Obstacle Avoiding Robot

Introduction

An *Obstacle Avoiding Robot* may be defined as a robot which can avoid any unwanted obstacle in its path and is capable of changing its path.

The basic tasks of an *Obstacle Avoiding Robot* can be divided into 3 sections, namely

- Navigation
- Processing
- Execution

The robot must be equipped with some means by which it can navigate through its surrounding to check if there is any obstacle or not. In this project we have used IR Sensors for the purpose of navigation .

After navigating the surrounding the robot must be capable of processing the input data from the navigation section. In this project the processing section is done with an EAB, which comes along with microcontroller PIC18F26K22.

While processing the robot takes decision according to the algorithm designed for it.

After processing the navigated data the robot must do some work (e.g. Movement etc). In this project we have used motor driver IC L293D to drive the motors/wheels according to the processed output from the EAB.

IR sensor

IR Sensor is composed of an IR emitter and an IR Receiver. When the IR Sensor is powered, the IR emitter emits infrared signal continuously and the IR Receiver is for receiving that infrared signal when it is echoed back after striking an obstacle. So, whenever the IR Receiver receives the infrared signal, the LED associated with it glows i.e. ON state and if the IR receiver does not receive any signal, the LED is in OFF state. So, this change in LED state can be measured with the EAB.

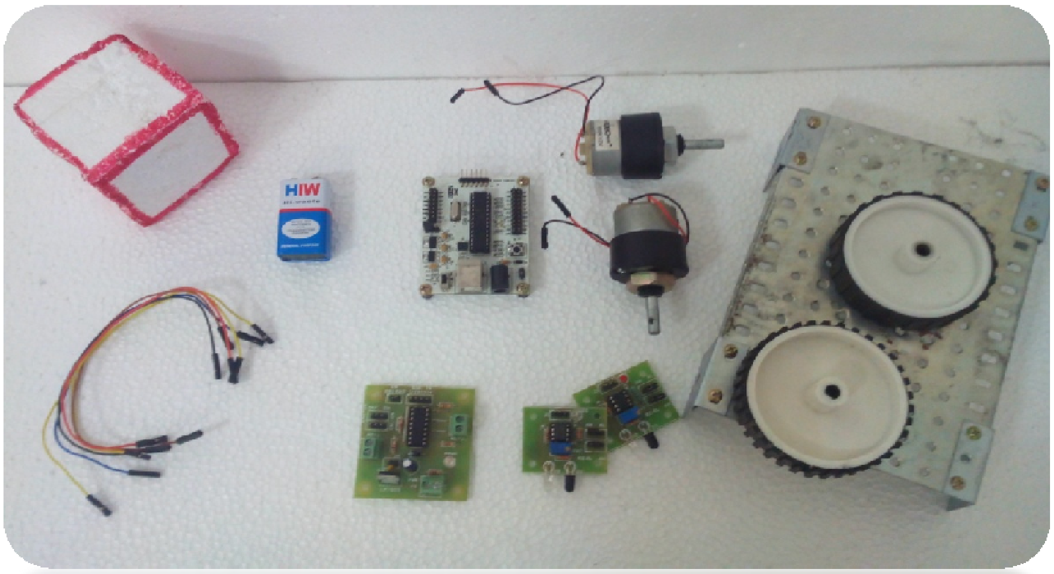
Motor Driver Board

The motor driver board consists of ICL293D. It has 4 inputs and 4 outputs. Since, the current from the EAB is not sufficient enough to drive the motors, so we have to use this Driver to increase the current by which we can drive the DC Motors.

Components

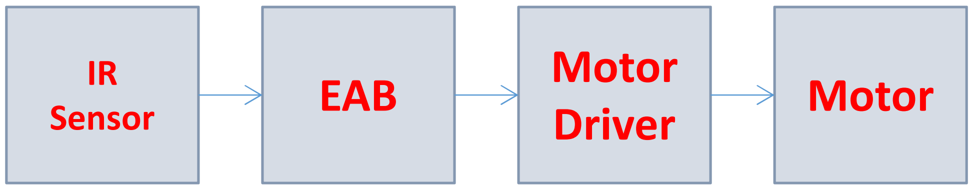
The Components required for building the Obstacle Avoiding Robot are:

- Embedded Application Board
- Motor Driver Board (containing L293D)
- IR Sensor (2 nos)
- DC Motors (2 nos)
- Chassis
- Connectors(Jumpers)
- 9V battery
- Wheels
- Obstacles



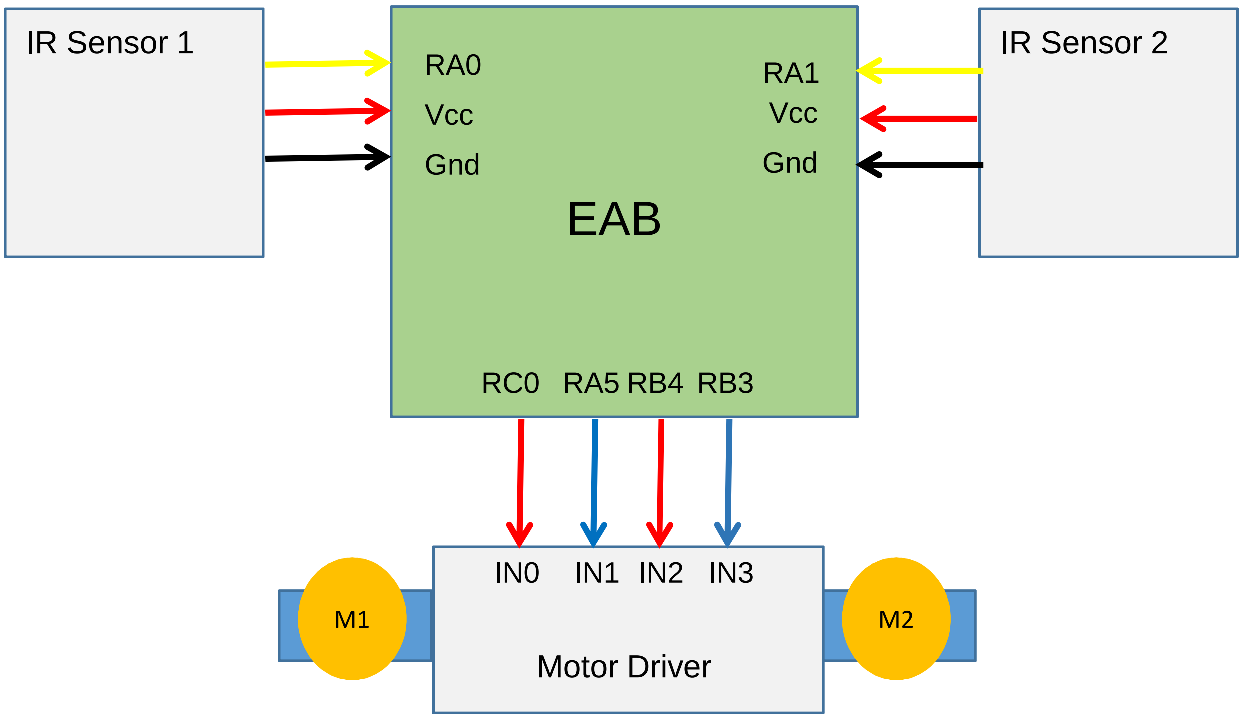
Block Diagram

Block level representation of the different blocks of the Obstacle Avoiding Robot.



Schematic Diagram

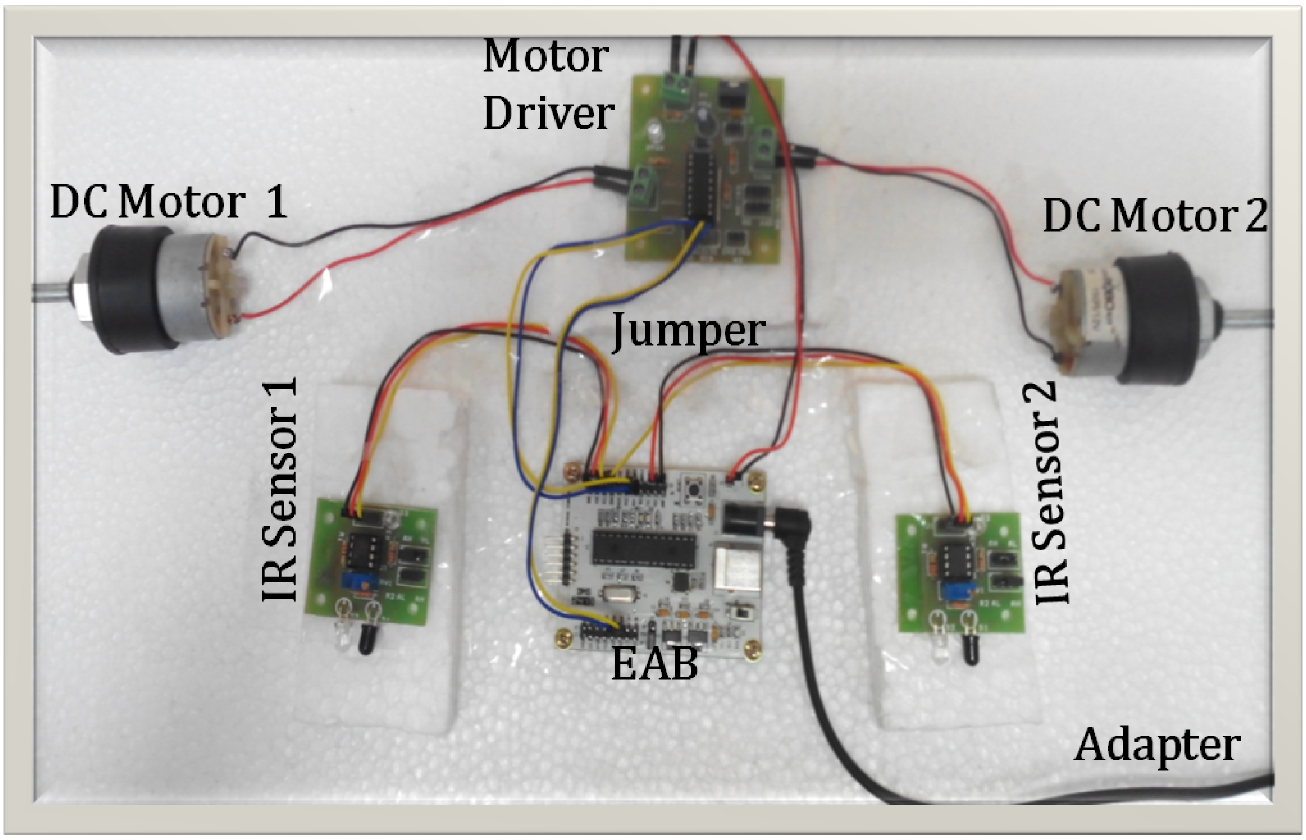
The Schematic diagram illustrates the circuit connections for designing the application.



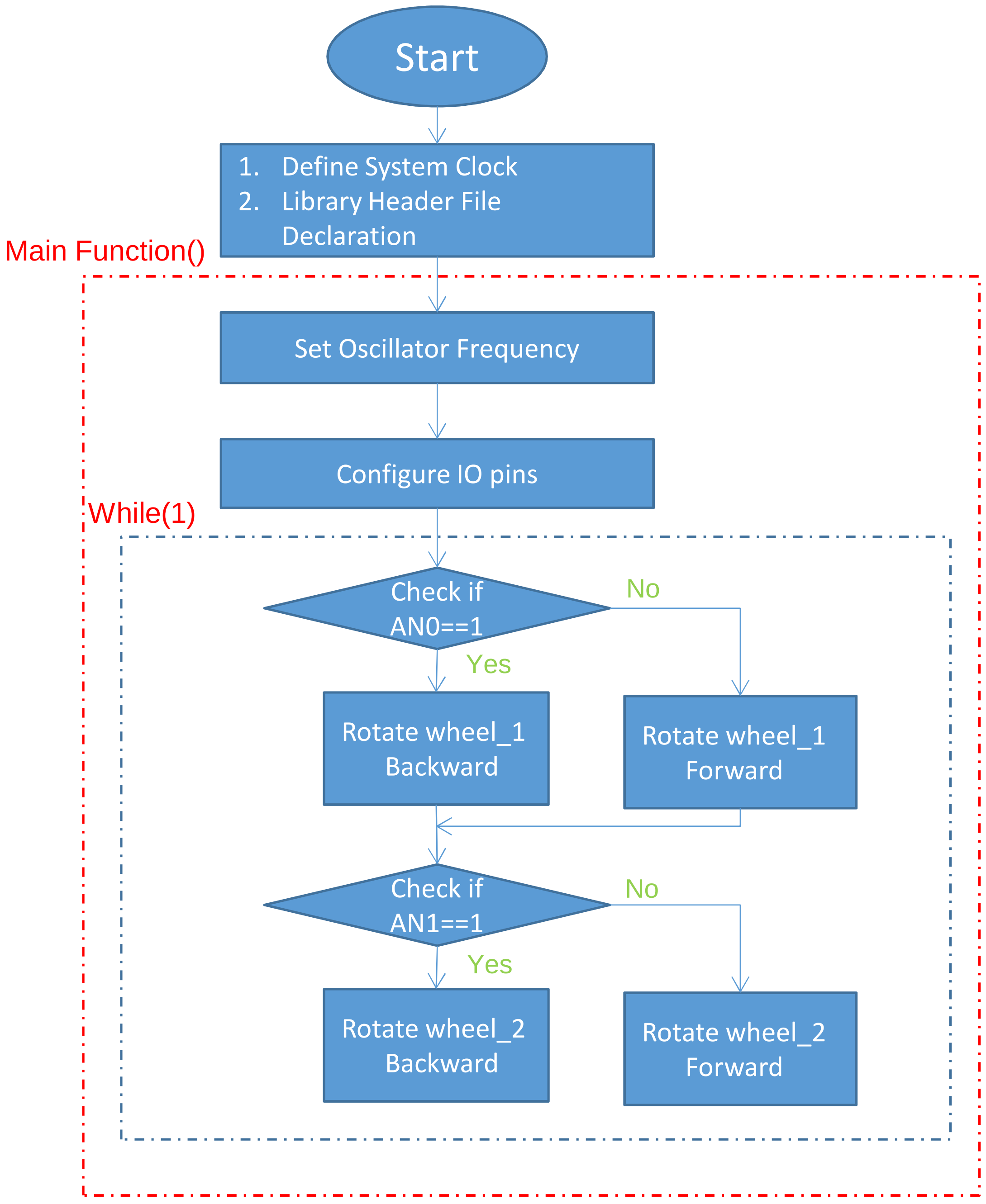
Connection Description

In this project we have used two IR Sensor modules as obstacle detector. One is placed in Left-Front corner and another is placed in Right-Front corner of the robot. Each module has 3-pins. These are GND, VCC and DATA pins. The GND and VCC pins are connected to the GND and VCC pins of an EAB. The DATA pin of IR Sensor1 is connected to RA0 pin of EAB while that of IR Sensor2 is connected to RA1 pin of EAB.

The out put of the EAB's are from the pins RC0, RA5, RB4 and RB3. These pins are responsible to control the robot and they are connected to the IN0, IN1, IN2 and IN3 pins of the Motor Driver board, respectively. Both the Motor Driver Board and the EABs are powered with 12V rechargeable battery.



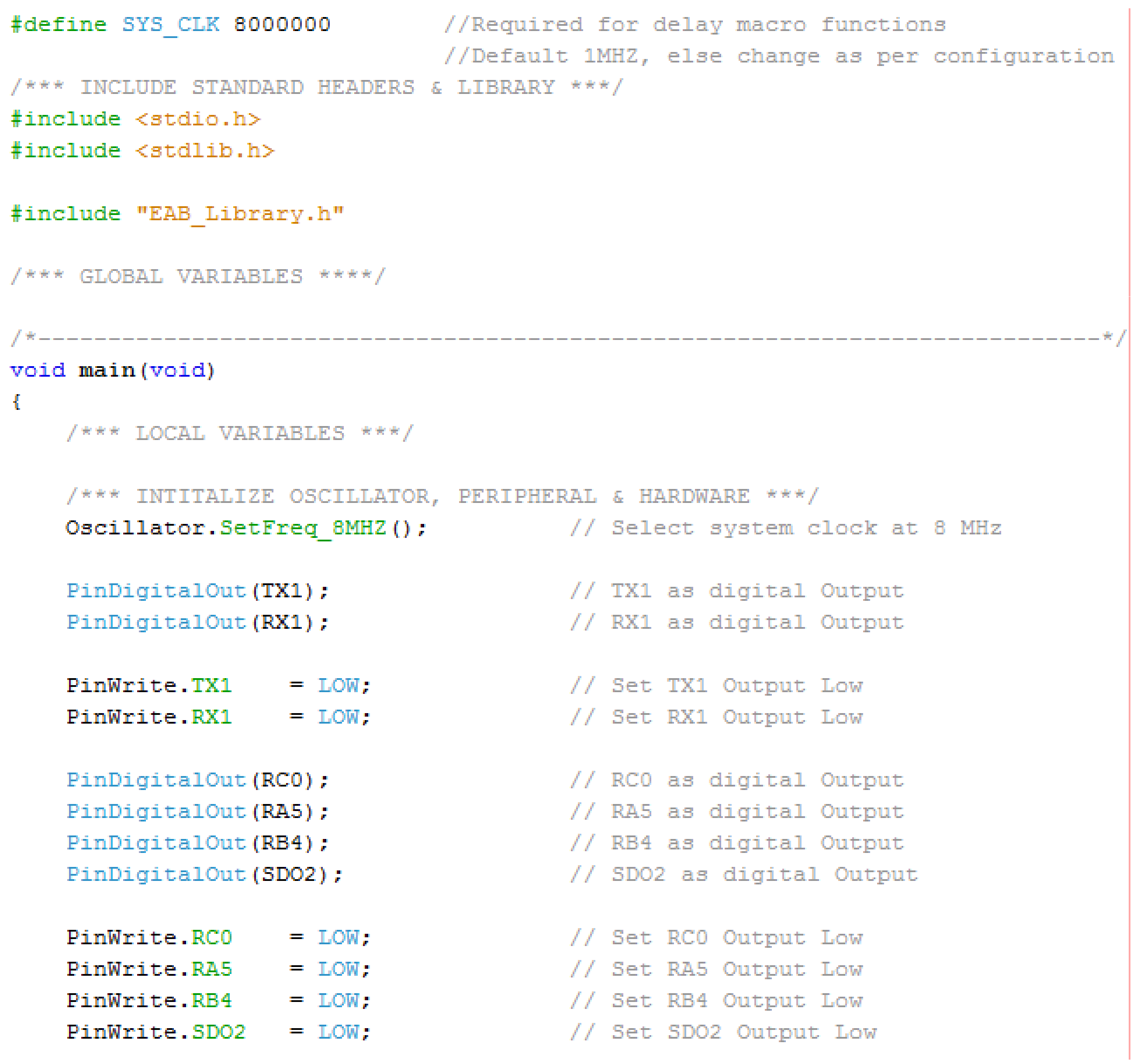
Code Flow Chart



Source Code

The Source code shown below is the firmware to be flashed in the microcontroller of the Embedded Application Board. The Source code is commented for better understanding of the user.

Refer to the EAB User Guide and EAB Programming Guide for more details on how to Flash(burn) program(Source Code) in the microcontroller of Embedded Application Board.



```
#define SYS_CLK 8000000        //Required for delay macro functions
                               //Default 1MHZ, else change as per configuration
/*** INCLUDE STANDARD HEADERS & LIBRARY ***/
#include <stdio.h>
#include <stdlib.h>

#include "EAB_Library.h"

/*** GLOBAL VARIABLES ****/

/*---------------------------------------------------------------------------*/
void main(void)
{
    /*** LOCAL VARIABLES ***/

    /*** INTITALIZE OSCILLATOR, PERIPHERAL & HARDWARE ***/
    Oscillator.SetFreq_8MHZ();           // Select system clock at 8 MHz

    PinDigitalOut(TX1);                  // TX1 as digital Output
    PinDigitalOut(RX1);                  // RX1 as digital Output

    PinWrite.TX1     = LOW;              // Set TX1 Output Low
    PinWrite.RX1     = LOW;              // Set RX1 Output Low

    PinDigitalOut(RC0);                  // RC0 as digital Output
    PinDigitalOut(RA5);                  // RA5 as digital Output
    PinDigitalOut(RB4);                  // RB4 as digital Output
    PinDigitalOut(SDO2);                 // SDO2 as digital Output

    PinWrite.RC0     = LOW;              // Set RC0 Output Low
    PinWrite.RA5     = LOW;              // Set RA5 Output Low
    PinWrite.RB4     = LOW;              // Set RB4 Output Low
    PinWrite.SDO2    = LOW;              // Set SDO2 Output Low
    PinDigitalIn(AN0);                        // AN0 as digital Input
    PinDigitalIn(AN1);                        // AN1 as digital Input

    PinWrite.AN0     = LOW;                   //Set AN0 Output Low
    PinWrite.AN1     = LOW;                   //Set AN1 Output Low


    /*** PLACE THE REPETITIVE TASKS IN THIS LOOP ***/
    while(1)
    {
         if(PinRead.AN0 == HIGH)              // Check Input State
        {
            PinWrite.RC0     = LOW;           // Set AN0 Output Low
            PinWrite.RA5     = HIGH;          // Set RA5 Output High
            PinWrite.TX1     = LOW;           // Set TX1 Output Low
        }
        if(PinRead.AN0 == LOW)                // Check Input State
        {
            PinWrite.RC0     = HIGH;          // Set RC0 Output high
            PinWrite.RA5     = LOW;           // Set RA5 Output Low
            PinWrite.TX1     = HIGH;          // Set TX1 Output high
        }
        if(PinRead.AN1 == HIGH)               // Check Input State
        {
            PinWrite.RB4     = LOW;           // Set RB4 Output Low
            PinWrite.SDO2    = HIGH;          // Set SDO2 Output High
            PinWrite.RX1     = LOW;           // Set RX1 Output Low
        }
        if(PinRead.AN1 == LOW)                // Check Input State
        {
            PinWrite.RB4     = HIGH;           //Set RB4 Output High
            PinWrite.SDO2    = LOW;           // Set SDO2 Output Low
            PinWrite.RX1     = HIGH;          // Set RX1 Output High
        }
    }
}
/*----------------------------------------------------------------------------*/
```

Application Notes

How to Operate

Follow the steps mentioned below in order to operate the project…

- Prepare an arena as you wish.
- Flash the code into microcontroller.
- Connect each and every part properly.
- Place the robot in a the arena.
- Power the EAB, Sensor Board and Motor Driver Circuit with 9V/12V DC. Carefully check the polarities and then connect .
- Switch ON the EAB.

Now you can see the Robot moving automatically, in the arena, without colliding any obstacles.

More Projects

Various other applications can be built using Obstacle Avoiding Robot.

Some of such applications are given below:

- Anti-collision system
- Alarm systems
- Animation
- Industrial Automation.