Experiment No: 3

Aim: Study of connectivity & configuration of Raspberry - Pi / Beagle Board Circuit with basic peripherals, LEDs, understanding GPIO and its use in program.

Theory:

Connectivity and configuration of Raspberry - Pi Guides to configure Raspberry Pi.

1. raspi-config:

The Raspberry Pi configuration tool in Raspbian, allowing you to easily enable features such as the camera & to change your specific settings such as keyboard layout.

2. config.txt:

The Raspberry Pi configuration file.

3. Wireless:

configuring your Pi to connect to a wireless network using the Raspberry Pi 3 and Pi zero w's in built wireless connectivity, or a USB wireless dongle.

4. Wireless access point:

configuring your Pi as a wireless network access point using the Raspberry Pi 3 & Pi zero w's. In built wireless connectivity, or a USB wireless dongle.

5. Audio config:

Switch your audio output between HDMI & the 3.5mm jack.

6. Camera config:
Installing and setting up the Raspberry Pi camera board.

7. External Storage config:
Mounting & setting up external storage on a Raspberry Pi.

8. Localisation:
Setting up your Pi to work in your local language / time zone.

9. Default pin configuration:
changing the default pin states.

10. Device Tree config:
Device Trees, overlays & parameters.

11. Kernel command line:
The linux kernel accepts a command line of parameters during boot. on the Raspberry Pi, this command line is defined in a file in the boot partition called cmdline.txt. This is a simple text file that can be edited using any text editor.

12. UART Configuration:
The SoCs used on the Raspberry Pis have two built-in UARTs, a PL011 & a mini UART. They are implemented using diff° hardware blocks, so they are slightly diff° charac? However both are 3.3V devices, which means extra care must be taken when connecting up to an RS232 or other system that utilizes diff° voltage levels. An adapter must be used

to convert the voltage levels between the two protocols.

13. Screensaver:

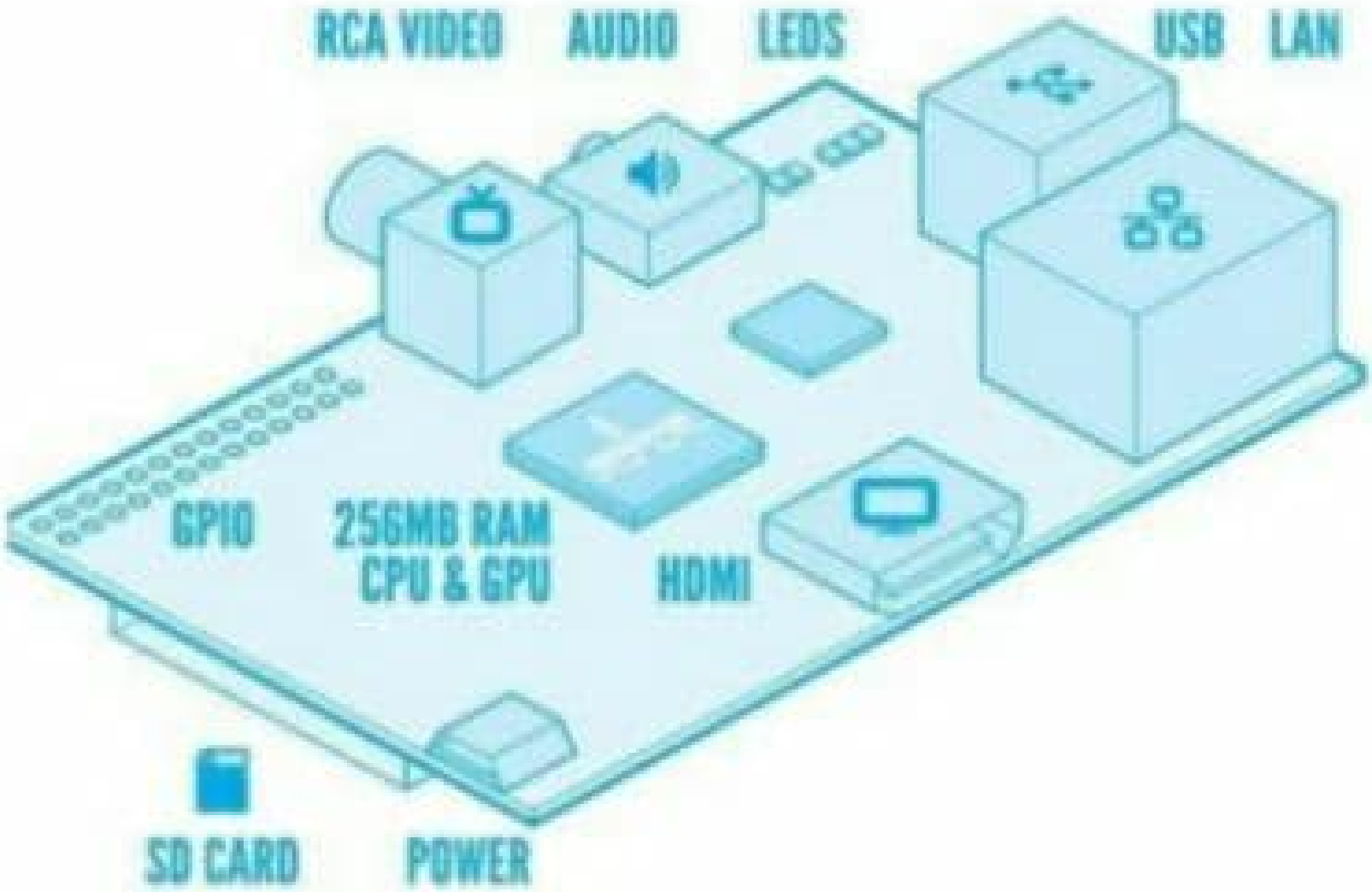
If you are using the Raspberry Pi solely on the console, you need to set the console blanking. The current setting, in seconds can be displayed using.

Add consoleblank=0 to turn screen blanking off completely, or edit it to set the number of seconds of inacitivity before the console with blank.

Connectivity of Raspberry Pi

Connectivity is truely superb for such a tiny device especially on the B version of the Raspberry Pi. There are two USB 2.0 ports that can be used to hook up peripherals or adapters, & this can be further expanded with a powered hub.

Diagram


RCA VIDEO
AUDIO
LEDS
USB
LAN
GPIO
256MB RAM
CPU & GPU
HDMI
SD CARD
POWER

for viedo, there's a full-size HDMI Pout, Making the Raspberry Pi compatiable with practically every monitor, Tv & other display out there. for older displays that don't support digital connectivity, the Raspberry Pi even has an analogue composite RCA video o/p. There are headers for further expansion, including the ability to hook up a camera or screen. All of these ports are found at the top of the board while the SD card reader is located at the bottom.

GPIO Mode:

The GPIO.BOARD option specifies that you are refering to the pins by the number of the pin the plug. i-e the numbers printed on the board 4 in the middle of the diagrams below.

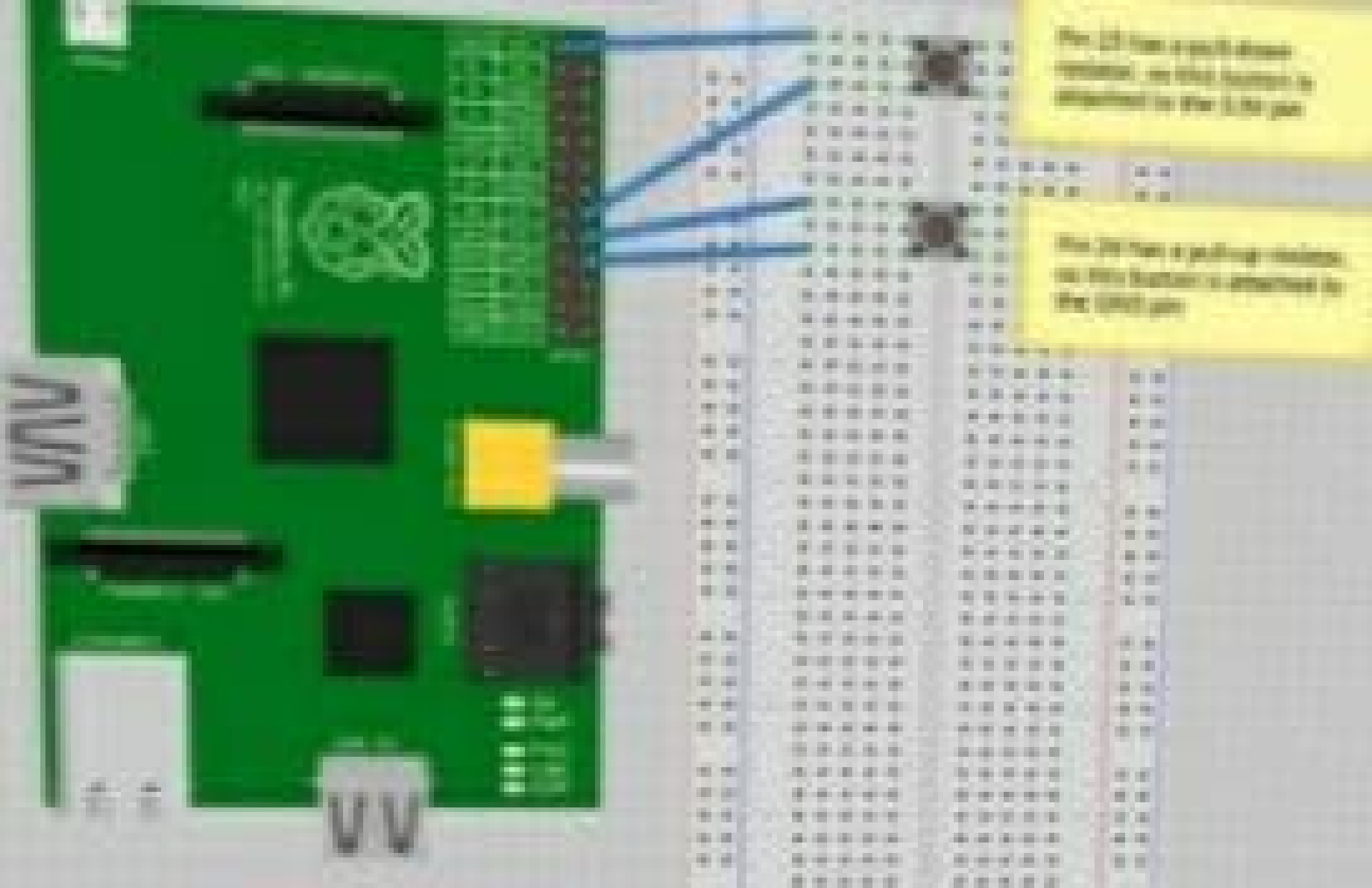
The GPIO.BCM option means that you are refering to the pins by the "Broadcom SOC channel" number, these are the number after "GPIO" in the green rectangles around the outside of the below diag:

— The Model B+ uses the same numbering as the Model B & 2.0, & adds new pins. (27-40).

— The Raspberry Pi zero, Pi 2B & Pi 3B use the same numbering as the B+.

Building a Circuit:

In the circuit show below two momentary switches are wired to GPIO pins 23 & 24. The switch on pin 23 is tied to 3.3V, while the switch on pin 24 is tied to ground. The reason for this is that the Raspberry Pi as internal pull-up & pull-down resistors that can be specified when the pin declarations are made.



To set up this pins, write:

- GPIO.setup(23, GPIO, PULL-UP-down = GPIO.PUD-DOWN).
- GPIO.setup(24, GPIO, PULL-UP-down = GPIO.PUD-UP)

This will enable a pull-down resistor on pin 23 & a pull-up resistor on pin 24. Now, let's check to see if we can read them. we'll also need to put these

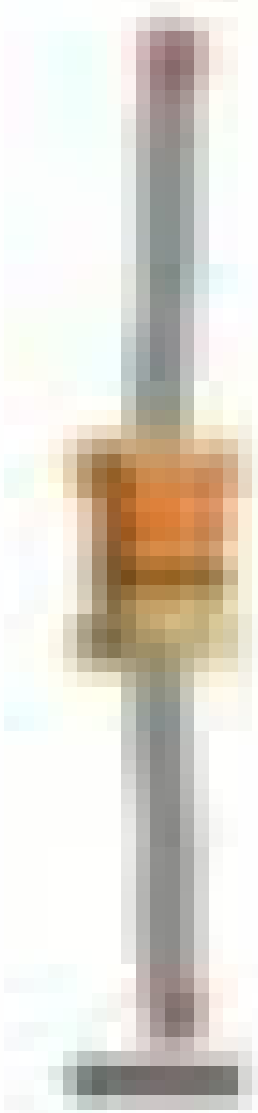
inside of a loop, so that it is constantly checking the pin voltage:

The code so far looks like this:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull-up-down=
    GPIO.PUD_DOWN)
GPIO.setup(24, GPIO.IN, pull-up-down=
    GPIO.PUD_UP)
while true:
if (GPIO.input(23)==1):
print("Button 1 Pressed")
if (GPIO.input(24)==0):
print("Button 2 pressed")
GPIO.cleanup().
```

The indents in Python are important when using loops, so be sure to include them. You also must run your skripts as "sudo" to access the GPIO pin. If you don't use this, then the GPIO pins will remain at whatever state they were last set.

Reuters



Resistor:

diag

You must ALWAYS use resistors to connect LED's up to the GPIO pins of the Raspberry Pi. The Raspberry Pi can only supply a small current. The LEDs will want to draw more, & if allowed to they will burn out the Raspberry Pi.

Resistors are a way of limiting the amount of electricity going through a circuit; specifically they limit the amount of 'current' that is allowed to flow. The ~~value~~ is also called ohm(Ω). The value of a resistor is marked with

Colored bands along the length of the resistor body:

Jumper wires:

jumper wires are used on breadboards to 'jump' from one connection to another.

- The ones you will be using in this circuit have different connectors on each end.
- The end with the 'pin' will go into the breadboard.
- The end with the piece of plastic with a hole in it will go onto the Raspberry Pi's GPIO pins.

Conclusion:

Thus we have studied Connectivity & Configuration of Raspberry Pi 4 also use of GPIO.

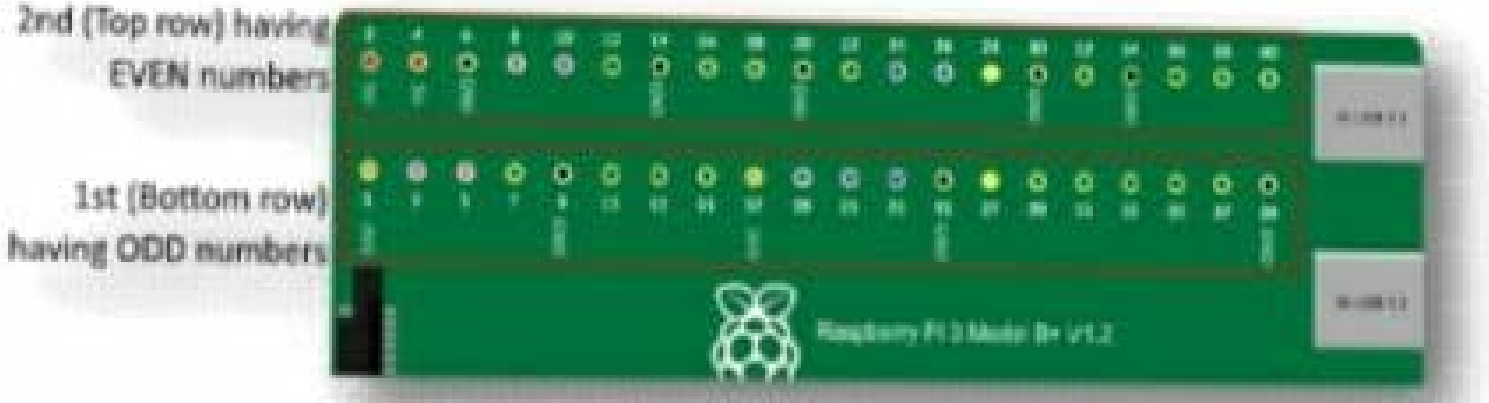
Jumper Wires

Understanding GPIO pins on Raspberry Pi board and its use in program.

Aim/objectives:

1. To understand the GPIO pins of Raspberry-Pi 3.
2. To program the GPIO pins of Raspberry-Pi-3 using Python.

1.Raspberry Pi 3 Model B is the latest version of raspberry pi board.
2.It is released on 29 February.
3. The above figure shows the Raspberry Pi 3 Model B and It's GPIO pins
4.General-purpose input/output (GPIO) is a generic pin on an integrated circuit orcomputer board whose behavior—including whether it is an input or outputpin—is controllable by the user at run time.



5. There are 40 pins available on board of Raspberry Pi 3 Model B.
6. The pins are arranged in a 2 X 20 fashion as shown in the figure above out of these 26 pins are GPIO pins. As you can observe, the numbers to the pins are given in zigzag Manner.
7. The first (bottom) row start with no so the pins in this row have odd numbers from 1 to 39.
8. The 2nd (top) row starts with number 2 the pins in this row have even numbers from 2 to 40.
9.
10. 26 pins are GPIO pins.
8 pins are Grounded (GND) pins.
2 pins are 5V Power Supply pins.
2 pins are 3.3V Power supply pins.
2 pins are not used.

11. Now if you're coming to the Raspberry Pi as an Arduino user, you're probably used to referencing pins with a single, unique number.
12. In Raspberry Pi there are two diff^n numbering schemes for referencing Pi pin.

Numbers:
1. Broadcom chip-specific pin numbers.
2. Physical pin numbers (BOARD).
You're free to use either number system. In a program, at a time, you can use only one number scheme.

Broadcom chip-specific pin numbers.
1. BCM - Commonly called "GPIO", these are the ones you probably want to use with RPi.GPIO.
2. The parameter used for this system.
3. This is a lower level way of working - it refers to the channel numbers on the Broadcom SOC.
4. To use this system, you have to always work with a diagram describing which channel number goes to which pin on the RPi.
5. your script could break between revisions of Raspberry Pi-boards.
6. In the system 26 GPIO pins are named as GPIO 01 to 26. BOARD.
7. This system uses physical-numbers corr^n to the pins physical location on the header.
8. The number printed on the board are physical numbering system.
9. The parameter used for this sysem.
10. The advantage of using the numbering system is that your h/w will always work regardless

of the board revision of the PRi.
11. you will not need to rewire your connector or change your code.
12. In this system.
26 GPIO pins are named bet 0 to 40.
To use the Python IDLE IDE for programming in Raspberry-Pi use the foll.

26 GPIO pins are named bet 0 to 40.
To use the Python IDLE IDE for programming in Raspberry-Pi use the foll.

- or open terminal window & type the command sudo idle 3.5 & press enter.
- Install all libraries required for Buzzer as given above

Raspberry Pi GPIO programming using Python.

1. The Raspberry Pi is often used in conjunction with other hardware to create interesting electronic projects.
2. The Pi 3 comes with 40 GPIO pins that you can use to interface with various hardware devices - for both receiving data from them.
3. To do this we have to program the GPIO pin. To include these libraries in the program, the command used is 'import'.
4. This way, we can write application to both read & also to control devices.
5. The default operating system used in Raspberry-Pi is Raspbian.
6. The python package used for Raspberry Pi GPIO programming is RPi.GPIO - It is already installed in Raspbian.
7. If you are using any other operating system, the package can be installed by using.

following command:
$ sudo pip install RPi.GPIO.

1. Import the RPi.GPIO library using the following command import RPi.GPIO as GPIO / import time.
2. Import the time library using the following command import time.
3. Set numbering scheme to be used. The method used for this is GPIO.setmode(). we GPIO.setmode(GPIO.BOARD).
4. Set the pin mode as INPUT or OUTPUT using the commands. GPIO.setsetup(channel, GPIO.IN) GPIO.setup(channel, GPIO.OUT).
5. Read input using following command. GPIO.input(pin no).
6. GPIO.output(pin no, state).
7. time.sleep()
8. GPIO.cleanup(). print("Exiting")
9. you must clean up the pin set-ups before your program exits otherwise those pin settings will persist, & that might cause trouble when you use the same pins in another program.
10. The Pi 'express its displeasure' with a warning.
11. To clean up the entire set of pins, invoke GPIO.cleanup().
12. If you want only a few pins to be cleaned up as GPIO.cleanup(channel_list)
13. Anyway, you can suppress the warning message by calling GPIO.setwarnings(False).
14. Save the program as with extension '.py'.
15. The IDE & named 'IDLE' used for programming is an interpreter & not a compiler. So to run the Python program.