**Experiment No. 5**

**Title:** Understanding the connectivity of Raspberry-Pi /Beagle board circuit with IR sensor. Write anapplication to detect obstacle and notify user using LEDs.

**Aim:** To understand the interface between IR sensor and Raspberry Pi.

Demonstrate when IR detects any obstacle it counts the number of times obstacle detected and displays on screen.

**Hardware Requirement**:

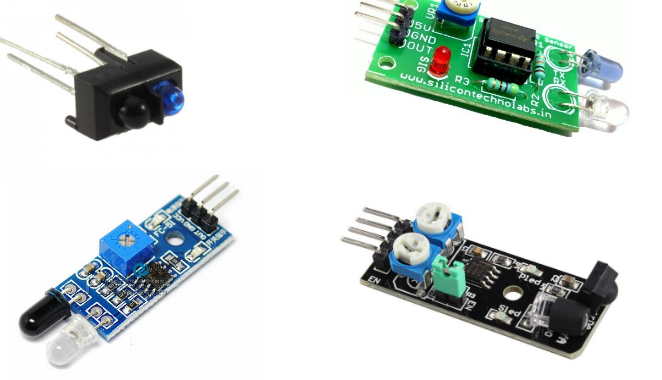
Raspberry Pi board with an SD card, IR Sensor

**Software Requirement:** Raspbian O.S, Python.

**Theory:**

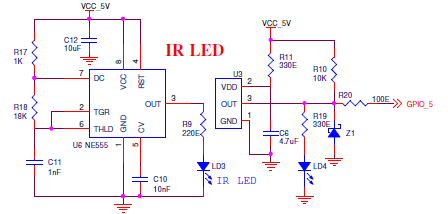
An infrared sensor is an electronic instrument which is used to sense certain characteristics of its surroundings by either emitting and/or detecting infrared radiation. Infrared sensors are also capable of measuring the heat being emitted by an object and detecting motion.

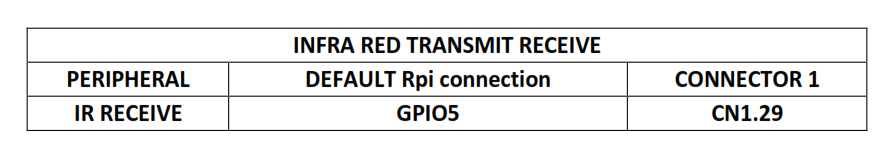
Infrared waves are not visible to the human eye. In the electromagnetic spectrum, infrared radiation can be found between the visible and microwave regions. The infrared waves typically have wavelengths between 0.75 and 1000μm.



IR transmitter and receiver pair is available on the Micro-Raspberry Pi board. The IR Transmitter transmits a signal at 38 KHz. A IC555 generates the 38KHz frequency which is then transmitted through the IR transmitter.

The IR receiver is a TSOP1738 sensor module. This IR sensor module consists of a PIN diode and a preamplifier which are embedded into a single package. The output of TSOP is active low and it gives +5V in off state. When IR waves, from a source, with a center frequency of 38 kHz incident on it, its output goes low. TSOP module has an inbuilt control circuit for amplifying the coded pulses from the IR transmitter. A signal is generated when PIN photodiode receives the signals. This input signal is received by an automatic gain control (AGC). For a range of inputs, the output is fed back to AGC in order to adjust the gain to a suitable level. The signal from AGC is passed to a band pass filter to filter undesired frequencies. After this, the signal goes to a demodulator and this demodulated output drives an npn transistor. The collector output of the transistor is obtained at pin 3 of TSOP module.





As soon as we execute the program IRLed.py, the sensor starts sensing if there is any obstacle around it.

If it found any kind of obstacle it send input to program and according to the logic if any input is detected the no of persons count gets increased and the count is displayed in screen.

**Conclusion:** We have successfully executed the program of displaying no. of obstacle found using IR sensor.And displayed on screen.

**Code :(IRLed.py)**

import time

import RPi.GPIO as GPIO

RUNNING = True

HIGH = 1

LOW = 0

DetectPin = 5

def InitSystem():

GPIO.setmode(GPIO.BCM)

GPIO.setup(DetectPin,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

return

def DetectPerson():

while True:

input\_state = GPIO.input(DetectPin)

time.sleep(0.3)

if input\_state == 0:

return LOW

else:

return HIGH

try:

print "\nCounting using IR LED\n"

print "-----------------------------------------------\n"

InitSystem()

count =0;

while RUNNING:

state = DetectPerson()

if state == LOW:

count+=1

print "person count =%d" %count

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

print "\Stopping Elevator"

# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()

**Code:**

import time

import RPi.GPIO as GPIO

RUNNING = True

HIGH = 1

LOW = 0

#Position LEDs

FL0\_LED\_POS = 10

FL1\_LED\_POS = 12

FL2\_LED\_POS = 13

FL3\_LED\_POS = 16

#CALL Position LEDs

FL0\_LED\_CALL = 6

FL1\_LED\_CALL = 7

FL2\_LED\_CALL = 8

FL3\_LED\_CALL = 9

#CALL Switch

FL0\_SW = 17

FL1\_SW = 18

FL2\_SW = 19

FL3\_SW = 20

#DIRECTION LED

LED\_D0 = 21

LED\_D1 = 22

LED\_D2 = 23

LED\_D3 = 24

LED\_D4 = 25

LED\_D5 = 26

LED\_D6 = 27

NO\_OF\_FLOORS = 4 # No of floors for Lift Simulation Operation

NO\_OF\_DIR\_LEDS = 7 # No of LEDs used for the lift direction (on Board)

DEFAULT\_LIFT\_POS = 0 # The floor no where lift is positioned when program is executed

DIR\_LED = [ LED\_D0,LED\_D1,LED\_D2,LED\_D3,LED\_D4,LED\_D5,LED\_D6]

FLOOR\_POS\_LED = [FL0\_LED\_POS,FL1\_LED\_POS,FL2\_LED\_POS,FL3\_LED\_POS]

FLOOR\_CALL\_LED =[FL0\_LED\_CALL,FL1\_LED\_CALL,FL2\_LED\_CALL,FL3\_LED\_CALL]

FLOOR\_SW =[FL0\_SW,FL1\_SW,FL2\_SW,FL3\_SW]

def InitElevator():

GPIO.setmode(GPIO.BCM)

GPIO.setup(FL0\_LED\_POS,GPIO.OUT)

GPIO.setup(FL1\_LED\_POS,GPIO.OUT)

GPIO.setup(FL2\_LED\_POS,GPIO.OUT)

GPIO.setup(FL3\_LED\_POS,GPIO.OUT)

GPIO.setup(FL0\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL1\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL2\_LED\_CALL,GPIO.OUT)

GPIO.setup(FL3\_LED\_CALL,GPIO.OUT)

GPIO.setup(LED\_D0,GPIO.OUT)

GPIO.setup(LED\_D1,GPIO.OUT)

GPIO.setup(LED\_D2,GPIO.OUT)

GPIO.setup(LED\_D3,GPIO.OUT)

GPIO.setup(LED\_D4,GPIO.OUT)

GPIO.setup(LED\_D5,GPIO.OUT)

GPIO.setup(LED\_D6,GPIO.OUT)

GPIO.setup(FL0\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL1\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL2\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

GPIO.setup(FL3\_SW,GPIO.IN,pull\_up\_down=GPIO.PUD\_UP)

for i in range(0,4):

GPIO.output(FLOOR\_POS\_LED[i],LOW)

for i in range(0,4):

GPIO.output(FLOOR\_CALL\_LED[i],LOW)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GoingUP():

for i in range(0,7):

GPIO.output(DIR\_LED[i],HIGH)

time.sleep(0.5)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GoingDOWN():

for i in range(0,7):

GPIO.output(DIR\_LED[6-i],HIGH)

time.sleep(0.5)

for i in range(0,7):

GPIO.output(DIR\_LED[i],LOW)

return

def GetFloorCall():

while True:

for call\_sw in range(0,4):

input\_state = GPIO.input(FLOOR\_SW[call\_sw])

time.sleep(0.1)

if input\_state == 0:

return call\_sw

try:

print "\nLift Operation Simulation using Python\n"

print "-----------------------------------------------\n"

InitElevator()

cur\_flr = DEFAULT\_LIFT\_POS # Variable for current lift floor (initially 0)

while RUNNING:

GPIO.output(FLOOR\_POS\_LED[cur\_flr],HIGH)

new\_flr = GetFloorCall()

if new\_flr > cur\_flr: # if (new floor > current floor) means lift is called to upper floor

tmp = cur\_flr # store current floor no into tmp variable

GPIO.output(FLOOR\_CALL\_LED[new\_flr],HIGH)

print "LIFT going UP to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (incremental); till it becomes destination

GoingUP() # Glow direction LEDs in upward direction

GPIO.output(FLOOR\_POS\_LED[tmp],LOW)

tmp += 1 # Increment tmp value by 1

GPIO.output(FLOOR\_POS\_LED[tmp],HIGH)

time.sleep(0.5) # Sleep for 0.5 second (500 ms)

elif new\_flr < cur\_flr: # if (new floor < current floor) means lift is called to lower floor

tmp = cur\_flr # store current floor no into tmp variable

GPIO.output(FLOOR\_CALL\_LED[new\_flr],HIGH)

GPIO.output(FLOOR\_CALL\_LED[cur\_flr],LOW)

print "LIFT going DOWN to floor #%d" %new\_flr # print destination floor

while (tmp != new\_flr): # Use tmp value (decremental); till it becomes destination

GoingDOWN() # Glow direction LEDs in downward direction

time.sleep(0.01) # Sleep for 10 ms

GPIO.output(FLOOR\_POS\_LED[tmp],LOW)

tmp -= 1 # Decrement tmp value by 1

GPIO.output(FLOOR\_POS\_LED[tmp],HIGH)

time.sleep(0.5) # sleep for 0.5 second (500 ms)

cur\_flr = new\_flr # Once lift reaches the destination; current floor points to destination floor no

GPIO.output(FLOOR\_CALL\_LED[cur\_flr],LOW)

time.sleep(0.1) # Sleep for 1 second

# If CTRL+C is pressed the main loop is broken

except KeyboardInterrupt:

RUNNING = False

print "\Stopping Elevator"

# Actions under 'finally' will always be called

finally:

# Stop and finish cleanly so the pins

# are available to be used again

GPIO.cleanup()