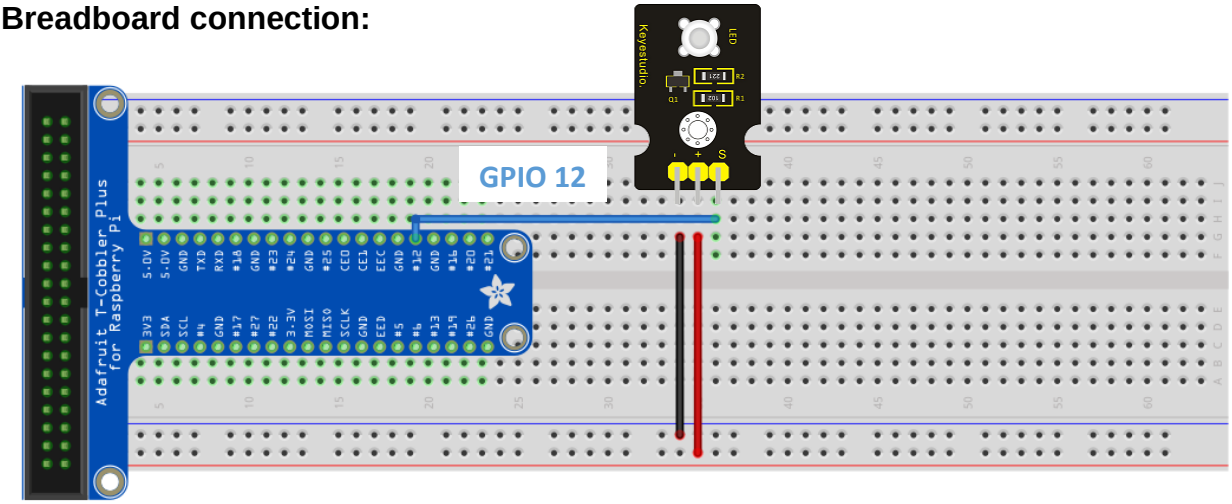
Keyestudio Sensor Kit (Python Version)

1. White LED Light

White LED light is a simple LED light that emits white light. It uses DC 3.3 – 5V as its working voltage.

Breadboard connection:



Python code:

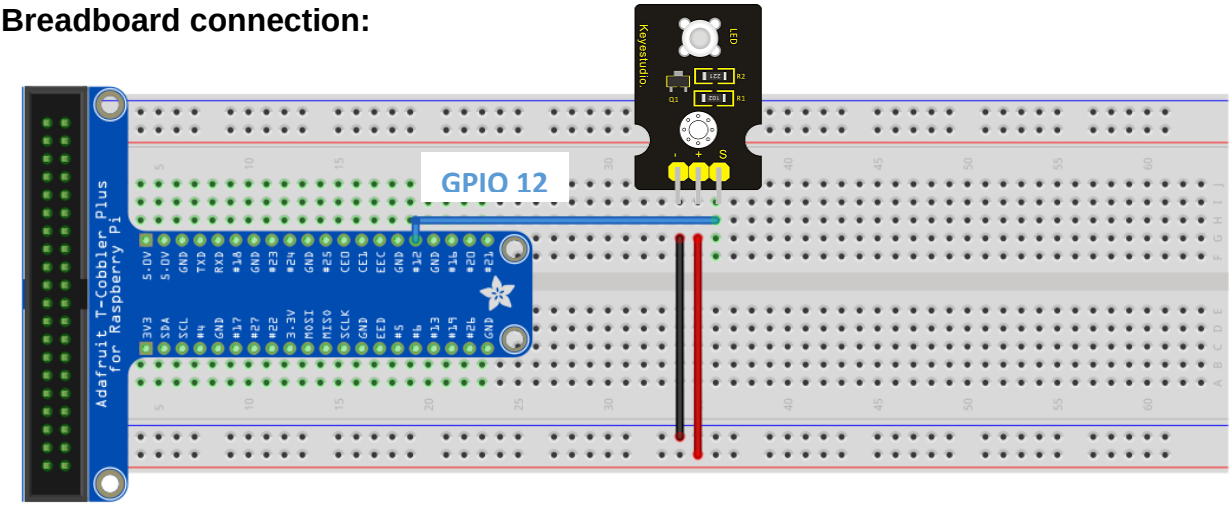
```
#Code to turn on KeyeStudio White LED

#KeyeStudio White LED = Ks0016
#Voltage = 3.3V - 5.0V
#LED color = White

#Imports LED from GPIOzero object
from gpiozero import LED
#Imports sleep from time library
from time import sleep

if __name__ == '__main__':
    Led_White = LED(12) #Sets Led_White as ouput signal on pin 12
```

```
    try:
        while(True): #Creates an infinite time loop
            Led_White.on() #outputs an on signal to pin 12
            print("White LED is on") #tells the user the led is on
            sleep(4) #forces the device to stay in this setting
            Led_White.off() #quits outputting a signal to pin 12
            print("White LED is off") #tells the user the LED is off
            sleep(2) #keeps the device in this state for 2 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

2. Red LED Light

Red LED light is a simple LED light that emits red light. It uses DC 3.3 – 5V as its working voltage.

Breadboard connection:



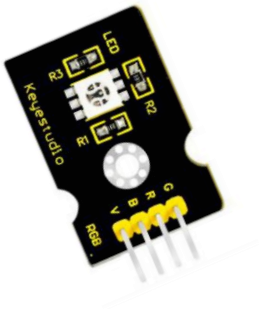
Python Code:

```
#Code to turn on KeyeStudio Red LED

#KeyeStudio Red LED
#Voltage = 3.3V - 5.0V
```

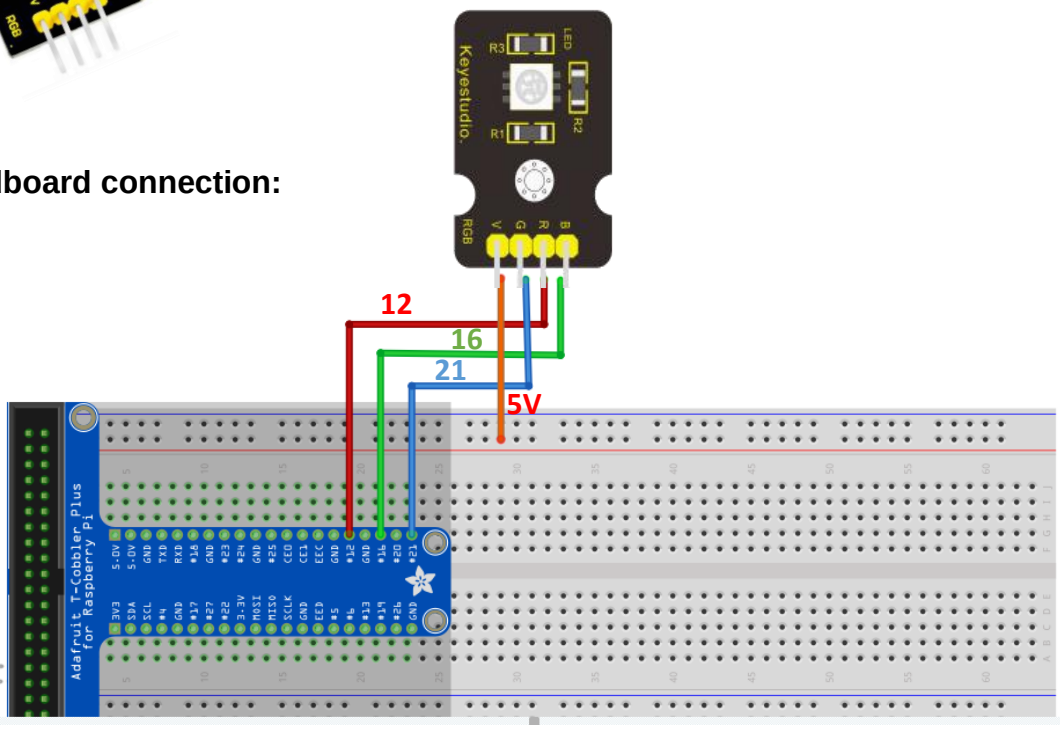
```
#LED color = Red


#Imports LED from GPIOzero object
from gpiozero import LED
#Imports sleep from time library
from time import sleep

if __name__ == '__main__':
    Led_Red = LED(12) #Sets Led_Red as ouput signal on pin 12
    try:
        while(True): #Creates an infinite time loop
            Led_Red.on() #outputs an on signal to pin 12
            print("White LED is on") #tells the user the led is on
            sleep(4) #forces the device to stay in this setting
            Led_Red.off() #quits outputting a signal to pin 12
            print("White LED is off") #tells the user the LED is off
            sleep(2) #keeps the device in this state for 2 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

3. RGB LED

RGB LED is a device consisting of 3 different LEDs (Red, Green and Blue). It uses DC 5V as its working voltage.



Breadboard connection:



Python Code:

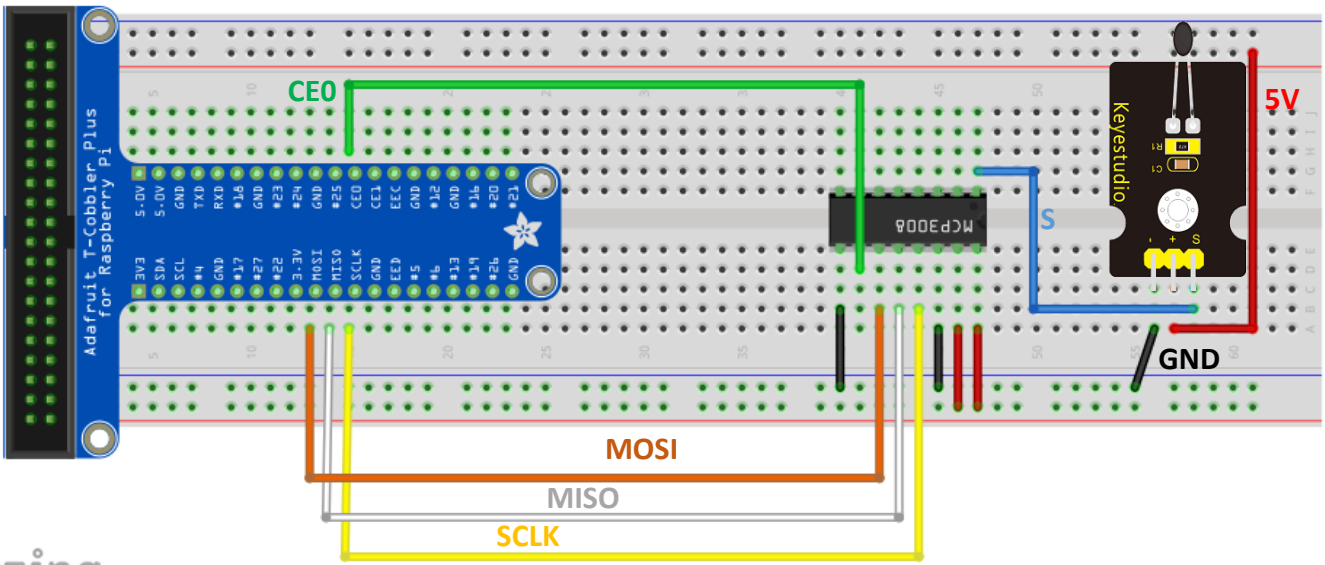
```
#Code to use the KeyeStudio RGB LED

#KeyeStudio RGB LED = Ks0032
#Primary Colors = Red, Green, Blue
#Voltage = 5V
#Input = Digital, Active High

#Imports RGBLED from the GPIOzero library
from gpiozero import RGBLED
#Imports sleep from the time library
from time import sleep

if __name__ == '__main__':
    #Selects the GPIO Pins for each LED
    led = RGBLED(12, 16, 21)
    try:
        while(True): #creates an infinite loop
            led.color = (0,1,1) #sets outputs as red on
            print("Red LED is on") #prints what led is on
            sleep(3) #forces the device to sleep for 3 seconds
            led.color = (1,0,1) #sets ouput as green on, turns red off
            print("Green LED is on") #prints what led is on
            sleep(3) #forces the device to sleep for 3 seconds
            led.color = (1,1,0) #sets output as blue on, turns green off
            print("Blue LED is on") #prints what led is on
            sleep(3) #forces the device to sleep for 3 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

4. Analog temperature

Analog temperature is a simple sensor which can sense temperature change in its surrounding. The sensor's operating voltage is 5V.

Since Raspberry Pi can't directly convert analog signal to digital signal, we need to use an IC chip called MCP3008 which will help to convert the analog signal to digital signal.

Breadboard connection:



Python Code:

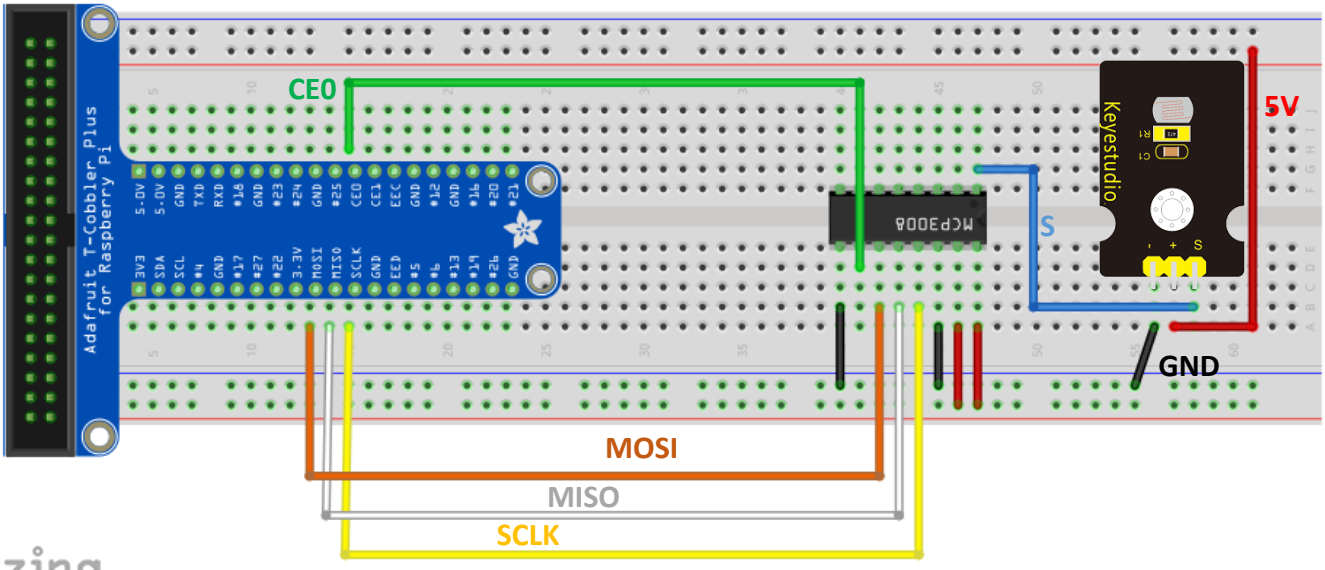
```
#Code for KeyStudio Analog Temperature Sensor with MCP3008 ADC Chip

#KeyStudio Analog Temperature Sensor = KY-013 Sensor
#Operating Voltage = 5 V
#Temperature measurement range = -55C to 125C / -67F to 257F

from time import sleep #import sleep from time library to be able to add
breaks
from gpiozero import MCP3008 #import MCP3008(the ADC converter chip) from the
gpiozero library

deg = chr(176)+'F' #for adding the degree symbol and F to the output
```

```
#sets up the channel number and SPI chip select device. the device is default
0.
tmp = MCP3008(channel=0, device=0) #define the object tmp for temperature
measurements

try:
    while True: #starts loop
        temerature = 0 #declare the temperature variable as zero
        temperature = ((180/1023)*tmp.raw_value)-55 #calculate the
temperature from the raw voltage value
        tempF = 0 #declare the tempF variable as zero
        tempF = (1.8*(temperature))+32 #convert into fahrenheit and store as
tempF

        print('{:.1F}' .format(tempF), deg, 10 * ' ') #print the temperature
in fahrenheit and shortens it to one decimal
        sleep(0.1) #force a sleep for 0.1 seconds to slow down the printing
so we can see it
except KeyboardInterrupt: #use control C to stop the outputs
    print ("All Done") #print a statement to show the interrupt went through
```

5. Photocell sensor

Photocell sensor is a common electronic sensor which has features of high sensitivity, quick response, and so on. Since it is a semiconductor, it is highly effective and easier to operate. The sensor's operating voltage is 5V.

Breadboard connection:

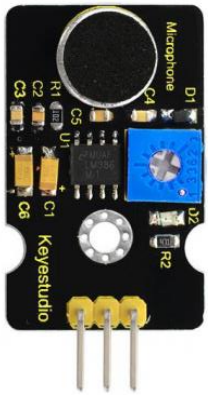


Python code:

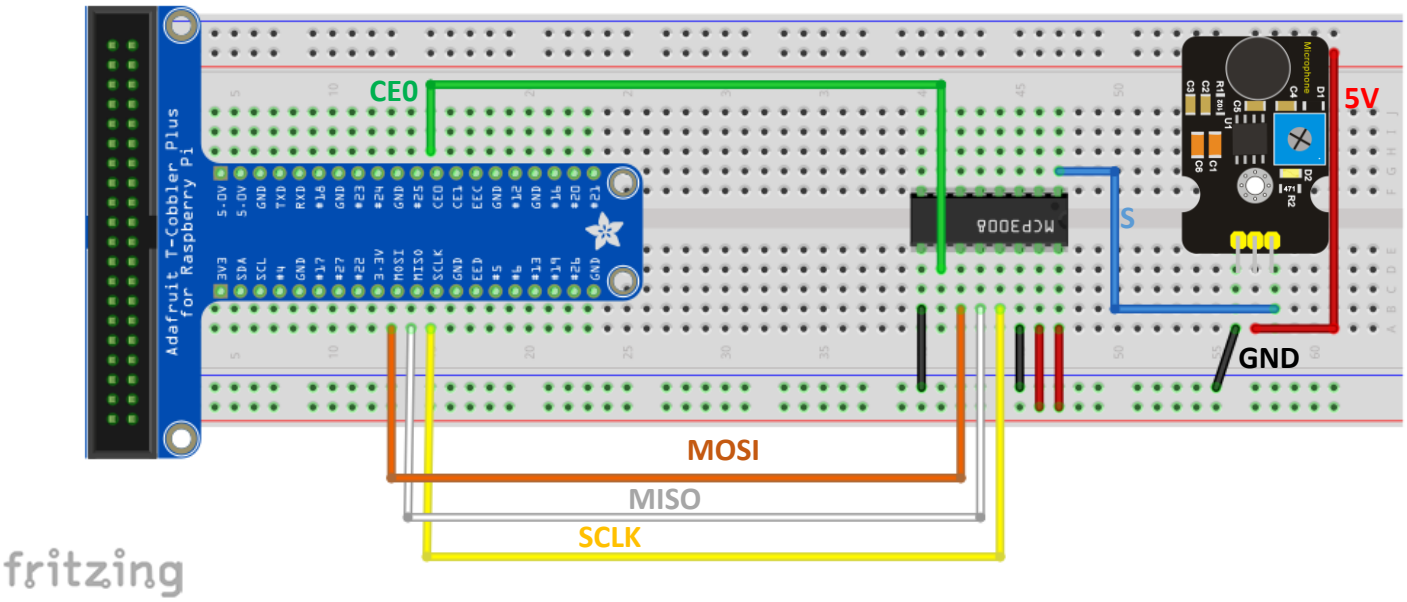
```
#Import the MCP3008 module from the gpiozero library
from gpiozero import MCP3008
#Import the sleep function from time library
from time import sleep
#Setting up the channel
analog_input = MCP3008(channel=0)
if __name__ == '__main__':
    try:
        while True:
            Ref = 4500 #Reference value of the photocell resistor
            MaxValue = 1023
            #Reading the analog value
            reading = analog_input.raw_value
            #converting the analog reading to the voltage
            voltage = (reading * Ref/MaxValue)
            #Finding the value of resistance of the photoresistor
            Resistance = 10000*((3.3/voltage)-1)
            #Displaying the required information
            print("Resistance={:.2f}".format(Resistance))
            #pausing the program for 1 second
            sleep(1)
    except KeyboardInterrupt:
            print('Exit')
```

6. Analog sound

Analog sound sensor is mainly used to detect the loudness in the environment. It uses DC 3.3V – 5V as its operating voltage. We need to use MCP3008 to find the analog sound.



Breadboard connection:



Python code:

```
'''
    For finding the relationship between ADC and Decibels,
    we used the following website as a reference:
    https://circuitdigest.com/microcontroller-projects/arduino-sound-level-
measurement
'''
#Import the MCP3008 module from the gpiozero library
from gpiozero import MCP3008
#Import the sleep function from time library
from time import sleep
#Setting up the channel
analog_input = MCP3008(channel=0)
reading = 0
if __name__ == '__main__':
    try:
        while True:
            VRef = 3.3
            MaxValue = 1023
            #Reading the analog value
            reading = analog_input.raw_value
            #converting the analog reading to the voltage
            voltage = reading * (VRef/MaxValue)
            #Converting ADC to dB using linear regression.
            #Please refer to the reference website on finding out how the
              below formula is derived.
            dB = (voltage + 83.2073) / 11.003
            #Displaying the required information
            print("Raw value:{:.2f}\tSound in
Decibels:{:.2f}".format(voltage, dB))
            #pausing the program for 1 second
```

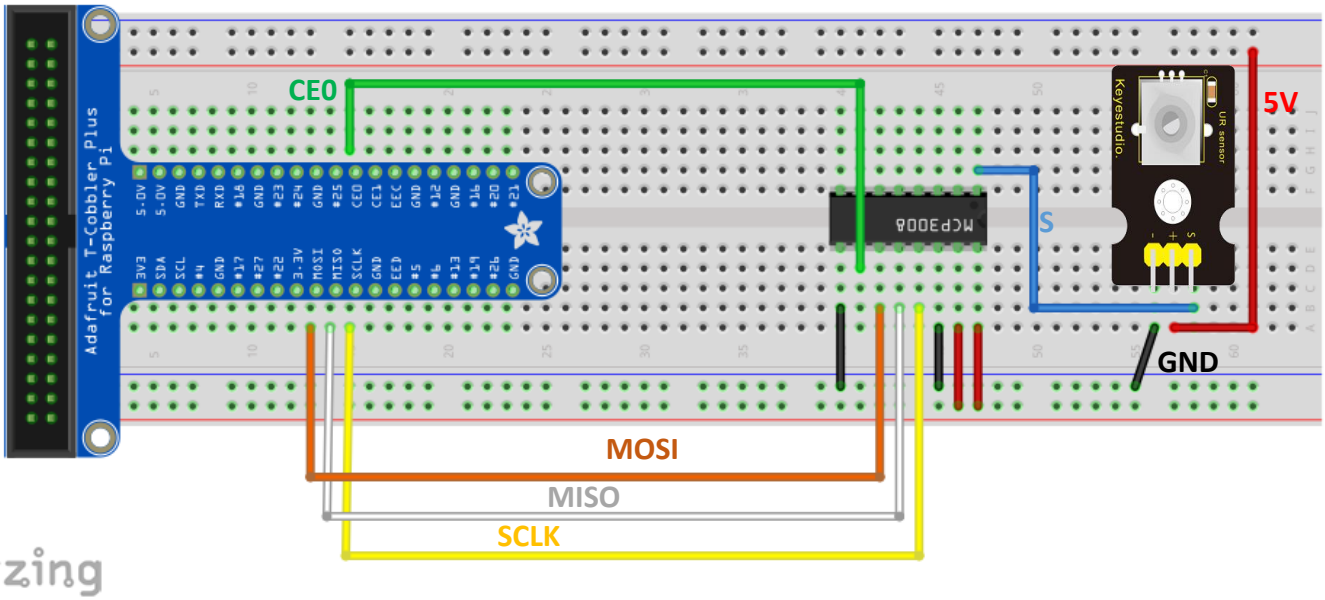
```
            sleep(1)
    except KeyboardInterrupt:
            print('Exit')
```

7. Analog rotation sensor

Analog rotation sensor is a potentiometer based rotator which helps to change the analog value by rotation. Its operating voltage is DC 3.3V – 5V.

We need to use MCP3008 to use the analog rotation sensor.

Breadboard connection:



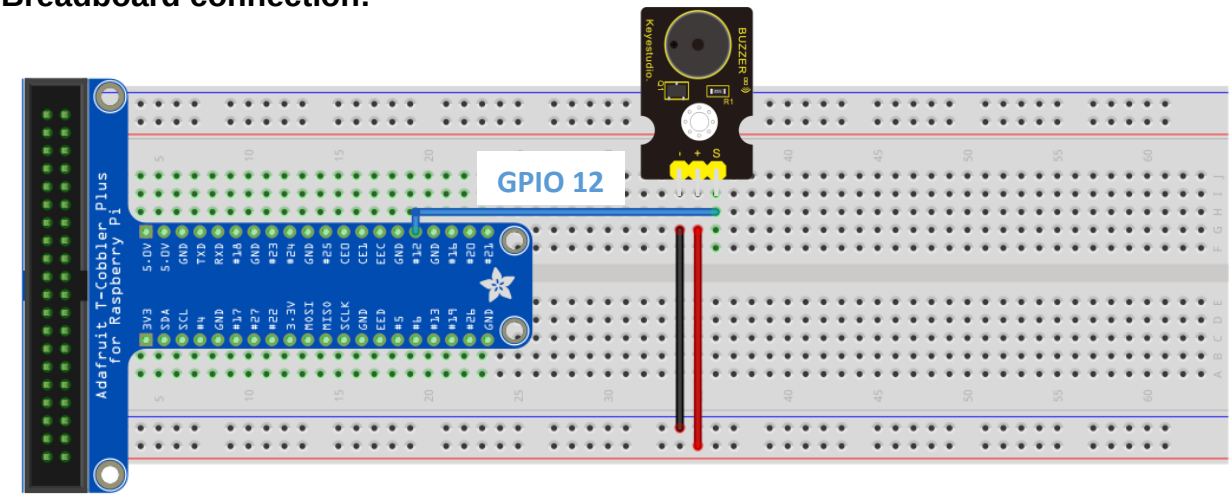
Python code:

```
#Module to find the Analog rotation ranging from 0-1023 using ADC
from gpiozero import MCP3008
#Import the sleep function from time library
from time import sleep
#Setting up the channel
analog_input = MCP3008(channel=0)
if __name__ == '__main__':
    try:
        while True:
            #Reading the analog value
            reading = analog_input.raw_value
            #Displaying the required information
            print("Analog Value:{0}".format(reading))
            #pausing the program for 1 second
            sleep(1)
    except KeyboardInterrupt:
            print('Exit')
```

8. Digital buzzer

Digital buzzer is a simple sound making module which can make sounds of different frequencies. They can be found in alarm devices, computers, timers, etc. Its operating voltage is 3.3V – 5V.

Breadboard connection:



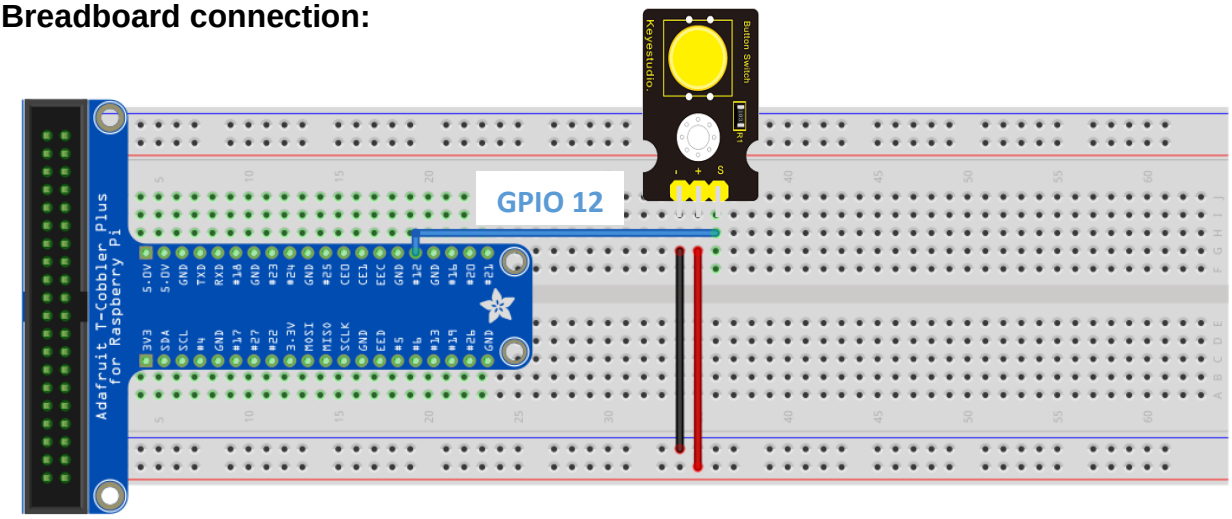
Python code:

```
#Code for KeyStudio Active Buzzer
#This code turns on the Active Buzzer after a second break for
#one second and then turns it off for 3 more seconds.

#KeyeStudio Buzzer Beeper = Ks0018
#Operating Voltage = 3.3V - 5.0V
#Interface Type: Digital

#Imports Buzzer from GPIOzero library
from gpiozero import Buzzer
#Imports sleep from time library
from time import sleep

if __name__ == '__main__':
    #sets Buzz_Beeper as buzzer object on pin 12
    Buzz_Beeper = Buzzer(12)
    try:
        while(True): #infinite loop
            sleep(1) #forces the device to sleep for 1 second
            Buzz_Beeper.on() #turns on the buzzer
            print("Buzzer is on.") #tells user the buzzer is on
            sleep(1) #forces the device to sleep for 1 second
            Buzz_Beeper.off() #turns off the buzzer
            print("Buzzer is off.") #tells user the buzzer is off
            sleep(3) #forces the device to sleep for 3 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

9. Digital Push button

Digital pushbutton is used to connect two points in a circuit when the button is pushed. When the pushbuttons aren't pushed, the circuits are disconnected. Its operating voltage is 3.3V – 5V.

Breadboard connection:



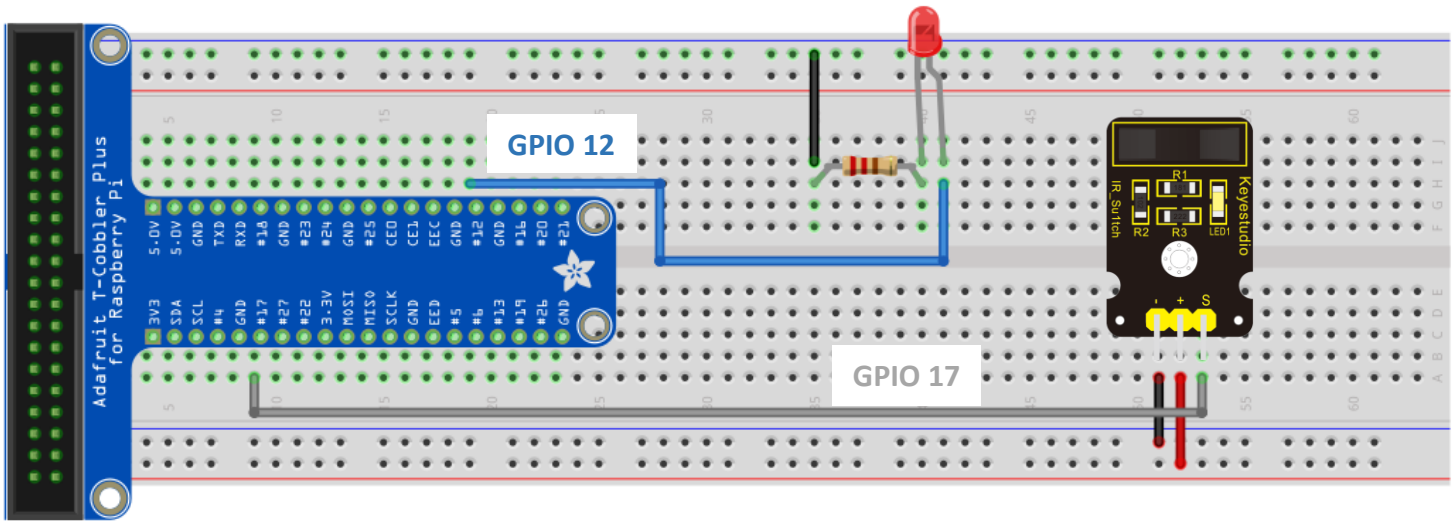
Python code:

```
#Code to read from KeyeStudio Push Button

#KeyeStudio Digital Push Button = Ks0029
#Interface = Digital
#Supply Voltage = 3.3V - 5V
#Simple and Easy to Operate

#Imports Button from GPIOzero library
from gpiozero import Button
#Imports sleep from the time library
from time import sleep

#this code reads for how long a capacitive touch button has been touched
if __name__ == "__main__":
    button = Button(12) #creates variable from button object on pin 12
    pn = 1 #sets a variables value as 1
    try:
        while(True): #creates an infinite loop
            button.wait_for_press() #waits for the button to be pressed
            #Will print out how many times the button has been pressed
            print("The button has been pressed this many times: ", pn)
            pn = pn + 1 #increments pn by 1 to tell how many times the
                        #the button has been pressed
            sleep(1) #waits for 1 second so no switch bounce
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

10. Photo Interrupter

Photo interrupter is a photo sensor that works by detecting light blockage when an object comes in between the two sides of the interrupter. It is also called an infrared switch, because it acts as an optical switch. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



Python code:

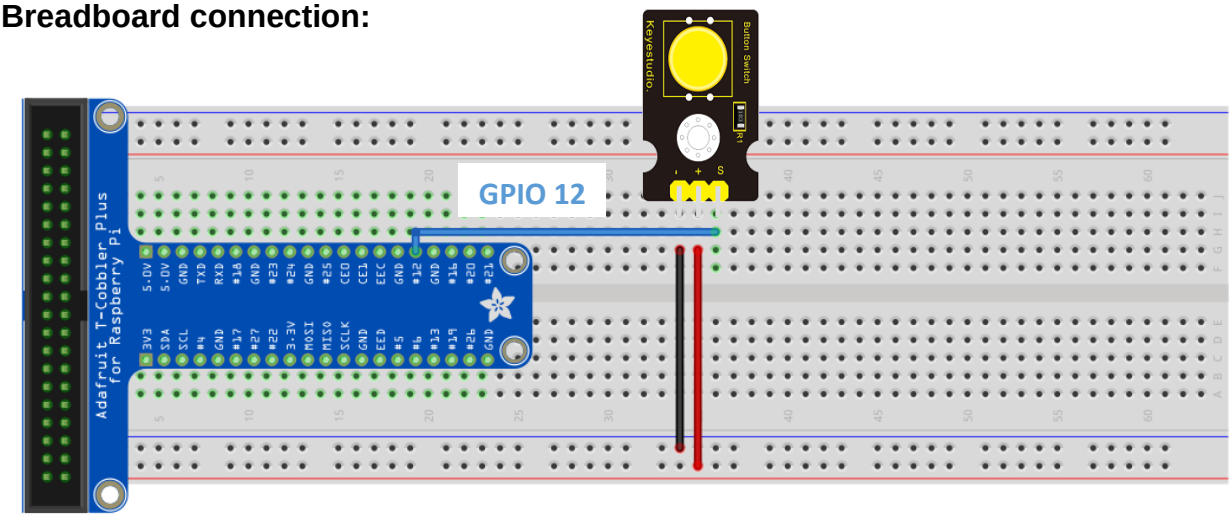
```
#Photo Interrupt module
#Import LED Module from gpiozero library
from gpiozero import LED
#A Module for sleeping the Pi
from time import sleep
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import DigitalInputDevice
if __name__ == '__main__':
    #Activating the function that controls the PhotoInterrupt Sensor
connected to pin 17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
    Led_Red = LED(12)
    try:
        while True:
```

```
            if(sensor.wait_for_active()):
                print("Signal Detected")
                Led_Red.on()
            if(sensor.wait_for_inactive()):
                Led_Red.off()
                print("Not detected")
    except KeyboardInterrupt:
           Led_Red.off()
           print("Cleaning up")
```

11. Capacitive Touch

Capacitive touch is an electrical sensor that is used as an alternative to push buttons. It is used to connect two points in a circuit when the yellow surface is touched. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



Python code:

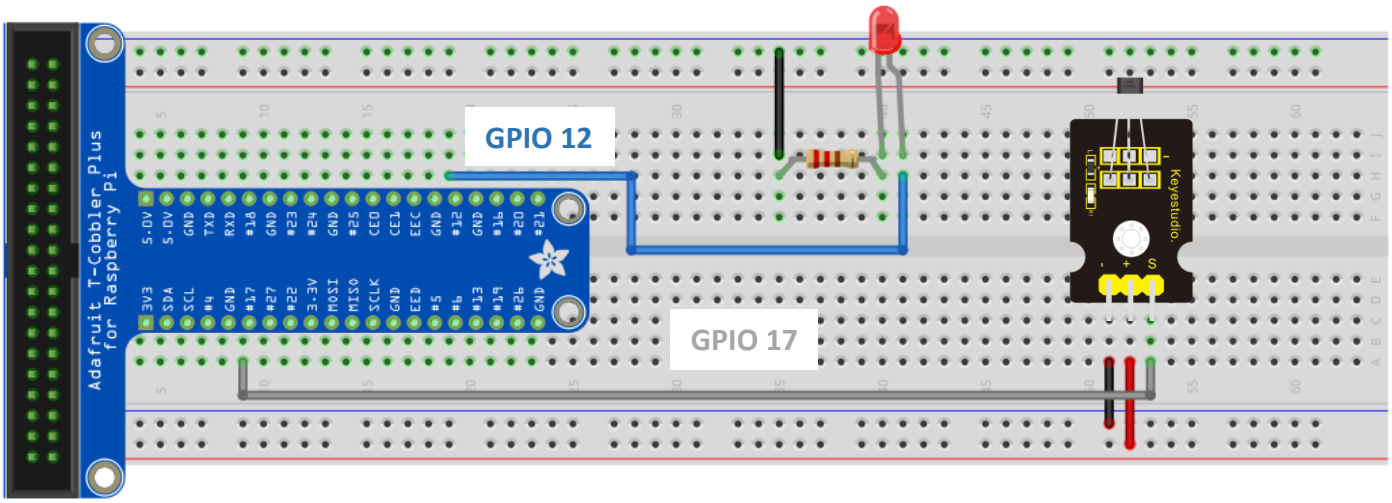
```
#Code for KeyeStudio Capactive Button
#This module demonstrates the capacitve touch button
#and tells the user how long the button has been held.

#KeyeStudio Capacitive Touch Sensor = Ks0031
```

```
#Operating Voltage = 3.3V to 5.0V
#Interface Type =  Digital

#imports Button from GPIOzero library
from gpiozero import Button
#imports sleep from Time Library to add breaks
from time import sleep

#this code reads for how long a capacitive touch button has been touched
if __name__ == "__main__":
    button = Button(12) #declares button as an object to read from pin 12
    pn = 1 #sets variable value as 1
    try:
        while(True): #creates an infinite loop
            button.wait_for_release() #stops the device proceeding until
button has been released
            print("The button has been touched for", pn, "seconds") #prints
the message in ""
            pn = pn + 1 #increments the variable for how long the button has
been pressed
            sleep(1) #forces the device to sleep for 1 second so a button
release can be read
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done\
```

12. Hall Magnetic Sensor

Hall magnetic sensor is a simple electric sensor which is used to detect the magnetic materials that are 3cm near to it. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



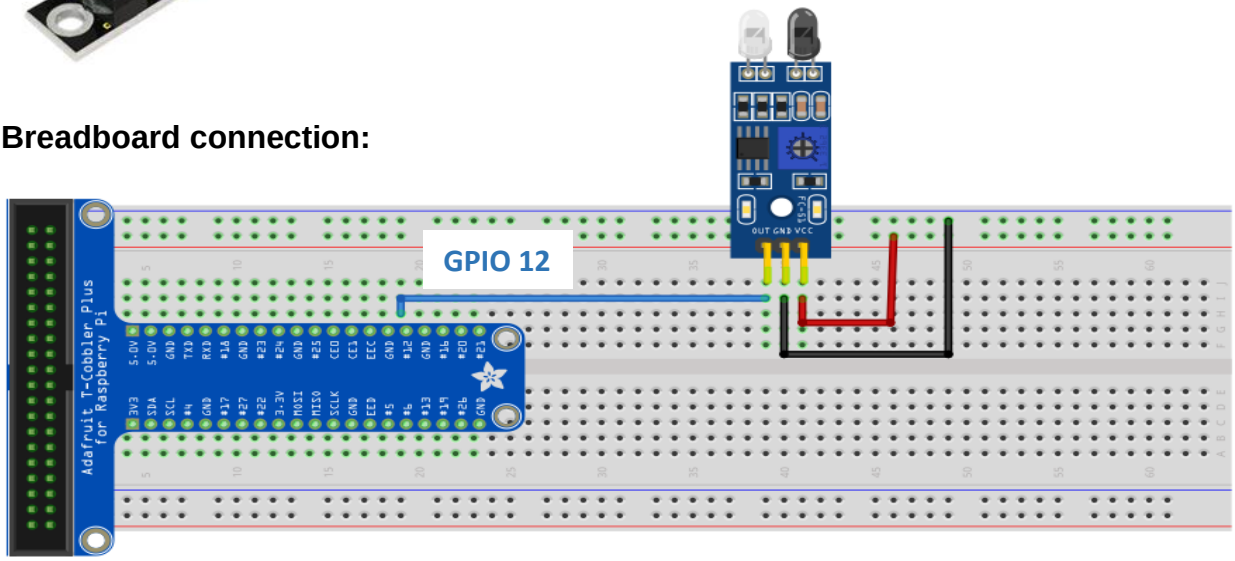
Python code:

```
#Magnetic Hall Module
#Import LED Module from gpiozero library
from gpiozero import LED
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import DigitalInputDevice
#A Module for sleeping the Pi
from time import sleep
if __name__ == '__main__':
    #Activating the function that controls the Sensor connected to pin 17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
    Led_Red = LED(12)
    try:
        while True:
        if(sensor.wait_for_active()):
                print("Magnet Detected")
                Led_Red.on()
        If(sensor.wait_for_inactive()):
                Led_Red.toggle()
        print("Not detected")
    except KeyboardInterrupt:
           print("Cleaning up")
```

13. Line tracking sensor

Line tracking sensor is a simple electric sensor that can detect white lines and black lines using the infrared mechanism. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



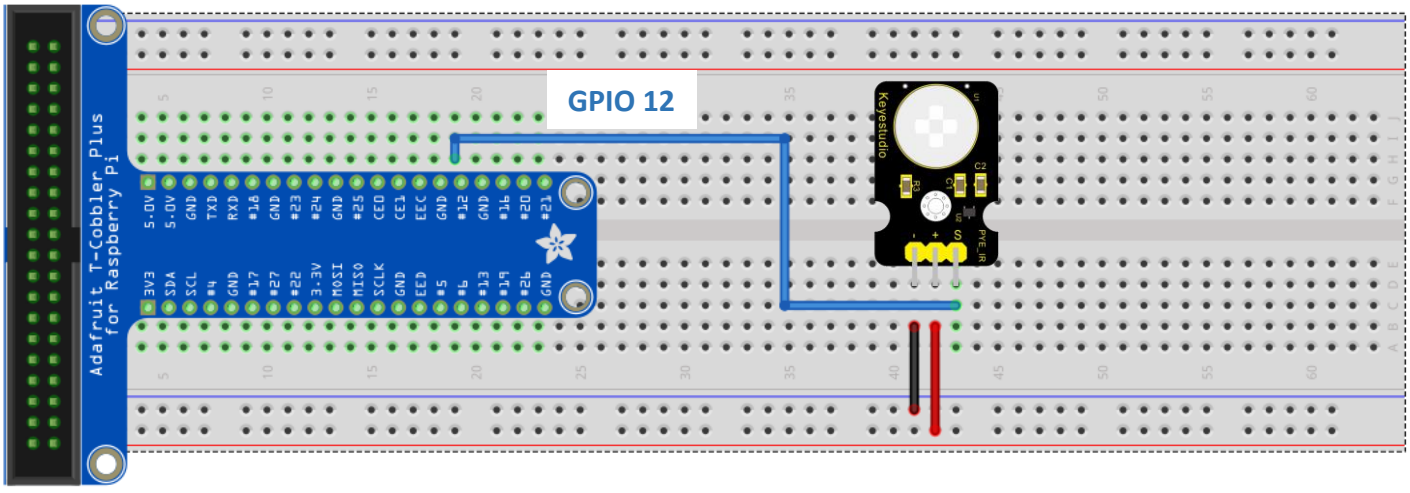
Python code:

```
#Code for KeyeStudio Line Tracking Sensor

#KeyeStudio Line Tracking Sensor = Ks0050
#Operating Voltage = +5V
#Operating Current = <10mA
#Operating Temperature Range = 0Celcius - 50Celcius

#Imports LineSensor from GPIOzero library
from gpiozero import LineSensor
#Imports pause from the signal library
from signal import pause

#when a line is detected, the led turns on, else it's dim
if __name__ == '__main__':
    #Creates sensor object as an input from pin 12
    sensor = LineSensor(12)
    #if line is detected, prints Line Detected and waits
    sensor.when_line = lambda: print('Line Detected')
    #if line isn't detected, prints No Line Detected and waits
    sensor.when_no_line = lambda: print('No Line Detected')
```

14. PIR Motion Sensor

PIR (Pyroelectric Infrared) motion sensor is an electronic sensor that can detect infrared signals from moving objects. It is one of the most commonly used sensors. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



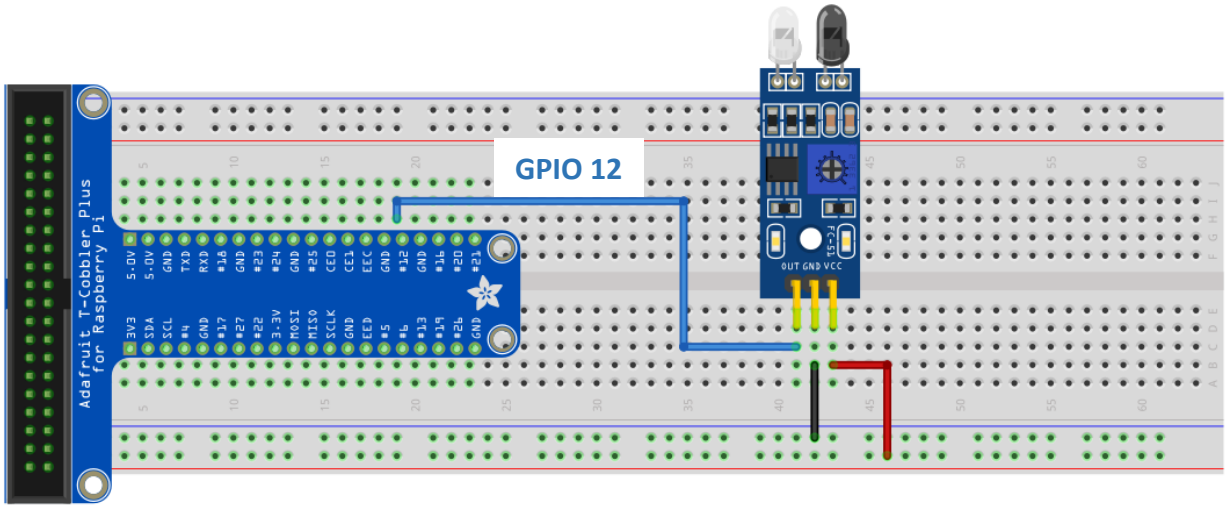
Python code:

```
#Code for KeyeStudio Motion Sensor

#KeyeStudio PIR Motion Sensor = Ks0052
#Input Voltage: 3.3V - 18V
#Working Current = 15 uA
#Working Temperature = -20C - +85C
#Detection Distance = 3-4 meters

#Imports the MotionSensor from the GPIOzero library
from gpiozero import MotionSensor
#Imports sleep from the time library
from time import sleep

if __name__ == "__main__":
    try:
        while(True): #Creates an infinite time loop
            pir = MotionSensor(12) #Sets pir as variable from motionsensor
                                    #to be read as an input from pin 12
            pir.wait_for_motion() #Waits for mo tion to be detected from
sensor
            print("Motion Detected") #Lets the user know motion was detected
            exit() #exits the program, must restart every time motion
detected
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

15. Infrared Obstacle avoidance

Infrared obstacle avoidance is an electronic sensor equipped with distance adjustment function which detects obstacle by transmitting infrared light. Its working voltage is DC 3.3 – 5V.

Breadboard connection:



Python Code:

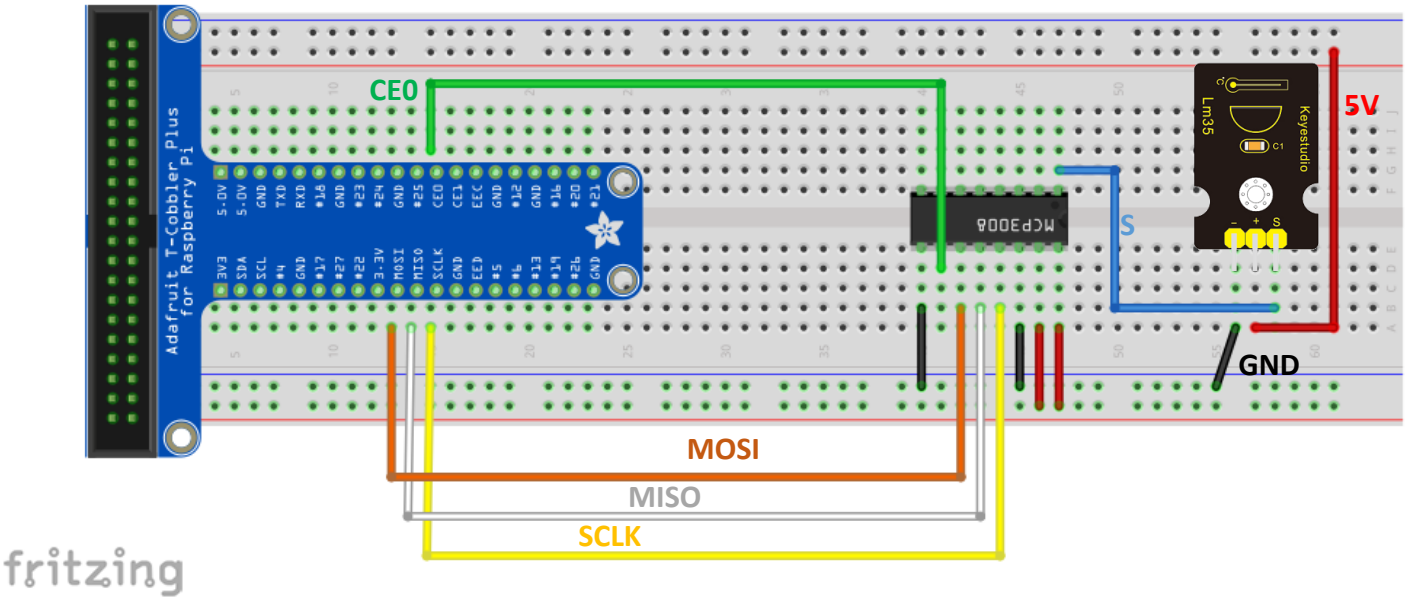
```
#Code for KeyeStudio Infrared Obstacle Avoidance Sensor

#KeyeStudio Infrared Obstacle Avoidance Sensor = Ks0051
#Operating Voltage = 3.3V - 5V
#Operating Current = >20mA
#Operating Temperature Range = -10Celcius - +50Celcius
#Detection Distance = 2-40cm
#IO Interface = 4 pin

#Imports LED and LineSensor from GPIOzero library
from gpiozero import LED, LineSensor
#Imports pause from the signal library
from signal import pause
#Imports GPIo from the RPi.GPIO library
import RPi.GPIO as GPIO
#Imports sleep from time library
from time import sleep

#Sets mode of the GPIO as BCM
GPIO.setmode(GPIO.BCM)
#Sets the internal resistor type of the device as Pull Down
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
#Sets the LED output as pin 21
engageSensor = LED(21)
#Turns on the infrared sensor
engageSensor.on()

try:
    while(True): #Starts an infinite loop
        output = GPIO.input(20) #Creates object with input from pin 20
        if(output != 1): #if statement checking whether the variable in
                         #output has changed from 1 to 0
            print("Object Ahead!") #prints message saying infrared object
detected
            sleep(1)
```

```
except KeyboardInterrupt: #allows user to exit the program
    print("All Done") #tells user the program is all done
```

16. LM35 Linear Temperature

LM35 linear temperature is a sensor that can be used to detect the ambient air temperature. It is the most commonly used temperature sensor. It can measure temperature up to 100 degree Celsius. We need to use MCP3008 to find the temperature.

Breadboard connection:



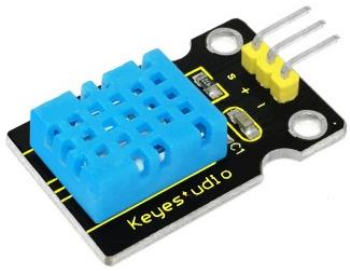
Python code:

```
#Import the MCP3008 module from the gpiozero library
from gpiozero import MCP3008
#Import the sleep function from time library
from time import sleep
#Setting up the channel
analog_input = MCP3008(channel=0)
```

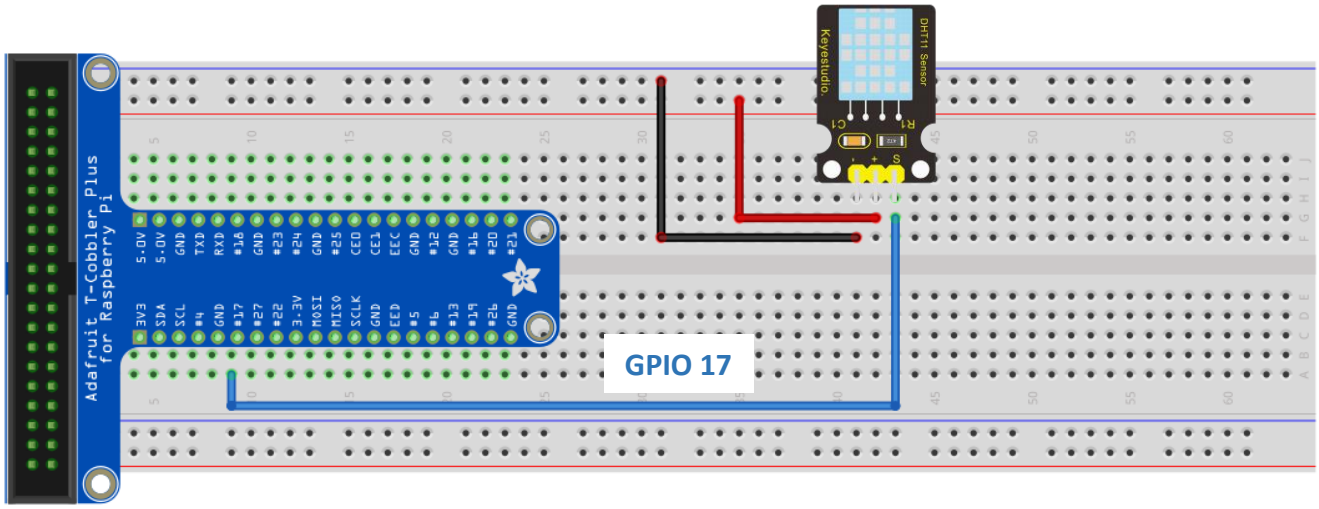
```
if __name__ == '__main__':
    try:
        while True:
            VRef = 3.3
            MaxValue = 1023
            #Reading the analog value
            reading = analog_input.raw_value
            #converting the analog reading to the voltage
            voltage = (reading * VRef/MaxValue)
            #converting the voltage to temperature to degree celsius
            temp_c = (voltage)/(10/1000)
            #converting the temperature to degree fahrenheit
            temp_f = (temp_c * 9.0) / 5.0 + 32
            #Displaying the required information
            print("Temp C={:.2f}\tTemp F={:.2f}".format(temp_c, temp_f))
            #pausing the program for 1 second
            sleep(1)
    except KeyboardInterrupt:
            print('Exit')
```

17. DHT11 Temperature and Humidity Sensor

DHT11 Temperature and Humidity Sensor refers to digital sensor that measures the temperature and humidity of the surrounding. Its working voltage is DC 5V.



Breadboard connection:



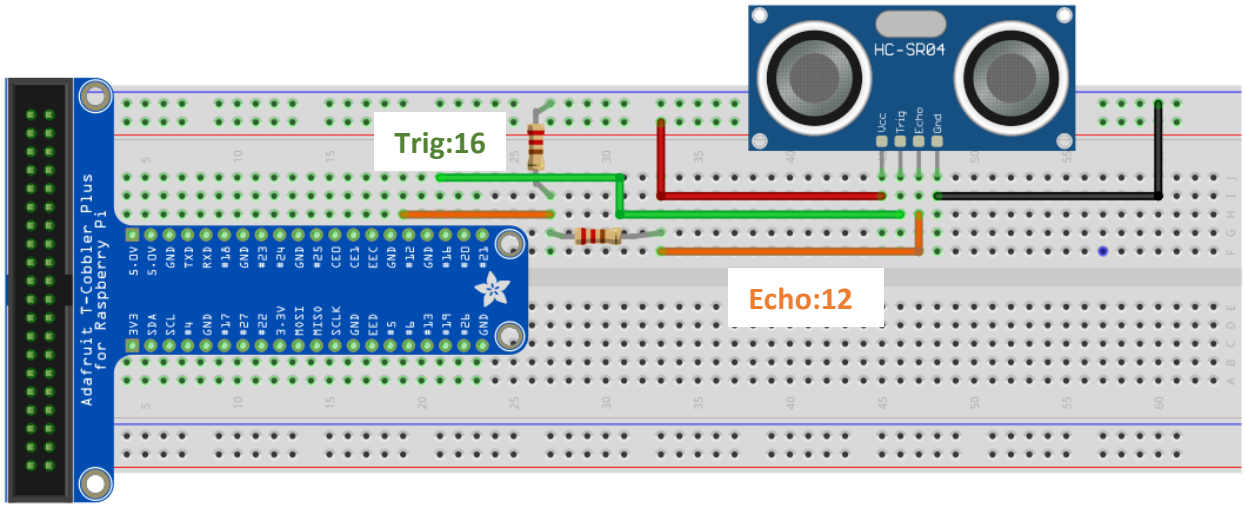
Python code:

```
#Code for KeyeStudio DHT11 Temperature and Humidity Sensor

#KeyeStudio DHT11 Temperature and Humidity Sensor = Ks0034
#Supply voltage = 3.3V
#Temperature Range = 0-50C
#Humidity Range = 20-90%
#Interface = Digital

from time import sleep #Imports sleep from time library
import Adafruit_DHT #Imports Adafruit_DHT library
import board #Imports board library

if __name__ == '__main__':
    DHT_Pin = 17 #Sets the input value frjom pin 17
    print("Measuring Temperature and Humidity")
    DHT_Sensor = Adafruit_DHT.DHT11 #Create a Temperature and Humidity Sensor
Object
    try:
        while True: #creates an infinite loop
            humidity, temperature = Adafruit_DHT.read(DHT_Sensor,DHT_Pin)
#creates variables from library
            if humidity is not None and temperature is not None: #if
statement testing if values are coming
                                                                 #in from the
sensor itself
                print("Temperature = {0:0.1f}C, Humidity =
{1:0.1f}%".format(temperature,humidity))
                #prints out the results from the sensor if coming in
            else:
                print("Sensor Failure") #prints out that the sensor isn't
getting any values
            sleep(3) #forces the device to sleep for 3 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

18. SR01 Ultrasonic Sensor

SR01 Ultrasonic sensor is a distance sensor capable of capturing Ultrasound reflections. Ultrasound reflections are the high frequency sounds above the audible range by human ears (20 Hz – 20 kHz). Its working voltage is DC 5V.

Breadboard connection:



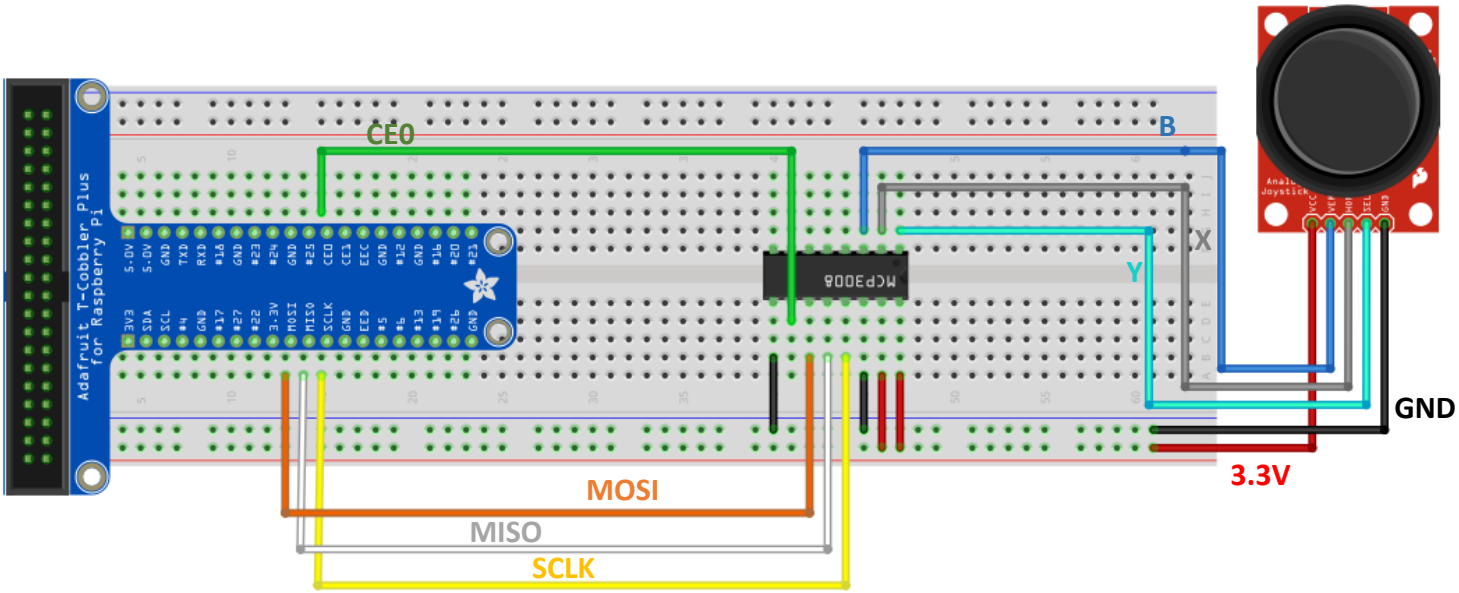
Python code:

```
#Code for the KeyeStudio UltraSound Ranger

#KeyeStudio SR01 UltraSonic Sensor = Ks0206
#Working Voltage = DC 5V
#Working Current = 15mA
#Working Frequency = 40 Hz
#Distance Range = 2centimeters - 5meters

#Imports DistanceSensor from the GPIOzero library
from gpiozero import DistanceSensor
#Imports sleep from the time library
from time import sleep

if __name__ == "__main__":
    sensor = DistanceSensor(echo = 12, trigger = 16) #sets echo input as pin
12 and trigger output as pin 16
    try:
        while(True): #creates an infinite time loop
            print('Distance: ', sensor.distance * 100) #outputs the distance
found
            sleep(1) #forces the device to sleep for 1 second
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

19. Joystick

Joystick sensor is majorly used to control X and Y position of the objects. It is mostly used in robotics. Its working voltage is DC 3.3V - 5V.

Breadboard connection:



Python code:

```
"""
This program may be used to connect a Joystick to the Raspberry Pi in a
script. To keep the
use open, no actions are preformed from inputs made on the joystick. The
program will return
the raw values produced by the joy stick which range from 0-1023 with the
neutral position
being 500-540. The button is state is either 0 or 1 with neutral being 0.
Serial Programmable Interface is used to achieve stable connection between
the ADC Chip and Pi.
One 16-pin ADC chip(MSP3008) may be used to control 4 joysticks
Made by Tyler James UWYO CEAS-ECE. Contact tjames15@uwyo.edu if needed.
```

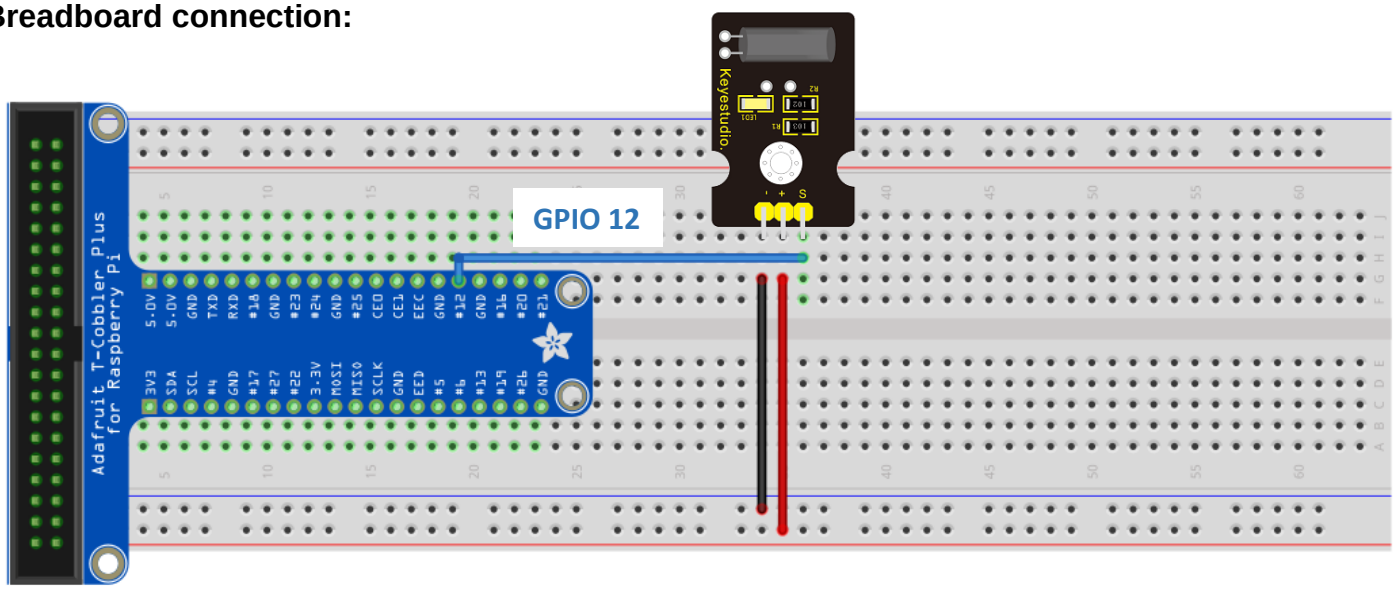
```
"""
#Import sleep from time
from time import sleep
#Import Serial Programmable Interface module from the Adafruit SPI library
import Adafruit_GPIO.SPI as SPI
#Import the MCP3008 module from the Adafruit library
import Adafruit_MCP3008
#Import the GPIO module from RPi.GPIO as io
import RPi.GPIO as io

#A class to instantiate channel objects
class Channel:
    def __init__(self,name,chnNumber,chnRef):
        self.__ChannelName = name #Give the Channel a name for future
referance
        self.__Channel = chnNumber - 1 #Setting the channel number to
computer numbering
        self.__RefValue = chnRef #The reference voltage value
    #Property method to access Channel Number
    @property
    def Number(self):
        return self.__Channel #Returning the Channel number in computer
numbering
    #Property method to access Channel Reference Value
    @property
    def RefValue(self):
        return self.__RefValue #Returning the reference voltage for the ADC
Chip

#Class to Instantiate a ADC Object
class ADC:
    def __init__(self,port,device):
        self.__SPI_Port = port #Where is it connected?
        self.__SPI_Dev = device #What device number is it?
        self.__adc = Adafruit_MCP3008.MCP3008(spi =
SPI.SpiDev(self.__SPI_Port,self.__SPI_Dev)) #Setting up the ADC chip
        self.__MaxRawValue = 1023 #The number of possible decimal values
        self.__Channels = [] #The number of channels

    def ChannelSetup(self,Name,Num,Ref):
        if(Num > 0 and Num <= 8):
            self.__Channels.append(Channel(Name,Num,Ref)) #Setting up the
channel on the ADC
        else:
            print('Channel number is not valid')

    def ReadChannel(self,Num):
        #Check whether the channel exists
        raw = 0 #Instantiating the raw value varible
        for chn in self.__Channels: #Checking the channel
            if(Num - 1) == chn.Number: #Polling the channel number to read
                raw = self.__adc.read_adc(chn.Number) #Raw value will be
digital output code or decimal value
        return raw #Returning value

#Class to make a joystick
class Joystick:
```

```
    def __init__(self,Port,Device,Channel,RefVolt,Button):
        self.__adc = ADC(Port,Device) #Setting up the ADC Chip for use
        self.__ChnY = Channel #Setting the Channel for the Y axis
        self.__ChnX = Channel+1 #Setting the Channel for the X axis
        self.__Y = self.__adc.ChannelSetup('Y',self.__ChnY,RefVolt) #Setting
up Channel Y
        self.__X = self.__adc.ChannelSetup('X',self.__ChnX,RefVolt) #Setting
up Channel X
        self.__Z = Button #Setting the Button pin
        io.setup(self.__Z,io.IN) #Setting button pin to output

    def Position(self):
        rawValueY = self.__adc.ReadChannel(self.__ChnY) #Getting the raw
value from Channel Y
        rawValueX = self.__adc.ReadChannel(self.__ChnX) #Getting the raw
value from Channel X
        rawValueZ = io.input(self.__Z) #Getting the state of the button
        return rawValueY,rawValueX,rawValueZ #Returning the values read


"""
To use the joystick in a script, call
'Joystick(Port,Device,Channel,RefVolt,Button)'
Port is saying where the ADC Chip is connected to the Pi
Device names the Chip
Channel is for the first channel used by the joystick as it uses two
channels, one for the y-axis and one for the x-axis
RefVolt is the referance voltage for the ADC Chip. Set this to the voltage
that is being used to power the joystick
Button is a digital input to which the button on the joystick is connected
to.


if __name__ == '__main__':
    io.setmode(io.BCM)
    joystick = Joystick(0,0,1,3.3,21)


    print('Reading Position')
    try:
        while True:
            rawValueY,rawValueX,rawValueZ = joystick.Position()
            print('Raw Value Y = {:4d} Raw Value X = {:4d} Raw Value Z =
{:1d}'.format(rawValueY,rawValueX,rawValueZ))
            sleep(0.10)


    except KeyboardInterrupt:
        print('Exit')
"""
```

20. Digital tilt sensor

Digital tilt sensor allows us to detect an object's orientation or inclination. It is common and easy to use. Its working voltage is DC 3.3V – 5V.

Breadboard connection:



Python code:

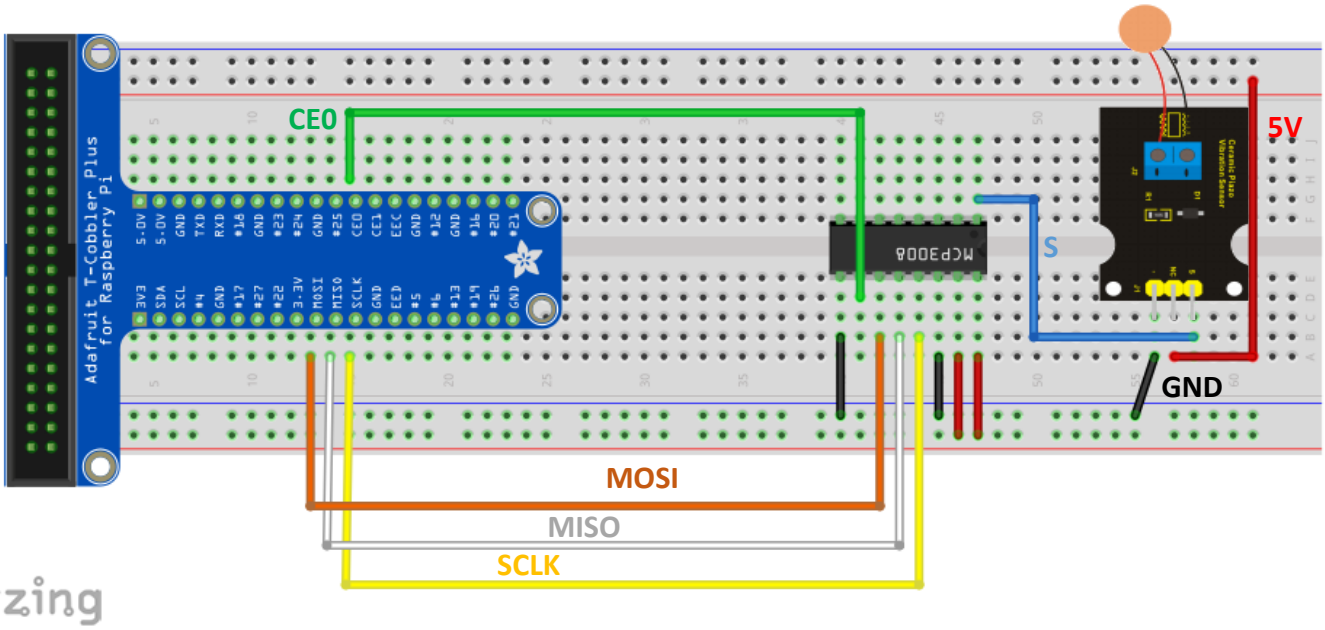
```
#Code for KeyeStudio Digital Tilt Sensor
#This code demonstrates the Tilt Sensor by telling the user
#when the sensor is tilted down vs when it is tilted up.

#KeyeStudio Digital Tilt Sensor = Ks0025
#Operating Voltage = 3.3V - 5.0V
#Interface Type = Digital

#Imports LED and LineSensor from GPIOzero Library
from gpiozero import LED
from gpiozero import LineSensor
#Imports sleep from time library
from time import sleep

if __name__ == '__main__':
    sensor = LineSensor(12) #sets sensor as object to read value from pin 12
    try:
        while(True): #starts an infinite loop
            sensor.when_line = lambda: print("The device is tilting up.")
#reads if the device has been tilted
            #if the device has been tilted, an LED one the sensor will light
up read
            sensor.when_no_line = lambda: print("The device is tilting
down.") #reads if device has been tilted back
```

```
            #if the device is tilted down, the LED on the device is very dim
    except KeyboardInterrupt: #allows the user to exit the program with Ctrl C
        print("All Done") #prints out the message
```

21. Analog Ceramic Vibration sensor

Vibration sensor is a simple sensor that detects the vibration of the surrounding. It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage is 3.3V – 5V.

Breadboard connection:

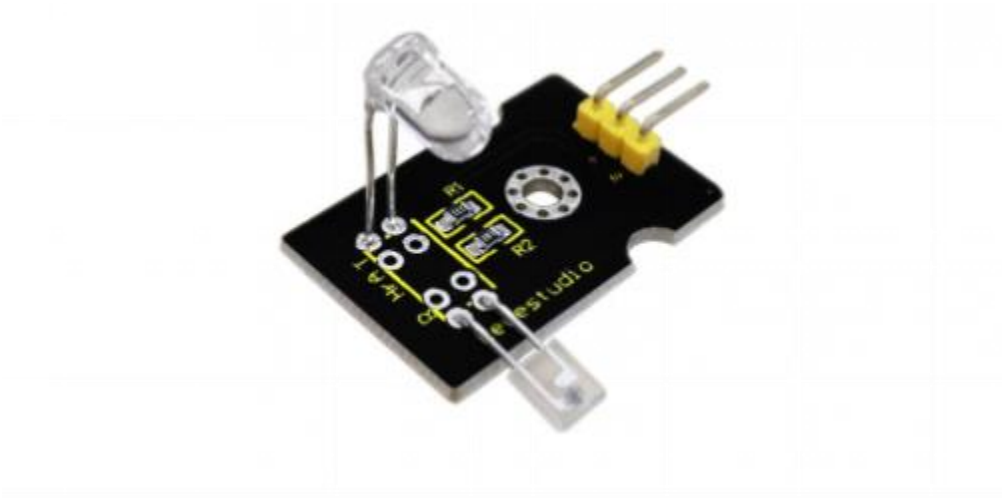


Python code:

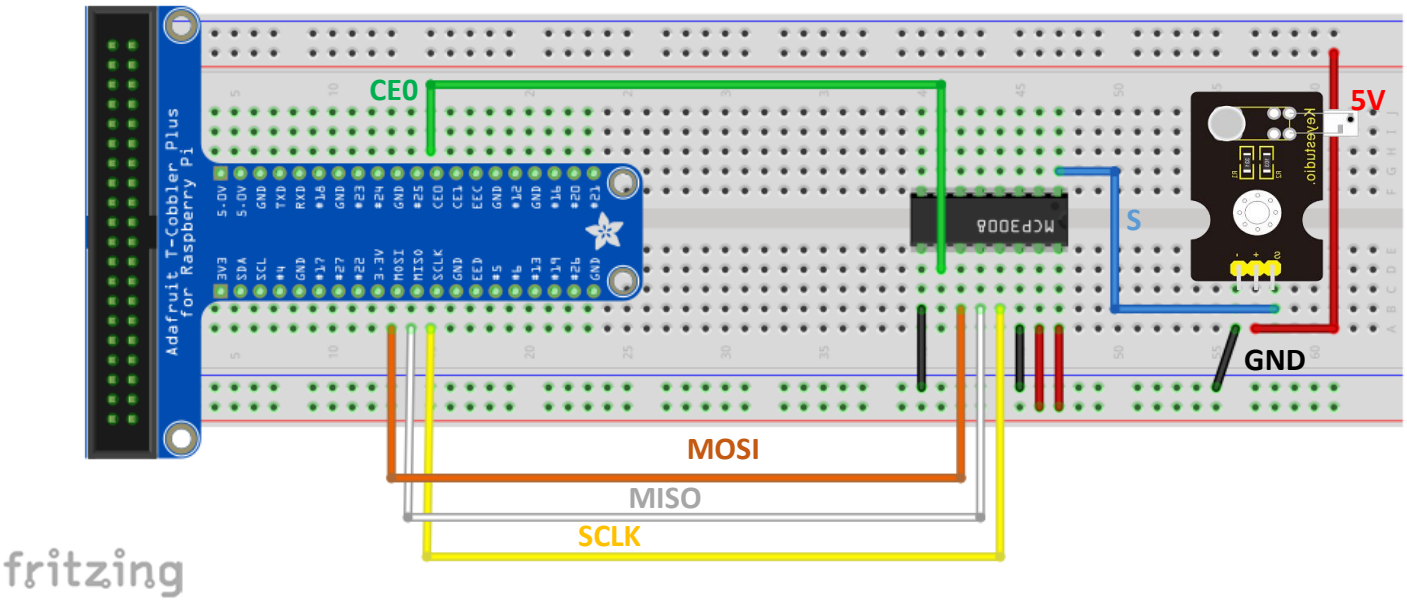
```
#Code for KeyeStudio Analog Ceramic Vibration Sensor

#KeyeStudio Analog Piezoelectric Ceramic Vibration Sensor = Ks0272
#Supply Voltage = 3.3V - 5V
#Working Current = <1mA
#Working Temperature Range = -10C - +70C
#Output Signal = Analog Signal

#Imports MCP3008 from GPIOzero library
from gpiozero import MCP3008
#Imports sleep from time library
from time import sleep

#defines variable vib as value from MCP3008
vib = MCP3008(channel = 0, device = 0)

try:
    while(True): #creates infinite time loop
        vibration = 0 #sets value to zero initially
        vibration = vib.raw_value #changes value to the sensor value
        print("The vibration detected is: ", vibration) #prints values
        sleep(1) #forces device to stop for 1 second
except KeyboardInterrupt: #allows user to exit the program
    print("All Done") #tells user the program is all done
```

22. Pulse Sensor

Pulse sensor is a simple electric sensor used to detect the pulse in our finger. The red LED flashes once the pulse is detected on the finger. It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage is 5V.



Breadboard connection:

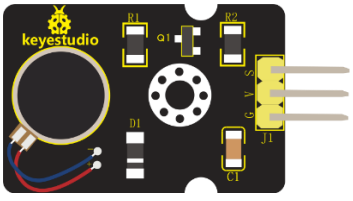


Python code:

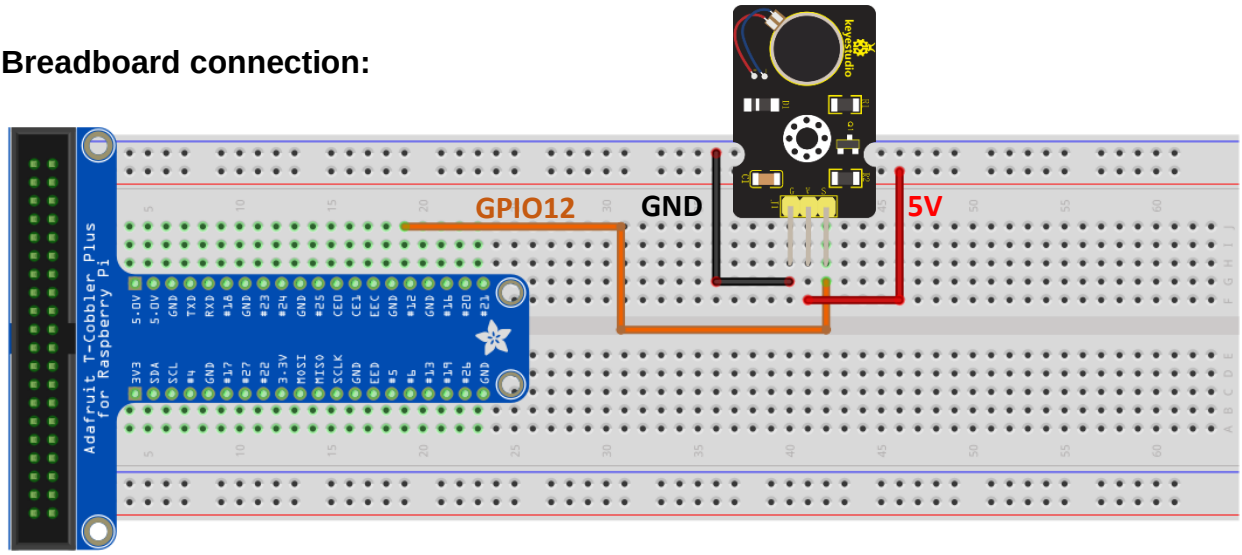
```
#Implementing a pulse measuring sensor
#Import a MCP3008 module from the gpiozero library
from gpiozero import MCP3008
from time import sleep

if __name__ == '__main__':
    PulseRate = MCP3008(0)
    while True:
        print(PulseRate.raw_value)
        sleep(1.0)
```

23. Vibration Motor Module

Vibration motor module is a simple electric sensor which vibrates whenever the output signal is high. It is a common sensor used in various electronic devices to produce vibration. Its working voltage is 5V.



Breadboard connection:

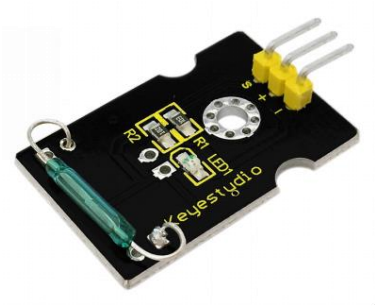


Python code:

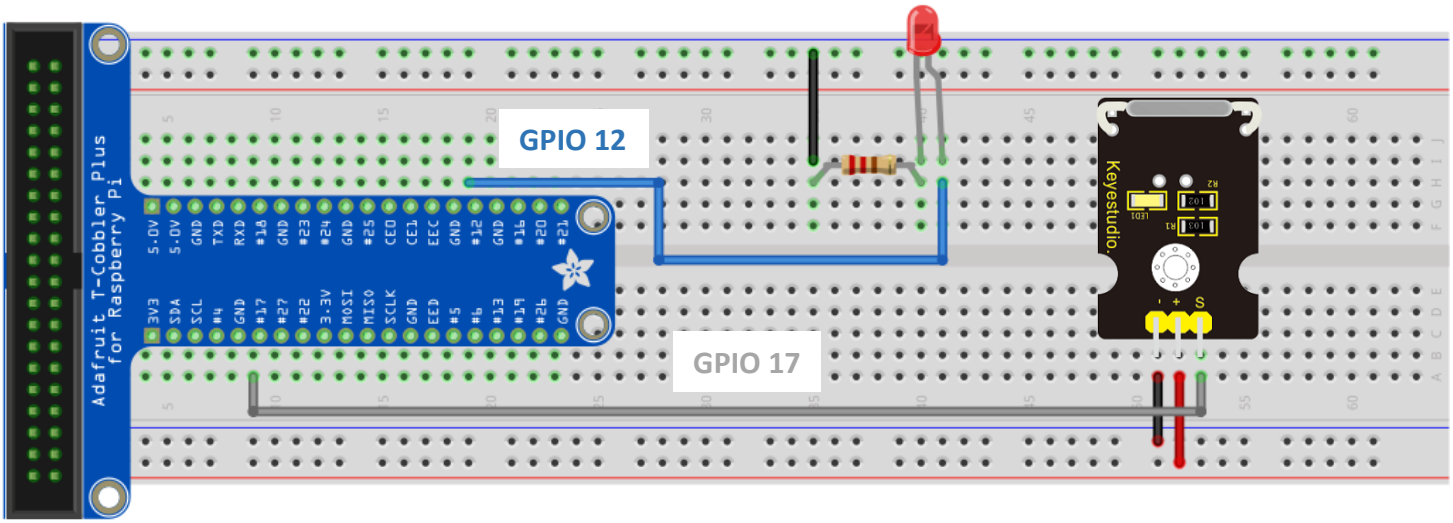
```
#Code to run KeyeStudio Motor Module.
#Program changes how fast the module vibrates using a potentiometer.

#KeyeStudio Motor Module = Ks0487
#Operating Voltage & Current: 5V DC Max / 35mA
#Max power: .5 W

#Time module, using sleep for a small delay
from time import sleep

#Motor Module does not have built in library
#Therefore must use PWMLED and the MCP chip for ADC with a potentiometer.
from gpiozero import PWMLED as Motor, MCP3008 as MCP#Import MCP3008 Object
and LED from gpiozero library

#Name analog input channel 0, pin connected to Pot
analog_input = MCP(channel = 0)
#"Motor" connected to Pin 5 of Pi
Mod = Motor(12)

if __name__ == "__main__":

    while (True):
            #set pin value to Potentiometer's value
            Mod.value = analog_input.value
            #Print binary value of Pot (0-1)
            print("Binary Value = {}".format(analog_input.value))
            #sleep for a tenth of a second
            sleep(.1)
```

24. Reed switch

Reed switch is an electronic sensor switch operated by an applied magnetic field. When a magnetic field is applied in the switch, the connection closes inside the tube and the current can flow. Its working voltage is 3.3V – 5V.



Breadboard connection:



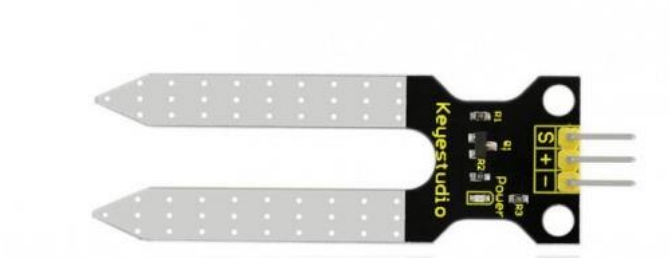
Python code:

```
#Reed Switch module
#Import LED Module from gpiozero library
from gpiozero import LED
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import DigitalInputDevice
#A Module for sleeping the Pi
from time import sleep
if __name__ == '__main__':
    #Activating the function that controls the ReedSwitch Sensor connected to
pin 17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
```

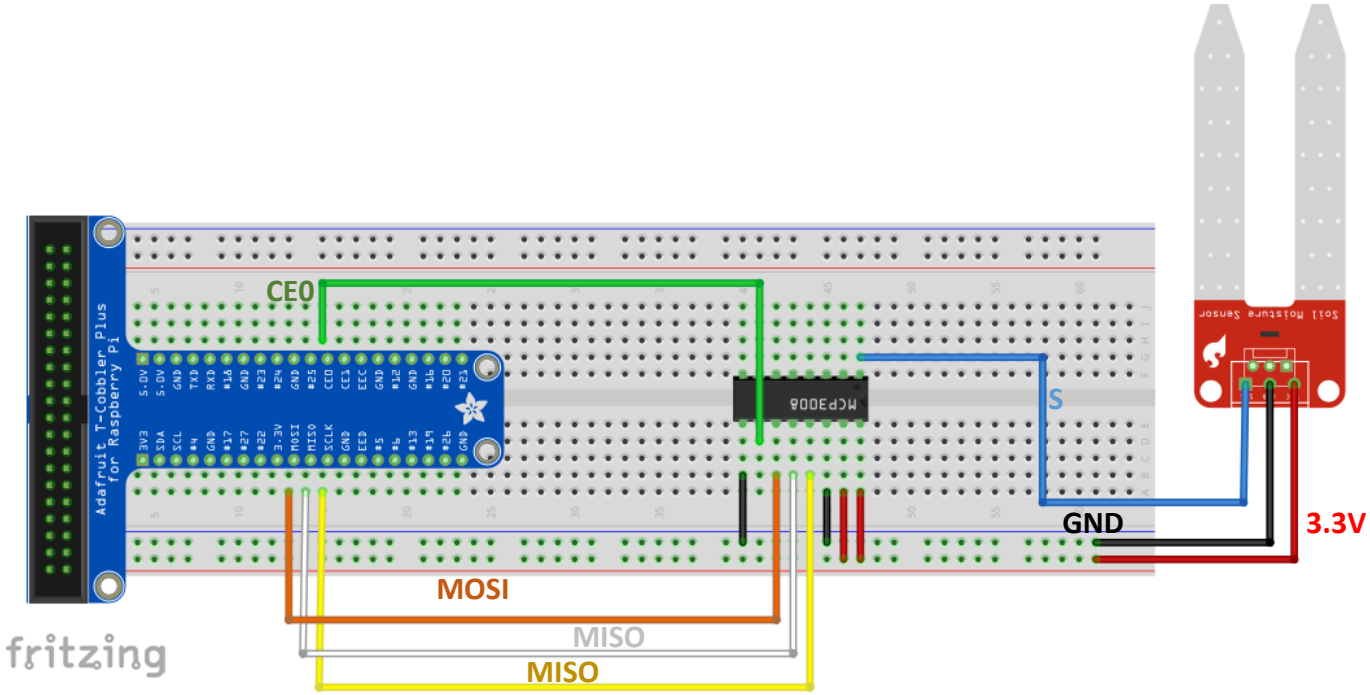
```
    Led_Red = LED(12)
    try:
        while True:
               if(sensor.wait_for_active()):
                   Led_Red.on()
                   print("Signal Detected")
               if(sensor.wait_for_inactive()):
                   Led_Red.toggle()
                   print("Not detected")
    except KeyboardInterrupt:
           print("Cleaning up")
```

25. Soil humidity sensor

Soil humidity sensor is an electronic sensor which is used to detect the soil humidity. It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage is 3.3V – 5V.



Breadboard connection:



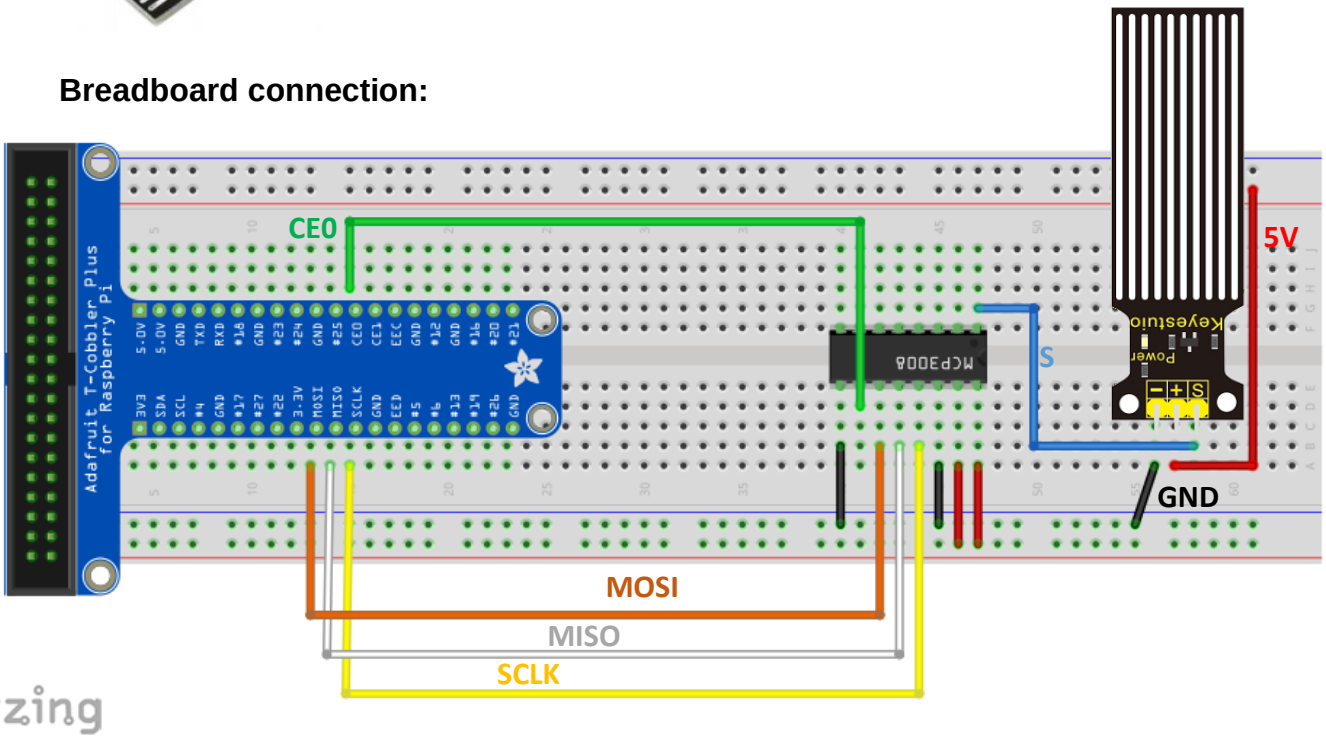
Python code:

```
#Program to find the moisture of a soil
#Import a MCP3008 module from the gpiozero library
from gpiozero import MCP3008
from time import sleep

if __name__ == '__main__':
    Voltage = MCP3008(0)
    RawValue = 0
    MAX_RAWVALUE = 1023
    VRef = 3.3
    MeasuredVoltage = 0.0
    moisture = 0.0
    while True:
        RawValue = Voltage.raw_value
        MeasuredVoltage = (VRef/MAX_RAWVALUE)*RawValue
        moisture = (MeasuredVoltage/3.3)*100
        print("Moisture is = {0:f} percent." .format(moisture))
        sleep(1.0)
```

26. Water sensor

Water sensor is an electronic sensor used to measure the water level and water drop. It is widely used in various devices. It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage is 3.3V – 5V.

Breadboard connection:



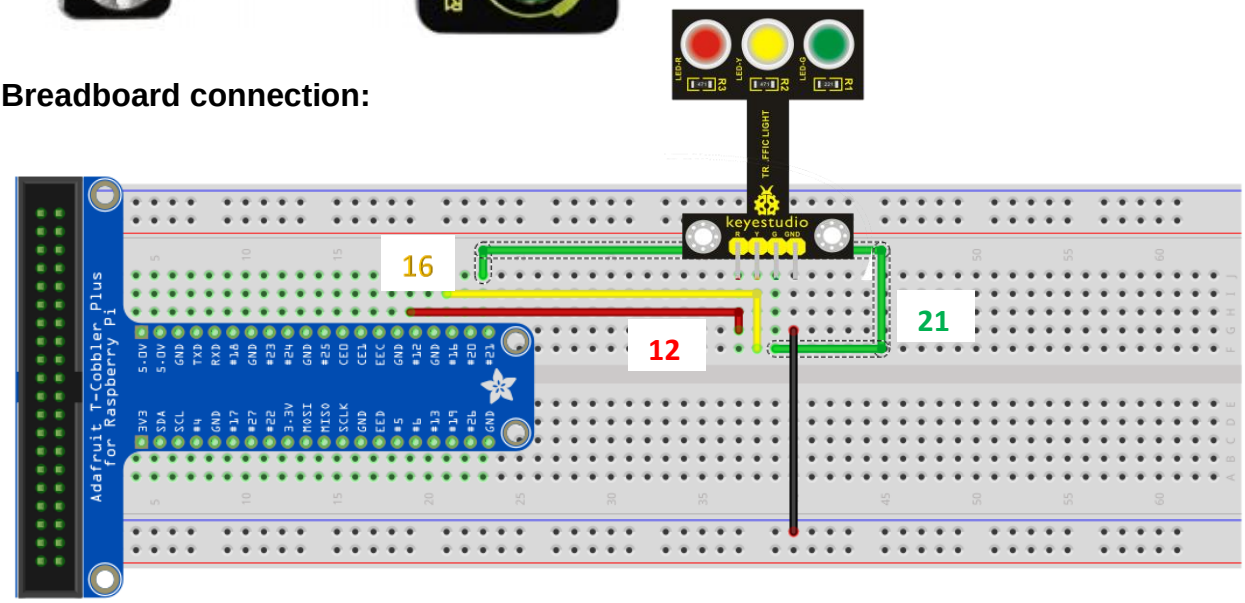
Python code:

```
#Implementing a Water measuring sensor
#Import a MCP3008 module from the gpiozero library
from gpiozero import MCP3008
from time import sleep

if __name__ == '__main__':
    Water = MCP3008(0)
    RawValue = 0
    MAX_RAWVALUE = 1023
    while True:
        RawValue = Water.raw_value
        print("Binary Value = {0:d}".format(RawValue))
        sleep(1.0)
```

27. Traffic light sensor

Traffic light sensor have three different colored LED lights (Red, Green and Blue) which helps to simulate the traffic light seen on the streets. Its working voltage is 3.3V – 5V.

Breadboard connection:



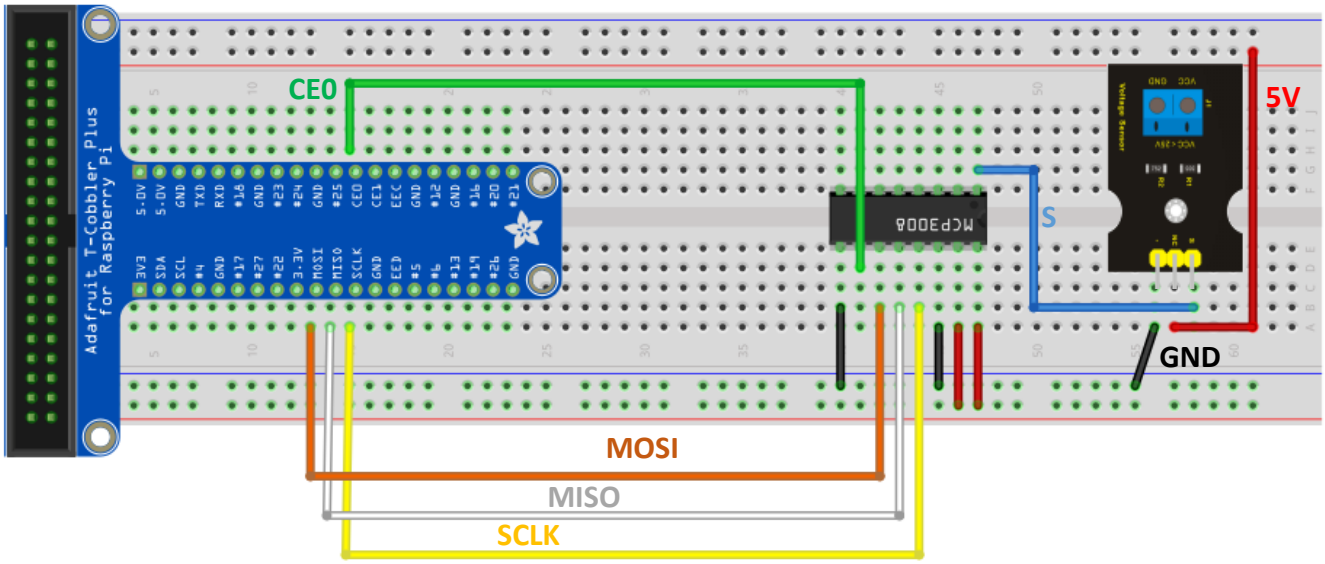
Python code:

```
#Simulating a Traffic Light behavior
#Import a Traffic Light Module from gpiozero library
from gpiozero import TrafficLights as TL
from time import sleep

if __name__ == '__main__':
    #Define Traffic Light Connections
    Red = 12
    Amber = 16
    Green = 21
    #Create a Traffic Light Object
    lights = TL(Red,Amber,Green)
    lights.green.on() #Turn on Green

    while True:
        sleep(5)
        #Turn off green and Turn on Amber
        lights.green.off()
        lights.amber.on()
        sleep(2)
        #Turn off Amber and Turn on Red
        lights.amber.off()
        lights.red.on()
        sleep(5)
        #Turn off Red and Turn on Green
        lights.red.off()
        lights.green.on()
```

28. Voltage Sensor

Voltage sensor is a simple electric sensor used to measure the voltage of the power supply. It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage ranges from DC 0V – 25V.

Breadboard connection:



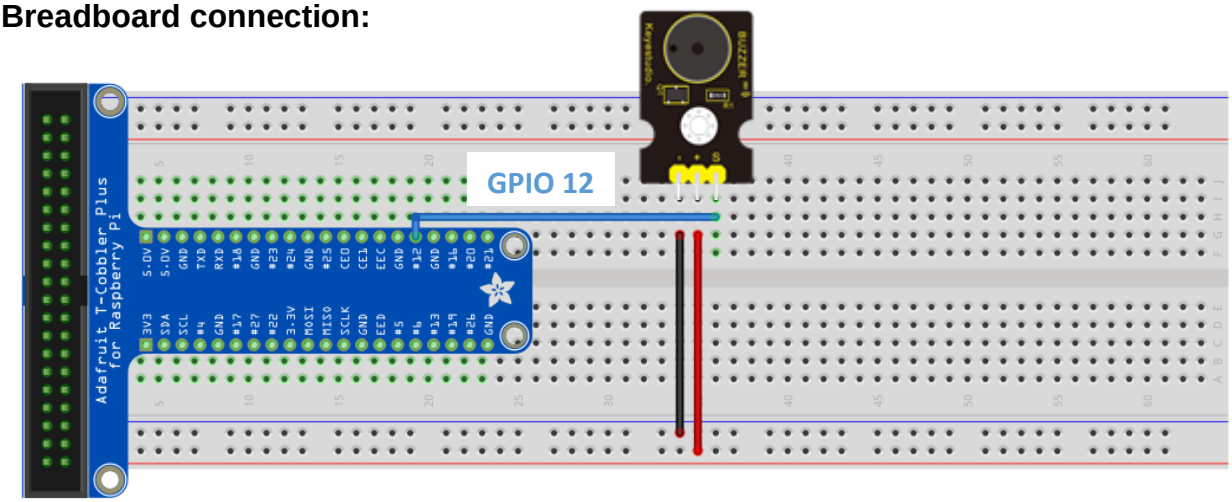
Python Code:

```
#Implementing a Voltage measuring sensor
#Import a MCP3008 module from the gpiozero library
from gpiozero import MCP3008
from time import sleep

if __name__ == '__main__':
    Voltage = MCP3008(0)
    RawValue = 0
    MAX_RAWVALUE = 1023
    VRef = 3.3
    MeasuredVoltage = 0.0
    while True:
        RawValue = Voltage.raw_value
        MeasuredVoltage = (VRef/MAX_RAWVALUE)*RawValue
        print("Binary Value = {0:d}\tVoltage:
{1:.2f}".format(RawValue,MeasuredVoltage))
        print("Binary Value = {0:d}\tVoltage:
{1:.2f}".format(RawValue,(VRef*Voltage.value)))
        sleep(1.0)
```

29. Passive Buzzer

Passive buzzer is an electronic sensor which produces different frequencies, enabling us to code the melody of a song easily. Its working voltage is DC 3.3V – 5V.

Breadboard connection:



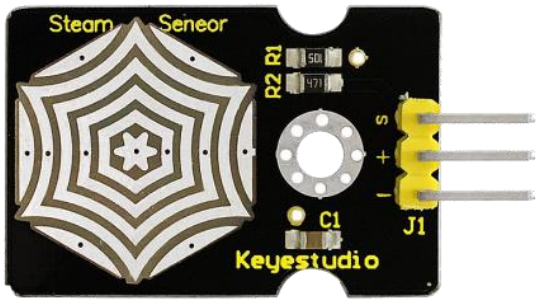
Python code:

```
#Code for KeyeStudio Passive Buzzer

#KeyeStudio Passive Buzzer Module = Ks0019
#Working Voltage = 3.3V - 5.0V
#Interface Type = Digital

#Imports TonalBuzzer from GPIOzero library
from gpiozero import TonalBuzzer
#Imports tones from GPIOzero library
from gpiozero.tones import Tone
#Imports sleep from time library
from time import sleep

if __name__ == '__main__':
    b = TonalBuzzer(12) #sets variable as output signal on pin 12
    try:
        while(True): #creates an infinite loop
            b.play(Tone("A4")) #tells buzzer which note to play
            b.play(Tone(220.0)) #This is in hertz
            b.play(Tone(60))
```

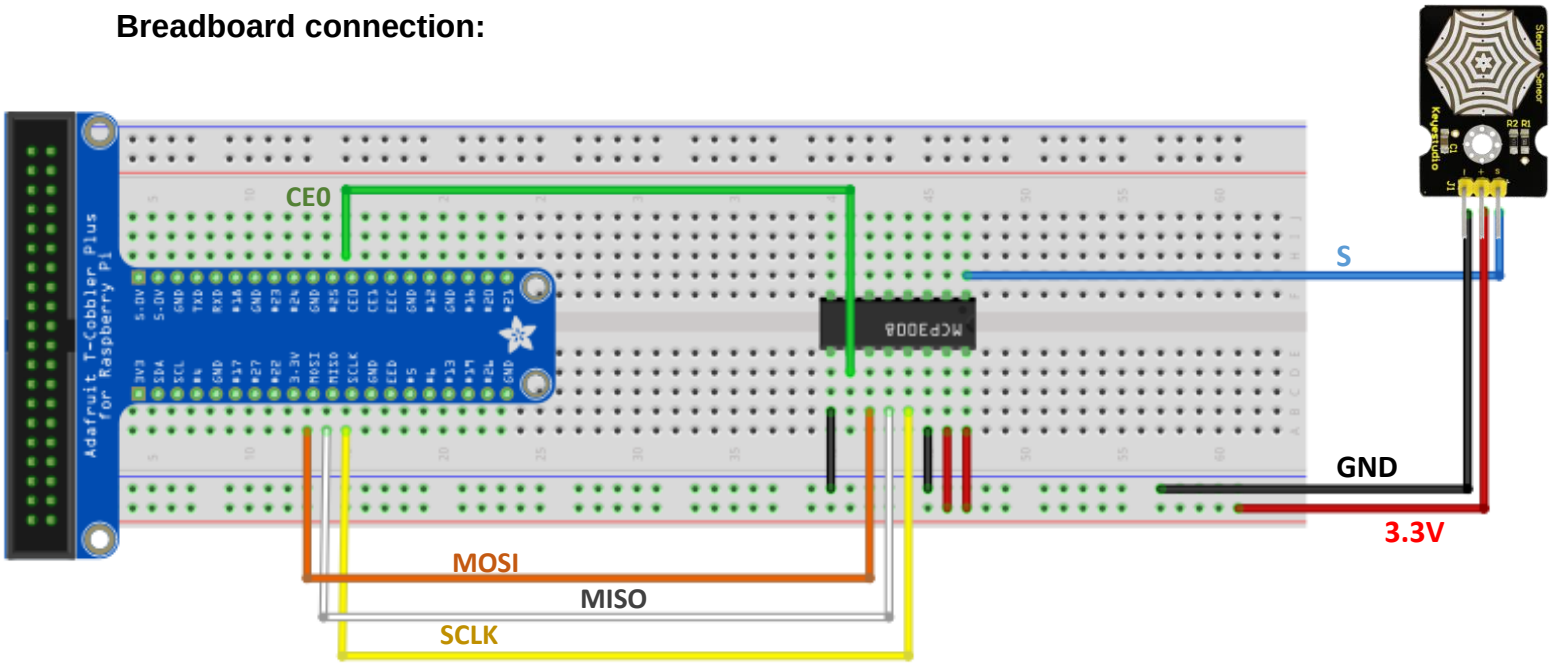
```
            print("Playing A4") #prints out which note is played
            sleep(1) #forces the device to play for 1 second
            b.play(Tone("A3")) #tells buzzer which note to play
            b.play(Tone(220.0)) #tells which hertz to play at
            b.play(Tone(60))
            print("Playing A3") #prints out which note is being played
            sleep(1) #forces buzzer to play for 1 second
            b.stop() #stops the buzzer from making any noice
            sleep(2) #forces the buzzer to sleep for 2 seconds
    except KeyboardInterrupt: #allows user to exit the program
        print("All Done") #tells user the program is all done
```

30. Steam sensor

Steam sensor is an electronic sensor used to detect rain water and steam. Since it is non-waterproof, it can only measure a drop of rain water and few pounds of steam. . It is an analog device and hence, we need to use MCP3008 pin to program it on Raspberry Pi. Its working voltage is 3.3V.



Breadboard connection:

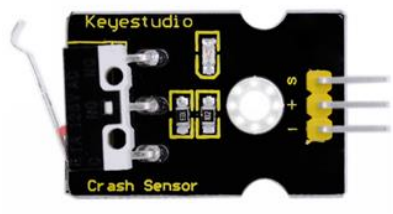


Python code:

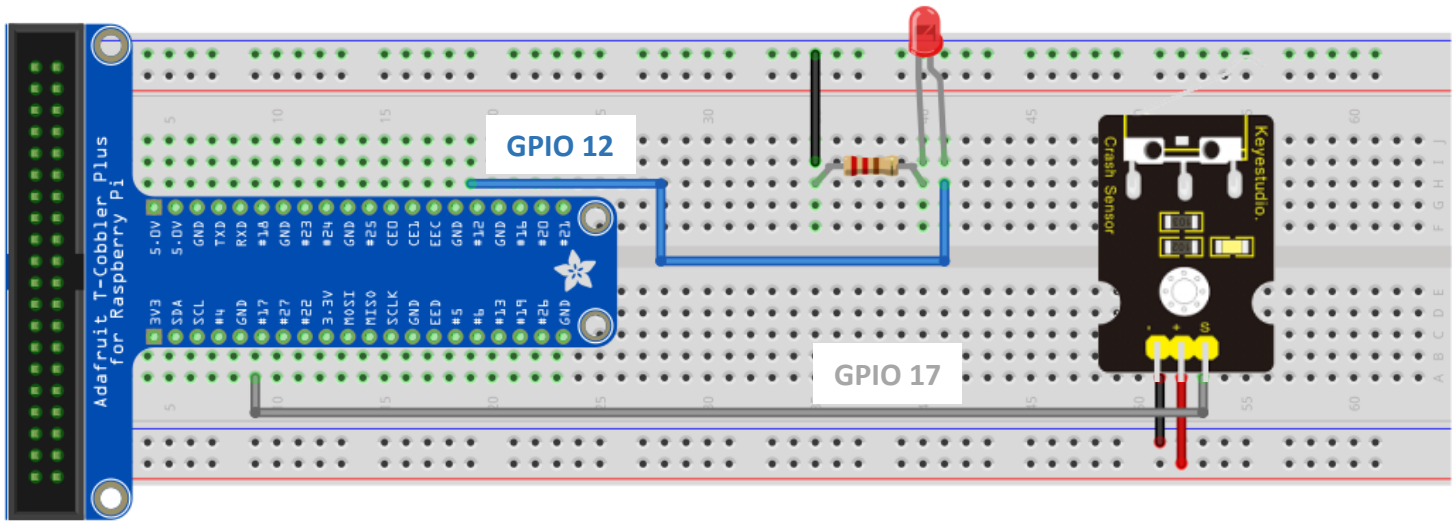
```
from gpiozero import MCP3008
from time import sleep

if __name__ == '__main__':
    Voltage = MCP3008(0)
    RawValue = 0
    MAX_RAWVALUE = 1023
    VRef = 3.3
    MeasuredVoltage = 0.0
    steam = 0.0
    while True:
        RawValue = Voltage.raw_value
        MeasuredVoltage = (VRef/MAX_RAWVALUE)*RawValue
        steam = (MeasuredVoltage/3.3)*100
        print("Steam is = {0:f} percent." .format(steam))
        sleep(1.0)
```

31. Collision Flash

Collision Flash/ Crash Sensor is an electronic micro switch which helps to detect collision by performing action on minimum physical force. It works on 3.3 – 5V power supply.



Breadboard connection:



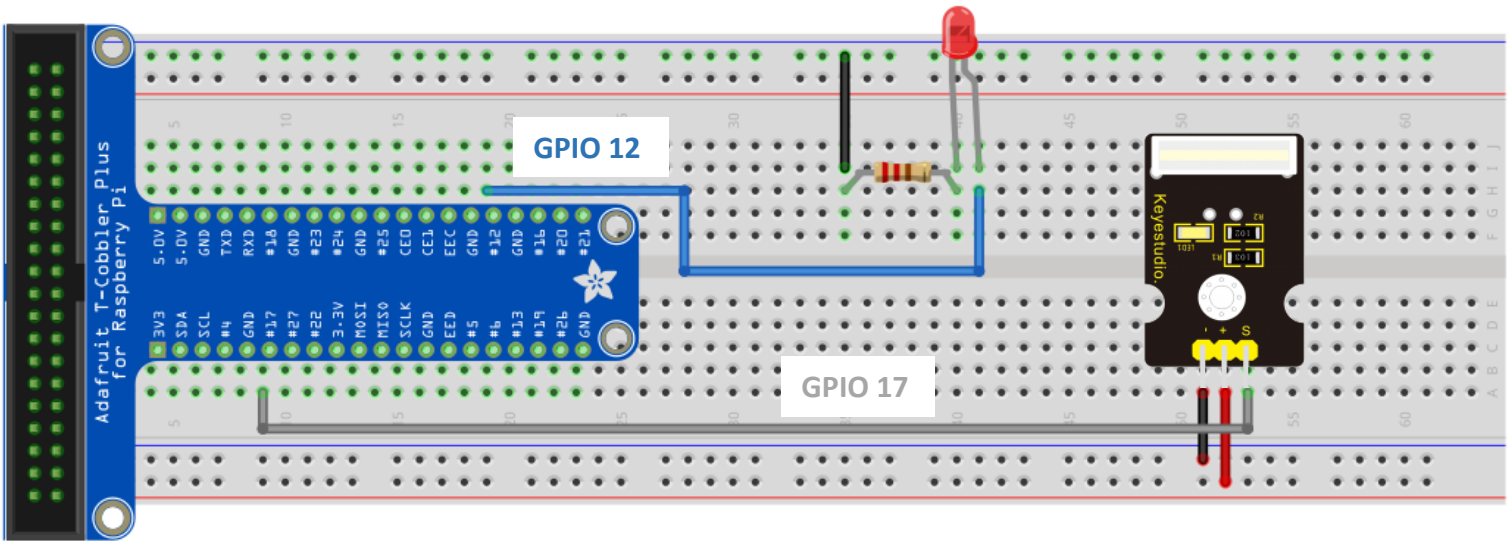
Python code:

```
#Crash sensor module
#Import LED Module from gpiozero library
from gpiozero import LED
#A Module for sleeping the Pi
from time import sleep
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import DigitalInputDevice
if __name__ == '__main__':
    #Activating the function that controls the Crash Sensor connected to pin
17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
    Led_Red = LED(12)
    try:
        while True:
            if(sensor.wait_for_active()):
                print("Signal Detected")
                Led_Red.on()
            if(sensor.wait_for_inactive()):
                Led_Red.off()
                print("Not detected")
    except KeyboardInterrupt:
           Led_Red.off()
           print("Cleaning up")
```

32. Knock sensor

Knock sensor is an electronic sensor which sends signal when a knock is felt by it. It is mostly used in car engines to find the maximum vibration produced. Its working voltage is 5V.

Breadboard connection:

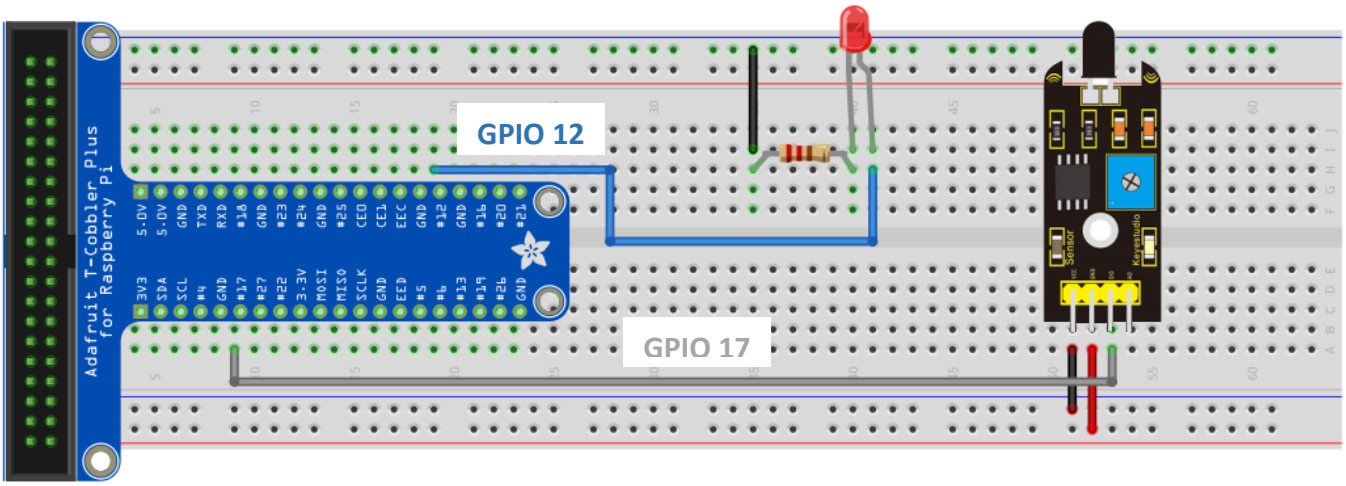


```
#Knock sensor module
#Import LED Module from gpiozero library
from gpiozero import LED
#A Module for sleeping the Pi
from time import sleep
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import DigitalInputDevice
if __name__ == '__main__':
    #Activating the function that controls the Knock Sensor connected to pin
17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
    Led_Red = LED(12)
    try:
        while True:
            if(sensor.wait_for_active()):
                print("Knock Signal Detected")
                Led_Red.on()
            if(sensor.wait_for_inactive()):
                Led_Red.off()
                print("Not detected")
    except KeyboardInterrupt:
           Led_Red.off()
           print("Cleaning up")
```

33. Flame Sensor

Flame sensor is an electronic sensor used to detect flames or lights with wavelength ranging from 750 nm – 1100 nm. It is mostly used as a fire alarm in buildings. Its supply voltage is 3.3V – 5V DC.

Breadboard connection:



```
#Flame alarm module
#Import LED Module from gpiozero library
from gpiozero import LED
#A Module for sleeping the Pi
from time import sleep
#Import DigitalInputDevice Module from gpiozero library
from gpiozero import InputDevice
if __name__ == '__main__':
    #Activating the function that controls the Flame sensor connected to pin
17
    sensor = DigitalInputDevice(17,pull_up=True)
    #Activating function that controls the red LED connected to pin 12
    Led_Red = LED(12)
    try:
        while True:
            if(sensor.wait_for_active()):
                print("Flame Detected")
                Led_Red.on()
            if(sensor.wait_for_inactive()):
                Led_Red.off()
                print("Not detected")
    except KeyboardInterrupt:
           Led_Red.off()
           print("Cleaning up")
```