7.2 Exemplary Device: Raspberry Pi

Raspberry Pi [104] is a low-cost mini-computer with the physical size of a credit card. Raspberry Pi runs various flavors of Linux and can perform almost all tasks that a normal desktop computer can do. In addition to this, Raspberry Pi also allows interfacing sensors and actuators through the general purpose I/O pins. Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".

7.3 About the Board

Figure 7.2 shows the Raspberry Pi board with the various components/peripherals labeled.

- **Processor & RAM** : Raspberry Pi is based on an ARM processor. The latest version of Raspberry Pi (Model B, Revision 2) comes with 700 MHz Low Power ARM1176JZ-F processor and 512 MB SDRAM.
- **USB Ports** : Raspberry Pi comes with two USB 2.0 ports. The USB ports on Raspberry Pi can provide a current upto 100mA. For connecting devices that draw current more than 100mA, an external USB powered hub is required.
- **Ethernet Ports** : Raspberry Pi comes with a standard RJ45 Ethernet port. You can connect an Ethernet cable or a USB Wifi adapter to provide Internet connectivity.
- **HDMI Output** : The HDMI port on Raspberry Pi provides both video and audio output. You can connect the Raspberry Pi to a monitor using an HDMI cable. For monitors that have a DVI port but no HDMI port, you can use an HDMI to DVI adapter/cable.
- **Composite Video Output** : Raspberry Pi comes with a composite video output with an RCA jack that supports both PAL and NTSC video output. The RCA jack can be used to connect old televisions that have an RCA input only.
- **Audio Output** : Raspberry Pi has a 3.5mm audio output jack. This audio jack is used for providing audio output to old televisions along with the RCA jack for video. The audio quality from this jack is inferior to the HDMI output.
- **GPIO Pins** : Raspberry Pi comes with a number of general purpose input/ouput pins. Figure 7.3 shows the Raspberry Pi GPIO headers. There are four types of pins on Raspberry Pi - true GPIO pins, I2C interface pins, SPI interface pins and serial Rx and Tx pins.
- **Display Serial Interface (DSI)** : The DSI interface can be used to connect an LCD panel to Raspberry Pi.
- **Camera Serial Interface (CSI)** : The CSI interface can be used to connect a camera module to Raspberry Pi.
- **Status LEDs** : Raspberry Pi has five status LEDs. Table 7.1 lists Raspberry Pi status LEDs and their functions.
- **SD Card Slot** : Raspberry Pi does not have a built in operating system and storage. You can plug-in an SD card loaded with a Linux image to the SD card slot. Appendix-A provides instructions on setting up New Out-of-the-Box Software (NOOBS) on Raspberry Pi. You will require atleast an 8GB SD card for setting up NOOBS.
- **Power Input** : Raspberry Pi has a micro-USB connector for power input.

Status LED	Function
ACT	SD card access
PWR	3.3V Power is present
FDX	Full duplex LAN connected
LNK	Link/Network activity
100	100 Mbit LAN connected

Table 7.1: Raspberry Pi Status LEDs

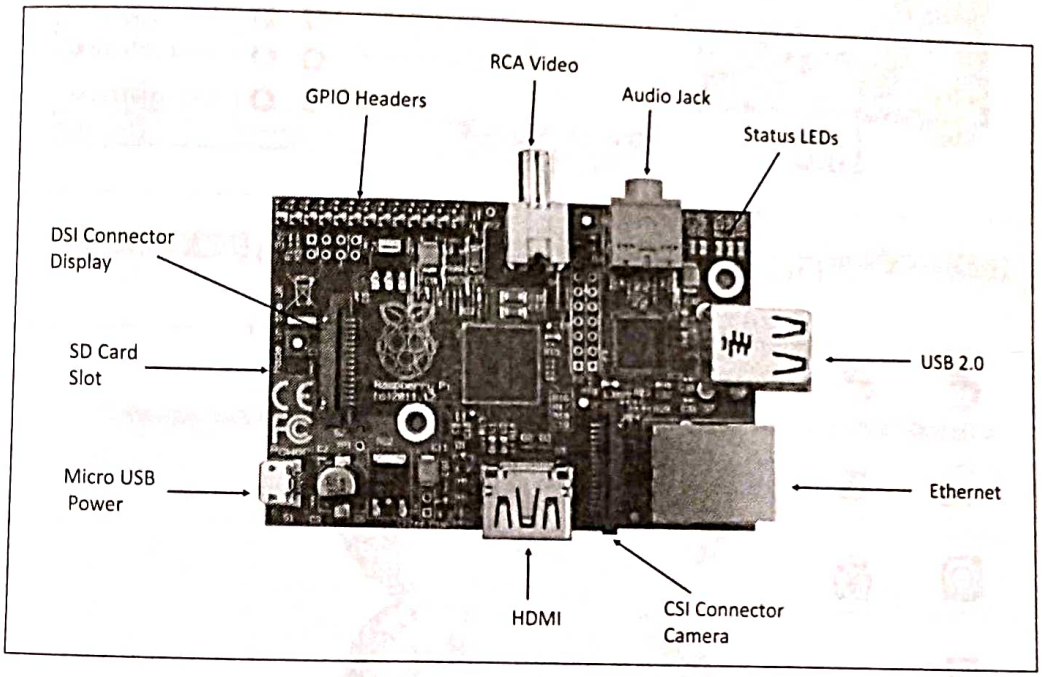

Figure 7.2: Raspberry Pi board

7.4 Linux on Raspberry Pi

Raspberry Pi supports various flavors of Linux including:

- **Raspbian** Raspbian Linux is a Debian Wheezy port optimized for Raspberry Pi. This is the recommended Linux for Raspberry Pi. Appendix-1 provides instructions on setting up Raspbian on Raspberry Pi.
- **Arch** : Arch is an Arch Linux port for AMD devices. advanced micro devices
- **Pidora** : Pidora Linux is a Fedora Linux optimized for Raspberry Pi.
- **RaspBMC** : RaspBMC is an XBMC media-center distribution for Raspberry Pi.
- **OpenELEC** : OpenELEC is a fast and user-friendly XBMC media-center distribution.
- **RISC OS** : RISC OS is a very fast and compact operating system.

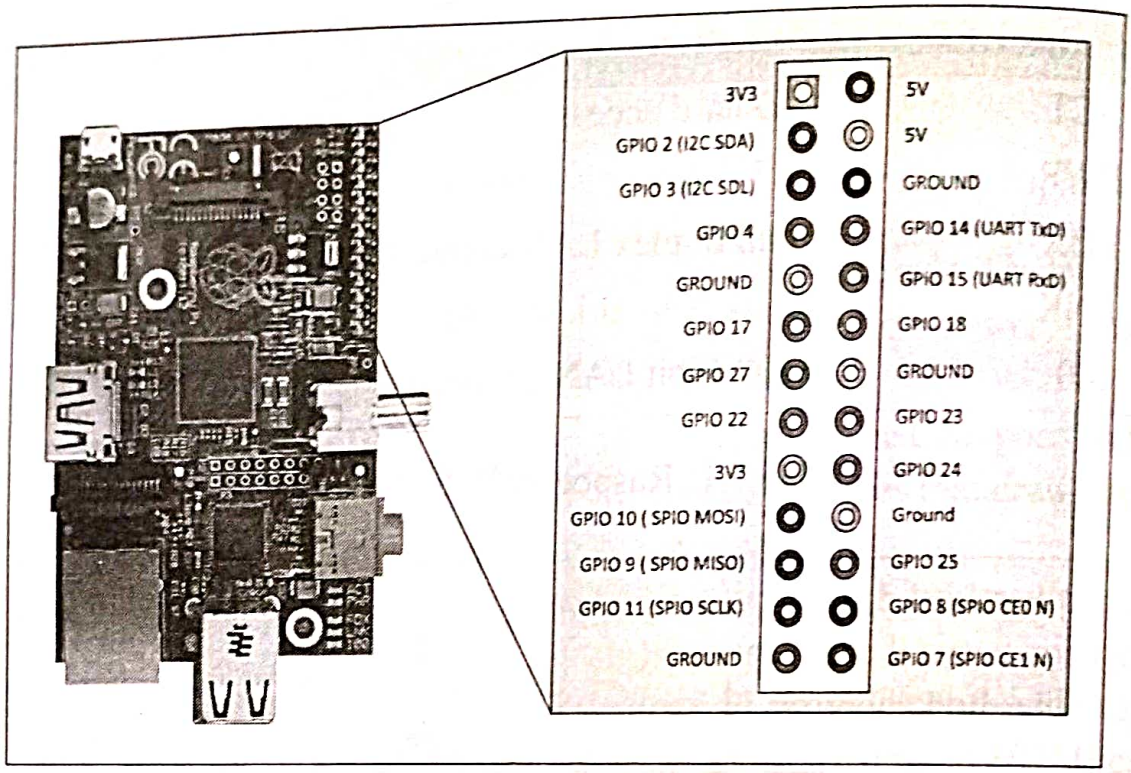

Figure 7.3: Raspberry Pi GPIO headers

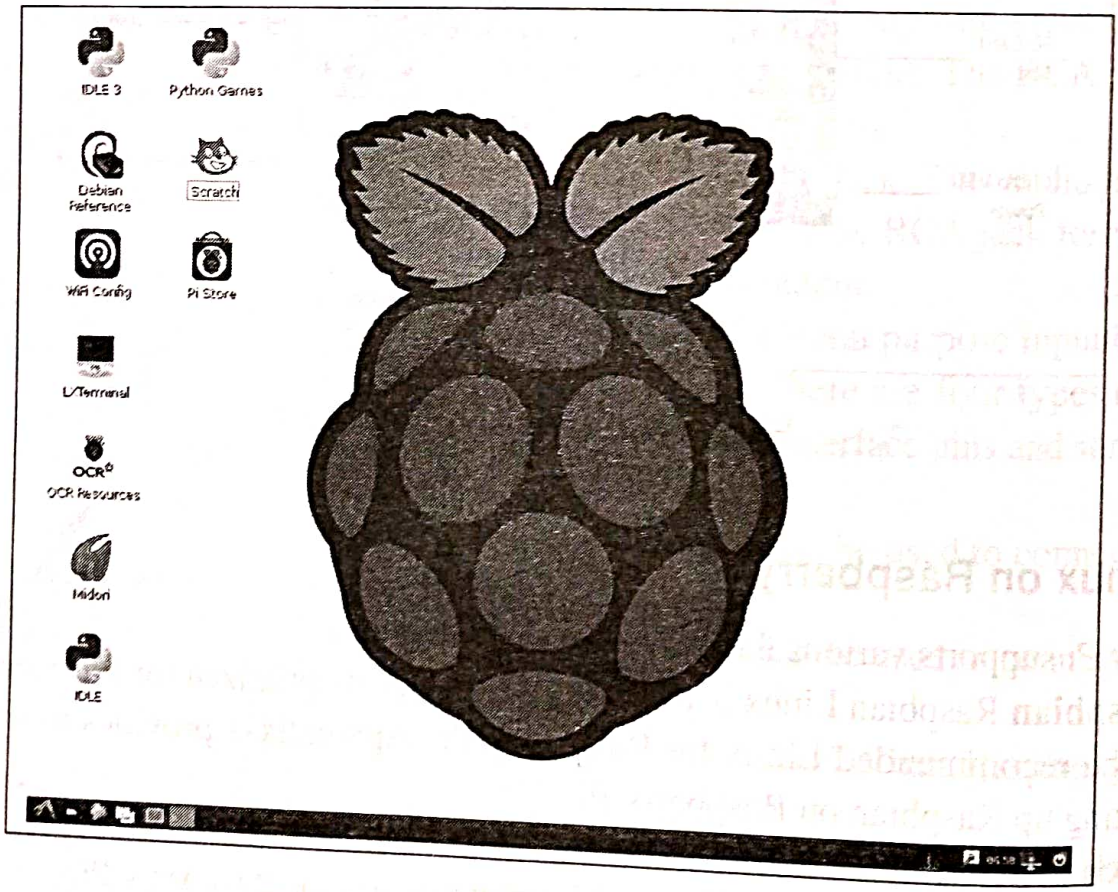

Figure 7.4: Rasbian Linux desktop

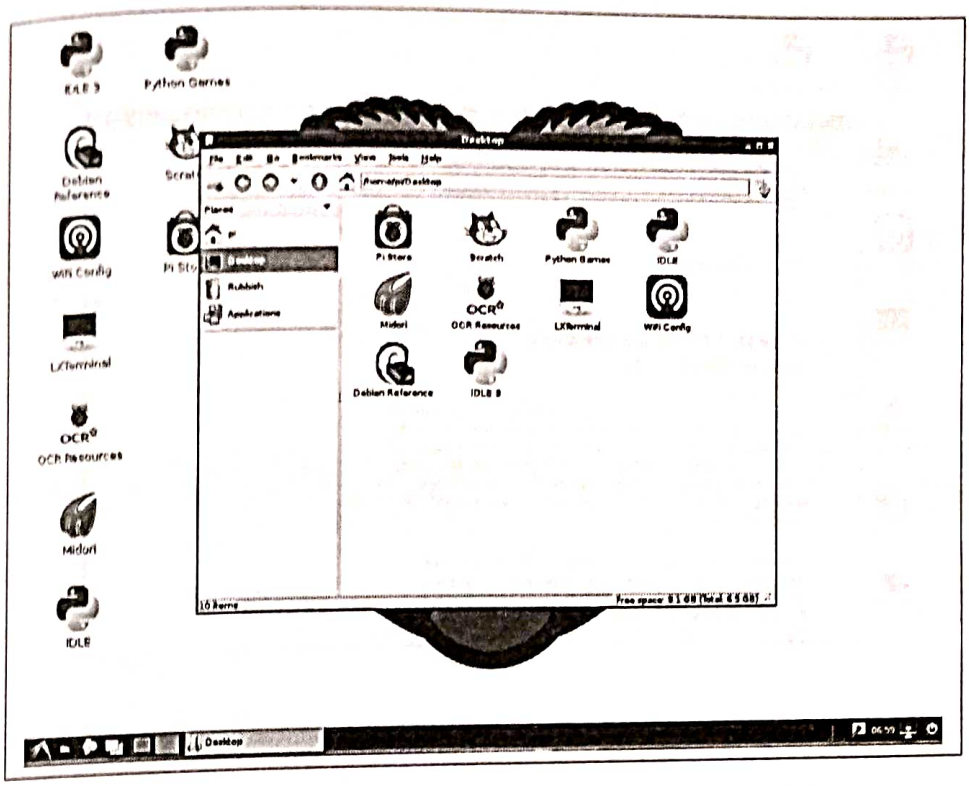

Figure 7.5: File explorer on Raspberry Pi

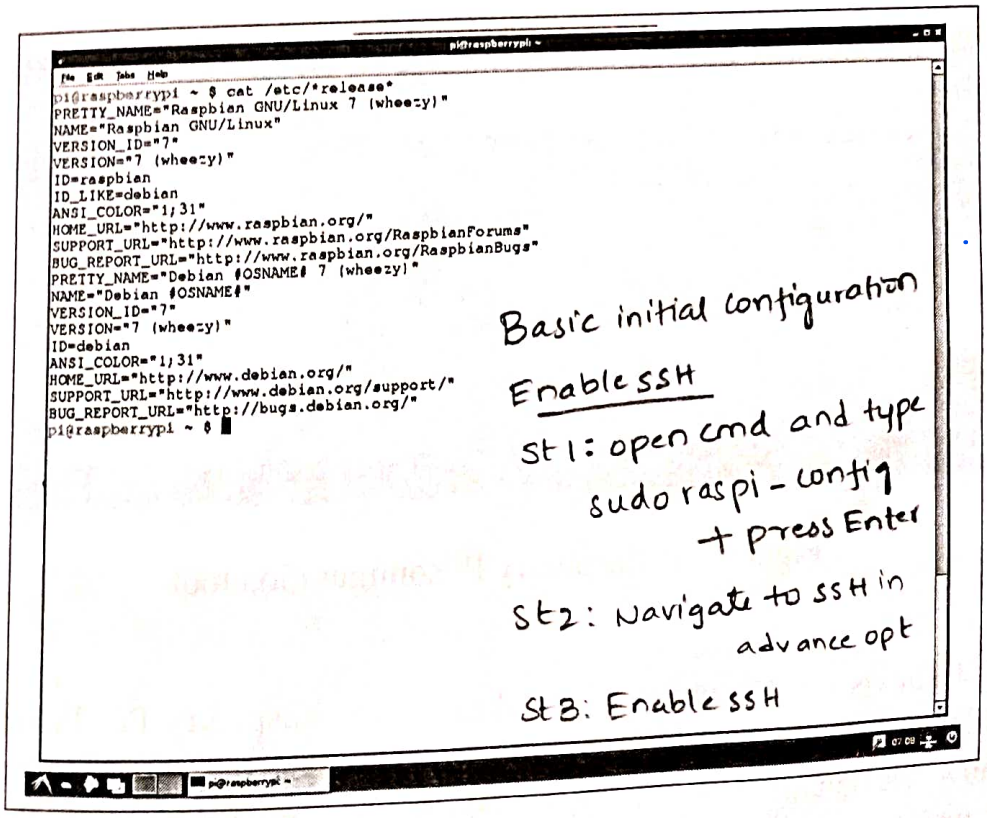

Figure 7.6: Console on Raspberry Pi

7.5 Raspberry Pi Interfaces

Raspberry Pi has serial, SPI and I2C interfaces for data transfer as shown in Figure 7.3.

7.5.1 Serial

The serial interface on Raspberry Pi has receive (Rx) and transmit (Tx) pins for communication with serial peripherals.

7.5.2 SPI

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used for communicating with one or more peripheral devices. In an SPI connection, there is one master device and one or more peripheral devices. There are five pins on Raspberry Pi for SPI interface:

- **MISO (Master In Slave Out)** : Master line for sending data to the peripherals.
- **MOSI (Master Out Slave In)** : Slave line for sending data to the master.
- **SCK (Serial Clock)** : Clock generated by master to synchronize data transmission
- **CE0 (Chip Enable 0)** : To enable or disable devices.
- **CE0 (Chip Enable 1)** : To enable or disable devices.

Serial data line (SDA): Sends the data between devices.
Serial clock (SCL): Provides a clock signal to synchronize the data transfer.
The single data line is used to both send and receive data,

7.5.3 I2C

The I2C interface pins on Raspberry Pi allow you to connect hardware modules. I2C interface allows synchronous data transfer with just two pins - SDA (data line) and SCL (clock line).

7.6 Programming Raspberry Pi with Python

In this section you will learn how to get started with developing Python programs on Raspberry Pi. Raspberry Pi runs Linux and supports Python out of the box. Therefore, you can run any Python program that runs on a normal computer. However, it is the general purpose input/output capability provided by the GPIO pins on Raspberry Pi that makes it useful device for Internet of Things. You can interface a wide variety of sensor and actuators with Raspberry Pi using the GPIO pins and the SPI, I2C and serial interfaces. Input from the sensors connected to Raspberry Pi can be processed and various actions can be taken, for instance, sending data to a server, sending an email, triggering a relay switch.

7.6.1 Controlling LED with Raspberry Pi

Let us start with a basic example of controlling an LED from Raspberry Pi. Figure 7.9 shows the schematic diagram of connecting an LED to Raspberry Pi. Box 7.1 shows how to turn

the LED on/off from command line. In this example the LED is connected to GPIO pin 18. You can connect the LED to any other GPIO pin as well.

Box 7.2 shows a Python program for blinking an LED connected to Raspberry Pi every second. The program uses the RPi. GPIO module to control the GPIO on Raspberry Pi. In this program we set pin 18 direction to output and then write *True/False* alternatively after a delay of one second.

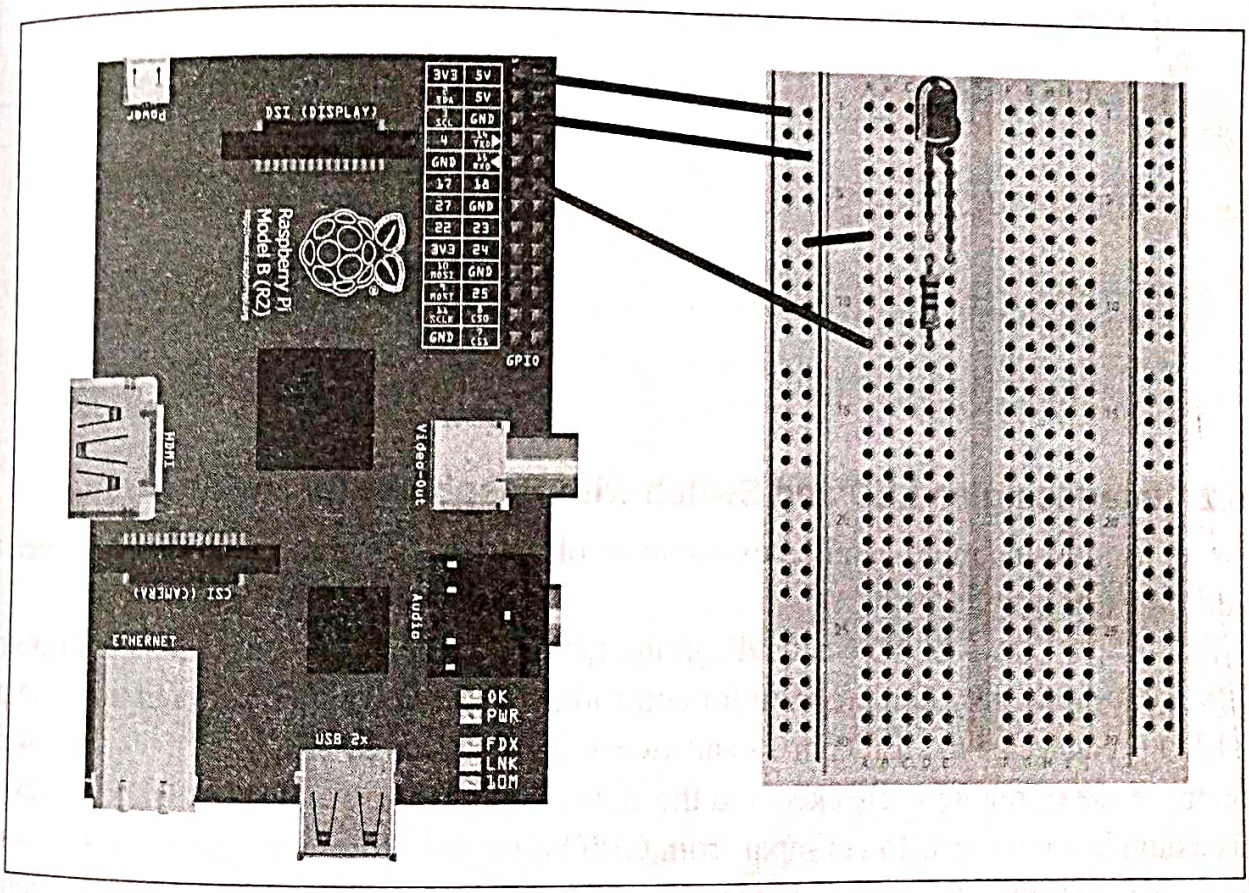

Figure 7.9: Controlling LED with Raspberry Pi

Box 7.1: Switching LED on/off from Raspberry Pi console

```
$echo 18 > /sys/class/gpio/export
$cd /sys/class/gpio/gpio18

#Set pin 18 direction to out
$echo out > direction

#Turn LED on
$echo 1 > value
```

```
#Turn LED off
$echo 0 > value
```

Box 7.2: Python program for blinking LED

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
   GPIO.output(18, True)
   time.sleep(1)
   GPIO.output(18, False)
   time.sleep(1)
```

7.6.2 Interfacing an LED and Switch with Raspberry Pi

Now let us look at a more detailed example involving an LED and a switch that is used to control the LED.

Figure 7.10 shows the schematic diagram of connecting an LED and switch to Raspberry Pi. Box 7.3 shows a Python program for controlling an LED with a switch. In this example the LED is connected to GPIO pin 18 and switch is connected to pin 25. In the infinite while loop the value of pin 25 is checked and the state of LED is toggled if the switch is pressed. This example shows how to get input from GPIO pins and process the input and take some action. The action in this example is toggling the state of an LED. Let us look at another example, in which the action is an email alert. Box 7.4 shows a Python program for sending an email on switch press. Note that the structure of this program is similar to the program in Box 7.3. This program uses the Python SMTP library for sending an email when the switch connected to Raspberry Pi is pressed.

Box 7.3: Python program for controlling an LED with a switch

```
from time import sleep
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
```

```
#Switch Pin
GPIO.setup(25, GPIO.IN)

#LED Pin
GPIO.setup(18, GPIO.OUT)

state=false

def toggleLED(pin):
  state = not state
  GPIO.output(pin, state)

while True:
  try:
    if (GPIO.input(25) == True):
      toggleLED(pin)
    sleep(.01)
  except KeyboardInterrupt:
    exit()
```

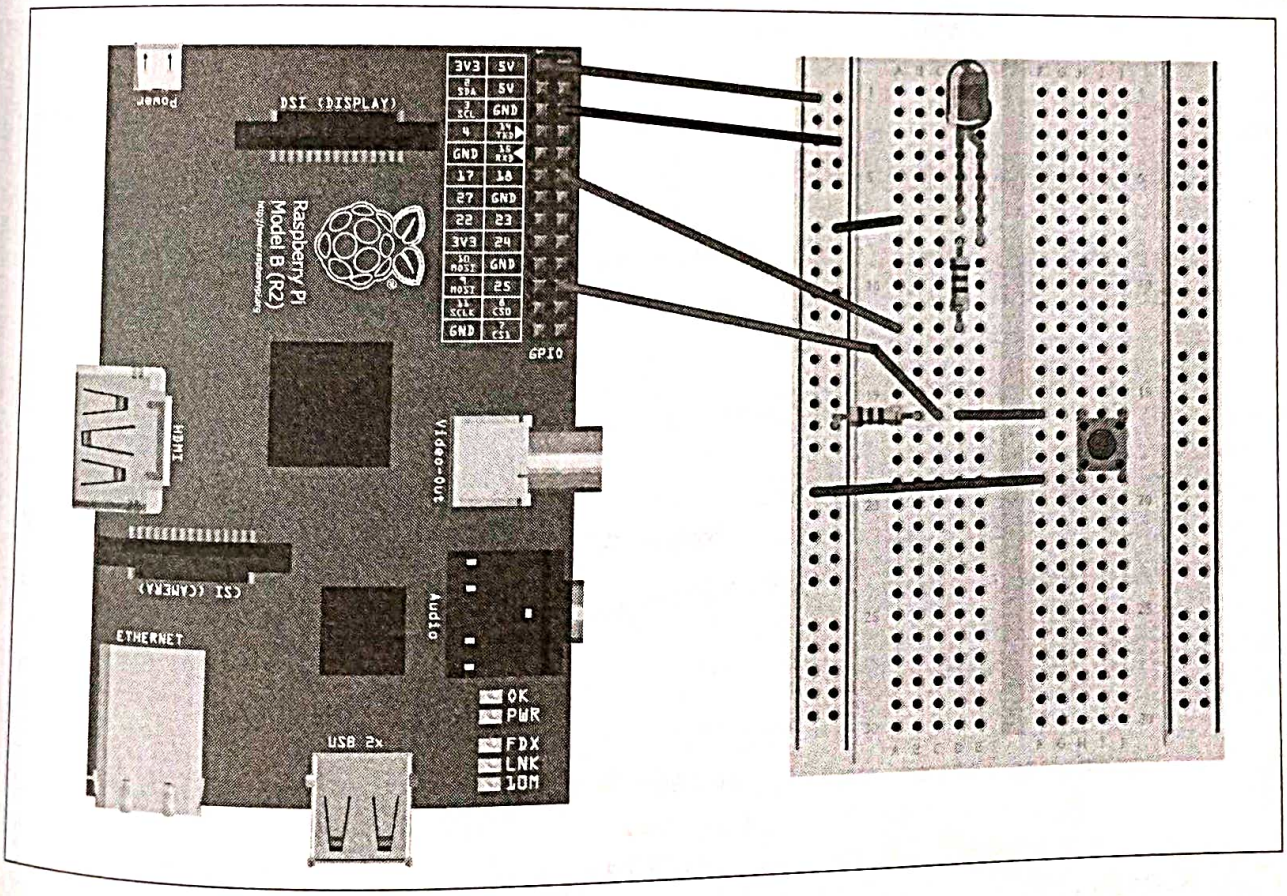

Figure 7.10: Interfacing LED and switch with Raspberry Pi

```
except KeyboardInterrupt:
    exit()
```

7.6.3 Interfacing a Light Sensor (LDR) with Raspberry Pi

So far you have learned how to interface LED and switch with Raspberry Pi. Now let us look at an example of interfacing a Light Dependent Resistor (LDR) with Raspberry Pi and turning an LED on/off based on the light-level sensed.

Figure 7.11 shows the schematic diagram of connecting an LDR to Raspberry Pi. Connect one side of LDR to 3.3V and other side to a 1μF capacitor and also to a GPIO pin (pin 18 in this example). An LED is connected to pin 18 which is controlled based on the light-level sensed.

Box 7.5 shows the Python program for the LDR example. The *readLDR()* function returns a count which is proportional to the light level. In this function the LDR pin is set to output and low and then to input. At this point the capacitor starts charging through the resistor (and a counter is started) until the input pin reads high (this happens when capacitor voltage becomes greater than 1.4V). The counter is stopped when the input reads high. The final count is proportional to the light level as greater the amount of light, smaller is the LDR resistance and greater is the time taken to charge the capacitor. LDR light ↑ Read Time ↑

Box 7.5: Python program for switching LED/Light based on reading LDR reading

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
ldr_threshold = 1000
LDR_PIN = 18
LIGHT_PIN = 25

def readLDR(PIN):
  reading=0
  GPIO.setup(LIGHT_PIN, GPIO.OUT)
  GPIO.output(PIN, False)
  time.sleep(0.1)
  GPIO.setup(PIN, GPIO.IN)
  while (GPIO.input(PIN)==False):
    reading=reading+1
  return reading
```

```
def switchOnLight(PIN):
   GPIO.setup(PIN, GPIO.OUT)
   GPIO.output(PIN, True)

def switchOffLight(PIN):
   GPIO.setup(PIN, GPIO.OUT)
   GPIO.output(PIN, False)


while True:
   ldr_reading = readLDR(LDR_PIN)
   if ldr_reading < ldr_threshold:
      switchOnLight(LIGHT_PIN)
   else:
      switchOffLight(LIGHT_PIN)

   time.sleep(1)
```

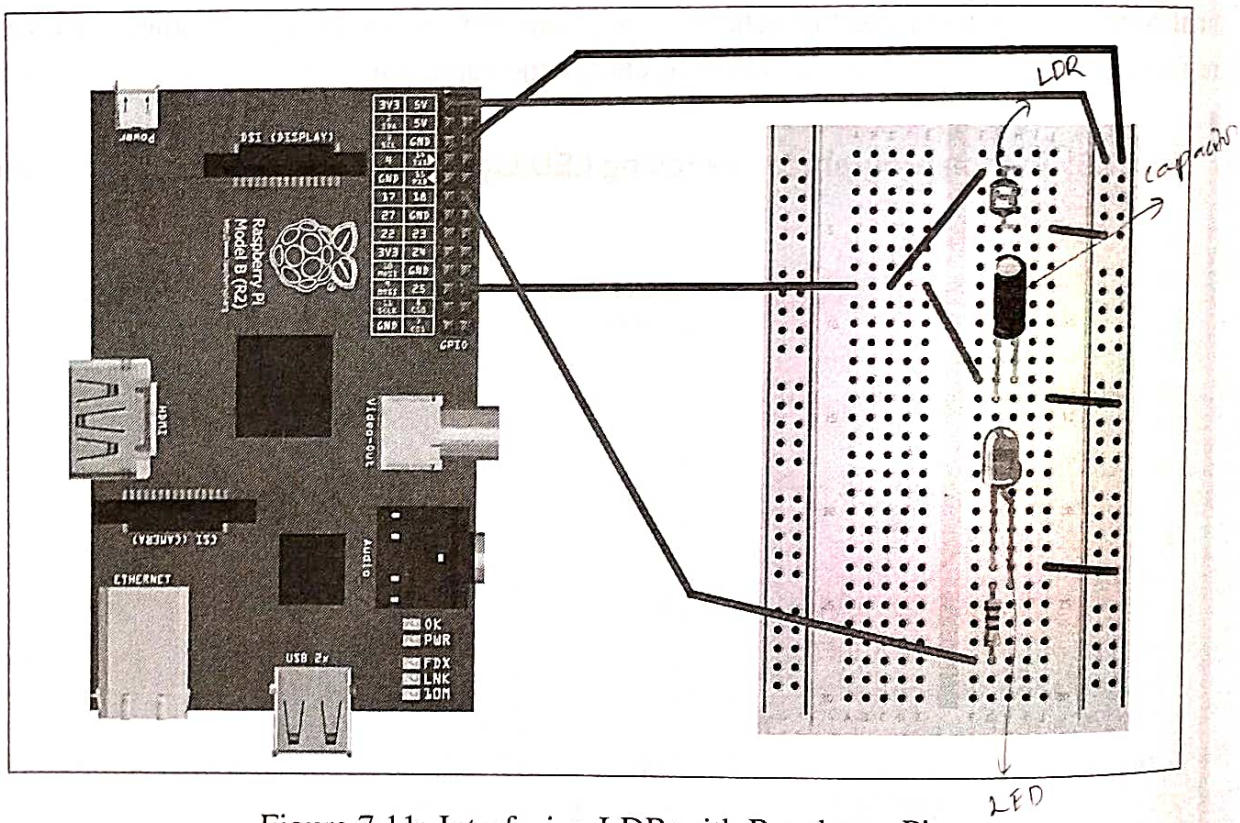

Figure 7.11: Interfacing LDR with Raspberry Pi