




ASSIGNMENT 2

PIR [PASSIVE INFRARED] MOTION SENSOR

MAITRIK PATEL [147497176], HAPPY PATEL [154566178]



Introduction

Humans and other animals emit radiation all the time. This is nothing to be concerned about, though, as the type of radiation we emit is infrared radiation (IR), which is pretty harmless at the levels at which it is emitted by humans. In fact, all objects at temperatures above absolute zero (-273.15C) emit infrared radiation.

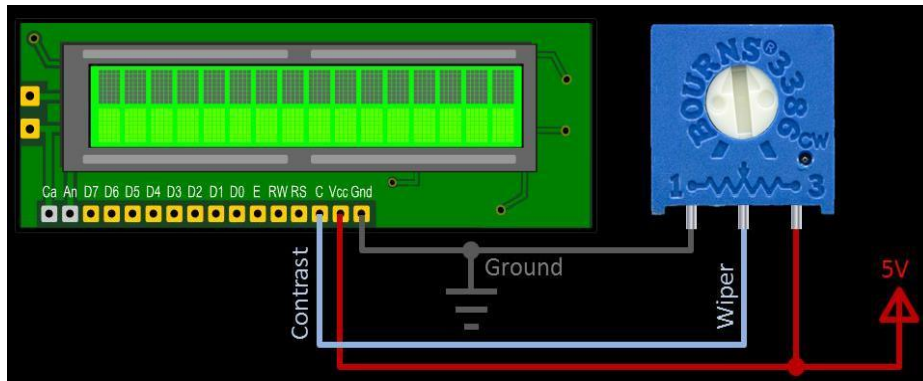
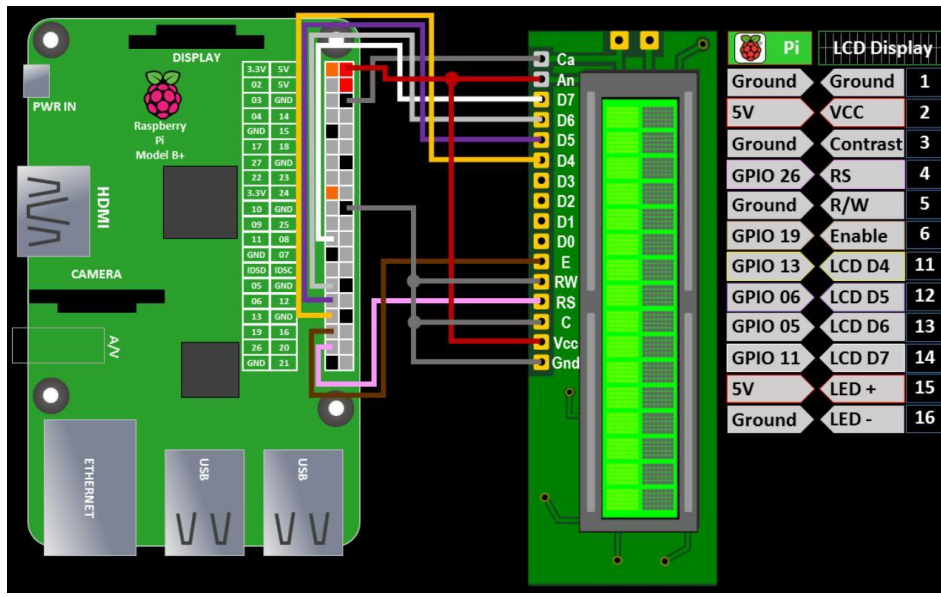
A PIR sensor detects changes in the amount of infrared radiation it receives. When there is a significant change in the amount of infrared radiation it detects, then a pulse is triggered. This means that a PIR sensor can detect when a human (or any animal) moves in front of it.

LCD Display

Step 1 – connect LCD Screen to PI

In order to send data to the LCD we are going to wire it up as follows

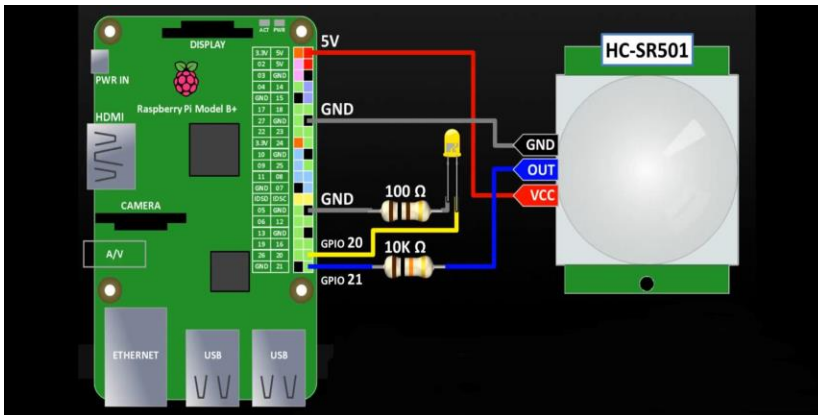
- Pin #1 of the LCD goes to ground
- Pin #2 of the LCD goes to +5V
- Pin #3 (Vo) of the LCD goes to ground
- Pin #4 (RS) connects to the Cobbler #26
- Pin #5 (RW) goes to ground
- Pin #6 (EN) connects to Cobbler #19
- Skip LCD Pins #7, #8, #9 and #10
- Pin #11 (D4) connects to cobbler #13
- Pin #12 (D5) connects to Cobbler #06
- Pin #13 (D6) connects to Cobber #05
- Pin #14 (D7) connects to Cobber #11
- Pin #15 (LED +) goes to +5V (red wire)
- Pin #16 (LED –) goes to ground (black wire)



Step 2 – connect HC-SR50, LED, BUZZER to PI

The setup is very simple since only one pin has to be activated during movement. The pins on the PIR are labelled:

- VCC to pin 2 (5V)
- OUT to pin 16 (GPIO 21)
- GND an pin 6 (ground)



The sensor is powered from one of the Pi's 5V pins, but it outputs 3.3V so it is safe to hook up directly to any of the Pi's GPIO pins. I'm using GPIO 21. The 10KΩ resistor is not necessary. It is only a precaution. I like to use resistors to protect the Pi GPIO inputs which can be damaged if accidentally connected to 5V or higher. The optional LED and BUZZER is connected in series with a resistor to GPIO 20. With a 100Ω resistor, GPIO 20 is sourcing about 11.8mA to power the LED. You don't want to exceed 16mA on any GPIO pin. LED's have very different current requirements so please check your datasheets. I always test the current on a breadboard with a multimeter prior to connecting to the Pi.

The code imports the Adafruit CharLCD library to drive the LCD display. It also uses the RPI library to handle the interrupts and requires version 0.5.2a or higher. The `GPIO.setup()` method is used to set GPIO 21 as an input. The internal pull down resistor is activated because the motion sensor output goes high upon detection. The pull down insures the default state of the pin is low. The pull down is not necessary if your sensor already pulls the output pin low in the no motion state. `GPIO.setup` is used again to configure GPIO 20 as an output to drive the LED indicator. The `motionSensor()` callback method is fired when the motion sensor output changes state (high to low or low to high). In either state the LCD display is cleared and the LED indicator is turned off. In addition, if there is a rising state which indicates motion, then the LCD display shows a message and the LED is turned on. The `GPIO.add_event_detect()` method initializes the interrupt which fires the above callback method. The `GPIO.both` parameter is passed because the

method should be called when motion is detected and when the HC-SR501 times out. I increased the bounce time to 300mS from 150mS in the video to optimize the performance. It is important to execute the `GPIO.cleanup()` method when the program exits to free up resources and prevent errors.

Step 3 – Code

Installing adafruit_charLCD Module

1. `sudo apt-get update`
2. `sudo apt-get install build-essential python-dev python-smbus python-pip git`
3. `sudo pip install RPi.GPIO`
4. `sudo pip install Adafruit_BBIO`
5. `cd ~`
6. `git clone https://github.com/adafruit/Adafruit_Python_CharLCD.git`
7. `cd Adafruit_Python_CharLCD`
8. `sudo python setup.py install`

Main Code

```
#!/usr/bin/python
#-----
# PROGRAM :PIRmotionsensor.py
# AUTHOR : MAITRIK PATEL [147497176], HAPPY PATEL
[154566178]
# DATE : 05/04/2019
# DESCRIPTION : PIR MOTION SENSOR
#-----

import signal
from time import sleep
import RPi.GPIO as GPIO
from Adafruit_CharLCD import Adafruit_CharLCD
lcd = Adafruit_CharLCD(rs=26, en=19, d4=13, d5=6, d6=5,
d7=11, cols=16, lines=2)
```

```
lcd.clear()
GPIO.setmode(GPIO.BCM)      # set up BCM GPIO numbering

# Set up input pin
GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Set up LED and BUZZER output
GPIO.setup(20, GPIO.OUT)

# Handler for SIGTERM signal, triggers system exit
def exit_program(signum, frame):
    exit(2)

# Callback function to run when motion detected
def motionSensor(channel):
    lcd.clear()
    GPIO.output(20, GPIO.LOW)
    if GPIO.input(21):      # True = Rising
        global counter
        counter += 1
        lcd.message('Motion
Detected\n{0}'.format(counter))
        GPIO.output(20, GPIO.HIGH)

# add event listener on pin 21
GPIO.add_event_detect(21, GPIO.BOTH,
callback=motionSensor, bouncetime=300)
counter = 0

try:
    signal.signal(signal.SIGTERM, exit_program)

    while True:
        sleep(1)          # wait 1 second

finally:
    # run on exit
```

```
GPIO.cleanup()          # clean up  
print "All cleaned up."
```

Reference

1. <https://www.raspberrypi-spy.co.uk/2012/07/16x2-lcd-module-control-using-python/>
2. <https://www.rototron.info/using-a-motion-detector-on-raspberry-pi/>
3. <https://youtu.be/dQY6hNA53oM>
4. <https://www.rototron.info/using-an-lcd-display-with-inputs-interrupts-on-raspberry-pi/>