Simple Infrared Obstacle Sensor

Chong Zan Kai  
ID: 1011101802  
Delta year of Computer Engineering  
Multimedia University, Cyberjaya.

Abstract

Infrared sensor is always one of the preferred choices in designing a low cost mobile robot. A simple infrared obstacle sensor can be easily built within a few hours by using a few components and a microcontroller. The microcontroller is used to generate infrared beam and process the signal from the infrared module. In this project the microcontroller being used is AT89C2051. However, it is presumed that other similar microcontroller should work fine as well.

Circuit Design and Algorithm

Infrared detector is one of the common sensors used for sensing the environment’s condition. However, the ambient light surrounding will introduce a lot of noises to the infrared detector. Hence, the infrared beam that used to detect the obstacle has to be modulated in certain frequency so that the BFP can eliminate the unwanted noises.

The infrared module that used in this circuit is IRM-8751, which is a generic IR module with BPF of 38kHz. Two IR emitters are put on the left and right of the IR module. Since the microcontroller has no sufficient current to drive a load like infrared emitter, transistors are used to provide a large current to drive the infrared emitters for a longer distance of detection. Besides that, it is always not a good idea to drive a load directly by using microcontroller.

Although the infrared beam has been modulated at a certain frequency, there are chances that the infrared module will be fault triggered by the ambient light’s noises. Hence, the microcontroller should take its responsibility to recognize the correct trigger. Since the IR beam should on and off at 833Hz, we can utilize the signal pattern for filtering process. The infrared module output is read into “valOn” just right after flashing the infrared beam for 600us. Then, after it idle for 600us, the infrared module will be read again to “valOff”. If “valOn” and “valOff” both giving the triggered value, then a conclusion can be made that it probably a noise trigger.
The scanning algorithm:

*Scan() function:*
1. Flash the left infrared emitter for 600us and turn off.
2. Read the IR modulator to valOn.
3. Sleep for another 600us.
4. Read the value on valOff.
5. If the valOn same as valOff, then that is the fault reading and return 0. Else, it is assume to be correct reading and return 1.
6. Go back to step 1 and flash the right infrared emitter this time.

Since both IR emitters will flash the IR beam at different time, the microcontroller will able to determine the location of the obstacle based on the received signal.

The IR carrier has to be in 38kHz. So the time period for the carrier is 26us, which it should on and off for 13us each time. The microcontroller will approximately use 1us to perform 1 cycle of operation. In other words, it should generate the IR pulse for every 13 cycles with the pulse width of 13 cycles. And for the same thing, the microcontroller will idle for 600us by executing 600 times of “NOP” (1 cycle) command.

Some functional pins are developed to utilize the capability of microcontroller:

<table>
<thead>
<tr>
<th>EN_LED (P1.7)</th>
<th>To enable sensor to run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRONT_LED (P1.6)</td>
<td>An obstacle is right in front of sensor.</td>
</tr>
<tr>
<td>LEFT_LED (P1.5)</td>
<td>An obstacle is on the left site of sensor.</td>
</tr>
<tr>
<td>RIGHT_LED (P1.4)</td>
<td>An obstacle is on the right site of sensor.</td>
</tr>
<tr>
<td>OBSATCLE_LED (P1.3)</td>
<td>An obstacle is detected somewhere.</td>
</tr>
</tbody>
</table>
The infrared obstacle detector circuit:

![Circuit Diagram]

The basic connection for the at89c2051 is not shown above.

**Part lists:**

Resistor- 2.2k *2  
Potentiometer- 2k*2  
2N2222 BJT transistor *2  
10mm IR emitter *2  
AT89c2051 microcontroller *1  
IRM-8751 infrared module *1

Microcontroller connection:  
Crystal 11.0592Mhz *1  
30pF * 2  
Resistor 10k*1  
Capacitor 10uF*1
**Limitation**

The range of detection will be shorted if there is a sunlight fall on the infrared module. Hence, it is advisable to cover the sensor in a box.

Since this is a simple circuit, it cannot measure the range of the obstacle detected. It can only tell if there is an obstacle in front. In fact, there is infrared range sensor in the market, GP1U52X by sharp, which can actually measure the range of the detected obstacle. However the price may be expensive for a beginner to afford.

**Reference**

*Mobile Robots Inspiration to Implementation* by Joseph L. Jones, Anita M. Flynn and Bruce A. Seiger.
Appendix

/*---------------------------------------------------------------
RedEyes version 3 for AT89x51
IR beam is 38khz

This version of sensor can determine the detected object is either on
left, right or front site of the sensor.

-EN_LED(low active) pin must be grounded to activate the sensor
to run.
---------------------------------------------------------------*/
#include <AT89X051.h>
#define DETECTOR P3_0
#define LEFT_IR P3_1
#define RIGHT_IR P3_2
#define EN_LED P1_7
#define FRONT_LED P1_6
#define LEFT_LED P1_5
#define RIGHT_LED P1_4
#define OBSTACLE_LED P1_3

void scan();
void flashLeft();
void flashRight();
void sleep();

void main(void)
{
    // int i, result;
    DETECTOR=1;
    LEFT_IR=0;
    RIGHT_IR=0;
    EN_LED=1;
    FRONT_LED=0;
    LEFT_LED=0;
    RIGHT_LED=0;
    OBSTACLE_LED=0;
    while(1)
    {
        while(EN_LED==0)
        {
            scan();
        }
    }
}

void scan()
{
    bit valOn, valOff;
    /*scan the left site*/
    flashLeft(); //flashing IR beam for 600us
    valOn=DETECTOR;
    sleep(); //wait 600us
    valOff=DETECTOR;
    if ( (!valOn && valOff )==1 )
#define DETECTOR P3_0
#define LEFT_IR P3_1
#define RIGHT_IR P3_2

void flashLeft()
{
    _asm
    mov ACC,#46

    flashLeftLoop:
        cpl _P3_1;1 cycle  //LEFT_IR P3_1
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        djnz ACC, flashLeftLoop 2 cycles, need 13cycles in loop.

    clr _P3_1 ;1 cycle

    _endasm;
}

void flashRight()
{
    _asm
    mov ACC,#46

    flashRightLoop:
        cpl _P3_2;1 cycle  //RIGHT_IR P3_2
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        djnz ACC, flashRightLoop 2 cycles, need 13cycles in loop.

    _endasm;
}
void sleep()
{
    _asm
    mov A, #30

    sleepLoop:
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        nop ;1 cycle
        djnz ACC, sleepLoop ;2 cycle, need 20 cycles in loop

    _endasm;
}