Single-board Computer - Raspberry-Pi & Programming

Dr. Soharab Hossain Shaikh

Associate Professor, CSE, SoET, BMU

1

Plan for the upcoming 4 Sessions

Monday, 20th September (completed yesterday)
- Setting the Context (Why RaspberryPi?)
- Introduction, Installation & Setup - RaspberryPi

Tuesday, 21st September (today)
- A Quick Intro to Python Programming
- Interfacing Sensors to RaspberryPi and GPIO Programming

Monday, 27th September (next week)
- Image Processing with Python - OpenCV
- Exploring new tools/libraries

Tuesday, 28th September (next week)
- Build Web-interface with Python - Flask
- Run a full-fledged applications with Raspberry-Pi

2

Plan for the upcoming 4 Sessions

Monday, 20th September (completed yesterday)
- Setting the Context (Why RaspberryPi?)
- Introduction, Installation & Setup - RaspberryPi

Tuesday, 21st September (today)
- A Quick Intro to Python Programming
- Interfacing Sensors to RaspberryPi and GPIO Programming

Monday, 27th September (next week)
- Image Processing with Python - OpenCV
- Exploring new tools/libraries

Tuesday, 28th September (next week)
- Build Web-interface with Python - Flask
- Run a full-fledged applications with Raspberry-Pi

3

Session - 2

- Intro to Python Programming
- Interfacing Sensors & GPIO Programming

4

Programming Language

- **Syntax** (Grammar/Rule for writing statements)

e.g. x = x + 5;

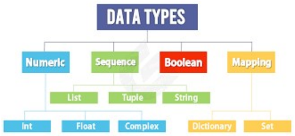
- **Semantics** (Meaning of a statement)

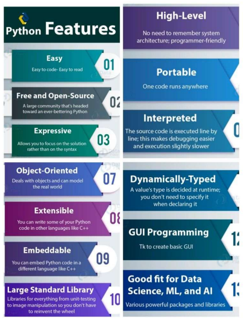
"The value of the variable x is incremented by 5"

- **Structured Programming:**
- **Sequence** ***(default sequential flow of execution of the statements)***
- **Conditional Execution** ***(e.g. if statement)***
- **Repetition/Iteration** ***(loops)***

5

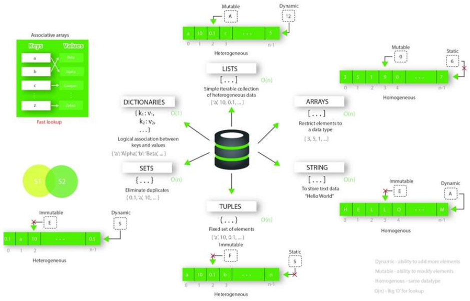
Python Programming




Python Features
Easy
Easy to code- Easy to read
01
Free and Open-Source
A large community that's headed toward an ever-bettering Python
02
Expressive
Allows you to focus on the solution rather than on the syntax
03
Object-Oriented
Deals with objects and can model the real world
07
Extensible
You can write some of your Python code in other languages like C++
08
Embeddable
You can embed Python code in a different language like C++
09
Large Standard Library
Libraries for everything from unit-testing to image manipulation so you don't have to reinvent the wheel
10
High-Level
No need to remember system architecture; programmer-friendly
Portable
One code runs anywhere
Interpreted
The source code is executed line by line; this makes debugging easier and execution slightly slower
Dynamically-Typed
A value's type is decided at runtime; you don't need to specify it when declaring it
GUI Programming
Tk to create basic GUI
12
Good fit for Data Science, ML, and AI
Various powerful packages and libraries
13

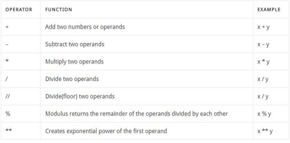
6

Python Built-in Data Structures

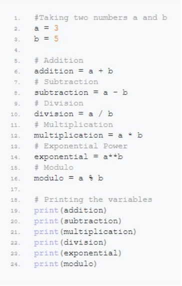


7

Arithmetic Operators

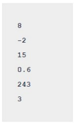


OPERATOR	FUNCTION	EXAMPLE
+	Add two numbers or operands	x + y
-	Subtract two operands	x - y
*	Multiply two operands	x * y
/	Divide two operands	x / y
//	Divide(floor) two operands	x / y
%	Modulus returns the remainder of the operands divided by each other	x % y
**	Creates exponential power of the first operand	x ** y



```
#Taking two numbers a and b
a = 3
b = 5

# Addition
addition = a + b
# Subtraction
subtraction = a - b
# Division
division = a / b
# Multiplication
multiplication = a * b
# Exponential Power
exponential = a**b
# Modulo
modulo = a % b

# Printing the variables
print(addition)
print(subtraction)
print(multiplication)
print(division)
print(exponential)
print(modulo)
```

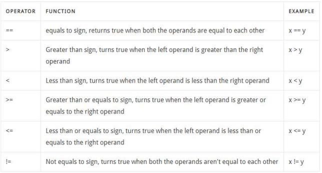
Output



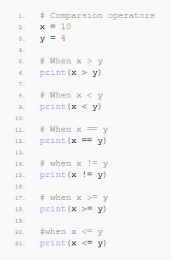
```
8
-2
15
0.6
243
3
```

8

Comparison Operators

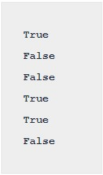


OPERATOR	FUNCTION	EXAMPLE
==	equals to sign, returns true when both the operands are equal to each other	x == y
>	Greater than sign, turns true when the left operand is greater than the right operand	x > y
<	Less than sign, turns true when the left operand is less than the right operand	x < y
>=	Greater than or equals to sign, turns true when the left operand is greater or equals to the right operand	x >= y
<=	Less than or equals to sign, turns true when the left operand is less than or equals to the right operand	x <= y
!=	Not equals to sign, turns true when both the operands aren't equal to each other	x != y



```
# Comparsion operators
x = 10
y = 4

# When x > y
print(x > y)

# When x < y
print(x < y)

# When x == y
print(x == y)

# when x != y
print(x != y)

# when x >= y
print(x >= y)

#when x <= y
print(x <= y)
```

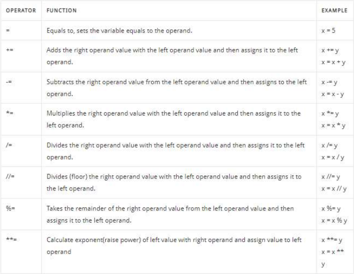
Output



```
True
False
False
True
True
False
```

9

Assignment Operators



OPERATOR	FUNCTION	EXAMPLE
=	Equals to, sets the variable equals to the operand.	x = 5

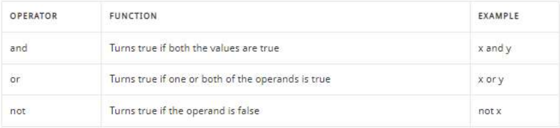
+=	Adds the right operand value with the left operand value and then assigns it to the left operand.	x += y
		x = x + y

-=	Subtracts the right operand value from the left operand value and then assigns to the left operand.	x -= y
		x = x - y

*=	Multiplies the right operand value with the left operand value and then assigns it to the left operand.	x *= y x = x * y
/=	Divides the right operand value with the left operand value and then assigns it to the left operand.	x /= y x = x / y
//=	Divides (floor) the right operand value with the left operand value and then assigns it to the left operand.	x //= y x = x // y
%=	Takes the remainder of the right operand value from the left operand value and then assigns it to the left operand.	x %= y x = x % y
**=	Calculate exponent(raise power) of left value with right operand and assign value to left operand	x **= y x = x ** y

10

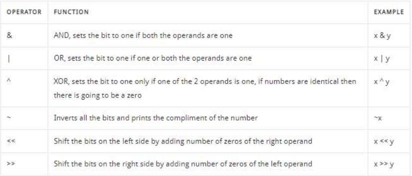
Logical Operators



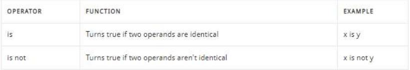
OPERATOR	FUNCTION	EXAMPLE
and	Turns true if both the values are true	x and y
or	Turns true if one or both of the operands is true	x or y
not	Turns true if the operand is false	not x

11

Bit-wise & Special Operators



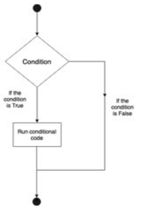
OPERATOR	FUNCTION	EXAMPLE
&	AND, sets the bit to one if both the operands are one	x & y
|	OR, sets the bit to one if one or both the operands are one	x | y
^	XOR, sets the bit to one only if one of the 2 operands is one, if numbers are identical then there is going to be a zero	x ^ y
~	Inverts all the bits and prints the compliment of the number	~x
<<	Shift the bits on the left side by adding number of zeros of the right operand	x << y
>>	Shift the bits on the right side by adding number of zeros of the left operand	x >> y

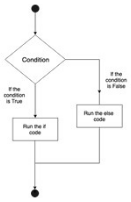


OPERATOR	FUNCTION	EXAMPLE
is	Turns true if two operands are identical	x is y
is not	Turns true if two operands aren't identical	x is not y

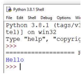
12

if Statement





```
x=10

if x<20:
    print('Hello')
else:
    print('Hi')
```



```
Python 3.8.1 (tags/v3
tel)] on win32
Type "help", "copyric
>>>
================== F
Hello
>>> |
```

Syntax:

```
if expression:
     statement
```

Syntax:

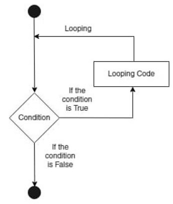
```
if expression:
     statement
else:
     statement
```

Syntax:

```
if expression1:
     statement
elif expression2:
     statement
elif expression3:
     statement
else:
     statement
```

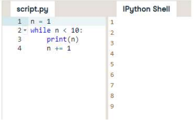
13

while Loop



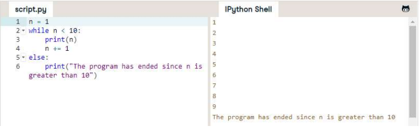
Syntax of While Loop

```
while expression:
     statements
```



```
n = 1
while n < 10:
    print(n)
    n += 1
```

```
1
2
3
4
5
6
7
8
9
```

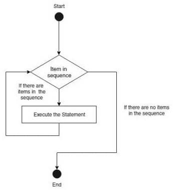


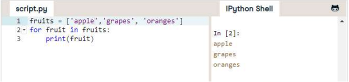
```
n = 1
while n < 10:
    print(n)
    n += 1
else:
    print("The program has ended since n is greater than 10")
```

```
1
2
3
4
5
6
7
8
9
The program has ended since n is greater than 10
```

14

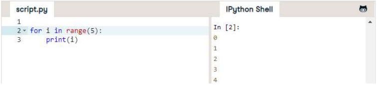
for Loop


Start
Item in sequence
If there are items in the sequence
Execute the Statement
If there are no items in the sequence
End



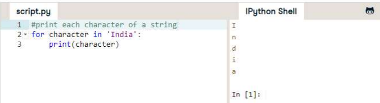
```
fruits = ['apple','grapes', 'oranges']
for fruit in fruits:
    print(fruit)
```

```
In [2]:
apple
grapes
oranges
```



```

for i in range(5):
    print(i)
```

```
In [2]:
0
1
2
3
4
```



```
#print each character of a string
for character in 'India':
    print(character)
```

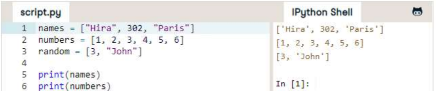
```
I
n
d
i
a

In [1]:
```

Syntax of For Loop

```
for variable in sequence:
     statement 1
     statement 2 ...
```

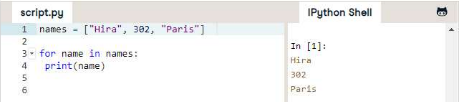
15

List



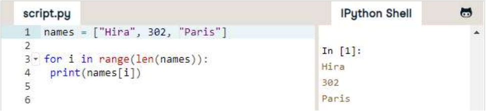
```
names = ["Hira", 302, "Paris"]
numbers = [1, 2, 3, 4, 5, 6]
random = [3, "John"]

print(names)
print(numbers)
```

```
['Hira', 302, 'Paris']
[1, 2, 3, 4, 5, 6]
[3, 'John']

In [1]:
```



```
names = ["Hira", 302, "Paris"]

for name in names:
  print(name)


```

```
In [1]:
Hira
302
Paris
```

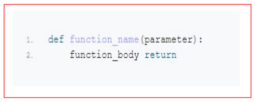


```
names = ["Hira", 302, "Paris"]

for i in range(len(names)):
  print(names[i])


```

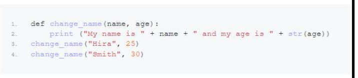
```
In [1]:
Hira
302
Paris
```

16

Function

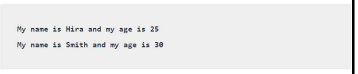


```
def function_name(parameter):
    function_body return
```

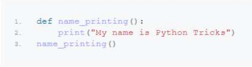


```
def change_name(name, age):
    print ("My name is " + name + " and my age is " + str(age))
change_name("Hira", 25)
change_name("Smith", 30)
```

Output

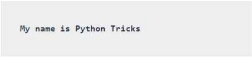


```
My name is Hira and my age is 25
My name is Smith and my age is 30
```



```
def name_printing():
    print("My name is Python Tricks")
name_printing()
```

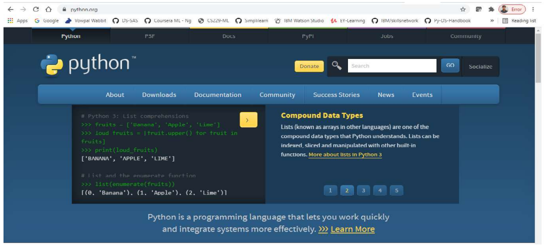
Output



```
My name is Python Tricks
```

17

Official Python Resources



```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. More about lists in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

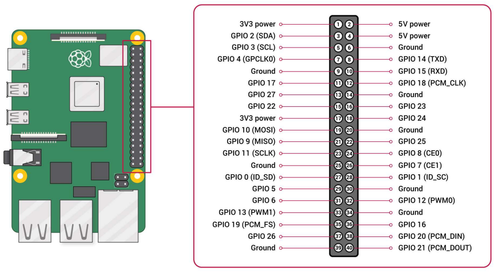
https://www.python.org/
https://docs.python.org/3/
https://docs.python.org/3/tutorial/index.html

18

Connecting Sensors with Raspberry-Pi & Programming the GPIO Pins

19

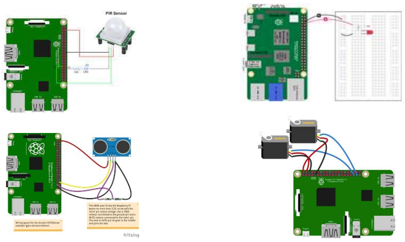
GPIO Pins



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

20

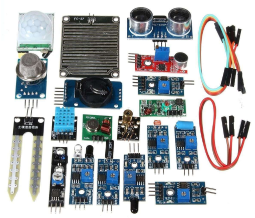
Connecting Sensors and Program them with GPIO



https://pypi.org/project/RPi.GPIO/
https://www.raspberrypi.org/documentation/usage/gpio/python/README.md

21

Sensors



1.KY-008 laser sensor module
Operating voltage: 5V; Specifications: 15 * 24 mm; Light source wavelength: 650 nm

2.Obstacle avoidance sensor module
VCC: 3.3V-5V external voltage (can be directly connected to the 5v and 3.3v microcontroller MCU)
GND: External GND; OUT: Small board digital output interface (0 and 1)
Connection method: VCC-VCC; GND-GND; OUT-IO; Board size: 3.2 x 1.4cm

3.Flame sensor module
Operating voltage: 3.3V-5V; Output form: DO digital switching output (0 and 1) and AO analog voltage output
Size: 3.2cm x 1.4cm ; Detect flame or wavelength range (approx): 760 nm to 1100 nm, lighter flame test distance is 80cm, the greater flame, the farther test distance ; Detection angle is about 60 degree, particularly sensitive to flame photometry
Adjustable sensitivity

4.315M wireless transceiver module
Operating voltage: DC5V; Quiescent current: 4mA
Receiving frequency: 315MHz ; Receiving sensitivity: -105DB; Size: 30 x 14 x 7mm

5.Emission head parameters
Transmission distance: 20-200 meter (different voltage, different effect)
Operating voltage: 3.5-12V; Dimension: 19 x 19mm; Work: AM; Transfer rate: 4KB / S
Transmitting power: 10mW; Transmission frequency: 315M; Pinout left → right: (DATA, VCC, GND)

6.One channel TCRT5000 tracing module
Detection distance: about 10mm; Operating voltage: 5V, black line output low level, white line output high level

7.Vibration sensor module
Operating voltage: 3.3V-5V; Output type: Digital switching output (0 and 1); Small PCB board size: 3.2cm x 1.4cm

8.MQ-2 gas sensor module
Dimension: 32 x 22 x 27mm; The main chip: LM393, ZYMQ-2 gas sensor
Operating voltage: 5V DC; Dual signal output (analog output, and TTL level output)
TTL output valid signal is low level. (When the output low level signal light can be directly connected microcontroller)
Analog output: 0 ~ 5V voltage, the higher the concentration the higher the voltage.

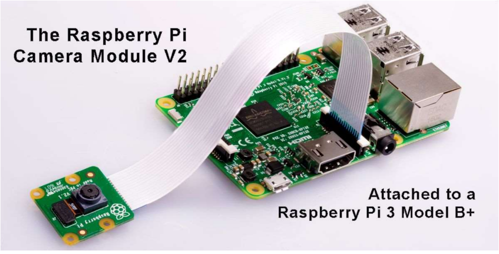
9.Sound sensor module
Operating voltage: 3.3V-5V; Small PCB board size: 3.4cm x 1.6cm

10.DHT11 temperature and humidity sensor module
VDD: 3.3-5.5V DC; DATA: Serial data; NC: No connection; GND: Grounding, negative power supply

11.HC-SR04 ultrasonic sensor module
Operating voltage: 5V; Ultra-small quiescent current: less than 2mA; Induction angle: not more than 15 degrees; Detection distance: 2cm-400cm; High precision: up to 0.3cm

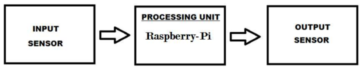
22

Camera connected to the R-Pi Board



23

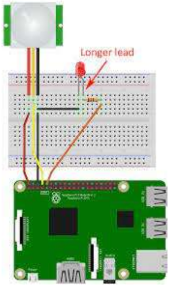
Sensors



INPUT SENSOR

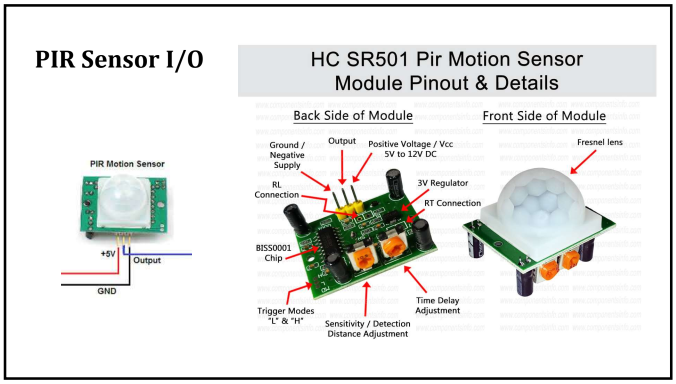
PROCESSING UNIT
Raspberry-Pi

OUTPUT SENSOR



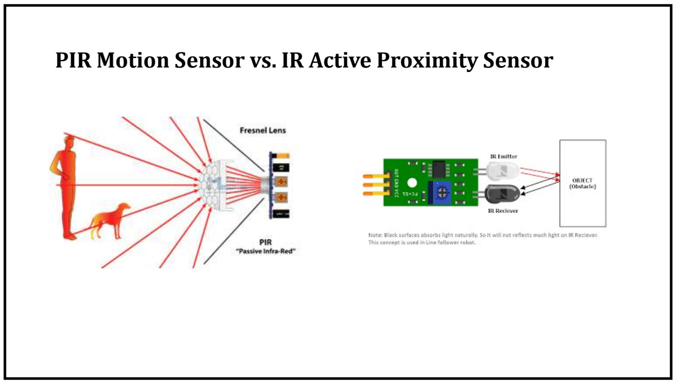
- Sensors can be powered up from the Raspberry-Pi board (5V/3.5V & GND).
- Input and Outputs pins of the Sensors can be connected with the GPIO pins of the Raspberry-Pi board.
- The GPIO pins can be programmed with e.g. Python to automate certain tasks (e.g. set an LED/Buzzer on when motion/proximity to objects detected).

24



PIR Sensor I/O

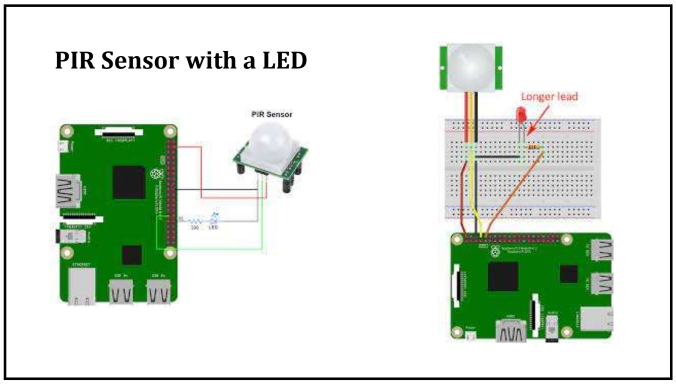
25



PIR Motion Sensor vs. IR Active Proximity Sensor

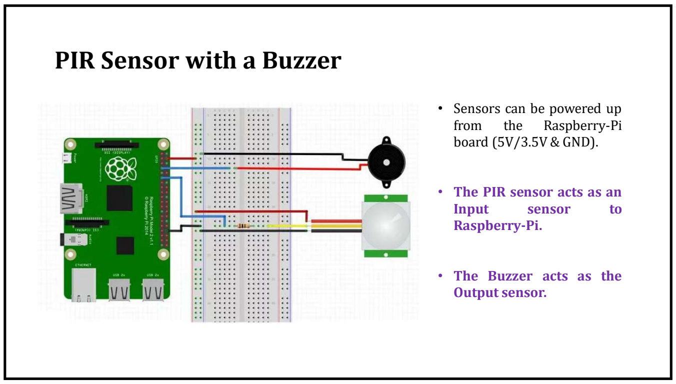
Note: Black surfaces absorbs light naturally. So it will not reflects much light on IR Reciever. This concept is used in Line follower robot.

26



PIR Sensor with a LED

27

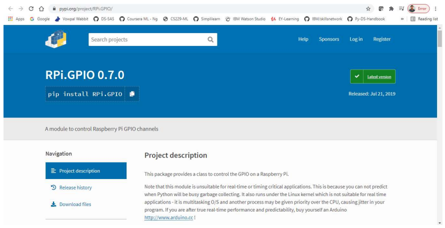


PIR Sensor with a Buzzer

- Sensors can be powered up from the Raspberry-Pi board (5V/3.5V & GND).
- **The PIR sensor acts as an Input sensor to Raspberry-Pi.**
- **The Buzzer acts as the Output sensor.**

28

GPIO Programming Library (Python)



RPi.GPIO 0.7.0

```
pip install RPi.GPIO
```

Released: Jul 21, 2019

A module to control Raspberry Pi GPIO channels

Navigation

Project description

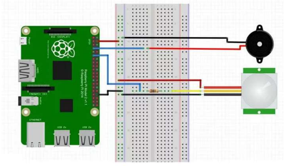
This package provides a class to control the GPIO on a Raspberry Pi.

Note that this module is unsuitable for real-time or timing critical applications. This is because you can not predict when Python will be busy garbage collecting. It also runs under the Linux kernel which is not suitable for real time applications - it is multitasking O/S and another process may be given priority over the CPU, causing jitter in your program. If you are after true real-time performance and predictability, buy yourself an Arduino http://www.arduino.cc !

https://pypi.org/project/RPi.GPIO/
https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/
https://sourceforge.net/p/raspberry-gpio-python/wiki/BasicUsage/
https://pinout.xyz/

29

Program the GPIO Pins

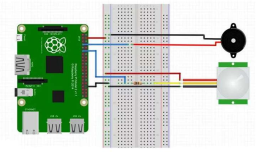


```
import RPi.GPIO as GPIO
import time

pir_sensor = 11
piezo = 7

GPIO.setmode(GPIO.BOARD)

GPIO.setup(piezo,GPIO.OUT)

GPIO.setup(pir_sensor, GPIO.IN)

current_state = 0
```

https://pimylifeup.com/raspberry-pi-motion-sensor/

30

Program the GPIO Pins



```
try:
    while True:
        time.sleep(0.1)
        current_state = GPIO.input(pir_sensor)
        if current_state == 1:
            print("GPIO pin %s is %s" % (pir_sensor, current_state))
            GPIO.output(piezo,True)
            time.sleep(1)
            GPIO.output(piezo,False)
            time.sleep(5)
except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

https://pimylifeup.com/raspberry-pi-motion-sensor/
https://raspi.tv/2013/rpi-gpio-basics-3-how-to-exit-gpio-programs-cleanly-avoid-warnings-and-protect-your-pi

31

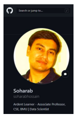
Summary

Today we have discussed the following:

- **A Quick Intro to Python Programming**
- **Interfacing different sensors with Raspberry-Pi**
- **GPIO pins and**
- **Programming the GPIO pins with Rpi.GPIO Python library**

32

33



Soharab
soharabhossain
Ardent Learner - Associate Professor, CSE, BMU | Data Scientist

Thank You

Class material will be uploaded here:
https://github.com/soharabhossain/JoE_2021

34