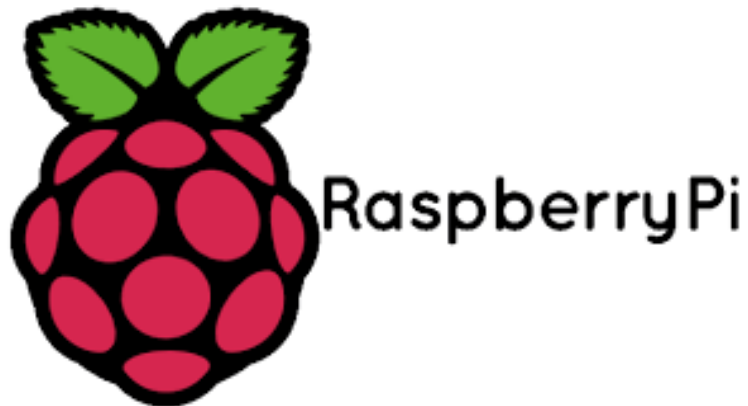


Birzeit University
Electrical and Computer Engineering Department
Interfacing Lab ENCS412

Exp. No. 7

Introduction to Raspberry Pi



Introduction

This is an introductory experiment about the credit card sized computer Raspberry Pi. This experiment is ideal for those who are interested in exploring the possibilities of Raspberry Pi as a computer. The experiment does not assume any prior knowledge on Raspberry Pi programming. However, knowledge of Linux operating system and Python programming language would greatly help you in getting up to speed.

What is Raspberry Pi?

The Raspberry Pi is a computer, very like the computers with which you're already familiar. It uses a different kind of processor you can install several versions of the Linux operating system that look and feel very much like Windows. If you want to, you can use the Raspberry Pi to surf the internet, send an email or write a letter using a word processor. But you can also do so much more.

This little board here is low cost, it's easily accessible, it's very simple to use. When you power it up you get a nice little desktop environment, it includes all of the things that you need to do to get started to learn programming. There's lots of information on the internet that you can take away and start programming code in to make things happen.

The great thing about these boards as well is in addition to software, you can play with hardware. So these little general purpose pins here allow access to the processor and you can hang off little hardware projects that you build and you can control via the code you are writing through the software application. So, this is a great tool for kids to learn how computers work at a grassroots level.

So there are various versions available of the Raspberry Pi currently. The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB. The system has Secure Digital (SD) (models A and B) or Micro SD (models A+ and B+) sockets for boot media and persistent storage.

In early February 2015, the next-generation Raspberry Pi, Raspberry Pi 2, was officially announced. The new computer board will initially be available only in one configuration (model B) and features a Broadcom BCM2836 SoC, with a quad-core ARM Cortex-A7 CPU 900MHz and a VideoCore IV dual-core GPU; 1 GB of RAM with remaining specifications being similar to those of the previous generation model B+.

Raspberry Pi Board

In this experiment we will use the most recent board which is the **Raspberry Pi 2 model B**.

Figure 1 below shows Raspberry Pi 2 model B board components.

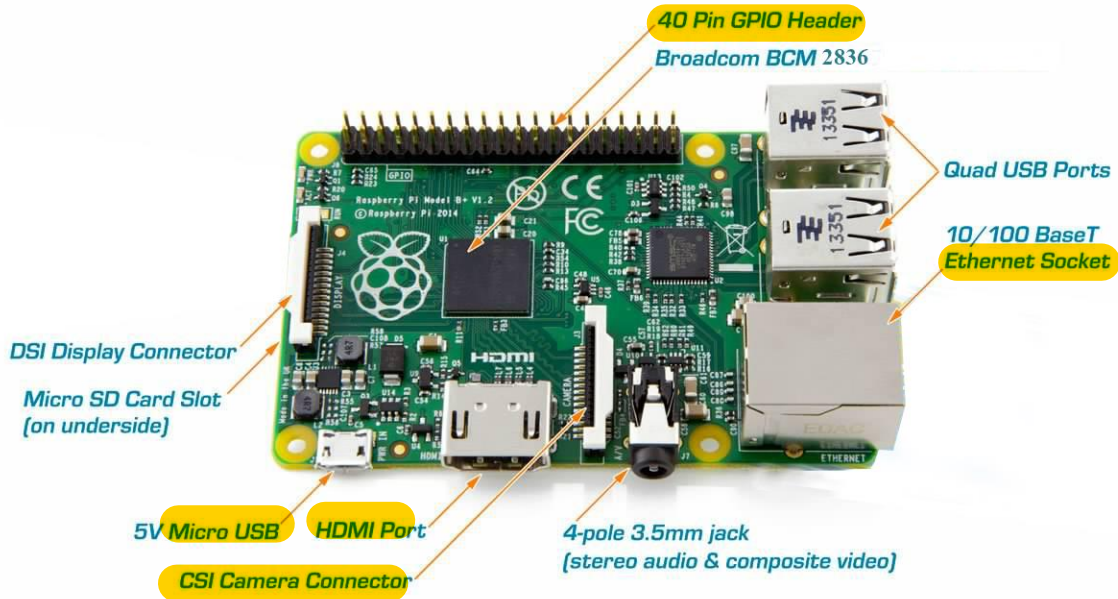


Fig.1: Raspberry Pi 2 board components.

As shown the board consists of the following:

2 USB ports, Ethernet out, camera connector, HDMI out, micro USB power, SD card, display connector, GPIO pins, video out, audio out and status LEDs.

And it has the following main features compared to model B+ and model A+:


	Raspberry Pi 2	Model B+	Model A+
			
SoC/CPU	BCM2836 Quadcore 900 MHz ARMv7	BCM2835 700 MHz ARMv6k	BCM2835 700 MHz ARMv6k
GPU	Broadcom VideoCore IV @ 250 MHz	Broadcom VideoCore IV @ 250 MHz	Broadcom VideoCore IV @ 250 MHz
RAM	1GB	512 MB	256 MB
Storage	MicroSD	MicroSD	MicroSD
USB	4	4	1
Ethernet	1	1	0
Video output	HDMI/Composite via RCA jack	HDMI/Composite via RCA jack	HDMI/Composite via RCA jack
Audio output	3.5 mm jack	3.5 mm jack	3.5 mm jack
GPIO	40	40	40

Fig.2: Raspberry Pi 2 board components.

GPIO PINs:

One powerful feature of the Raspberry Pi is the row of **GPIO (general purpose input/output) pins** along the edge of the board, next to the video out socket.

These pins are a **physical interface between the Raspberry Pi and the outside world**. At the simplest level, you can think of them **as switches** that **you** can **turn on or off (input)** or that the **Raspberry Pi** can **turn on or off (output)**. **26** of the 40 pins are **GPIO pins**; the **others** are **power or ground pins**.

GPIO **voltage levels** are **3.3v** and are **not 5v tolerant**. There is **no over-voltage protection on the board**. A **voltage near 3.3 V** is interpreted as a **logic one** while a **voltage near zero volts** is a **logic zero**.

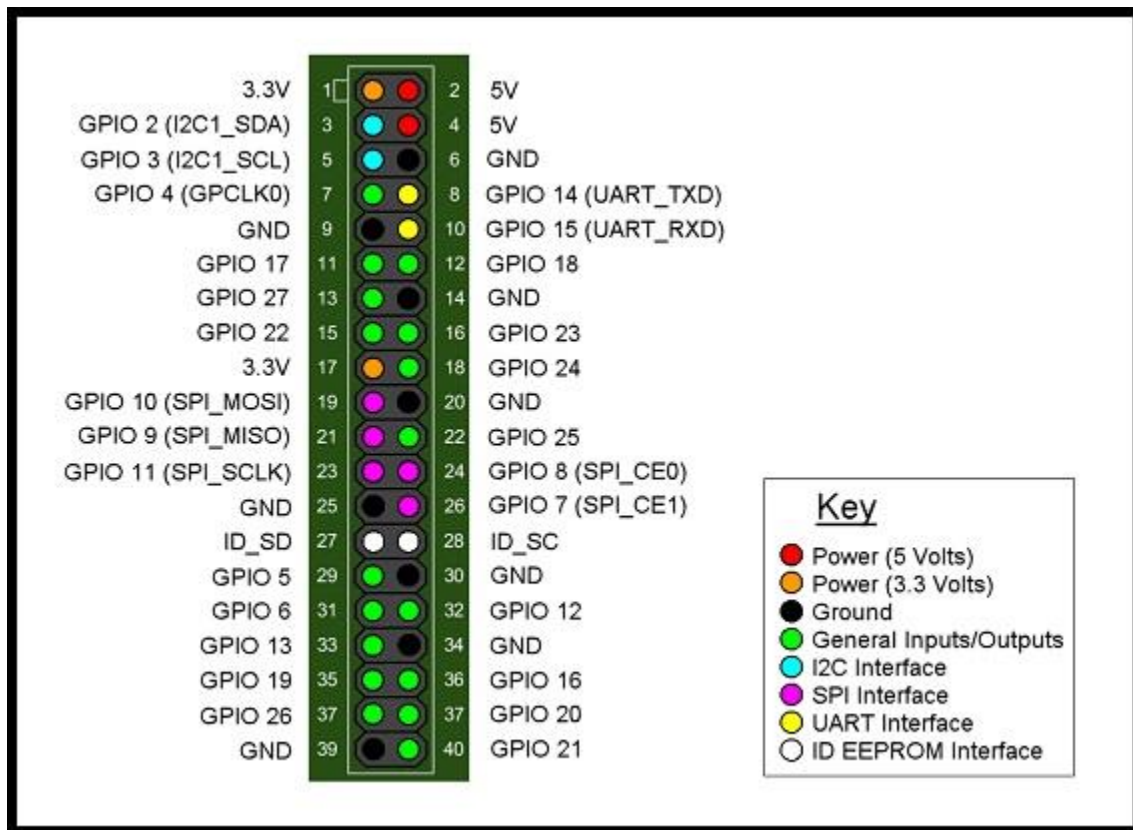


Fig. 3: Raspberry Pi 2 GPIO pins.

What are they for? What can I do with them?

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from

another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere and those devices can send data back. **Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.**

Raspberry Pi programming Language

The Raspberry Pi Foundation recommends **Python** as a language **for learners**. **Any language** which will **compile for ARMv6** can be used with the Raspberry Pi, though; so you are not limited to using **Python**. **C, C++, Java, Scratch**, and **Ruby** all come installed by **default on the Raspberry Pi**.

Python is a dynamic, interpreted (bytecode-compiled) language. There are **no type declarations of variables, parameters, functions, or methods in source code**. **This makes the code short and flexible**, and you **lose the compile-time type checking of the source code**. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

To learn how to program using Python languages see this link:

<http://www.tutorialspoint.com/python/index.htm>

PIR sensor



Fig. 4: PIR (motion detector) sensor

PIR sensors allow you to sense motion, almost always used to **detect whether a human has moved in or out of the sensors range**. They are **small, inexpensive, low-power, easy to use** and **don't wear out**. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, **"Passive Infrared"**, **"Pyroelectric"**, or **"IR motion sensors"**.

PIRs are basically **made of a pyroelectric sensor**, which can **detect levels of infrared radiation**. **Everything emits some low level radiation, and the hotter something is, the more radiation is emitted**. The sensor in a **motion detector** is actually **split in two halves**.

The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

Gas sensor

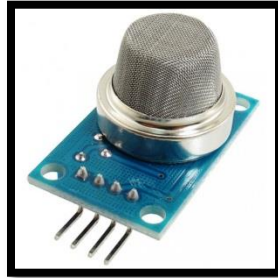


Fig. 5: Gas sensor

This is a simple-to-use liquefied petroleum gas (LPG) sensor, suitable for sensing LPG (composed of mostly propane and butane) concentrations in the air. The MQ-6 can detect gas concentrations anywhere from 200 to 10000ppm.

This sensor has a high sensitivity and fast response time. The sensor's output is an analog resistance. The drive circuit is very simple; all you need to do is power the heater coil with 5V, add a load resistance, and connect the output to an ADC.

Operating System

1-Installing impeded operating system:

In this experiment we will be using the raspberry pi with the Rasbian as the operating system for the raspberry pi. In order to do that, we need to go through a few easy steps.

We need 8 GB class 4(or higher) memory card, card reader, laptop or computer connected to internet, mouse, keyboard and screen.

1-Formatting the SD card:

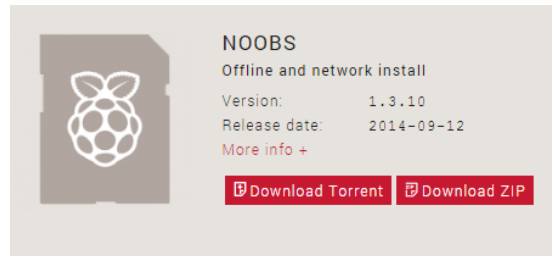
We need to clear all the files from the SD card; this can be done by downloading a program on the computer from www.sdcard.org. Just go to the download page and select “SD Formatter for Windows or MAC Download”.

After finishing the download and the program setup insert the SD card in the card reader and format it using the program.

2-Downloading NOOBS:

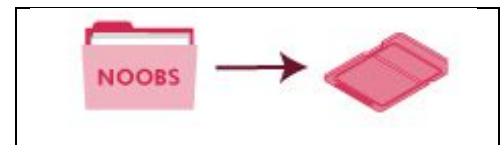
To setup Raspbian, NOOBS must be downloaded first (also you can download Raspbian without the NOOBS).

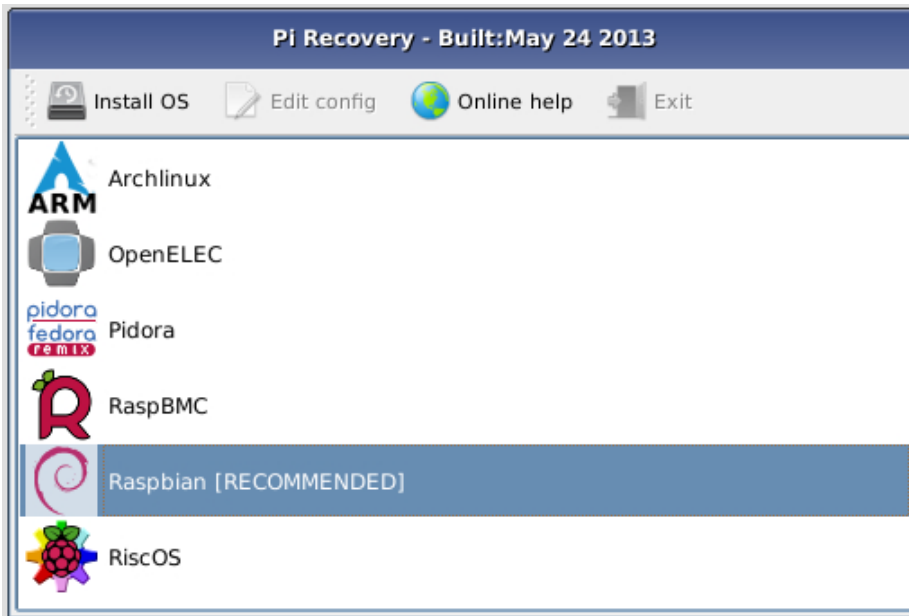
Go to the download page of www.raspberrypi.org and download NOOBS on your computer. Then open the file and copy all the files from the NOOBS folder to the SD card, this can take up to 20 minutes.



3- Installing Raspbian:

After copying the files, safely remove the SD card and insert it in its place in the raspberry pi. Connect the mouse, keyboard and screen to the raspberry pi then power it. Raspberry pi will start booting.





After the boot you will see a list on the screen of the possible operating systems than can be setup (as the above picture). Choose Raspbian and press OK. This should start the setup. It will take up to 30 minutes.



Note: while setup Raspbian it worth to see the notes which will appear on the screen.

PIR Motion Detector

In the first part we will connect the PIR and led to the raspberry pi, our goal is to turn on the led when a motion is detected.

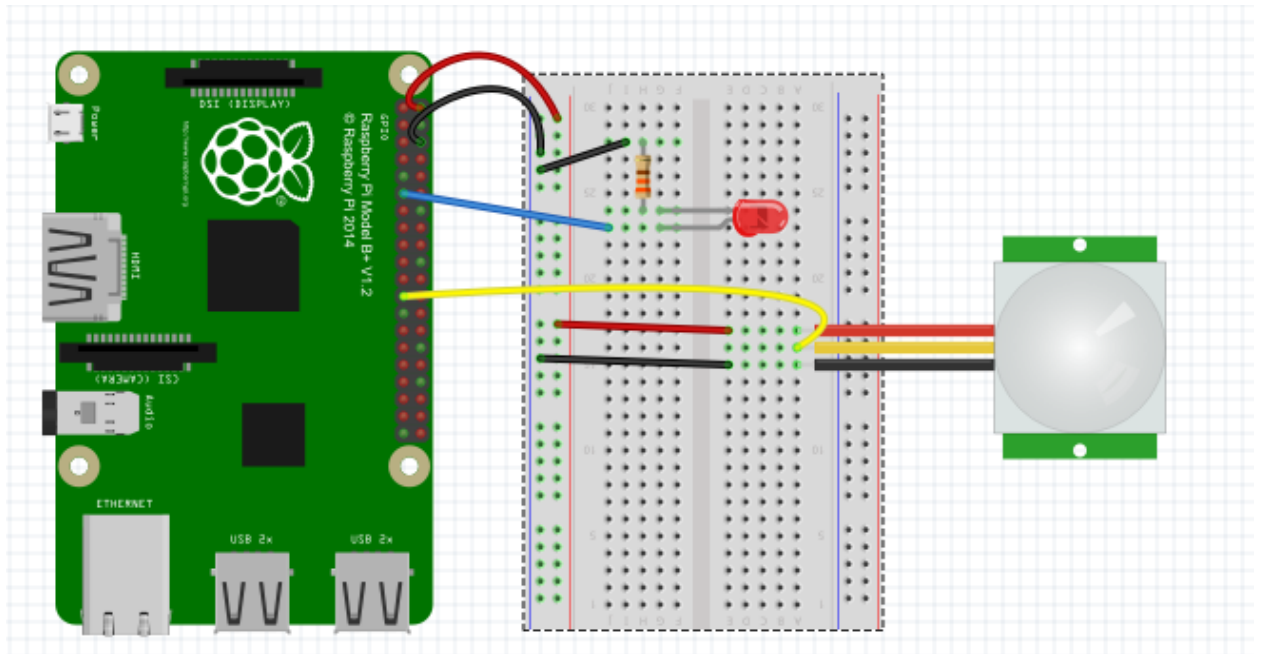
In the second part we will add LDR to the circuit, this will turn on the led if there is a motion detected and there is no light detected.

Procedure:

1-PIR with led:

Connect PIR sensor and a led to the raspberry pi as in the picture:

- 1- Connect vcc on pin#1
- 2- Connect gnd on pin#6
- 3- Connect output pin of PIR to pin#23
- 4- Connect 300ohm resistor with the led(+leg) then connect it with pin#11
- 5- Connect the short (-) leg to GND



6-Then open LXterminal from tools bar to run the following program, to do this go to the location of the saved code which is:

```
Cd /home/pi
```

Writing the code:

- 1- Create new folder

```
mkdir "folder name"
```

- 2- Enter the folder

```
cd "folder name"
```

- 3- Open the editor

```
sudo nano "program name".py
```

- 4- Write the following code in the editor:

```
1 import time
2 import RPi.GPIO as GPIO #import GPIO library
3 GPIO.setmode(GPIO.BOARD) #configuration GPIO pins
4 GPIO.setup(23, GPIO.IN) #define pin #23 as input pin
5 GPIO.setup(11, GPIO.out) #define pin #11 as output pin
6
7 while True:
8     if (GPIO.input(23) == True): #true means high
9         printn " Motion Detected "
10        GPIO.output(11,1) #turn on led 1 is high
11    else:
12        printn " There is No Motion "
13        GPIO.output(11,0) #turn of led 0 is low
14    time.sleep(1) #delay for one second
15
```

- 5- To save the code press “ctrl+x”, then press “y” then enter.

To run the code write the following command:

```
sudo python program_Name.py
```

2-PIR with led and LDR:

Since raspberry pi can't read analog signal and because LDR only gives analog, output we need to find a way to connect the LDR to it without using ADC.

The way we do this is by taking advantage of a basic electronic property of resistors and capacitors. It turns out that if you take a capacitor that is initially storing no voltage, and then connect it to power (like 3.3V) through a resistor, it will charge up to the power voltage slowly. The bigger the resistor, the slower it is.

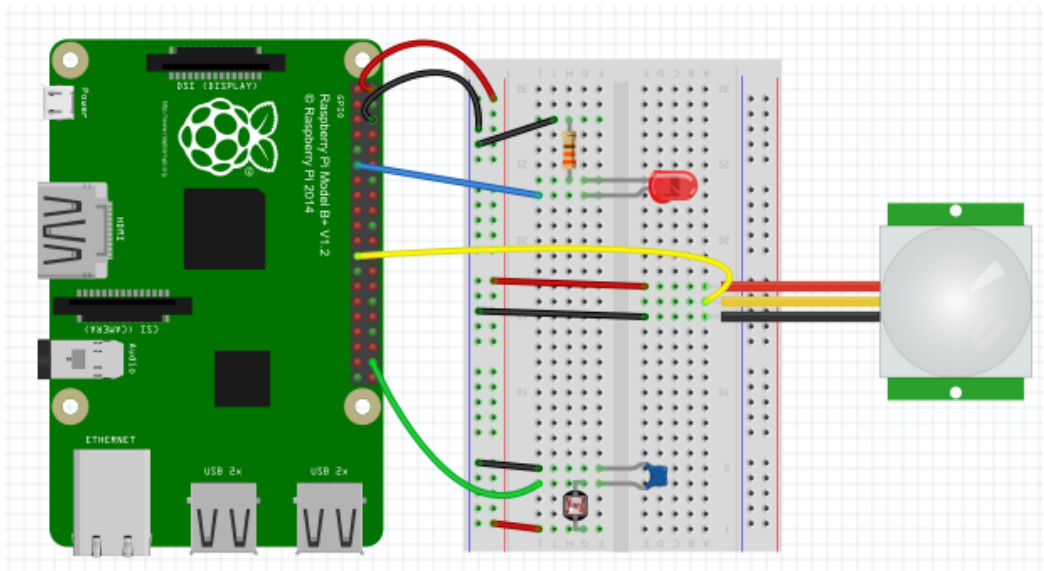
Note:

This technique only works with sensors that act like resistors. It cannot be used with sensors that have a pure analog output like IR distance sensor.

It's not nearly as precise as an ADC and it's a little flakey (since it depends on the Pi timing itself which can vary based on how 'busy' the computer is)

Procedure:

Connect the LDR and 1uf capacitor as in the following picture:



Then run code as you did in the first code (in step #6):

```

1  import time
2  import RPi.GPIO as GPIO
3  import os
4  DEBUG = 1
5
6  GPIO.setmode(GPIO.BOARD)
7  GPIO.setup(11, GPIO.OUT)
8
9  def RCtime (RCpin):
10     reading = 0
11     GPIO.setup(RCpin, GPIO.OUT)
12     GPIO.output(RCpin, GPIO.LOW)
13     time.sleep(0.1)
14
15     GPIO.setup(RCpin, GPIO.IN)
16     # This takes about 1 millisecond per loop cycle
17     while (GPIO.input(RCpin) == GPIO.LOW):
18         reading += 1
19     return reading
20
21 def motion (mpin) :
22     GPIO.setup(mpin, GPIO.IN)
23     motionstate = False
24     motionstate = GPIO.input(mpin)
25     return motionstate
26
27
28 while True:
29     if (motion(23) == True and RCtime(38)>20):
30         # print " Motion Detected " +"Light Off "
31         GPIO.output(11,True)
32         time.sleep(2)
33     elif motion(23) == True and RCtime(38)<20:
34         # print " Motion Detected "+"Light On"
35         GPIO.output(11,False)
36
37     elif motion(23) == False and RCtime(38)>20:
38         # print " There is No Motion "+"Light Off"
39         GPIO.output(11,False)
40
41     time.sleep(0.1)
42

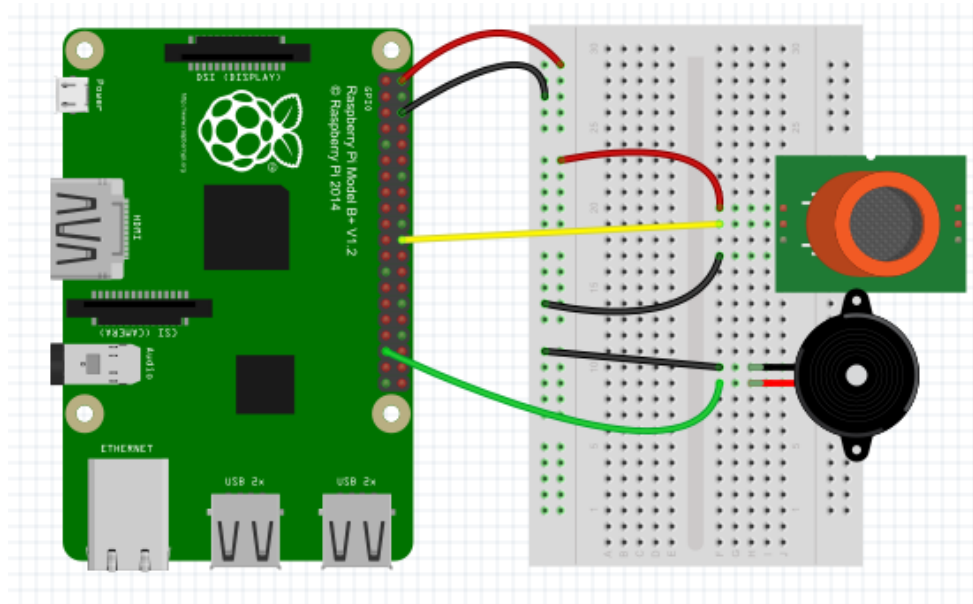
```

Gas Sensor

The second program will be to run a gas sensor, the object of this program is turn on a fan if the gas sensor detect gas.

Procedure:

To do that, connect the components as following:

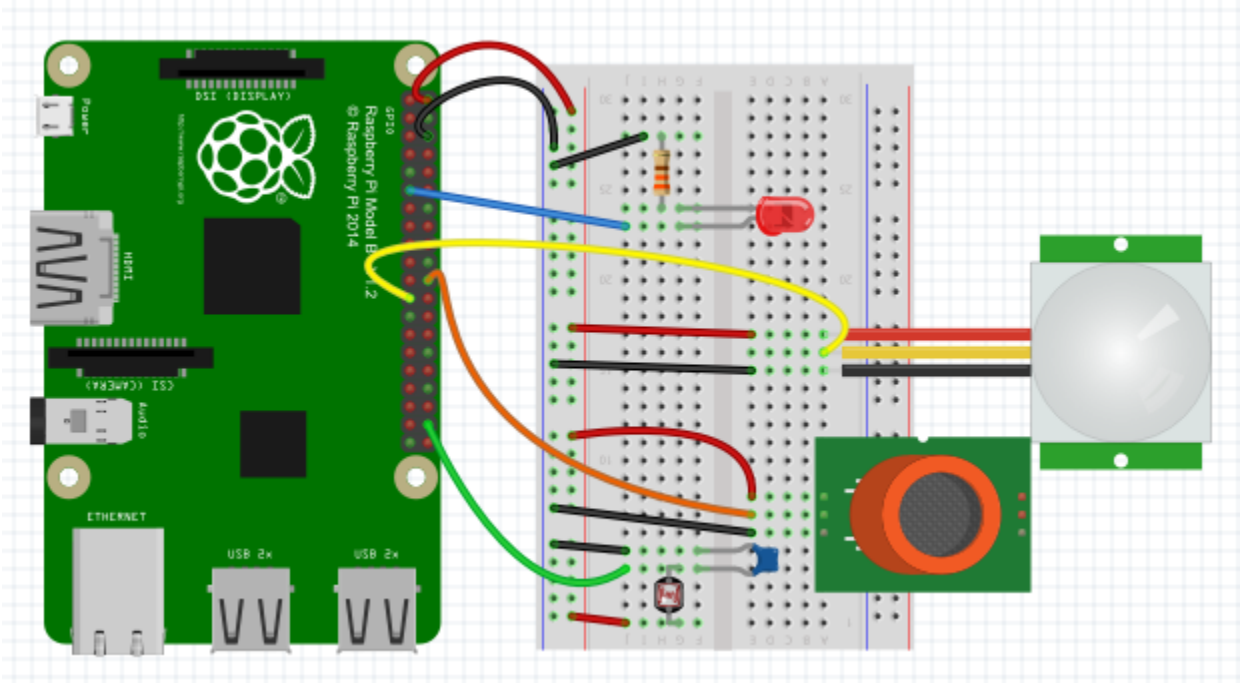


Complete the following code and run it as you did with the first code (in step #6):

```
1 import time
2 import RPi.GPIO as GPIO
3
4
5 GPIO.setmode(GPIO.BOARD)
6 GPIO.setup( , GPIO.OUT)
7
8 def GasDet (Gaspin) :
9     GPIO.setup(Gaspin, GPIO.IN)
10    Gasstate = False
11    Gasstate = GPIO.input (Gaspin)
12    return Gasstate
13
14 while True:
15
16     time.sleep(1)
17
```

Running the two programs together:

Now we will build the two circuits together and run the two programs. Just connect the components on the same pins as before. The full circuit should be as following:



In order to run two programs (or more) you should type the name of the program that you wrote and saved using the command “sudo” and “&” as following:

```
Sudo python motion.py&
```

```
Sudo python gas.py&
```

To stop any program you should write the command

```
Sudo kill "number of program"
```

To stop all the programs write this command

```
Sudo killall sudo python programName.py
```