24.HC-SR501_Sensor

Introduction

In this lesson, we will use PIR to sense the movement of pedestrians, and use servos, LED, buzzer to simulate the work of the sensor door of the convenience store. When the pedestrian appears within the sensing range of the PIR, the indicator light will be on, the door will be opened, and the buzzer will play the opening bell.

Hardware Required

- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * HC-SR501 PIR SENSOR
- ✓ 1 * LED
- ✓ 1 * 40-pin Cable
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard
- ✓ 1 * Resistor (220Ω)

Principle

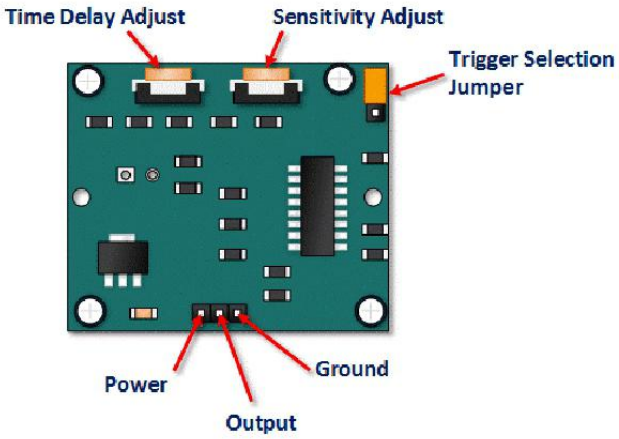
HC-SR501 PIR SENSOR

PIR sensors are more complicated than many of the other sensors explained in this tutorial (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output.

The PIR sensor itself has two slots. Each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor).

When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or an animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

24.HC-SR501_Sensor



Pin or Control	Function
Time Delay Adjust	Sets how long the output remains high after detecting motion.... Anywhere from 5 seconds to 5 minutes.
Sensitivity Adjust	Sets the detection range.... from 3 meters to 7 meters
Trigger Selection Jumper	Set for single or repeatable triggers.
Ground pin	Ground input
Output Pin	Low when no motion is detected.. High when motion is detected. High is 3.3V
Power Pin	5 to 20 VDC Supply input

HC SR501 PIR Functional Description

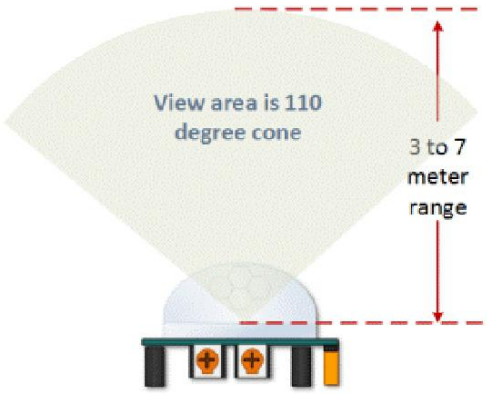
The SR501 will detect infrared changes and if interpreted as motion, will set its output low. What is or is not interpreted as motion is largely dependent on user settings and adjustments.

Device Initialization

The device requires nearly a minute to initialize. During this period, it can and often will output false detection signals. Circuit or controller logic needs to take this initialization period into consideration.

Device Area of Detection

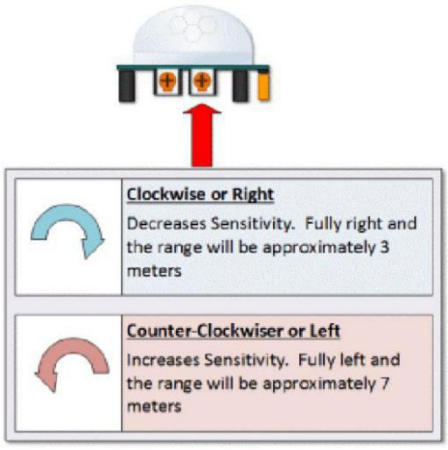
The device will detect motion inside a 110 degree cone with a range of 3 to 7 meters.



HC SR501 View Area

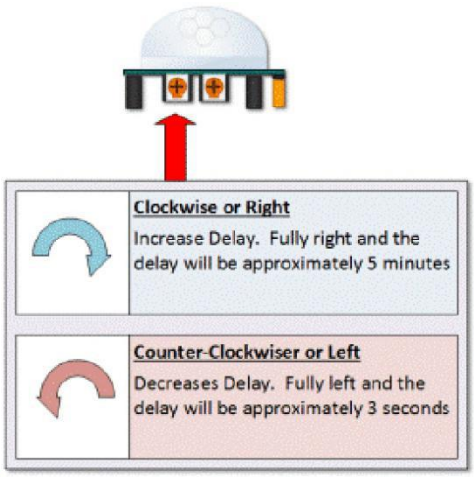
PIR Range (Sensitivity) Adjustment

As mentioned, the adjustable range is from approximately 3 to 7 meters. The illustration below shows this adjustment.



HC SR501 Sensitivity Adjust Time Delay Adjustment

The time delay adjustment determines how long the output of the PIR sensor module will remain high after detection motion. The range is from about 3 seconds to five minutes.



HC SR501 Time Delay Adjustment

3 Seconds Off After Time Delay Completes – IMPORTANT

The output of this device will go LOW (or Off) for approximately 3 seconds AFTER the time delay completes. In other words, ALL motion detection is blocked during this three second period.

For Example:

Imagine you're in the single trigger mode and your time delay is set 5 seconds.

The PIR will detect motion and set it high for 5 seconds.

After five seconds, the PIR will sets its output low for about 3 seconds.

During the three seconds, the PIR will not detect motion.

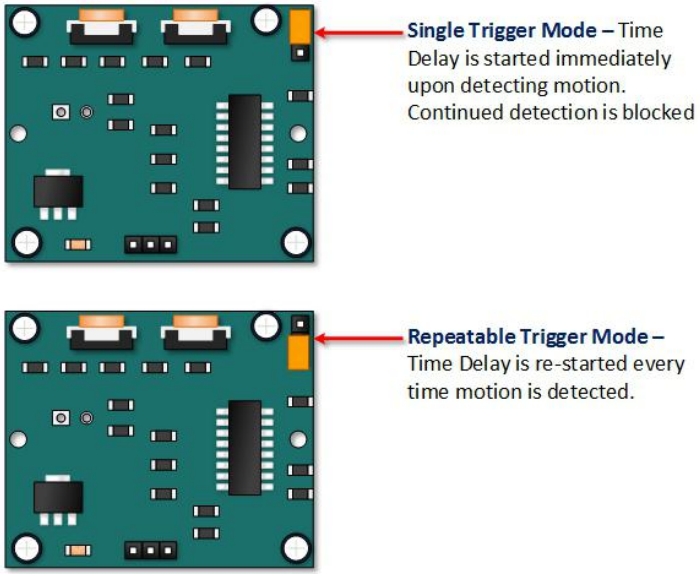
After three seconds, the PIR will detect motion again and detected motion will once again set the output high.

Trigger Mode Selection Jumper

The trigger mode selection jumper allows you to select between single and repeatable triggers. The affect of this jumper setting is to determine when the time delay begins.

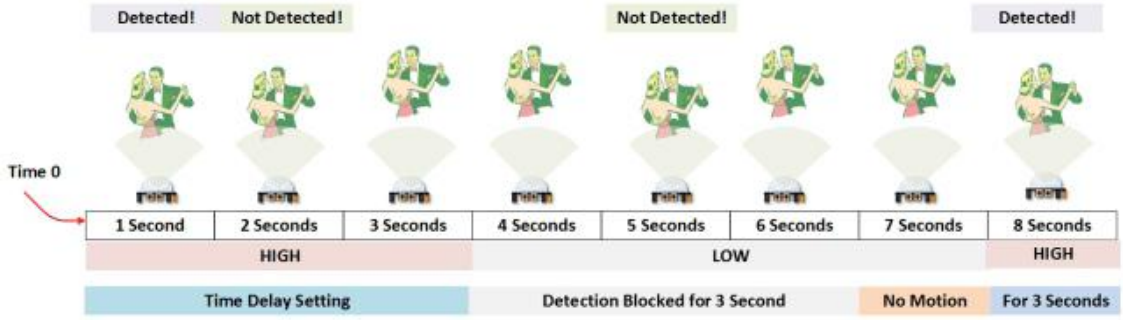
Application Examples

Imagine that you want to control lighting on a dance floor based upon where the dancers are dancing. Understanding how the time delay and trigger mode interact will be necessary to controlling that lighting in the manner that you want.



Example One

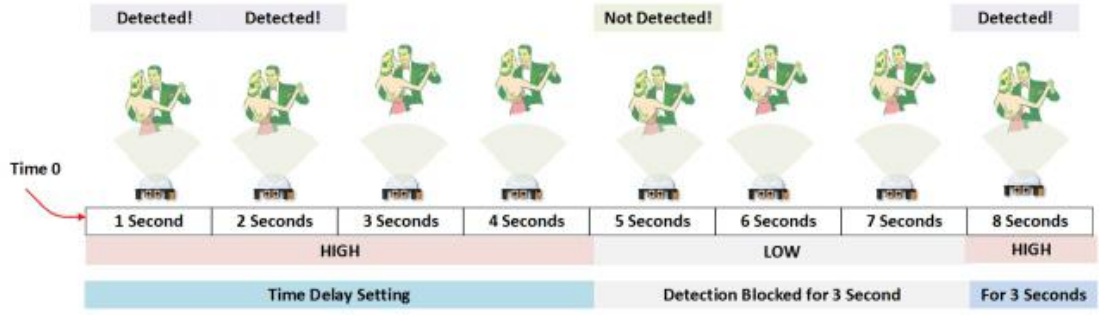
In this first example, the time delay is set to three seconds and the trigger mode is set to single. As you can see in the illustration below, the motion is not always detected. In fact, there is a period of about six seconds where motion can not be detected. Feel free to click on the image to enlarge.



Example Two

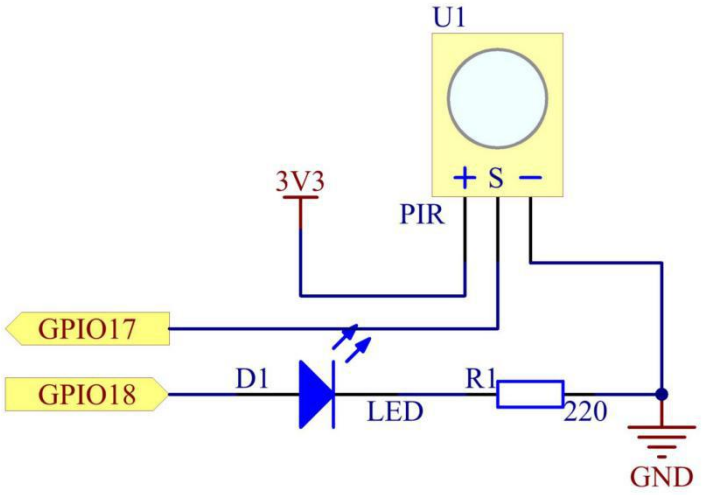
In the next example, the time delay is still at three seconds and the trigger is set to repeatable. In the illustration below, you can see that the time delay period is restarted.

However, after that three seconds, detection will still be blocked for three seconds. As I mentioned previously, you could override the 3 second blocking period with some creative code, but do give that consideration. Some of the electronics you use may not like an on and then off jolt. The three seconds allows for a little rest before starting back up.



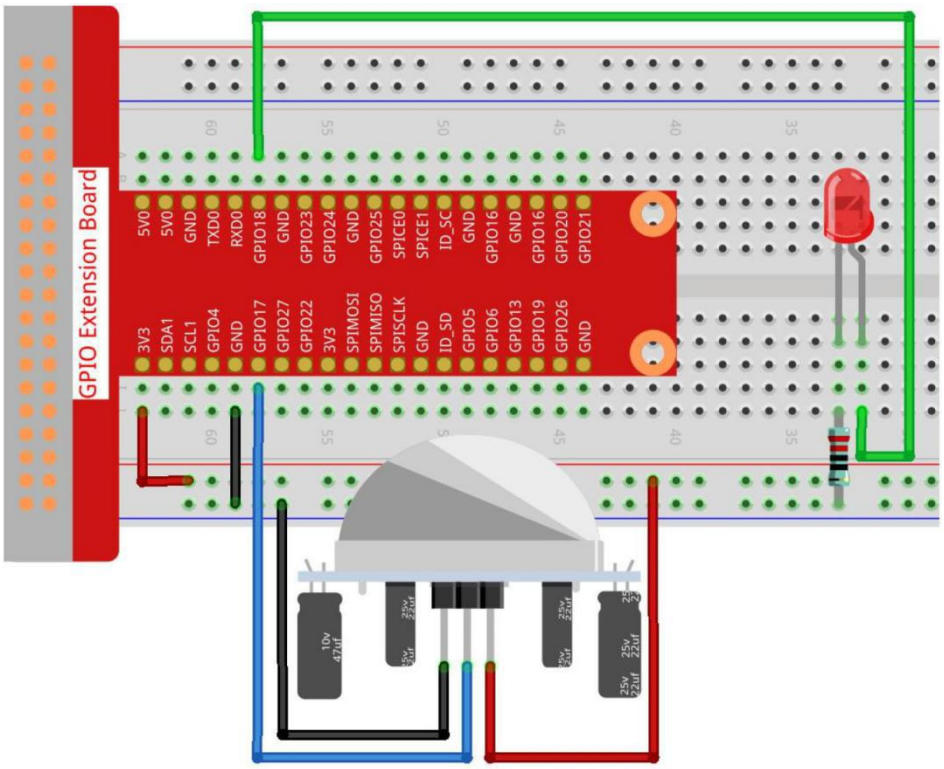
Schematic Diagram

T-Board Name	physical	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18



Experimental Procedures

Step 1: Build the circuit.



For C Language Users

Step2: Go to the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/24.HC-SR501
```

Step 3: Compile the code.

```
gcc 24.HC-SR501.c -o HC-SR501.out -lwiringPi
```

Step 4: Run the executable file .

```
sudo ./HC-SR501.out
```

After the code runs, PIR detects surroundings and lights an LED if it senses someone walking by. There are two potentiometers on the PIR module: one is to adjust sensitivity and the other is to adjust the detection distance. In order to make the PIR module work better, you need to try to adjust these two potentiometers.

Code

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

#define ledPin      1       //define the ledPin
#define sensorPin 0         //define the sensorPin

int main(void)
{
    if(wiringPiSetup() == -1){ //when initialize wiring failed,print messageto screen
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(ledPin, OUTPUT);    //control the led
    pinMode(sensorPin, INPUT);    //detect the light

    while(1){

        if(digitalRead(sensorPin) == HIGH){ //if read sensor for high level
            digitalWrite(ledPin, HIGH);      //led on
```

```
            printf("led on...\n");
        }
        else {
            digitalWrite(ledPin, LOW);      //led off
            printf("...led off\n");
        }
    }

    return 0;
}
```

Code Explanation

```
        if(digitalRead(sensorPin) == HIGH){ //if read sensor for high level
            digitalWrite(ledPin, HIGH);      //led on
            printf("led on...\n");
        }
```

The PIR works like a button, and we read its value to control the LED.

For Python Language Users

Step 3: Go to the folder of the code.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 4: Run the executable file.

```
sudo python3 24.HC-SR501.py
```

After the code runs, PIR detects surroundings and lights an LED if it senses someone walking by. There are two potentiometers on the PIR module: one is to adjust sensitivity and the other is to adjust the detection distance. In order to make the PIR module work better, you need to try to adjust these two potentiometers.

Code

The code here is for Python3, if you need for Python2, please open the code with the

suffix py2 in the attachment.

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO

ledPin = 18       # define the ledPin
sensorPin = 17       # define the sensorPin

def setup():
    GPIO.setmode(GPIO.BCM)           # Numbers GPIOs by physical location
    GPIO.setup(ledPin, GPIO.OUT)     # Set ledPin's mode is output
    GPIO.setup(sensorPin, GPIO.IN)       # Set sensorPin's mode is input

def loop():
    while True:
        if GPIO.input(sensorPin)==GPIO.HIGH:
            GPIO.output(ledPin,GPIO.HIGH)
            print ('led on ...')
        else :
            GPIO.output(ledPin,GPIO.LOW)
            print ('led off ...')

def destroy():
    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.
                    destroy()
```

Code Explanation

```
        if GPIO.input(sensorPin)==GPIO.HIGH:
            GPIO.output(ledPin,GPIO.HIGH)
            print ('led on ...')
```

The PIR works like a button, and we read its value to control the LED.

Phenomenon Picture