KUMAN
www.kumantech.com

Infrared Obstacle Avoidance Module

Overview

"Obstacle Avoidance Module" is the conventional name for this small module which packages an infrared emitter and receiver into a mobile, low cost, and adjustable proximity sensor. When the sensor is facing a nearby obstacle, the emitted beam of (invisible) infrared light bounces off the obstacle and is reflected back to the receiver, which reports it as an obstacle. When no obstacle is present, the infrared beam does not bounce back, and so the receiver sees no reflection and therefore reports no obstacle. These sensors can be combined with microcontroller logic (like your Raspberry Pi) to create obstacle avoidance systems in wheeled robots and similar contexts.

In this experiment, you'll program the Raspberry Pi to switch on an LED when the obstacle avoidance module detects an obstacle.

Experimental Materials

Raspberry Pi	x1
Breadboard	x1
Obstacle Avoidance Module	x1
LED (3-pin)	x1
Resistor(330Ω)	x1
Dupont jumper wires	

Experimental Procedure

1. If you have not done so already, prepare your development system by installing the Python interpreter, RPi.GPIO library, and wiringPi library as described in READ_ME_FIRST.TXT.

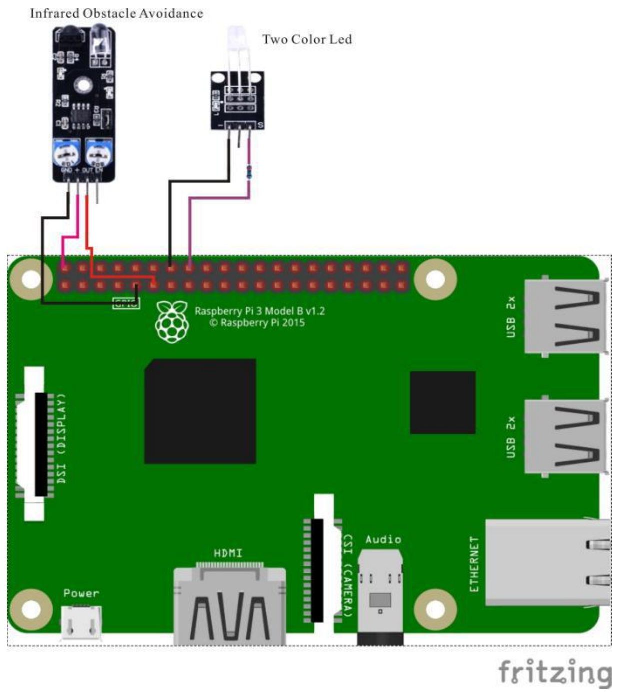
2. Install the obstacle avoidance module and three-pin LED on your breadboard, and use the resistor and Dupont jumper wires to connect them as illustrated in

the Wiring Diagram below. Note you will connect only two of the three pins on the LED, and only three of the four pins on the obstacle avoidance module.

3. Execute the sample stored in this experiment's subfolder.
   If using C, compile and execute the C code:
```
cd Code/C
gcc obstacle.c -o obstacle.out -lwiringPi
./obstacle.out
```
   If using Python, launch the Python script:
```
cd Code/Python
python obstacle.py
```

4. Make experimental observations. As you move the obstacles in front of the sensor, the LED illuminates. You can adjust the sensitivity of its distance detector by adjusting the **righthand** onboard potentiometer, labeled "205" just above the sensor's EN pin. The effective sensing range is ~2cm to 40cm. (The **lefthand** onboard potentiometer, labeled "103" above the GND pin, controls the frequency of the infrared transmitter. You will likely have no need to adjust this.)

Wiring Diagram



Obstacle Avoidance module position:

"OUT"	↔	Raspberry Pi pin 11
"+"	↔	Raspberry Pi +5V
"GND"	↔	Raspberry Pi GND

LED pin position:

"S”	↔	Raspberry Pi pin 16 (through resistor)
"-"	↔	Raspberry Pi GND

Sample Code

Python Code

```
#!/usr/bin/env python
import RPi.GPIO as GPIO

ObstaclePin = 11
LedPin = 16

def setup():
   GPIO.setmode(GPIO.BOARD)  # Numbers GPIOs by physical location
   GPIO.setup(ObstaclePin,
GPIO.IN, pull_up_down=GPIO.PUD_UP)
   GPIO.setup(LedPin, GPIO.OUT)

def loop():
   while True:
      if (0 == GPIO.input(ObstaclePin)):
         print "Barrier is detected !"
         GPIO.output(LedPin, True)
      else:
         GPIO.output(LedPin, False)
```

```
def destroy():
   GPIO.cleanup()              # Release resource

if __name__ == '__main__':   # Program start from here
   setup()
   try:
      loop()
   except KeyboardInterrupt:
      destroy()
```

C Code

```
#include <wiringPi.h>
#include <stdio.h>

#define ObstaclePin 0
#define LedPin    4

int main(void)
{
   if(wiringPiSetup() == -1)
   {
      printf("setup wiringPi failed !\n");
      return -1;
   }

   pinMode(LedPin, OUTPUT);
   while(1)
   {
      if(0 == digitalRead(ObstaclePin))
      {
         printf("Barrier detected!\n");
         digitalWrite(LedPin, HIGH);
      }
      else
      {
         digitalWrite(LedPin, LOW);
      }
   }
   return 0;
}
```