



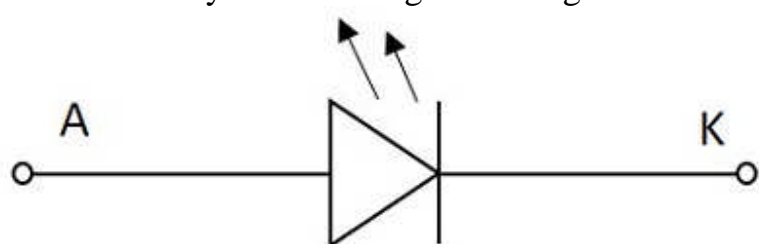
LAB MANUAL 1

Interfacing Digital Sensors

LED – Light Emitting Diodes

This one is the most popular diodes used in our daily life. This is also a normal PN junction diode except that instead of silicon and germanium, the materials like gallium arsenide, gallium arsenide phosphide are used in its construction.

The figure below shows the symbol of a Light emitting diode.



Symbol of LED

LED

Like a normal PN junction diode, this is connected in forward bias condition so that the diode conducts. The conduction takes place in a LED when the free electrons in the conduction band combine with the holes in the valence band. This process of re-combination emits light. This process is called as Electroluminescence. The color of the light emitted depends upon the gap between the energy bands. The LEDs for non-visible Infrared light are used mostly in remote controls.



Reference:

https://www.tutorialspoint.com/basic_electronics/basic_electronics_optoelectronic_diodes.htm

LED Interfacing using GrovePI & Raspberry-PI

Steps-

1. Connect Grove-PI Hat to RaspberryPI.
2. Connect Led to D7 port.
3. Run below code to get output from sensor

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/1_LED_Interfacing.py

Code-

```
from grovepi import *  
  
led = 7  
pinMode(led,"OUTPUT")  
  
ip = input()  
if ip == "H":  
    digitalWrite(led,1)  
elif ip == "L":  
    digitalWrite(led,0)
```

LED Interfacing using DFRobot Hat & RaspberryPI

In the first project, we will learn about the following contents:

1. Internal circuit resolution of digital LED light emitting module
2. Basic uses of Thonny Python IDE
3. Basic Python code for operating GPIO.

Hardware

1. Gravity: 37 Pcs Sensor Set
2. Raspberry Pi 4 Model B
3. IO Expansion HAT for Raspberry Pi 4B/3B+
4. 8GB + SanDisk Class10 SD/MicroSD Memory Card
5. 5V@3A USB Power Supply

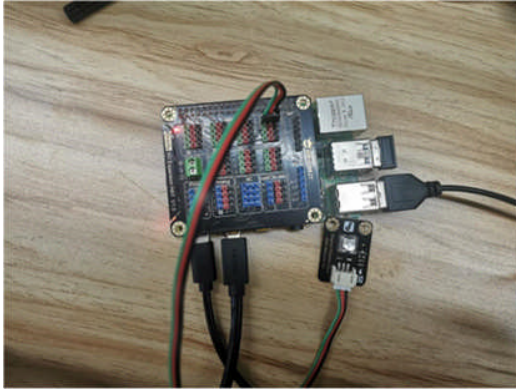
Connection

- Connect the necessary peripherals such as the screen, power, keyboard and mouse to your Raspberry Pi.



Device Setup

- Install the IO expansion board on the Raspberry Pi. And connect the LED light-emitting module to the digital port 12 of the expansion board. Then start it up.



- Open Thonny Python IDE to copy the following program into it

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/1_led_control.py

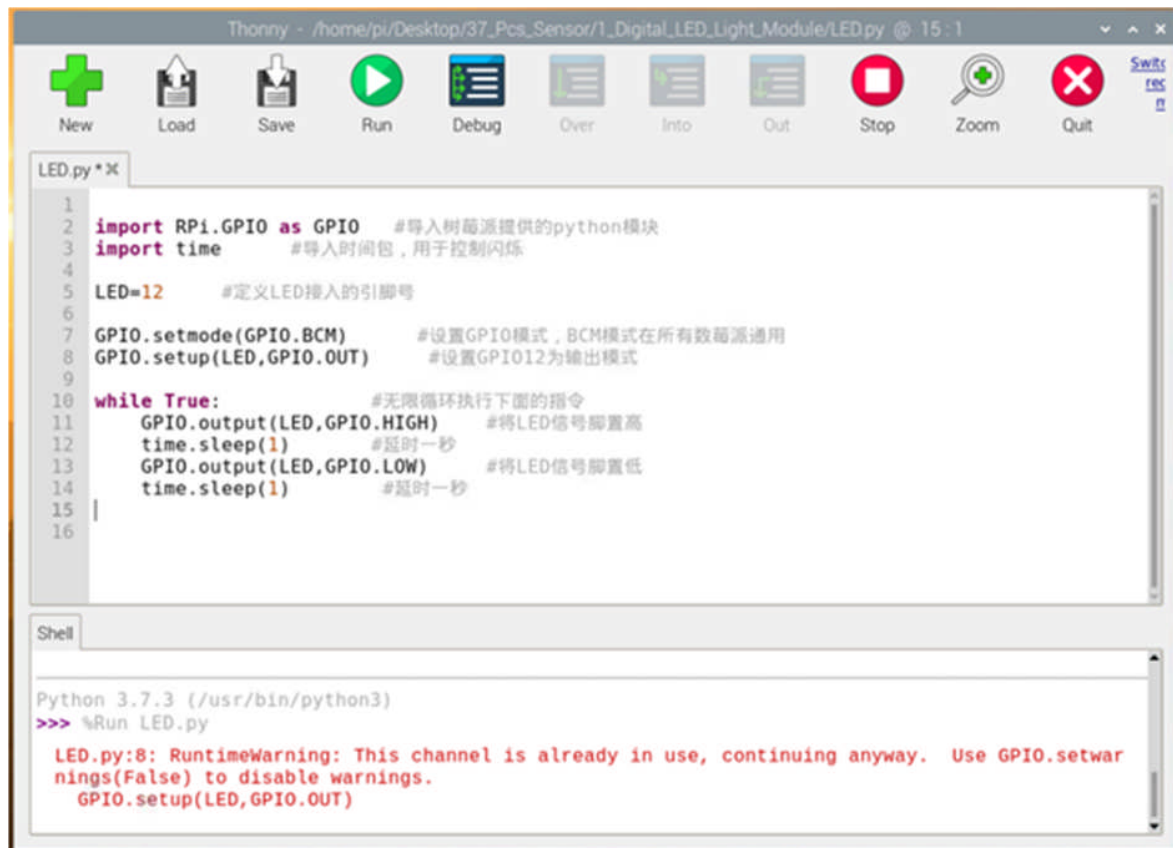
Code

```
import RPi.GPIO as GPIO
import time
import atexit

LED=12

atexit.register(GPIO.cleanup)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)

while True:
    GPIO.output(LED,GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED,GPIO.LOW)
    time.sleep(1)
```



```
Thonny - /home/pi/Desktop/37_Pcs_Sensor/1_Digital_LED_Light_Module/LED.py @ 15.1

New Load Save Run Debug Over Into Out Stop Zoom Quit

LED.py *X
1
2 import RPi.GPIO as GPIO #导入树莓派提供的python模块
3 import time #导入时间包，用于控制闪烁
4
5 LED=12 #定义LED接入的引脚号
6
7 GPIO.setmode(GPIO.BCM) #设置GPIO模式；BCM模式在所有树莓派通用
8 GPIO.setup(LED,GPIO.OUT) #设置GPIO12为输出模式
9
10 while True: #无限循环执行下面的指令
11     GPIO.output(LED,GPIO.HIGH) #将LED信号脚置高
12     time.sleep(1) #延时一秒
13     GPIO.output(LED,GPIO.LOW) #将LED信号脚置低
14     time.sleep(1) #延时一秒
15
16

Shell

Python 3.7.3 (/usr/bin/python3)
>>> %Run LED.py

LED.py:8: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED,GPIO.OUT)
```

[VNC Viewer](#)

- Click 'Save' to set the file name and the path you like to save
- Click 'Run', then you can see the LED is flickering

Buzzer

There are many ways to communicate between the user and a product. One of the best ways is audio communication using a buzzer



Reference: <https://www.instructables.com/How-to-use-a-Buzzer-Arduino-Tutorial/>

What is a Buzzer?

An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-' symbol or short terminal and it is connected to the GND terminal.



Reference: <https://www.elprocus.com/wp-content/uploads/Buzzer-Pin-Configuration-272x300.jpg>

How to use a Buzzer?

A buzzer is an efficient component to include the features of sound in our system or project. It is an extremely small & solid two-pin device thus it can be simply

utilized on breadboard or PCB. So, in most applications, this component is widely used.

There are two kinds of buzzers commonly available like simple and readymade. Once a simple type is power-driven then it will generate a beep sound continuously. A readymade type looks heavier & generates a Beep. Beep. Beep. This sound is because of the internal oscillating circuit within it.

This buzzer uses a DC power supply that ranges from 4V – 9V. To operate this, a 9V battery is used but it is suggested to utilize a regulated +5V/+6V DC supply. Generally, it is connected through a switching circuit to switch ON/OFF the buzzer at the necessary time interval.

Buzzer Interfacing using GrovePI & Raspberry-PI

Steps-

1. Connect Grove-PI Hat to RaspberryPI.
2. Connect Buzzer to D2 port.
3. Connect Button to D4 port
4. Run below code to get output from sensor

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/2_Buzzer_Control.py

Code-

```
import time
from grovepi import *
import math
buzzer_pin = 2    #Port for buzzer
button = 4
pinMode(buzzer_pin,"OUTPUT")
pinMode(button,"INPUT")

while True:
    a=digitalRead(button)
    print(a)
    if (a==0):
        digitalWrite(buzzer_pin,0)
    else:
```



```
digitalWrite(buzzer_pin,1)
```

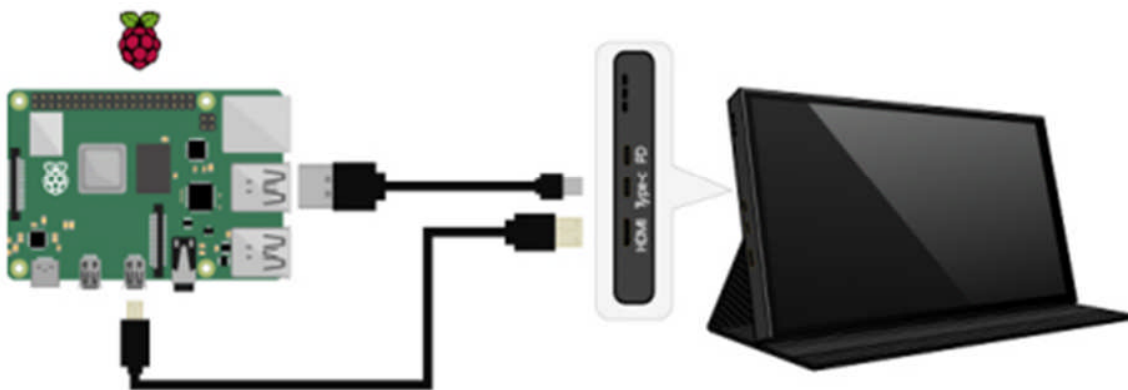
Buzzer Interfacing using DFRobot & Raspberry-PI

Hardware Required

1. Gravity: 37 Pcs Sensor Set
2. Raspberry Pi 4 Model B
3. IO Expansion HAT for Raspberry Pi 4B/3B+
4. 8GB + SanDisk Class10 SD/MicroSD Memory Card
5. 5V@3A USB Power Supply

Connection

- Connect the necessary peripherals such as the screen, power, keyboard and mouse to your Raspberry Pi.



[Device Setup](#)

- Install the Raspberry Pi IO expansion board on your 'Pi' and connect the Buzzer to digital port 12 on the expansion board, and the digital push button module to the digital port 8. Then power on.
- By analyzing the schematic circuit diagram of this module, we can see that when release the button, the signal output pin is at low level and the indicator light is off. When the button is pressed, the signal pin is at high level and the indicator light is on. They are corresponding to the two states of the LED light-emitting module. In this case, the button can be used to control the LED light through a simple judgment by the Raspberry Pi.
- Open Thonny Python IDE to copy the following program into it

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/2_buzzer_button.py

Code

```
import RPi.GPIO as GPIO
import time
import atexit
```

```
Buzzor=12
Button=8
```

```
atexit.register(GPIO.cleanup)
GPIO.setmode(GPIO.BCM)
GPIO.setup(Buzzor,GPIO.OUT)
GPIO.setup(Button,GPIO.IN)
```

```
while True:
    if GPIO.input(Button):
        GPIO.output(Buzzor,GPIO.HIGH)
    else :
        GPIO.output(Buzzor,GPIO.LOW)
    time.sleep(0.1)
```

- Click 'Save' to set the file name and the save path you like
- Click 'Run', you can see that the light is on and off with the button pressed and released.

Digital Vibration Sensor

The vibration sensor is also called a piezoelectric sensor. These sensors are flexible devices which are used for measuring various processes. This sensor uses the piezoelectric effects while measuring the changes within acceleration, pressure, temperature, force otherwise strain by changing to an electrical charge. This sensor is also used for deciding fragrances within the air by immediately measuring capacitance as well as quality.

Vibration Sensor Working Principle

The working principle of vibration sensor is a sensor which operates based on different optical otherwise mechanical principles for detecting observed system vibrations.

The sensitivity of these sensors normally ranges from 10 mV/g to 100 mV/g, and there are lower and higher sensitivities are also accessible. The sensitivity of the sensor can be selected based on the application. So it is essential to know the levels of vibration amplitude range to which the sensor will be exposed throughout measurements.

Interfacing Vibration Sensor with GrovePI & RaspberryPI

Github link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/3_Vibration_Detection.py

Code-

```
import time
from grovepi import *
import math
buzzer_pin = 2    #Port for buzzer
vibration_sensor = 4
pinMode(buzzer_pin,"OUTPUT")
pinMode(vibration_sensor,"INPUT")
digitalWrite(buzzer_pin,0)
while True:
    a=digitalRead(vibration_sensor)
    print(a)
    if (a==0):
        digitalWrite(buzzer_pin,0)
    else:
        digitalWrite(buzzer_pin,1)
```

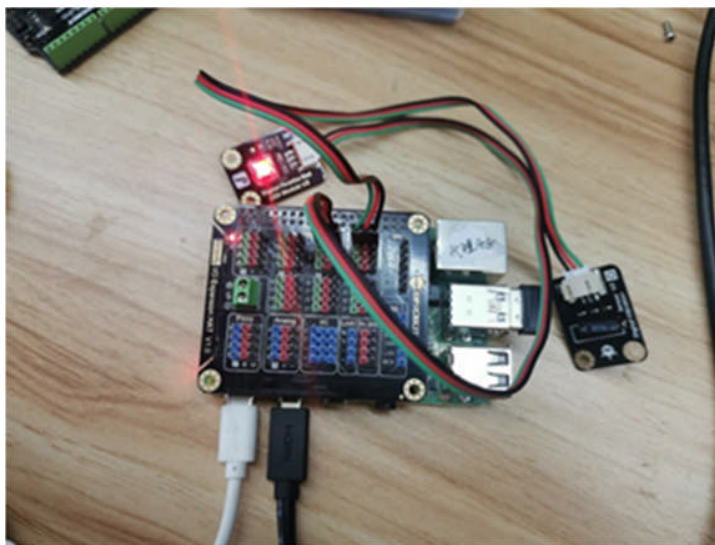
Interfacing Vibration Sensor with DFRobot Hat & RaspberryPI

It is a great idea to use a vibration switch to turn on and off the circuit through vibration to generate a signal. Despite a simple structure, you can make full use of

this vibration sensor with creative thinking, like step counting, Crash warning light and so on. As long as you have ideas, the usage of simple components will change endlessly. As long as you have ideas, the usage of simple components can be varied an infinite number of times.

Use Digital Vibration Sensor on Your Raspberry Pi

- Connect the digital LED light-emitting module to the pin 12 on extension board, and connect the digital vibration sensor to the pin 8.



- Open Thonny Python IDE to copy the following program into it

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/8_DFR_digital_vibration_sensor.py

Code-

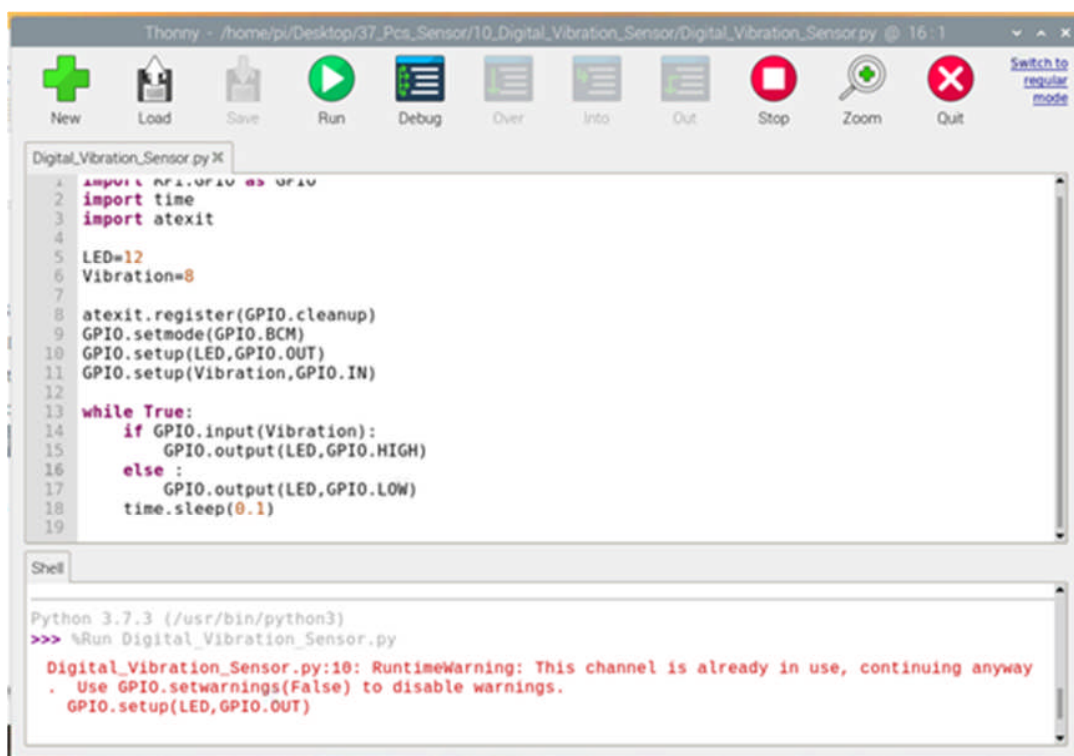
```
import RPi.GPIO as GPIO
import time
import atexit
```

```
LED=12
```

```
Vibration=8
```

```
atexit.register(GPIO.cleanup)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)
GPIO.setup(Vibration,GPIO.IN)
```

```
while True:
    if GPIO.input(Vibration):
        GPIO.output(LED,GPIO.HIGH)
    else :
        GPIO.output(LED,GPIO.LOW)
    time.sleep(0.1)
```



- Click 'Run', you can observe that when the sensor detects vibration, the LED will go out.

Heart Rate Monitor Sensor

A person's heartbeat is the sound of the valves in his/her's heart contracting or expanding as they force blood from one region to another. The number of times the heart beats per minute (BPM), is the heartbeat rate and the beat of the heart that can be felt in any artery that lies close to the skin is the pulse.

Two Ways to Measure a Heartbeat

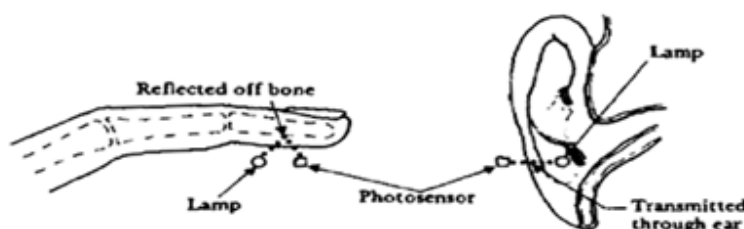
- **Manual Way:** Heartbeat can be checked manually by checking one's pulses at two locations- wrist (the **radial pulse**) and the neck (**carotid pulse**). The procedure is to place the two fingers (index and middle finger) on the wrist (or neck below the windpipe) and count the number of pulses for 30 seconds and then multiplying that number by 2 to get the heartbeat rate. However, pressure should be applied minimum and also fingers should be moved up and down till the pulse is felt.
- **Using a sensor:** Heart Beat can be measured based on optical power variation as light is scattered or absorbed during its path through the blood as the heartbeat changes.

Principle of Heartbeat Sensor

The heartbeat sensor is based on the principle of **photoplethysmography**. It measures the change in volume of blood through any organ of the body which causes a change in the light intensity through that organ (avascular region). In the case of applications where the heart pulse rate is to be monitored, the timing of the pulses is more important. The flow of blood volume is decided by the rate of heart pulses and since light is absorbed by the blood, the signal pulses are equivalent to the heartbeat pulses.

There are two types of photoplethysmography:

- **Transmission:** Light emitted from the light-emitting device is transmitted through any vascular region of the body like earlobe and received by the detector.
- **Reflection:** Light emitted from the light-emitting device is reflected by the regions.

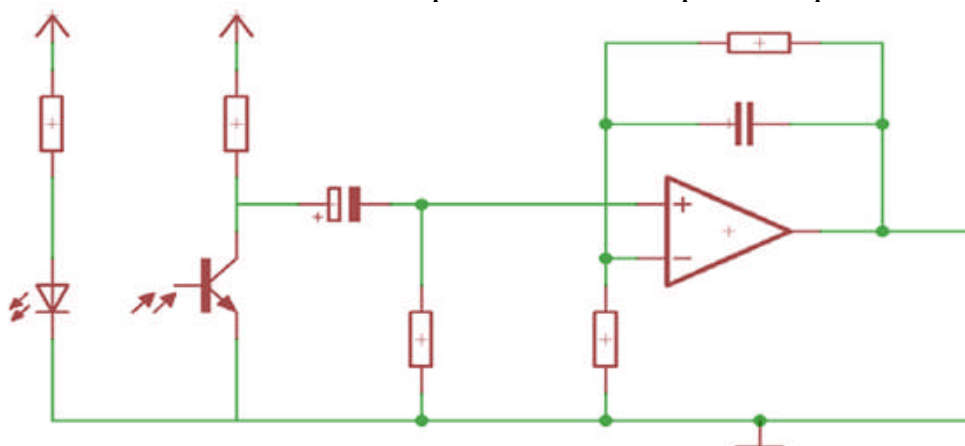


Reflection mechanism

Working of a Heartbeat Sensor

The basic heartbeat sensor consists of a light-emitting diode and a detector like a light detecting resistor or a photodiode. The heartbeat pulses cause a variation in the flow of blood to different regions of the body. When tissue is illuminated with the light source, i.e. light emitted by the led, it either reflects (a finger tissue) or transmits the light (earlobe). Some of the light is absorbed by the blood and the transmitted or the reflected light is received by the light detector. The amount of light absorbed depends on the blood volume in that tissue. The detector output is in the form of the electrical signal and is proportional to the heartbeat rate.

This signal is a DC signal relating to the tissues and the blood volume and the AC component synchronous with the heartbeat and caused by pulsatile changes in arterial blood volume is superimposed on the DC signal. Thus the major requirement is to isolate that AC component as it is of prime importance.



Amplification of optical signal from blood

To achieve the task of getting the AC signal, the output from the detector is first filtered using a 2 stage HP-LP circuit and is then converted to digital pulses using a comparator circuit or using simple ADC. The digital pulses are given for calculating the heartbeat rate, given by the formula-

BPM(Beats per minute) = $60 \cdot f$, Where f is the pulse frequency

Interfacing Hear beat sensor with GrovePI & Raspberry

Github link

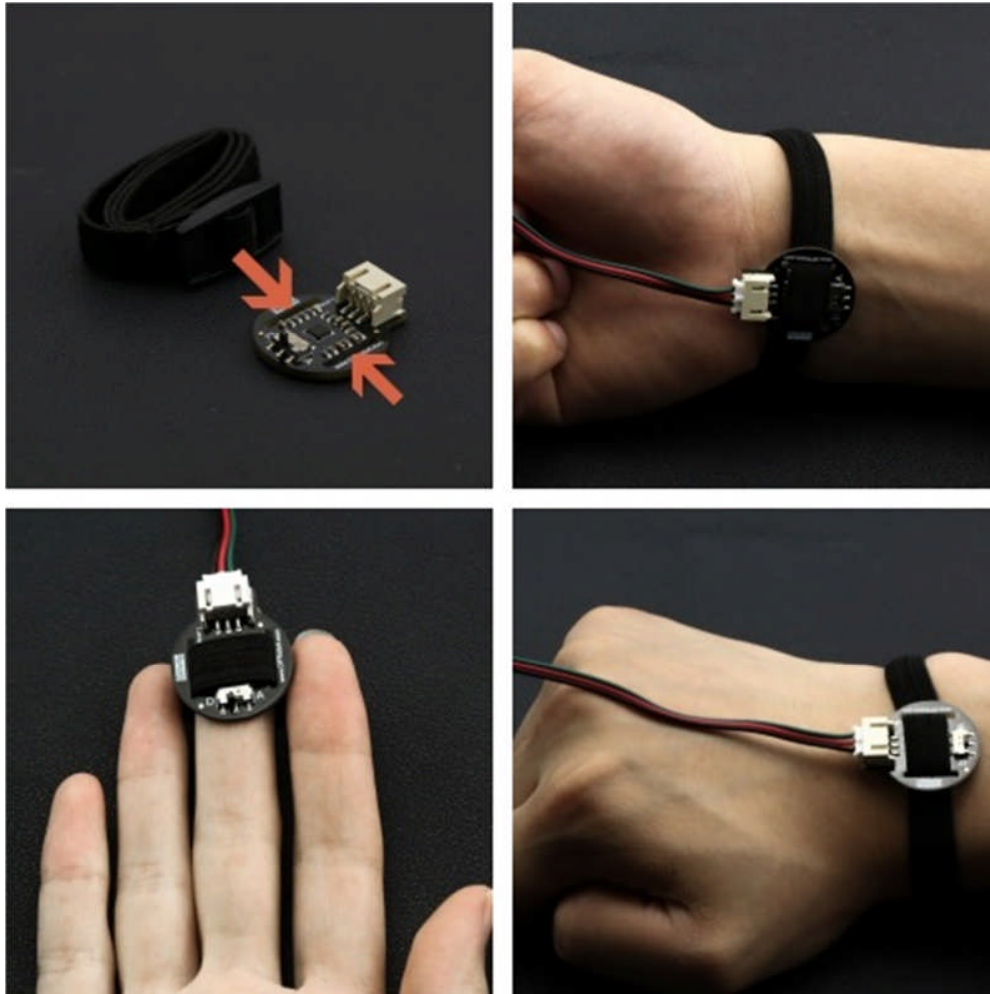
https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/4_Heart_beat_sensor.py

Code-

```
import time
from grovepi import *
import math
LED = 2    #Port for buzzer
heart_beat_sensor = 4
pinMode(LED,"OUTPUT")
pinMode(heart_beat_sensor,"INPUT")
digitalWrite(LED,0)
while True:
    a=digitalRead(heart_beat_sensor)
    print(a)
    if (a==0):
        digitalWrite(LED,0)
    else:
        digitalWrite(LED,1)
```

Interfacing Heart beat sensor with DFRobot Hat & RaspberryPI

The DFRobot heart rate sensor, despite a tiny size of just a thumb, can monitor human heart rates. Compatible with RaspberryPI, this plug-and-play module is really convenient to use that it's equipped with Gravity 3-Pin interface. It uses PPG (Photo Plethysmo Graphy) to measure heart rate, a low-cost optical technology that monitors the human heart rate by detecting blood oxygen levels in subcutaneous capillaries. Besides, this technology features fast response, stable performance and strong adaptability. As being equipped with two mounting holes, the sensor can be wrapped on your finger, wrist, earlobe or other areas where it has contact with your skin.



Heart beat Sensor

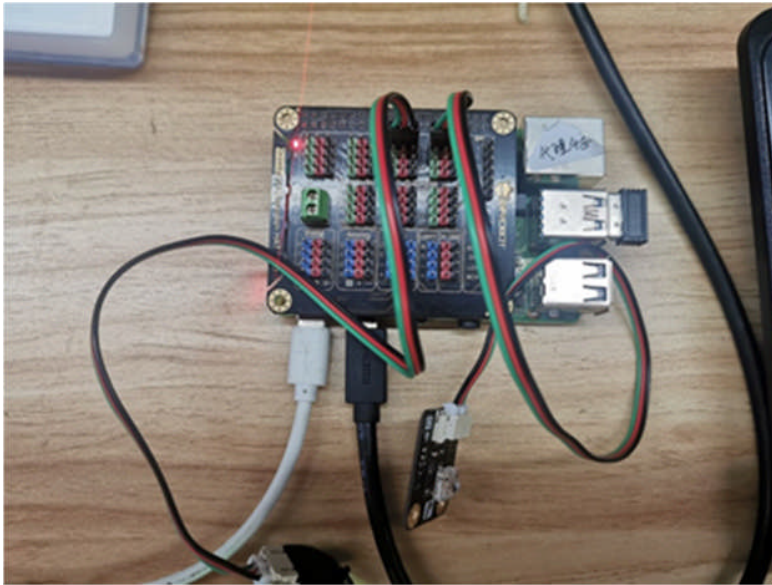
The module has two signal output modes: square wave and pulse wave, which can be freely switched through the on-board switch. The pulse wave outputs a continuous heart rate waveform, while the square wave outputs a corresponding square wave according to heart rates.

Precautions

1. This is a static heart rate sensor. Please do not move or press too tightly during its measurement.
2. This product is NOT a professional medical device and should not be used to diagnose or treat medical conditions.

Use Heart Rate Sensor on Your Raspberry Pi

- Connect the digital LED light-emitting module to the pin 12 on extension board, the heart rate sensor to the pin 8, and the analog light sensor to the analog port 0.



- Open Thonny Python IDE and copy the following program into it.

DIGITAL Sensing (Ensure switched to Digital mode of sensor)

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/10_DFR_Hearbeat_Sensor.py

Code

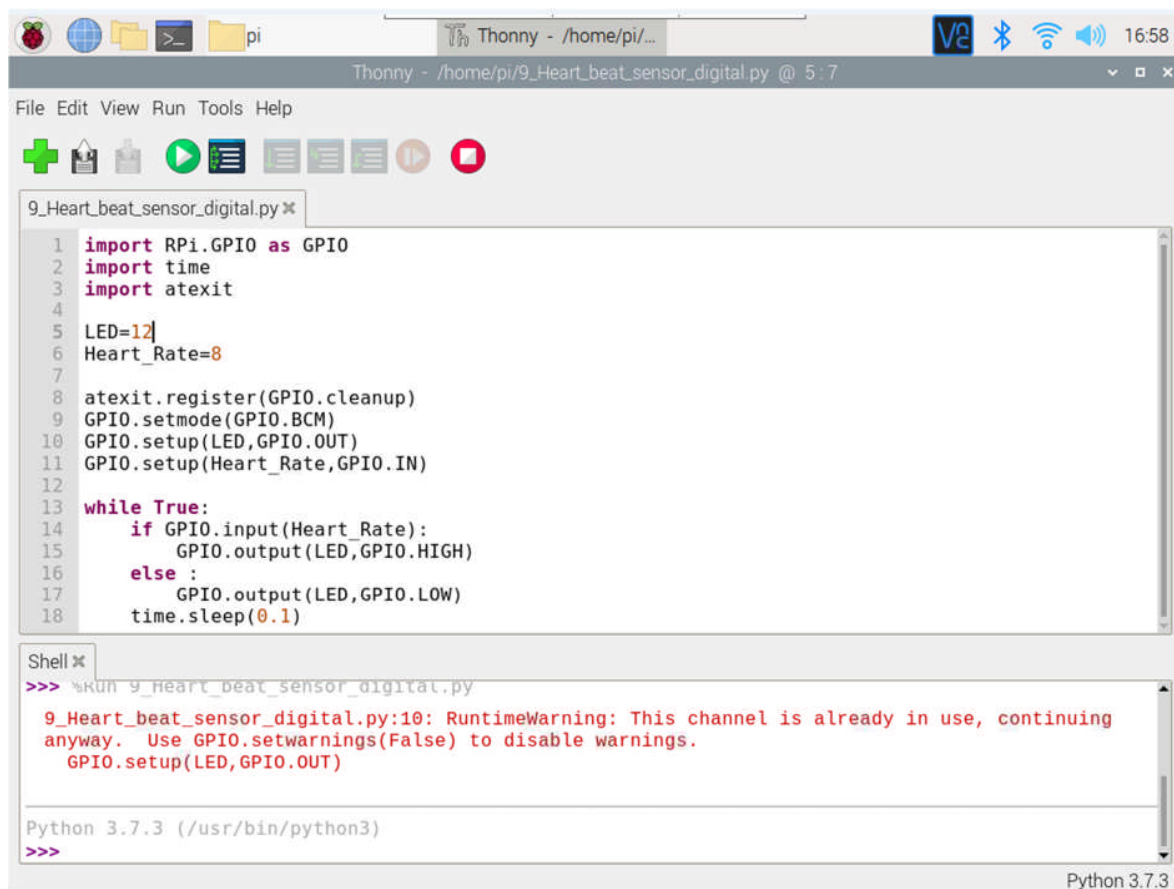
```
import RPi.GPIO as GPIO
import time
import atexit
```

```
LED=12
Heart_Rate=8
```

```
atexit.register(GPIO.cleanup)
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)
GPIO.setup(Heart_Rate,GPIO.IN)
```

```
while True:
    if GPIO.input(Heart_Rate):
        GPIO.output(LED,GPIO.HIGH)
    else :
        GPIO.output(LED,GPIO.LOW)
```

`time.sleep(0.1)`

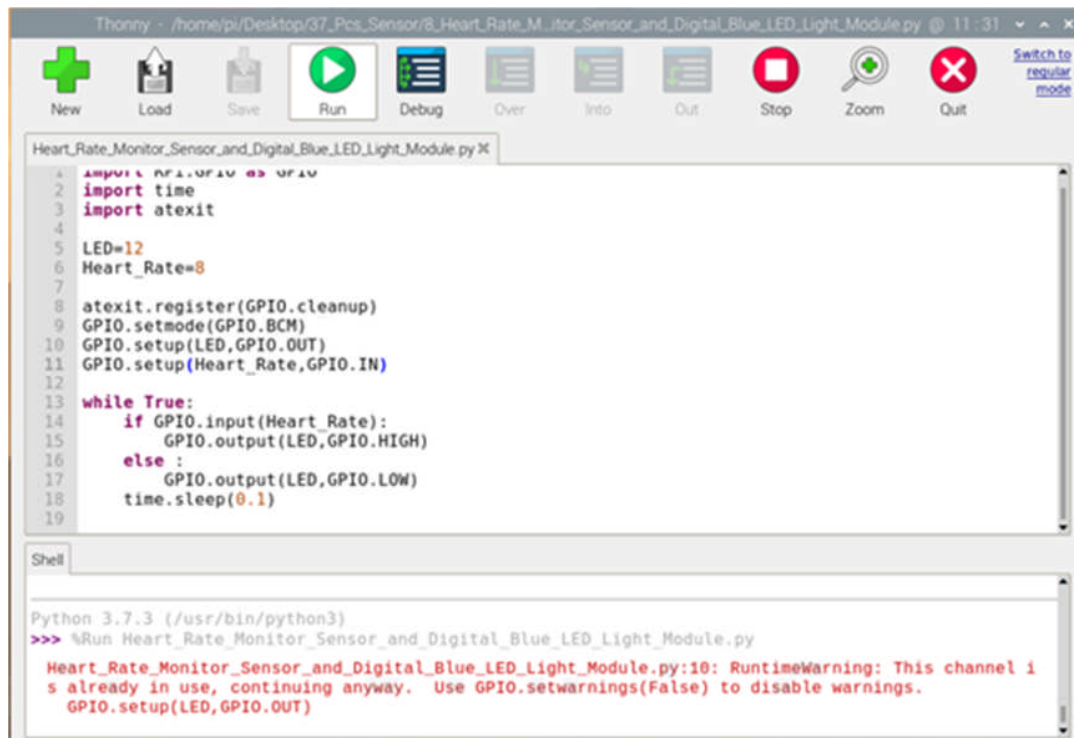


```
Thonny - /home/pi/...
Thonny - /home/pi/9_Heart_beat_sensor_digital.py @ 5:7
File Edit View Run Tools Help
+ [Icons]
9_Heart_beat_sensor_digital.py x
1 import RPi.GPIO as GPIO
2 import time
3 import atexit
4
5 LED=12
6 Heart_Rate=8
7
8 atexit.register(GPIO.cleanup)
9 GPIO.setmode(GPIO.BCM)
10 GPIO.setup(LED,GPIO.OUT)
11 GPIO.setup(Heart_Rate,GPIO.IN)
12
13 while True:
14     if GPIO.input(Heart_Rate):
15         GPIO.output(LED,GPIO.HIGH)
16     else :
17         GPIO.output(LED,GPIO.LOW)
18     time.sleep(0.1)

Shell x
>>> %RUN 9_Heart_beat_sensor_digital.py
9_Heart_beat_sensor_digital.py:10: RuntimeWarning: This channel is already in use, continuing
anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED,GPIO.OUT)

Python 3.7.3 (/usr/bin/python3)
>>>
```

Python 3.7.3



```
Thonny - /home/pi/Desktop/37_Pcs_Sensor/8_Heart_Rate_Monitor_Sensor_and_Digital_Blue_LED_Light_Module.py @ 11:31
New Load Save Run Debug Over Info Out Stop Zoom Quit Switch to regular mode

Heart_Rate_Monitor_Sensor_and_Digital_Blue_LED_Light_Module.py X
1 import sys, RPi.GPIO as GPIO
2 import time
3 import atexit
4
5 LED=12
6 Heart_Rate=8
7
8 atexit.register(GPIO.cleanup)
9 GPIO.setmode(GPIO.BCM)
10 GPIO.setup(LED,GPIO.OUT)
11 GPIO.setup(Heart_Rate,GPIO.IN)
12
13 while True:
14     if GPIO.input(Heart_Rate):
15         GPIO.output(LED,GPIO.HIGH)
16     else :
17         GPIO.output(LED,GPIO.LOW)
18     time.sleep(0.1)
19
Shell
Python 3.7.3 (/usr/bin/python3)
>>> %Run Heart_Rate_Monitor_Sensor_and_Digital_Blue_LED_Light_Module.py
Heart_Rate_Monitor_Sensor_and_Digital_Blue_LED_Light_Module.py:10: RuntimeWarning: This channel is
already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(LED,GPIO.OUT)
```

Then the LED will go on and off with your heartbeat.

Conductivity Switch Sensor

A Conductivity Sensor is a device designed to measure the electrical conductivity of a solution. Electrical conductivity is a measure of a material's ability to conduct an electric current, and in the context of a conductivity sensor, it is used to assess the concentration of ions in a solution. The higher the concentration of ions in the solution, the better it can conduct electricity.

Conductivity is a measure of a material's ability to conduct an electric current. It is a fundamental property in the field of electrical physics and plays a crucial role in various scientific, industrial, and environmental applications. The unit of conductivity is the siemens per meter (S/m) in the International System of Units.

Conductivity is related to the presence and mobility of charged particles, specifically ions, in a substance. In a material with high conductivity, electric charges can move freely, allowing the transmission of an electric current. In contrast, materials with low conductivity restrict the flow of electric charges.

Working principle of conductivity sensor

The working principle of a conductivity sensor is based on the electrical conductivity of a solution, which is a measure of its ability to conduct an electric current. Here's a simplified explanation of the working principle:

1. **Electrodes:** A typical conductivity sensor consists of two or more electrodes made of conductive materials. These electrodes are usually placed at a fixed distance from each other.
2. **Electrical Potential:** An electrical potential (voltage) is applied across the electrodes. This creates an electric field in the solution between the electrodes.
3. **Ion Movement:** In a solution, various ions (charged particles) may be present, contributing to the conductivity of the solution. These ions can be positively charged (cations) or negatively charged (anions). When the electrical potential is applied, these ions start to move towards the oppositely charged electrode.
4. **Current Flow:** As ions move through the solution towards the electrodes, they carry an electric current. The higher the concentration of ions in the solution, the greater the current flow, and consequently, the higher the conductivity.
5. **Measurement:** The sensor measures the current flowing between the electrodes. The conductivity of the solution is then calculated using Ohm's Law ($I = V/R$), where I is the current, V is the voltage, and R is the electrical resistance. Conductivity (σ) is often expressed in Siemens per meter (S/m) or microsiemens per centimeter ($\mu\text{S/cm}$).
6. **Temperature Compensation:** Since conductivity is also temperature-dependent, many conductivity sensors include temperature compensation mechanisms. This ensures that variations in temperature do not significantly affect the accuracy of the conductivity measurement.

The Conductivity Sensor Works by applying a voltage, inducing ion movement in the solution, and measuring the resulting electric current. The measured current is then used to calculate the conductivity of the solution, providing information about the concentration of ions and, by extension, the solution's electrical conductivity.

Interfacing Conductivity Switch Sensor with GrovePI & RaspberryPI

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/5_Conductivity_Sensor.py

Code

```
import time
from grovepi import *
import math
buzzer_pin = 2    #Port for buzzer
conductivity_sensor = 4
pinMode(buzzer_pin,"OUTPUT")
pinMode(conductivity_sensor,"INPUT")
digitalWrite(buzzer_pin,0)
while True:
    a=digitalRead(conductivity_sensor)
    print(a)
    if (a==0):
        digitalWrite(buzzer_pin,0)
    else:
        digitalWrite(buzzer_pin,1)
```

Interfacing Conductivity Switch Sensor with DFRobot & RaspberryPI

The conductivity switch is to conduct through two exposed conductors. When the two sides are connected through a certain medium, they are turned on. And the sensor can detect whether there is an object connection between the two poles. When the resistance value of the connected object is less than 10M, it can output a high level. You can use it to make a fruit piano, music wind chime, handshake sensor light and other interesting applications.

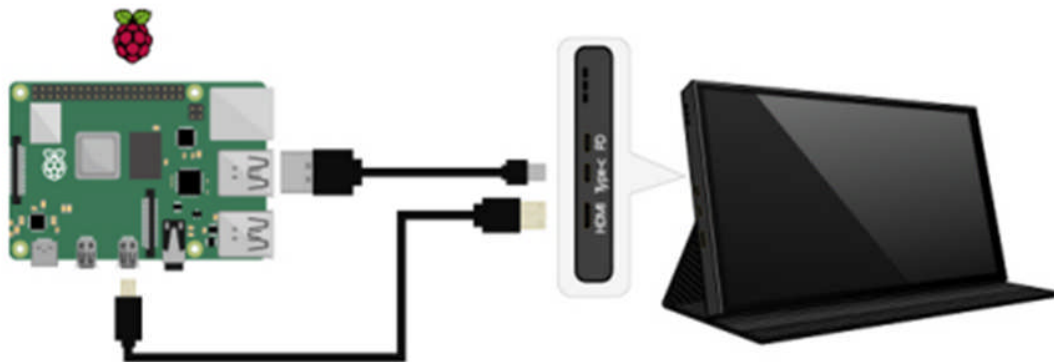
Hardware

- Gravity: 37 Pcs Sensor Set
- Raspberry Pi 4 Model B
- IO Expansion HAT for Raspberry Pi 4B/3B+
- 8GB + SanDisk Class10 SD/MicroSD Memory Card

Learning Contents

Connection

- Connect the Raspberry Pi correctly to devices such as the screen, power, keyboard and mouse.

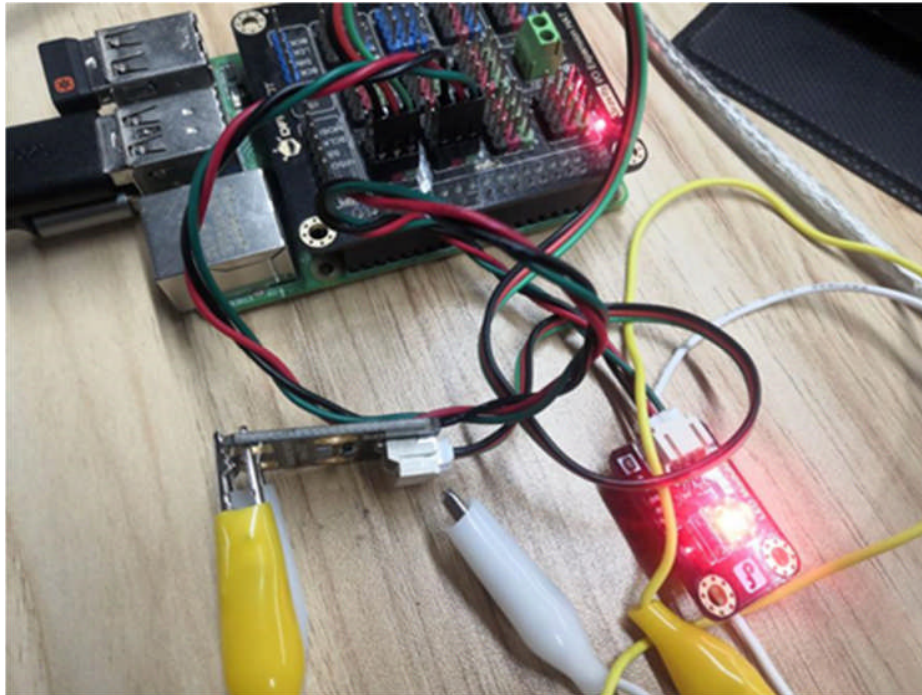


[Device Setup](#)

- Install the Raspberry Pi IO expansion board on the Raspberry Pi and connect the LED light module to the No. 12 digital port on the board, and the conductivity switch sensor module to the No. 8 digital port. When we touch to connect the two alligator clips with our hands or other conductive objects, we can see the corresponding phenomenon.



Conductivity Sensor



- Open Thonny Python IDE to copy the following program into it
Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/11_DFR_conductivity_sensor.py

Code

```
import RPi.GPIO as GPIO
import time
```

```
LED=12    # Define the pin number to which the LED is connected
sensor = 8    # Define the PIN to which the touch sensor is connected
```

```
GPIO.setmode(GPIO.BCM)    Set GPIO mode, BCM mode is common to all RPI
GPIO.setup(sensor,GPIO.IN) # Set GPIO12 to output mode
GPIO.setup(LED,GPIO.OUT)   # Set GPIO8 to input mode
GPIO.output(LED,GPIO.HIGH) #Define LED original value
```

```
while True:    Execute the following commands in an infinite loop
    key = GPIO.input(sensor)
    if (key):    # Judge whether the button is pressed
        GPIO.output(LED,GPIO.LOW)
```

```

else :
    GPIO.output(LED,GPIO.HIGH)
time.sleep(0.01)

```

```

File Edit View Run Tools Help

9_Heart_beat_sensor_digital.py 10_conductivity_sensor.py

1 import RPi.GPIO as GPIO # Import the python module provided by the Raspberry Pi
2 import time # Import time package to control flicker
3
4 LED=12 # Define the pin number to which the LED is connected
5 sensor = 8 # Define the pin number to which the touch sensor is connected
6
7 GPIO.setmode(GPIO.BCM) # Set GPIO mode, BCM mode is common to all Raspberry Pi
8 GPIO.setup(sensor,GPIO.IN) # Set GPIO12 to output mode
9 GPIO.setup(LED,GPIO.OUT) # Set GPIO8 to input mode
10 GPIO.output(LED,GPIO.HIGH) #Define LED original value
11
12 while True: # Execute the following commands in an infinite loop
13     key = GPIO.input(sensor)
14     if (key ): # Judge whether the button is pressed
15         GPIO.output(LED,GPIO.LOW) # Button pressed, start the micro vibrator
16     else : # If GPIO8 is low (that is, the button is released), execute t
17         GPIO.output(LED,GPIO.HIGH) # Not start the micro vibrator
18     time.sleep(0.01) # Delay one second, here is to control the frequency of th

Shell x
>>> %Run 10_conductivity_sensor.py
10_conductivity_sensor.py:9: RuntimeWarning: This channel is already in use, continuing anyway
. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(LED,GPIO.OUT) # Set GPIO8 to input mode

Python 3.7.3 (/usr/bin/python3)
>>>
Python 3.7.3

```

You can observe that sensor is able to detect any short circuit (or current drawing activity) happening in circuitry

Digital Touch Sensor

The human body has five sense elements which are used to interact with our surroundings. Machines also need some sensing elements to interact with there surroundings. To make this possible sensor was invented. The invention of the first manmade sensor, thermostat, dates back to 1883. In 1940s infrared sensors were introduced. Today we have sensors that can sense motion, light, humidity, temperature, smoke, etc...Analog and digital both types of sensors are available today. Sensors have brought a revolutionary change in the size and cost of various control systems. One of such sensor which can detect touch is the Touch sensor.

What is a Touch Sensor?

Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc... Touch sensors offer an intuitive user interface.



Touch Sensor

Touch sensors are also known as Tactile sensors. These are simple to design, low cost and are produced in large scale. With the advance in technology, these sensors are rapidly replacing the mechanical switches. Based on their functions there are two types of touch sensors- Capacitive sensor and Resistive sensor

Capacitive sensors work by measuring capacitance and are seen in portable devices. These are durable, robust and attractive with low cost. Resistive sensors don't depend on any electrical properties for operation. These sensors work by measuring the pressure applied to their surface.

Working Principle of Touch Sensor

Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they get activated and acts as a closed switch. When the pressure or contact is removed they act as an open switch. Capacitive touch sensor contains two parallel conductors with an insulator between them. These conductors plates act as a capacitor with a capacitance value C_0 .

When these conductor plates come in contact with our fingers, our finger acts as a conductive object. Due to this, there will be an uncertain increase in the capacitance.

A capacitance measuring circuit continuously measures the capacitance C_0 of the sensor. When this circuit detects a change in capacitance it generates a signal. The resistive touch sensors calculate the pressure applied on the surface to sense the touch. These sensors contain two conductive films coated with indium tin oxide, which is a good conductor of electricity, separated by a very small distance.

Across the surface of the films, a constant voltage is applied. When pressure is applied to the top film, it touches the bottom film. This generates a voltage drop which is detected by a controller circuit and signal is generated thereby detecting the touch.

Interfacing Touch Sensor with GrovePI & RaspberryPI

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/6_Touch_Sensor.py

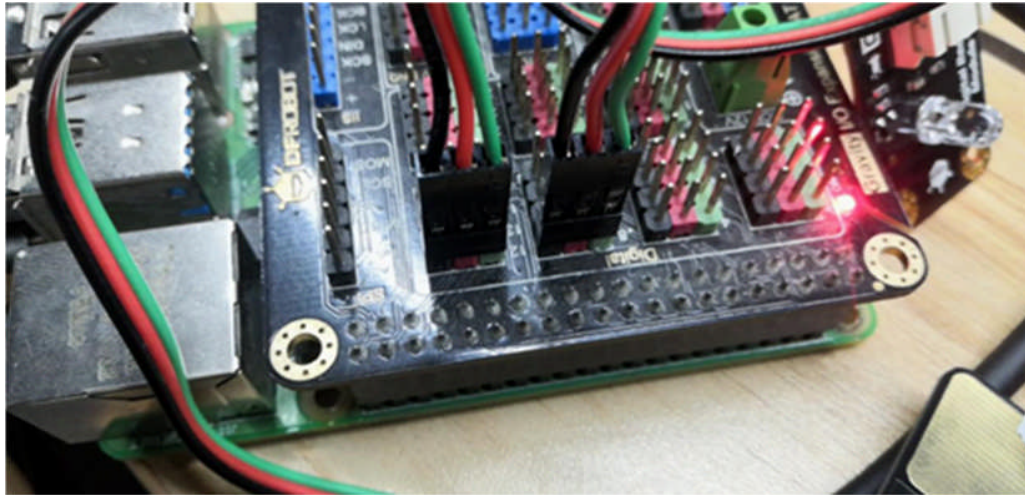
Code

```
import time
from grovepi import *
import math
buzzer_pin = 2    #Port for buzzer
touch_sensor = 4
pinMode(buzzer_pin,"OUTPUT")
pinMode(touch_sensor,"INPUT")
digitalWrite(buzzer_pin,0)
while True:
    a=digitalRead(touch_sensor)
    print(a)
    if (a==0):
        digitalWrite(buzzer_pin,0)
    else:
        digitalWrite(buzzer_pin,1)
```

Interfacing Touch Sensor with DFRobot & RaspberryPI

Steps

- Connect the Raspberry Pi correctly to devices such as the screen, power, keyboard and mouse.
- Install the Raspberry Pi IO expansion board on the Raspberry Pi, connect the LED light-emitting module digital port 12 on the board, and the digital touch sensor to port 8, then turn on the Pi.



*If there is a metal object or a finger touches the metal piece, pin 8 inputs a high level and triggers a high level on the pin 12, and the LED is on. When there is no metal object or finger touch, the pin 12 is low and the LED is off. *

Program

- Open Thonny Python IDE to copy the following program into it

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/14_DFR_touch_sensor.py

Code

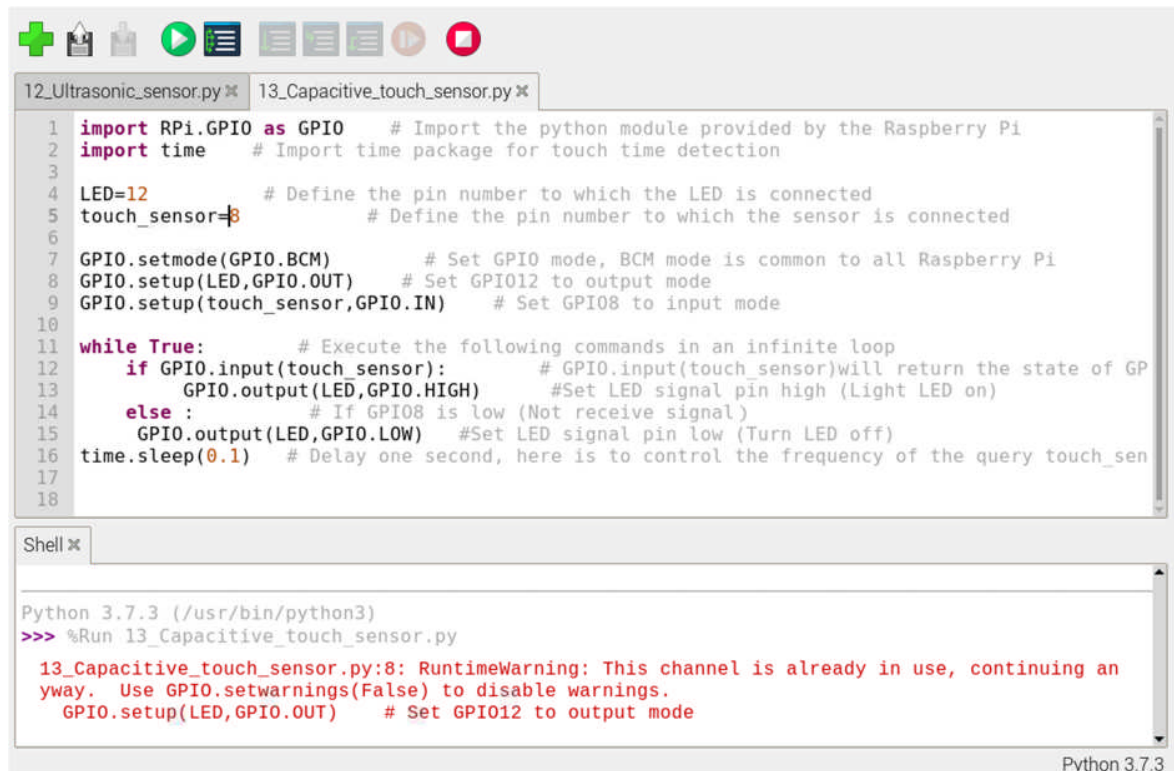
```
import RPi.GPIO as GPIO
import time # Import time package for touch time detection

LED=12      # Define the pin number to which the LED is connected
touch_sensor=8 # Define pin number to which sensor is connected

GPIO.setmode(GPIO.BCM) # Set GPIO mode, BCM mode is common
GPIO.setup(LED,GPIO.OUT) # Set GPIO12 to output mode
GPIO.setup(touch_sensor,GPIO.IN) # Set GPIO8 to input mode

while True: # Execute following commands in infinite loop
    if GPIO.input(touch_sensor):
        GPIO.output(LED,GPIO.HIGH) #Set LED signal pin high
    else : # If GPIO8 is low (Not receive signal)
        GPIO.output(LED,GPIO.LOW) #Set LED signal pin low
```

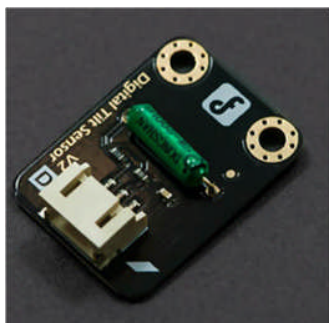
`time.sleep(0.1) # Delay one second.`



```
12_Ultrasonic_sensor.py 13_Capacitive_touch_sensor.py
1 import RPi.GPIO as GPIO # Import the python module provided by the Raspberry Pi
2 import time # Import time package for touch time detection
3
4 LED=12 # Define the pin number to which the LED is connected
5 touch_sensor=8 # Define the pin number to which the sensor is connected
6
7 GPIO.setmode(GPIO.BCM) # Set GPIO mode, BCM mode is common to all Raspberry Pi
8 GPIO.setup(LED,GPIO.OUT) # Set GPIO12 to output mode
9 GPIO.setup(touch_sensor,GPIO.IN) # Set GPIO8 to input mode
10
11 while True: # Execute the following commands in an infinite loop
12     if GPIO.input(touch_sensor): # GPIO.input(touch_sensor) will return the state of GP
13         GPIO.output(LED,GPIO.HIGH) #Set LED signal pin high (Light LED on)
14     else : # If GPIO8 is low (Not receive signal)
15         GPIO.output(LED,GPIO.LOW) #Set LED signal pin low (Turn LED off)
16     time.sleep(0.1) # Delay one second, here is to control the frequency of the query touch_sen
17
18
Shell
Python 3.7.3 (/usr/bin/python3)
>>> %Run 13_Capacitive_touch_sensor.py
13_Capacitive_touch_sensor.py:8: RuntimeWarning: This channel is already in use, continuing an
away. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(LED,GPIO.OUT) # Set GPIO12 to output mode
Python 3.7.3
```

Digital Steel Ball Inclination Sensor

This is a digital module based on a steel ball switch. It utilizes the characteristics of the steel ball to roll it towards the bottom through gravity, thereby closing or opening the switch. So it can also be used as a simple tilt sensor.



[Tilt Sensor](#)

This module can also be used in combination with the Raspberry Pi sensor expansion board. In this case, it can be used to make very interesting interactive works, which is safer than using a mercury switch.

Operating Principle

It utilizes the characteristics of the steel ball to roll it towards the bottom through gravity, thereby closing or opening the switch. This sensor has a cylindrical design that allows it to detect inclination in all 360 directions. It is a versatile and reliable choice for tilt detection in a variety of applications.

Interfacing Tilt Sensor with GrovePI & RaspberryPI

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/GrovePI_Codes/7_Tilt_Sensor.py

Code

```
import time
from grovepi import *
import math
LED = 2    #Port for buzzer
Tilt_sensor = 4
pinMode(LED,"OUTPUT")
pinMode(Tilt_sensor,"INPUT")
digitalWrite(LED,0)
while True:
    a=digitalRead(Tilt_sensor)
    print(a)
    if (a==0):
        digitalWrite(LED,0)
    else:
        digitalWrite(LED,1)
```

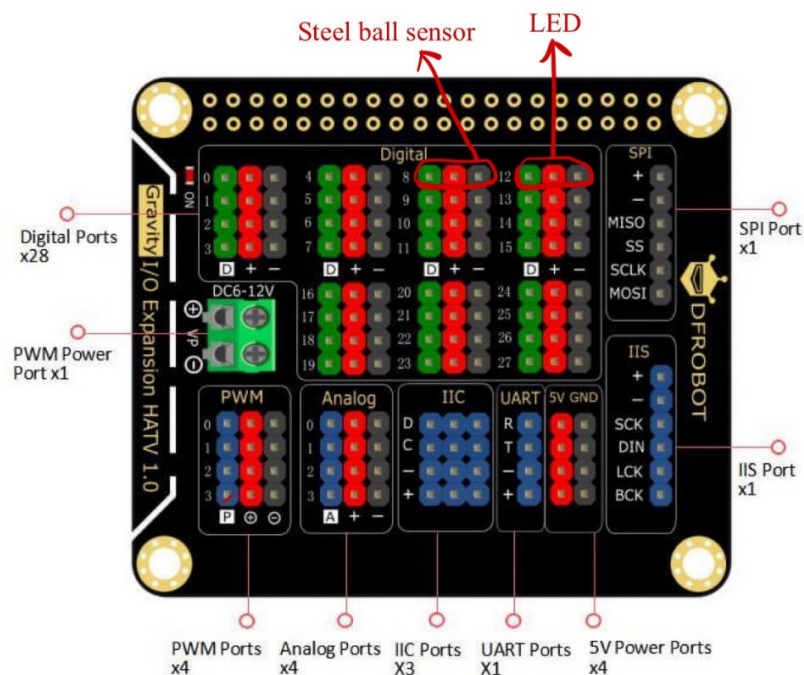
Interfacing Tilt Sensor with DFRobot & RaspberryPI

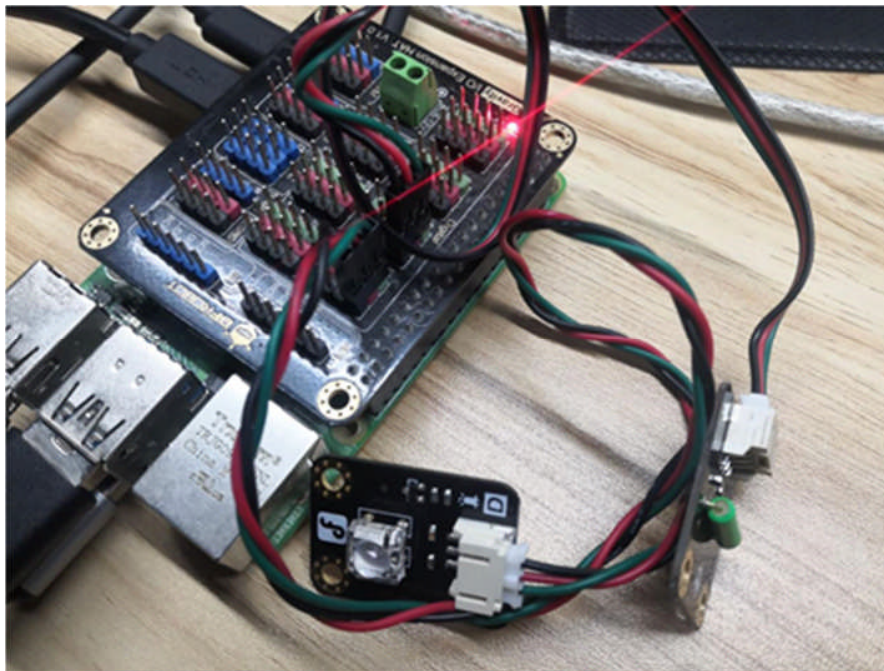
Steps

- Connect the Raspberry Pi correctly to devices such as the screen, power, keyboard and mouse.



- Connect this sensor to pin 8 on the expansion board. For convenience, connect a led switch to pin 12.





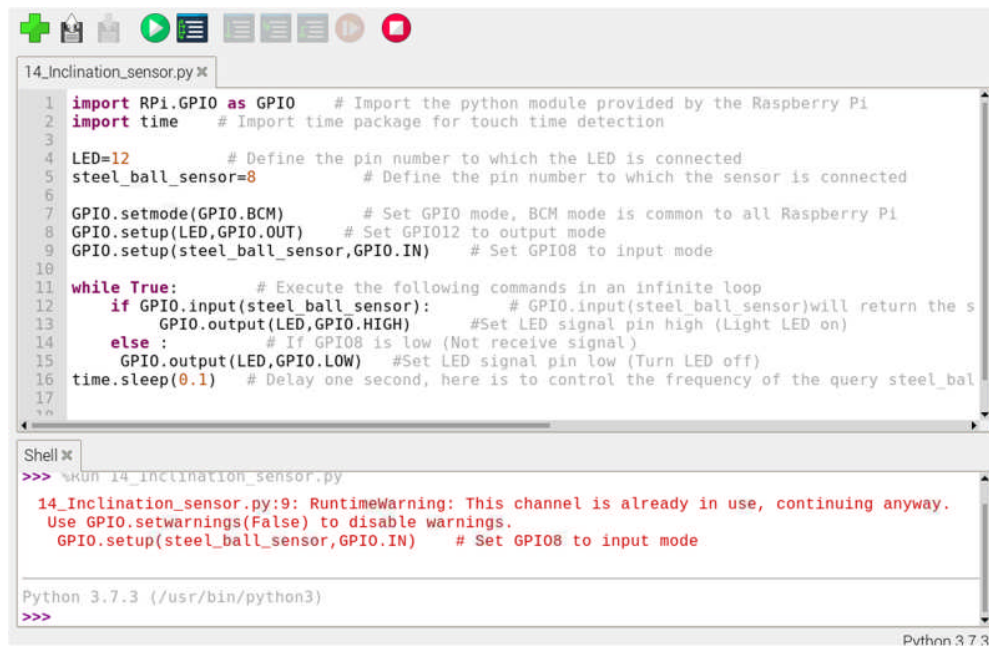
Program

Github Link

https://github.com/Code-Unnati/Advance-Course/blob/master/Module-2/Unit-3/DFRobot_IoT_Codes/15_DFR_ball_inclination_sensor.py

- Open Thonny Python IDE to copy the following program into it

```
import RPi.GPIO as GPIO
import time
LED=12
steel_ball_sensor=8
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED,GPIO.OUT)
GPIO.setup(steel_ball_sensor,GPIO.IN)
while True:
    if GPIO.input(steel_ball_sensor):
        GPIO.output(LED,GPIO.HIGH)
    else :      # If GPIO8 is low (Not receive signal)
        GPIO.output(LED,GPIO.LOW)
    time.sleep(0.1) # Delay one second
```



The image shows a code editor window titled '14_Inclination_sensor.py' with a Python script. The script imports the RPi.GPIO module as GPIO and the time module. It defines two pins: LED=12 and steel_ball_sensor=8. The LED pin is configured as an output mode, and the steel_ball_sensor pin is configured as an input mode. A while loop runs indefinitely, checking the input of the steel_ball_sensor. If it is high, the LED is turned on (GPIO.HIGH). If it is low, the LED is turned off (GPIO.LOW). A sleep function is used to delay the loop for 0.1 seconds.

```
1 import RPi.GPIO as GPIO # Import the python module provided by the Raspberry Pi
2 import time # Import time package for touch time detection
3
4 LED=12 # Define the pin number to which the LED is connected
5 steel_ball_sensor=8 # Define the pin number to which the sensor is connected
6
7 GPIO.setmode(GPIO.BCM) # Set GPIO mode, BCM mode is common to all Raspberry Pi
8 GPIO.setup(LED,GPIO.OUT) # Set GPIO12 to output mode
9 GPIO.setup(steel_ball_sensor,GPIO.IN) # Set GPIO8 to input mode
10
11 while True: # Execute the following commands in an infinite loop
12     if GPIO.input(steel_ball_sensor): # GPIO.input(steel_ball_sensor) will return the s
13         GPIO.output(LED,GPIO.HIGH) #Set LED signal pin high (Light LED on)
14     else : # If GPIO8 is low (Not receive signal)
15         GPIO.output(LED,GPIO.LOW) #Set LED signal pin low (Turn LED off)
16     time.sleep(0.1) # Delay one second, here is to control the frequency of the query steel_bal
17
18
```

Below the code editor is a terminal window titled 'Shell'. It shows the command to run the script: `>>> python 14_inclination_sensor.py`. The output shows a runtime warning: `14_Inclination_sensor.py:9: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.` followed by the line `GPIO.setup(steel_ball_sensor,GPIO.IN) # Set GPIO8 to input mode`. The terminal also shows the Python version: `Python 3.7.3 (/usr/bin/python3)`.