000) // white below LS and black below RS
            {
            PORTB=0b00010000; // only left motor rotates forward
                              // so robot turns right
            }
```

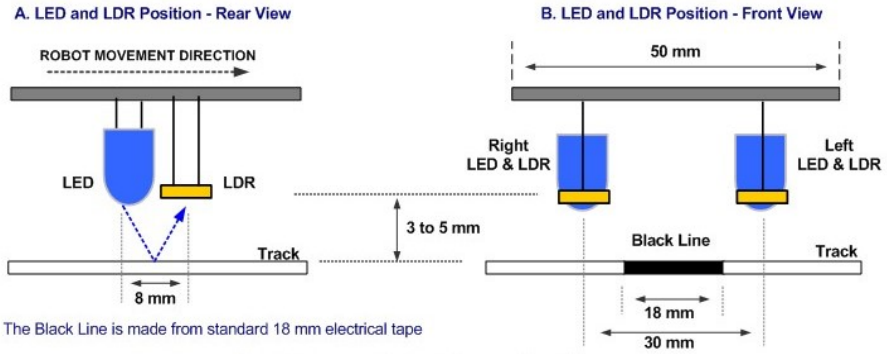
```
        if(s==0b0000001) // white below RS and black below LS
        {
        PORTB=0b00000010; // only right motor rotates forward
                          // so robot turns left
        }
        else // white surface below both the sensors
        {
        PORTB=0b00000000; // both motors are OFF and robot stops
        }
} // while loop closed
} // main function closed
```

HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First read the program carefully. Understand the steps as taught to you.
2. Connect your kit to USB port. Burn the 'hex' file into your kit.
3. Connect both the IR sensors, to PC3 & PC0 in your kit.
4. Now make a black track of oval shape. Make this track particularly on smooth surface.
5. Now according to the width of track adjust the distance between two sensors.
6. Keep them apart from each other as required. Place your robot on the track such that the two sensors will be on white. Now connect battery and switch on the kit. Your robot will follow the track.
7. Is it working? Nice! You did it.

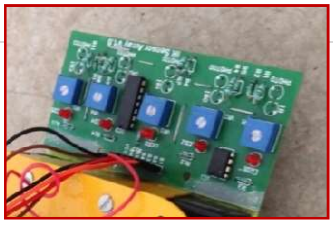
IMPORTANT NOTE

1. If the robot is not following the track correctly, then adjust the sensitivity of the IR sensor by turning adj. screw ANTICLOCKWISE to decrease sensitivity of the sensors.
2. For this, first keep black surface below both sensors. The indicators of the sensors must remain OFF.
3. So by trial and error, adjust the sensitivity and then check it on black track.
4. Remember, when you are done correctly with this adjustment, your robot must follow the track correctly.



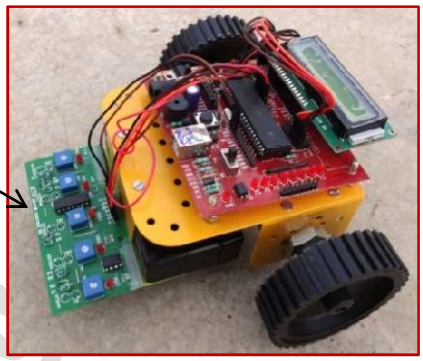
Mechanism of sensors behind the line following robot

Straight track following U-turn robot



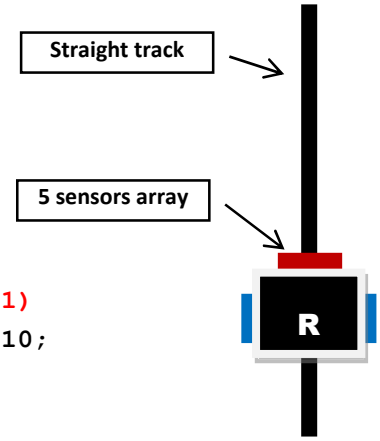
```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the 5 sensors array to PC4 & PC0 in your kit.
         Connect LEFTMOST SENSOR to PC4 and RIGHTMOST SENSOR to PC0
*/
#define F_CPU 12000000UL
#include <avr/io.h>
int main()
  {
      DDRB=0b00011110;
      DDRC=0b0000000;
      int s=0;

      while(1)
        {
      s=PINC&0b0011111; // masking the sensor status
      if(s==0b0000000)
        {
        PORTB=0b00000000; // STOP
        }

      if(s==0b0011011)
        {
        PORTB=0b00010010; // GO FORWARD
        }

      if((s==0b0000111)||(s==0b0001111))
        {
        PORTB=0b00001010; // POWER LEFT
        }

      if((s==0b0010011)||(s==0b0010111))
        {
        PORTB=0b00000010; // SOFT LEFT
        }

      if((s==0b0011100)||(s==0b0011110))
        {
        PORTB=0b00010100; // POWER RIGHT
        }
```

5 sensors array



PC4	PC3	PC2	PC1	PC0	Value	Action
0	0	0	0	0	0	STOP
1	1	0	1	1	27	FORWARD
0	0	1	1	1	7	POWER LEFT
0	1	1	1	1	15	POWER LEFT
1	0	0	1	1	19	SOFT LEFT
1	0	1	1	1	23	SOFT LEFT
1	1	0	0	1	25	SOFT RIGHT
1	1	1	0	0	28	POWER RIGHT
1	1	1	0	1	29	SOFT RIGHT
1	1	1	1	0	30	POWER RIGHT
1	1	1	1	1	31	U-TURN

```
if((s==0b0011001)||(s==0b0011101))
  {
  PORTB=0b00010000; // SOFT RIGHT
  }
      if(s==0b0011111)
      {
            PORTB=0b00010100;
            _delay_ms(300);
                        if(s==0b0011011)
                        PORTB=0b00010010;
                        else
                        continue;
      }
      } // while closed
} // main closed
```

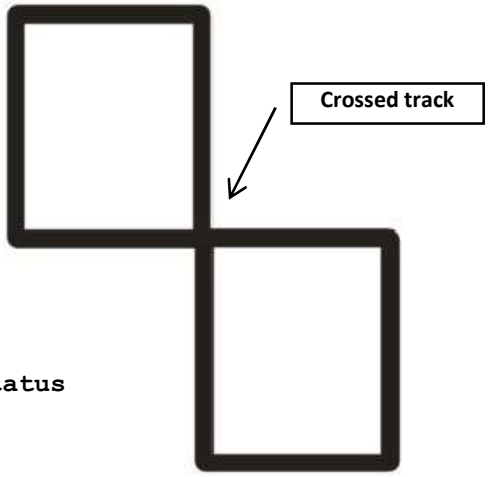


HOW TO USE AND RUN THIS PROGRAM IN YOUR KIT?

1. First you will require the 5 sensor array card, which is readily available at us. You can purchase it by contacting us at: http://www.vsagar.org/contact-us.
2. Now connect the card to **PC4 & PC0** in your kit. Connect LEFTMOST SENSOR to **PC4** and RIGHTMOST SENSOR to **PC0**.
3. Now keep a black paper or sheet below all the sensors. Switch on the robot battery supply and adjust the sensitivity of all the sensors such that their indicator LEDs will just turn OFF.
4. Once this adjustment is done, burn the code given above into your robot microcontroller.
5. Now fix a long black track *(approx. 1m length)* on smooth floor or plywood using black tape.
6. Then keep the robot on the track such that its center IR sensor will be exactly on black track.
7. Switch on the battery supply of the robot. Your robot will start following the track up to the end.
8. When it reaches at the end of the track, all the sensor indicator LEDs will glow and the robot will take U-turn, until its middle sensor is again on black track.
9. Again it will follow the black track to the other end and will take U-turn.
10. In this way, it will continue traversing the black track to-and-fro.
11. *Is it working? Nice!*
12. *Now don't forget to give your valuable feedback to us on our website: http://www.vsagar.org/contact-us.*

Crossed track following robot (using 2 sensors)

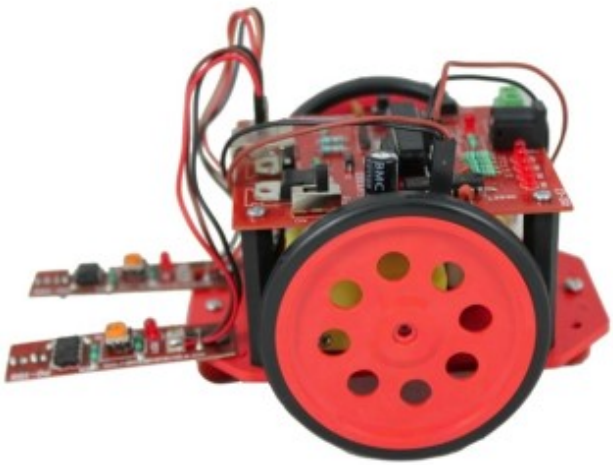
```
/*
      Applicable to ATMega8/16/32/128
      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the 2 sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/

#define F_CPU 12000000UL // defining clock
#include <avr/io.h>
#include <util/delay.h>

      int main()
      {
      DDRB=30; // PB4-PB1 set for motors
      DDRC=0; // PORTC is set as input port
      int s=0; // variable to store sensor status

      while(1) // infinite loop
      {
      s=PINC&0b0001001; // masking PC3 and PC0 to record sensor status

      if(s==0) // both sensors are on black
      {
      PORTB=0; // stop for a bit to show that I am thinking!
      _delay_ms(1000);
      PORTB=18; // move forward - 00010010
      _delay_ms(1000);
      }

      if(s==8)
      {
      PORTB=20; // power right turn - 00010100
      }

      if(s==1)
      {
      PORTB=10; // power left turn - 00001010
      }

      if(s==9)
      {
```



```
PORTB=18; // move forward - 00010010
}

} // while closed

} // main closed
```

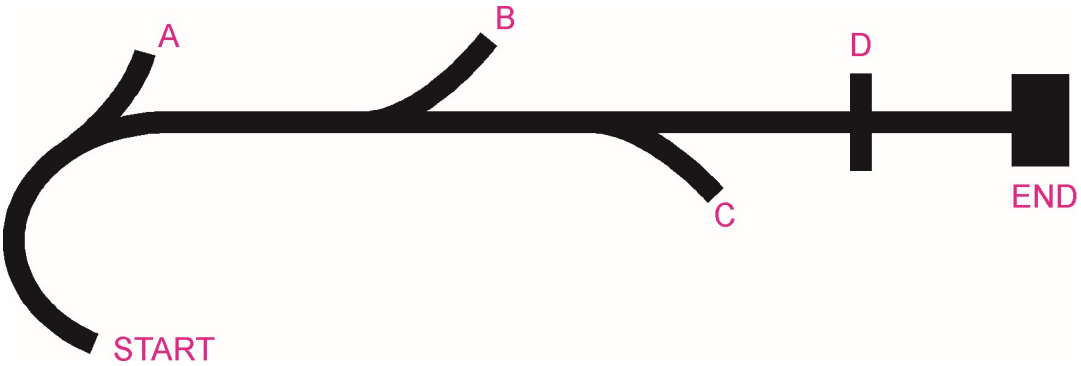
HOW TO USE AND RUN THIS PROGRAM ON YOUR KIT?

1. First read the program carefully and understand the logic used in it to take decision particularly on crossing track point.
2. Then burn your program in your kit and create a crossed track as shown in the above diagram.
3. Switch on your robot and keep it on the track such that the two IR sensors will be apart from each other and will have white surface below them.
4. Your robot will move forward. On the turns it will take the turns properly.
5. When it will arrive at the crossed track, it will stop and will take the necessary decision and then will go straight.
6. In this way, it will follow the track endlessly.
7. *Now I will give you one task here:*
    a. *When your robot arrives at the crossing track, suppose we want to turn it rather than going straight.*
    b. *Then what will you do? What particular modifications are needed in the above program? Can you work out the same…?*



Simple robot used for the above program

Diverted track following robot




```
/*
      Applicable to ATMega8/16/32/128

      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the 2 sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/
#define F_CPU 1200000UL // defining clock frequency for accurate delay
#include <avr/io.h> // includes input/output header file
#include <util/delay.h> // includes delay header file
int main(void)
      {
      DDRB=0b00011110; //PORTB as output Port connected to motors
      DDRC=0b0000000; //PORTC Input port connected to Sensors
      int LS=0, RS=0;
      int counter=1; // a counter is initiated at '1'
            while(1) // infinite loop
            {
     LS=PINC&0b0001000; // masking PC3 to receive left sensor status
     RS=PINC&0b0000001; // masking PC0 to receive right sensor status

      if((LS==0b0001000) & (RS==0b0000001)) // both sensors ON
                  {
                        PORTB=0b00010010; // move forward
                  }

      if((LS==0b0000000)&(RS==0b0000001))
                  {
                        PORTB=0b00010000; // turn right
                  }
```

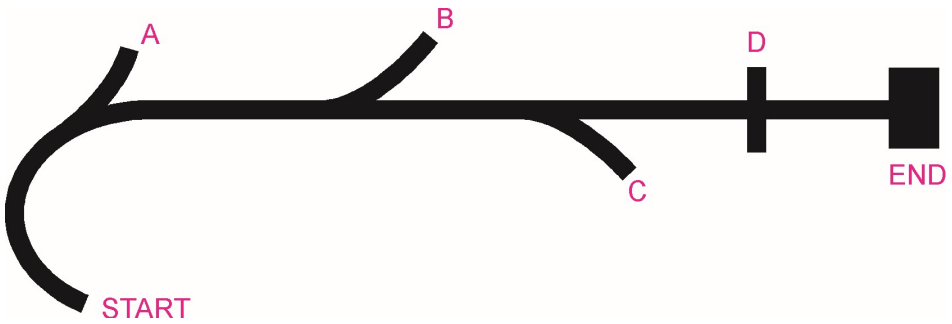
```
if((LS==0b0001000)&(RS==0b0000000))
            {
                  PORTB=0b00000010; // turn left
            }

if((LS==0b0000000)&(RS==0b0000000)&(counter==1)) // both sensors OFF
            {
                  PORTB=0b00000000; // stop
                  _delay_ms(300);
                  PORTB=0b00010000; // turn right
                  _delay_ms(100);
                  counter++; // here value of counter=2
                  LS=172;  // random values stored
                  RS=183;
            }

if((LS==0b0000000)&(RS==0b0000000)&(counter==2)) // both sensors OFF
            {
                  PORTB=0b00000000; // stop
                  _delay_ms(300);
                  PORTB=0b00010000; // turn right
                  _delay_ms(100);
                  counter++; // here value of counter=3
                  LS=182;
                  RS=177; // random values stored

            }

if((LS==0b0000000)&(RS==0b0000000)&(counter==3)) // both sensors OFF
            {
                  PORTB=0b00000000; // stop
                  _delay_ms(300);
                  PORTB=0b00000010; // turn left
                  _delay_ms(100);
                  counter++; // counter=4
                  LS=203;
                  RS=289;
            }
```

continued on next page…

```
if((LS==0b0000000)&(RS==0b0000000)&(counter==4)) // both sensors OFF
            {
                  PORTB=0b00000000; // stop
                  _delay_ms(300);
                  PORTB=0b00010010; // move forward
                  _delay_ms(100);
                  counter++; // counter=5
                  LS=158;
                  RS=196; // random values stored

            }

if((LS==0b0000000)&(LS==0b0000000)&(counter==5)) // both sensors OFF
            {
                  PORTB=0b00000000; // stop
                  break;
            }

      } // while closed

} // main closed
```

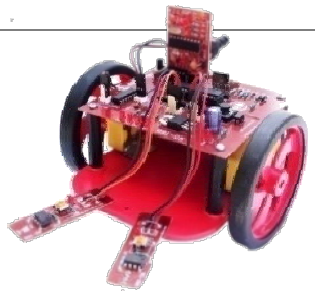

HOW TO USE AND RUN THIS PROGRAM ON YOUR KIT?

1. First read the program carefully and understand the logic used in it to take decision particularly on each diversion of the track at points A, B, C and finally at point D.
2. A counter is initiated in the program to create number of combinations.
3. From starting point the robot traverses the black track using simple BLFR logic.
4. When it comes to diversion 'A', it stops first since its both sensors are on black. Then it takes decision to turn right.
5. When it comes to diversion 'B', it again stops and then turns right in the same way.
6. When it comes to diversion 'C', it turns left. Finally when it reaches crossing 'D', it stops and then takes the decision of going forward for about 100ms.
7. Lastly it stops permanently at the END point, since the **`while`** loop is broken.



Suggested diversion track for the above program

INTELLIGENT APPLICATIONS OF SENSORS



Edge avoiding robot

```
/*
      Applicable to ATMega8/16/32/128

      *** CONNECTION DETAILS OF KIT ***
      1) The 2 motors in your kit, are connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the 2 sensors to PC3 & PC0 in your kit.
         Connect LEFT SENSOR to PC3 and RIGHT SENSOR to PC0
*/

#define F_CPU 1200000UL // defining clock frequency for accurate delay
#include <avr/io.h> // includes input/output header file
#include <util/delay.h> // includes delay header file
int main(void)
      {
      DDRB=0b00011110; //PORTB as output Port connected to motors
      DDRC=0b0000000; //PORTC Input port connected to Sensors
      int S=0;
            while(1) // infinite loop
            {
            S=0b00001001; // masking the status of both sensors

                  if(S==9) // both sensors are on white
                  {
                  PORTB=18; // move forward
                  }

                  if(S==0) // both sensors outside the edge
                  {
                  PORTB=0; // stop
                  _delay_ms(1000);
                  PORTB=12; // move backward
                  _delay_ms(500); // adjust it as per requirement
                  PORTB=16; // turn right
                  _delay_ms(300); // adjust as required
                  S=9;
                  }
```

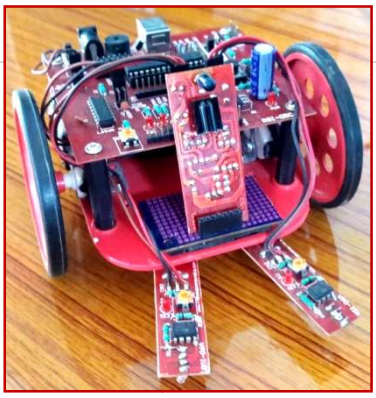
```
        if(S==8) // LS on white, RS is outside the edge
        {
        PORTB=0; // stop
        _delay_ms(1000);
        PORTB=12; // move backward
        _delay_ms(500); // adjust it as per requirement
        PORTB=16; // turn right
        _delay_ms(300); // adjust as required
        S=9;
        }

        if(S==1) // RS on white, LS is outside the edge
        {
        PORTB=0; // stop
        _delay_ms(1000);
        PORTB=12; // move backward
        _delay_ms(500); // adjust it as per requirement
        PORTB=2; // turn left **** NOTE THIS STEP ****
        _delay_ms(300); // adjust as required
        S=9;
        }
} // while closed
} // main closed
```

HOW TO USE AND RUN THIS PROGRAM ON YOUR KIT?

1. You can test the working of this program on a smooth surface of a table. The table-top should be off-white or white, but should not be black.
2. First read the program carefully and understand the logic used in it to take decision particularly when it reaches at the EDGE of table.
3. When its both sensors are on white it moves in forward direction.
4. When it reaches at the edge of table, its both sensors are out-of-the-table surface and they sense it AS BLACK SURFACE, due to depth at table edge.
5. So it stops first and then moves backward and then turns in a particular direction. After that it senses that its both sensors are on white, so moves forward.
6. When any one of the two sensors are out of the edge of table same action takes place, but the direction of its turn is different as given in the program.
7. *So? Did it work?*
8. *If its working properly, do not forget to give us feedback on our website: http://www.vsagar.org/contact-us*

Note: *You can use the same program as simple obstacle avoiding robot on plane surface with number of black strips pasted on the surface within the path of robot.*

Edge avoider + obstacle avoider robot



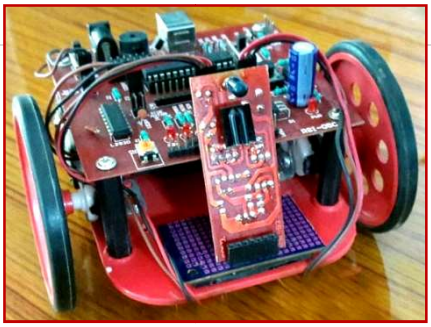
```
/*
      Applicable to ATMega8/16/32/128

      *** CONNECTION DETAILS OF KIT ***
      1) Motors connected to PB4-PB1, as follows:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect the 3 sensors to PC3, PC1 & PC0 in your kit.
         See the note given at the end of this program.
*/

#define F_CPU 1200000UL // defining clock frequency for accurate delay
#include <avr/io.h> // includes input/output header file
#include <util/delay.h> // includes delay header file
int main(void)
      {
      DDRB=0b00011110; //PORTB as output Port connected to motors
      DDRC=0b0000000; //PORTC Input port connected to Sensors
      int S=0;
            while(1) // infinite loop
            {
            S=0b00001011; // masking the status of both sensors

                  if(S==9) // LS & RS are on white
                           // and MS is not sensing any obstacle i.e MS=0
                  {
                  PORTB=18; // move forward
                  }

                  { // obstacle avoiding logic

                  if(S==11) // LS & RS are on white
                                // but MS is sensing obstacle
                                // on table-top i.e. MS=1
                  {
                  PORTB=0; // stop
                  _delay_ms(1000);
                  PORTB=12; // move backward
                  _delay_ms(500); // adjust it as per requirement
                  PORTB=16; // turn right
                  _delay_ms(300); // adjust as required
                  S=9;
                  }
                  } // end of obstacle avoiding logic
```

```
                { // edge avoiding logic
                if(S==0) // both sensors outside the edge
                {
                PORTB=0; // stop
                _delay_ms(1000);
                PORTB=12; // move backward
                _delay_ms(500); // adjust it as per requirement
                PORTB=16; // turn right
                _delay_ms(300); // adjust as required
                S=9;
                }

                if(S==8) // LS on white, RS is outside the edge
                {
                PORTB=0; // stop
                _delay_ms(1000);
                PORTB=12; // move backward
                _delay_ms(500); // adjust it as per requirement
                PORTB=16; // turn right
                _delay_ms(300); // adjust as required
                S=9;
                }

                if(S==1) // RS on white, LS is outside the edge
                {
                PORTB=0; // stop
                _delay_ms(1000);
                PORTB=12; // move backward
                _delay_ms(500); // adjust it as per requirement
                PORTB=2; // turn left **** NOTE THIS STEP ****
                _delay_ms(300); // adjust as required
                S=9;
                }
                } // end of edge avoiding logic
        } // while closed
} // main closed
```


HOW TO USE AND RUN THIS PROGRAM ON YOUR KIT?

1. You can test the working of this program on a smooth surface of a table. The table-top should be off-white or white, but should not be black.

Note: *Connect LEFT SENSOR to PC3, MIDDLE SENSOR to PC1 and RIGHT SENSOR to PC0 of PORTC pins. Also fix middle sensor vertically and the other two horizontally as shown in the image above. So middle sensor will sense the obstacles in between the path and LS & RS will detect the edges of the table-top.*



Wall following robot

```
/*
      Applicable to ATMega8/16/32/128

      *** CONNECTION DETAILS OF KIT ***
      1) Motors connections to PB4-PB1:
         Left motor:  PB4 -> (+) and PB3 -> (-)
         Right motor: PB1 -> (+) and PB1 -> (-)
      2) Connect only one sensor (WS) to PC0 in your kit.
         Fix the sensor vertically to detect the presence of wall.
*/
#define F_CPU 1200000UL // defining clock frequency for accurate delay
#include <avr/io.h> // includes input/output header file
#include <util/delay.h> // includes delay header file
int main()
      {
      DDRB=0b00011110; //PORTB as output port connected to motors
      DDRC=0b0000000; //PORTC input port connected to Sensors
      int WS=0;

            while(1) // infinite loop
            {
            WS=0b00000001; // masking the status of wall sensor

                  if(WS==0) // there is no wall in front of sensor
                  {
                  PORTB=18; // move forward
                  }

                  if(WS==1)
                  {
                  PORTB=20; // power right turn
                  _delay_ms(300);
                        if(WS==0)
                        {
                        PORTB=18;
                        }
                        else
                        continue;
                  }

            } // while closed
      } // main closed
```