PROJECT REPORT

Submitted By

Roshan G – MDM14B028

D Sasi Kumar – MFD14I002

Akash Yadav – MDM14B010

Gautham Manoj – MFD14I003

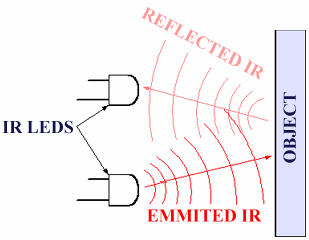
Introduction:

IR Proximity sensor Construction:

There are 2 parts in an IR Proximity sensor, transmitter and receiver. Transmitter(Forward Bias) will emit IR radiation up to 20 cms as potential difference across the terminals. Receiver (Reverse Bias) is a photodetector, which detects the emitted IR radiation and generates a corresponding potential difference across the terminals.

Working of IR proximity sensor

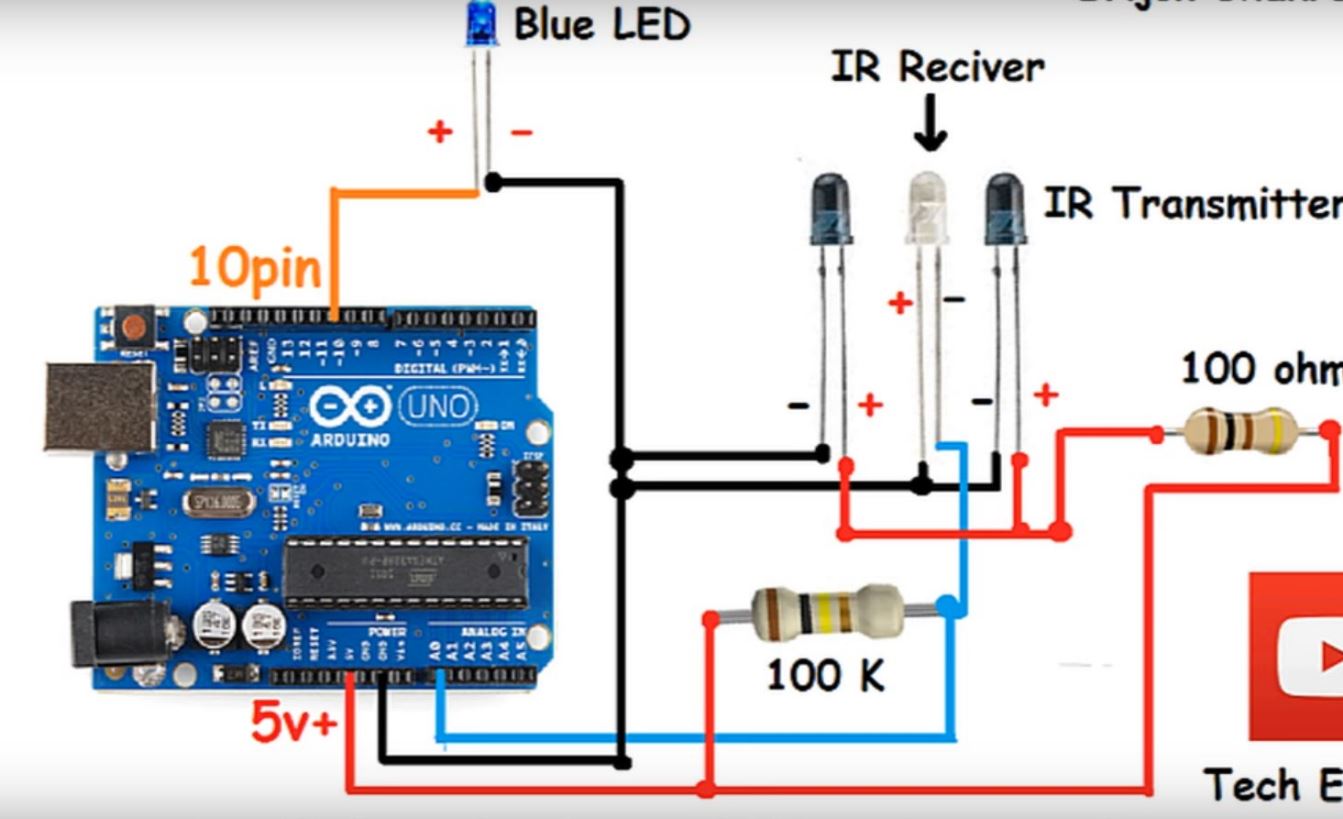
An IR proximity sensor works by applying a voltage difference to 4 IR light emitting diodes (LED’s) which in turn, emit infrared light. This light propagates through the air and once it hits an object it is reflected back towards the other LED (Receiver). The electrical property of LED is used here which is the fact that LED produce a potential difference across its leads when exposed to light, similar to a photocell but much lower output current. If the object is close, the reflected light will be stronger than if the object is further away. The sensing unit used here is Arduino UNO which takes output from receiver LED’s and detects the reflected infrared light.



Apparatus used

* IR LED+ Photodetector pair (4 Nos).
* Arduino UNO.
* 8-bit LCD display.
* Resistors (100k,370 ohms)
* Breadboard (2 Nos).
* Jumper wires.
* Potentiometer (BOURNS 3296).

CIRCUIT



In the actual circuit, there are 4 IR led pairs. With that we are finding the obstacle in its range for each pair. At a time only one ir led pair will work, which has obstacle above within its range (20 cms).

With the help of this sensor outputs, we can determine which pattern is drawn above those IR LED’s.

The 4 IR LED’s acts like nodes to the draw pattern, so with these 4 nodes we can detect the following patterns.

1. Point
2. Line
3. Triangle
4. Quadrilateral

CONDITION: The starting and ending point of the pattern should be same.

These patterns are recognized based on number of inputs to IR LED’s. The conditions for displaying above patterns are:

|  |  |
| --- | --- |
| Nof\_det[4] array | DISPLAYING PATTERN |
| 2,0,0,0 | Point |
| 2,1,0,0 | Line |
| 2,1,1,0 | Triangle |
| 2,1,1,1 | Quadrilateral |

At a time the obstacle should be recognized by only one IR LED pair. For not to detect two inputs at a time, an insulating tape is wrapped around the photodetector (IR receiver). We are using a variable in the code for counting the no. of inputs to the IR LED.

We also added a led for each IR LED pair to indicate whether the obstacle present or not. i.e., whenever there is an obstacle in the range of IR LED, the indicator LED will blink once.

The pattern which is drawn is shown in a 16 x 2 LCD display in words.

Algorithm:

1. Create arrays for sensor outputs(sen\_out[4]) and the number of times a sensor detects the object(nof\_det[4]).
2. Collect the sensor output voltages and store them in their corresponding Sensor outputs array.
3. Define a minvalue for the sensor ouput, and increment the sensor’s corresponding value by 1 in the nof\_det array, whenever the sen\_out value is less than minvalue.
4. Declare a delay of 1000 micro seconds, so that an object is not detected multiple times.
5. Repeat the above steps until any element of the nof\_det array has a value more than one.
6. Use bubble sort method to sort the nof\_det array in decreasing order.
7. Find out the position appearance of the first zero in the sorted array.
8. If no zero is present, then the pattern is a quadrilateral. If first zero is at the last position, then the pattern is a triangle.
9. If first zero is at the third position, then the pattern is a line. Or If first zero is at the second position, then no pattern is detected only a Point is detected.
10. Display the detected pattern. And repeat the process for another round of pattern detection.

APPLICATIONS:

1. To control a system by drawing patterns.
2. For non-contact operations, helps to detect movement even at a distance.

FURTHER ENHANCEMENTS:

1. Increasing the no of IR led+ Photodetector pairs, will help in detection of complex and large no. of patterns.
2. Placing the nodes in such a manner to avoid unwanted detection of objects, while drawing a particular pattern.

CODE:

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int peakValue = 200;

int sensorValue[4]={0,0,0,0};

int a[4]={0,0,0,0}; //set your own value based on your sensors

int e[4];

int d=0;

int n=4;

int swap;

int j=0;

void printarray(int b[],int len) // helps in printing array

{

for(int i=0;i<len;i++)

{

Serial.print(a[i]);

Serial.print("\t");

}

Serial.println("");

}

int exitconditioncheck(int b[], int len) // to get the exit conditon

{

int c=0;

for(int i=0;i<len;i++)

{

if(a[i]==2)

{

c++;

break;

}

}

return(c);

}

void setup() {

Serial.begin(9600);

pinMode(13, OUTPUT);

pinMode(13, OUTPUT);

pinMode(13, OUTPUT);

pinMode(13, OUTPUT);

lcd.begin(16, 2);

// Print a message to the LCD.

lcd.print("pattern recognition");

}

void loop() {

digitalWrite(13,HIGH);

digitalWrite(13,HIGH);

digitalWrite(13,HIGH);

digitalWrite(13,HIGH);

Serial.println("selectpoint");

lcd.setCursor(0, 0);

// Print a message to the LCD.

lcd.print("pattern recognition");

lcd.setCursor(0, 1);

lcd.print("Select a point");

for(;;)

{

sensorValue[0] = analogRead(A0); // to read the values of sensor

//Serial.println(sensorValue[0]);

//Serial.println(sensorValue[0]);

sensorValue[1] = analogRead(A2);

// Serial.println(sensorValue[1]);

sensorValue[2] = analogRead(A3);

//Serial.println(sensorValue[2]);

sensorValue[3] = analogRead(A4);

//Serial.println(sensorValue[3]);

if (sensorValue[0] < peakValue) // check if it's higher than the current peak:

{a[0]=a[0]+1;

printarray(a,4);

lcd.setCursor(0, 1);

lcd.print("detected!!!!!!");

digitalWrite(13, LOW);

delay(500);

/\* lcd.setCursor(0, 1);

lcd.print("Selectpoint");\*/

delay(1000);

digitalWrite(13,HIGH);

break;

}

if (sensorValue[1] < peakValue) {

a[1]=a[1]+1;

printarray(a,4);

lcd.setCursor(0, 1);

lcd.print("detected!!!!!!");

digitalWrite(13, LOW);

delay(500);

/\* lcd.setCursor(0, 1);

lcd.print("Selectpoint");

delay(1000); \*/

digitalWrite(13,HIGH);

break;

}

if (sensorValue[2] < peakValue) {

a[2]=a[2]+1;

printarray(a,4);

lcd.setCursor(0, 1);

lcd.print("detected!!!!!!");

digitalWrite(13, LOW);

delay(500);

/\* lcd.setCursor(0, 1);

lcd.print("Selectpoint");\*/

delay(1000);

digitalWrite(13,HIGH);

break;

}

if (sensorValue[3] < peakValue) {

a[3]=a[3]+1;

printarray(a,4);

lcd.setCursor(0, 1);

lcd.print("detected!!!!!!");

digitalWrite(13, LOW);

delay(500);

/\* lcd.setCursor(0, 1);

lcd.print("Selectpoint");\*/

delay(1000);

digitalWrite(13,HIGH);

break;

}

}

d=exitconditioncheck(a,4); // check for exit condition

if(d==1) // after exit condition has been satisfied

{

for(int i=0;i<4;i++)

{

e[i]=a[i]; // creating a new array.

}

for (int m = 0 ; m < ( n - 1 ); m++) // sorting the new array

{

for (int k = 0 ; k < n - m - 1; k++)

{

if (e[k] < e[k+1]) /\* For decreasing order use < \*/

{

swap=e[k];

e[k]=e[k+1];

e[k+1]=swap;

}

}

}

printarray(e,4);

if(e[1]==0) // pattern recognition

{Serial.println("Point");

lcd.setCursor(0, 0);

lcd.print("ptern recognised");

lcd.setCursor(0, 1);

lcd.print("point!!!!");

delay(2000);

j++;}

else if(e[2]==0)

{Serial.println("lin");

lcd.setCursor(0, 0);

lcd.print("ptern rgnsed");

lcd.setCursor(0, 1);

lcd.print("line!!!!!");

delay(2000);

j++;}

else if(e[3]==0)

{Serial.println("Tri");

lcd.setCursor(0, 0);

lcd.print("ptern recognised");

lcd.setCursor(0, 1);

lcd.print("Triangle!");

delay(2000);

j++;}

else if(e[4]!=0)

{Serial.println("rectangle");

lcd.setCursor(0, 0);

lcd.print("ptern recognised");

lcd.setCursor(0, 1);

lcd.print("rectangle");

delay(2000);

j++;}

if(j>0) // to deinitialize the values so that new pattern can be created

{

for(int i=0;i<4;i++)

{

a[i]=0;

}

j=0;}

d=0;

}

}