IR Receiver

Overview  
 This course will use the Raspberry Pi to control the infrared receiving head to receive the infrared signal, and control the LED light on and off according to the captured infrared signal.

Experimental Materials

RaspberryPi \*1

Breadboard \*1

InfraredEmitter \*1

InfraredReceiver \*1

Led \*1

Dupont Line

Ready to work  
1. Install python interpreter in your Raspberry Pi system  
2. Install the RPi.GPIO library in your Raspberry Pi system  
3. Install the wiringPi library in your Raspberry Pi system  
See the attached "Installing a Python Interpreter and Corresponding Libraries in a Raspberry Pi System" for details.

Product description

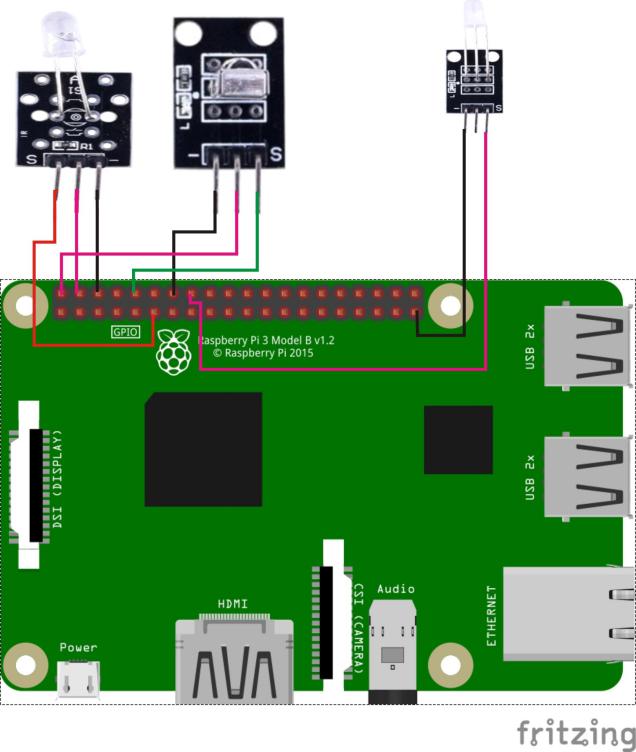
I. Introduction:

Infrared remote control is currently one of the most widely used communications and remote control methods.The infrared remote control device has the characteristics of small size, low power consumption, strong functions, and low cost. It is also used on recorders, audio equipment, air-conditioner,toys and other small electronic devices after the color television and video recorder.The infrared receiving head is a device that integrates optical signal (invisible light) into electric signal, receives, amplifies, and demodulates the integrated device. The internal IC has completed demodulation, and the output is a digital signal.



Characteristic parameters

◆Strong anti-interference: epoxy resin package plus shielding anti-jamming design  
◆ Wide operating voltage: 2.7-5.5V  
◆Low power consumption: Wide angle; Long distance reception  
◆ Output Match TTL: CMOS Level  
  
Wiring diagram



Sample code

1.Python code

#!/usr/bin/env python

import RPi.GPIO as GPIO

import time

IrReceiverPin = 10

IrEmissionPin = 11

LedPin = 16

Led\_status = 1

def setup():

GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location

GPIO.setup(LedPin, GPIO.OUT) # Set LedPin's mode is output

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

GPIO.setup(IrEmissionPin, GPIO.OUT) # Set IrPin's mode is output

GPIO.output(LedPin, GPIO.LOW) # Set LedPin low to off led

def swLed(ev=None):

global Led\_status

Led\_status = not Led\_status

GPIO.output(LedPin, Led\_status) # switch led status(on-->off; off-->on)

print("receive signal!\n")

time.sleep(0.1)

GPIO.output(LedPin, False)

def loop():

GPIO.add\_event\_detect(IrReceiverPin, GPIO.FALLING, callback=swLed) # wait for falling

while True:

print '...IrPin high'

GPIO.output(IrEmissionPin, GPIO.HIGH) # IrPin on

time.sleep(0.5)

print 'IrPin low...'

GPIO.output(IrEmissionPin, GPIO.LOW) # IrPin off

time.sleep(0.5)

def destroy():

GPIO.output(LedPin, GPIO.LOW) # led off

GPIO.cleanup() # Release resource

if \_\_name\_\_ == '\_\_main\_\_': # Program start from here

setup()

try:

loop()

except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the child program destroy() will be executed.

print("KeyboardInterrupt.\n")

destroy()

2.C code

#include <wiringPi.h>

#include <stdio.h>

#define IrEmissionPin 0

#define IrReceiverPin 16

#define LedPin 4

int cnt = 0;

void myISR(void)

{

printf("Recevied infrared. cnt = %d\n", ++cnt);

if(digitalRead(LedPin) == HIGH)

{

digitalWrite(LedPin, LOW);

}

else

{

digitalWrite(LedPin, HIGH);

}

delay(100);

digitalWrite(LedPin, LOW);

}

int main(void)

{

if(wiringPiSetup() == -1)

{

printf("setup wiringPi failed !\n");

return -1;

}

pinMode(LedPin, OUTPUT);

pinMode(IrEmissionPin, OUTPUT);

pinMode(IrReceiverPin, INPUT);

pullUpDnControl(IrReceiverPin, PUD\_UP)

if(wiringPiISR(IrReceiverPin, INT\_EDGE\_FALLING, &myISR) == -1)

{

printf("setup ISR failed !");

return -1;

}

while(1)

{

digitalWrite(IrEmissionPin, HIGH);

printf("IrEmissionPin is set High\n");

delay(500);

digitalWrite(IrEmissionPin, LOW);

printf("IrEmissionPin is set Low\n");

delay(500);

}

return 0;

}

Experimental phenomena

Whenever the infrared receiver receives an infrared signal, the status of the led lamp will change.