

INTERNET OF THINGS JOURNAL

Ankit Patel

53003170058

TY.BSC(IT) DIV-B

2019-2020

53003170058

Internet of Things

Academic Year: 2019-2020

Course: B. Sc. – Information Technology

Sem: V

Content

Sr. No.	Practical Details	Date	Pg. No.	Signature
1.	Raspberry Pi Hardware Preparation and Installation	3/7/2019	3	
2.	Displaying Different LED Patterns with Raspberry Pi	10/7/2019	6	
3.	Displaying Time over 4-digit 7 segment display	17/7/2019	10	
4.	Capturing Images with Raspberry Pi and Pi Camera	24/7/2019	13	
5.	Capturing video with Raspberry Pi and Pi Camera	7/8/2019	17	
6.	GPS Module Interfacing with Raspberry Pi	28/8/2019	20	
7.	RFID Module Interfacing with Raspberry Pi	18/9/2019	23	

Practical No. 1

Raspberry Pi Hardware Preparation and Installation

Aim: To study Raspberry Pi Hardware Preparation and Installation. **Hardware Guide:**

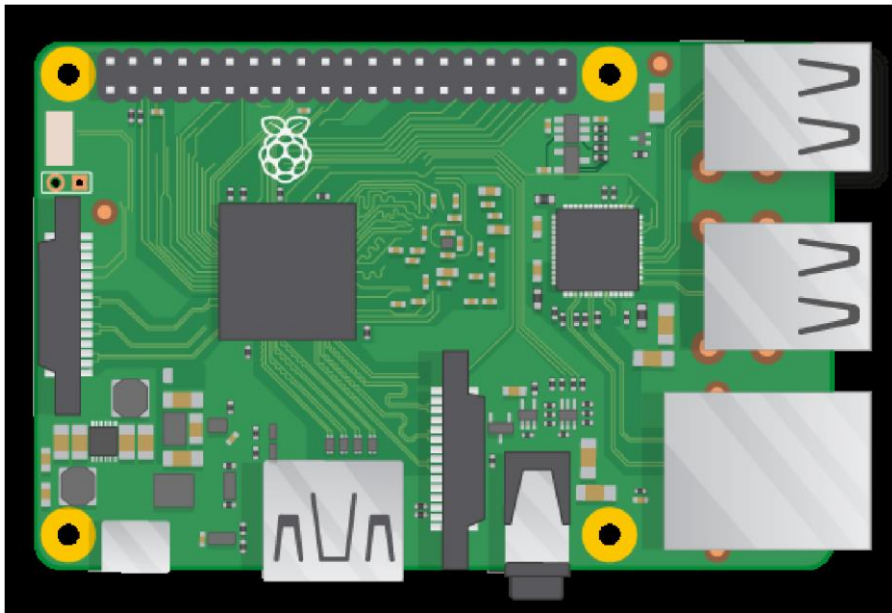
For getting started with raspberry pi for the first time you will require the following hardware

1. Raspberry Pi (latest Model)
2. Monitor or TV
3. HDMI cable
4. Ethernet cable
5. USB keyboard
6. USB mouse
7. Micro USB power supply
8. 8GB or larger microSD card
9. SD Card Reader

Raspberry Pi 3 model B:

The Raspberry Pi 3 is the third generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016. Compared to the Raspberry Pi 2 it has:

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)



Like the Pi 2, it also has:

- 4 USB ports
- 40 GPIO pins

- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- Video Core IV 3D graphics core
- The Raspberry Pi 3 has an identical form factor to the previous Pi 2 (and Pi 1 Model B+) and has complete compatibility with Raspberry Pi 1 and 2.

Monitor or TV:

- A monitor or TV with HDMI in can be used as a display with a Raspberry Pi. Most modern television sets and monitors have an HDMI port, and are the easiest to get working with the Raspberry Pi. You can use an HDMI cable to connect the Raspberry Pi directly to the television or monitor.
- Some older monitors have a DVI port. These work well with the Raspberry Pi, although you'll need an HDMI-to-DVI adapter to attach to an HDMI cable, or a one-piece HDMI-to-DVI cable. Some old monitors have a VGA port. These can be trickier to use as you'll need an HDMI-to-VGA converter, which can change digital video to analogue video. A simple port adapter won't work.

HDMI to HDMI Cable:

Connect Raspberry Pi to a Monitor or TV with a HDMI to HDMI cable.

Ethernet Cable:

Ethernet cable will allow your Pi to connect with the internet. It is also useful for headless setup of Raspberry Pi

USB Keyboard and Mouse:

Any standard USB keyboard and mouse can be used with the Raspberry Pi. This plug and play devices will work without any additional driver. Simply plug them into the Raspberry Pi and they should be recognised when it starts up.

Power Supply:

It is recommended that you use a 5V, 2A USB power supply for all models of Raspberry Pi.

SD Card:

The latest version of Raspbian, the default operating system recommended for the Raspberry Pi, requires an 8GB (or larger) micro SD card. SD card will store the operating systems as well as all the file and applications created by you.

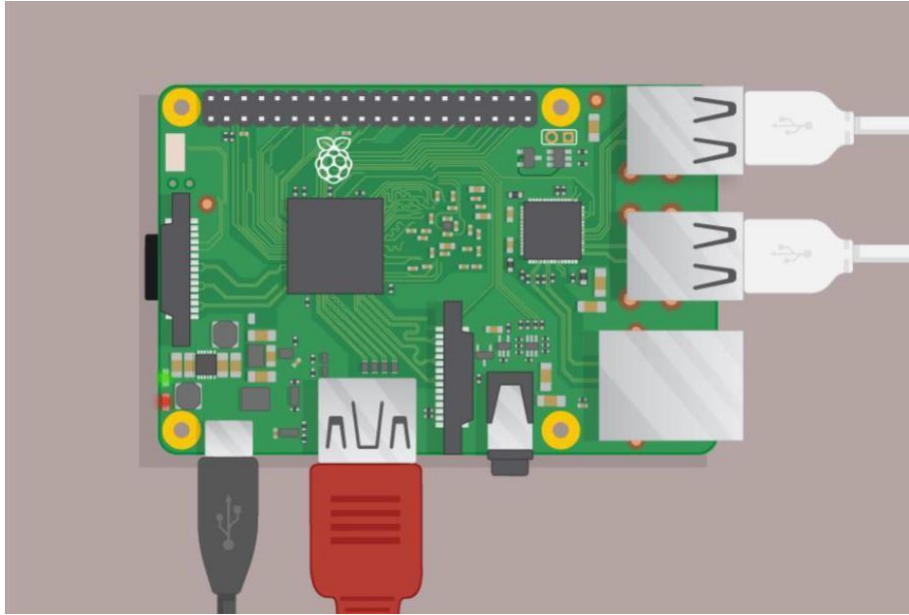
Installation Guide:

Now since you have all the required hardware, we will now learn how to get the operating system onto your microSD card so that you can start using software on your Raspberry Pi **Get Raspbian OS on your microSD card:** Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more.

1. To download Raspbian log on to raspberrypi.org and click on the download, then click on Raspbian and lastly download the RASPBIAN JESSIE WITH DESKTOP file. You can choose either the Torrent file or ZIP file.
2. The downloaded file will be in zip format. To unzip the file, you will require an unzip tool. You can use any unzipping tool viz. WINRAR, 7ZIP etc. After unzipping the file, you will find a disc image file in the unzipped folder.
3. Now format the SD Card before writing the disc image file on the SD card. You can use SD Formatter tool or any other tool of your wish.
4. To write the image file of the operating system on the SD card you will require a Disk Imager tool. For this you can use Win32 Disk Imager tool.
5. Once the image is written on the SD Card, your untitled SD card will now have the name boot. Your SD Card will now hold the Raspbian Operating system required for the first-time setup.

Plugging in Raspberry Pi:

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi. It will only fit one way.
2. Next, plug your keyboard and mouse into the USB ports on the Raspberry Pi.
3. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc).
4. Connect your HDMI cable from your Raspberry Pi to your monitor or TV.
5. If you intend to connect your Raspberry Pi to the internet, plug an Ethernet cable into the Ethernet port, or connect a WiFi dongle to one of the USB ports (unless you have a Raspberry Pi 3).
6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your Raspberry Pi.

**Conclusion:**

In this practical session I have installed the raspberry pi and also the hardware connection was done with the help of Raspberry Pi (latest Model) , Monitor or TV , USB keyboard ,USB mouse ,Micro USB power supply ,8GB or larger microSD card ,SD Card Reader etc.

Practical No. 2 Displaying Different LED Patterns with Raspberry Pi

Aim: To display different LED patterns with Raspberry Pi

Hardware Guide:

Along with the basic setup you will require the following components to get started with the GPIO pins as follows: 1. LED

2. Resistor
3. Connecting wires
4. Breadboard

GPIO

One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the top edge of the board.

These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Of the 40 pins, 26 are GPIO pins and the others are power or ground pins (plus two ID EEPROM pins which you should not play with unless you know your stuff!)



What are they for? What can we do with them?

You can program the pins to interact in amazing ways with the real world. Inputs don't have to come from a physical switch; it could be input from a sensor or a signal from another computer or device, for example. The output can also do anything, from turning on an LED to sending a signal or data to another device. If the Raspberry Pi is on a network, you can control devices that are attached to it from anywhere** and those devices can send data back. Connectivity and control of physical devices over the internet is a powerful and exciting thing, and the Raspberry Pi is ideal for this.

Working of GPIO pins

Output


When we use a GPIO pin as an output. Each pin can turn on or off, or go HIGH or LOW in computing terms. When the pin is HIGH it outputs 3.3 volts (3v3); when the pin is LOW it is off.

Input

GPIO outputs are easy; they are on or off, HIGH or LOW, 3v3 or 0v. Inputs are a bit trickier because of the way that digital devices work. Although it might seem reasonable just to connect a button across an input pin and a ground pin, the Pi can get confused as to whether the button is on or off. It might work properly, it might not. It's a bit like floating about in deep space; without a reference, it would be hard to tell if you were going up or down, or even what up or down meant! Therefore, you will see phrases like "pull up" and "pull down" in Raspberry Pi GPIO tutorials. It's a way of giving the input pin a reference so it knows for certain when an input is received.

Warning: Randomly plugging wires and power sources into your Pi, however, may kill it. Bad things can also happen if you try to connect things to your Pi that use a lot of power.

Circuit Diagram:



		Physical Pins				
GPIO#	2nd func	pin#	pin#	2nd func	GPIO#	
N/A	+3V3	1	2	+5V	N/A	
GPIO2	SDA1 (I2C)	3	4	+5V	N/A	
GPIO3	SCL1 (I2C)	5	6	GND	N/A	
GPIO4	GCLK	7	8	TXD0 (UART)	GPIO14	
N/A	GND	9	10	RXD0 (UART)	GPIO15	
GPIO17	GEN0	11	12	GEN1	GPIO18	
GPIO27	GEN2	13	14	GND	N/A	
GPIO22	GEN3	15	16	GEN4	GPIO23	
N/A	+3V3	17	18	GEN5	GPIO24	
GPIO10	MOSI (SPI)	19	20	GND	N/A	
GPIO9	MISO (SPI)	21	22	GEN6	GPIO25	
GPIO11	SCLK (SPI)	23	24	CE0_N (SPI)	GPIO8	
N/A	GND	25	26	CE1_N (SPI)	GPIO7	
EEPROM	ID_SD	27	28	ID_SC	EEPROM	
GPIO5	N/A	29	30	GND	N/A	
GPIO6	N/A	31	32	-	GPIO12	
GPIO13	N/A	33	34	GND	N/A	
GPIO19	N/A	35	36	N/A	GPIO16	
GPIO26	N/A	37	38	N/A	GPIO20	
N/A	GND	39	40	N/A	GPIO21	

Software Guide:

Raspbian OS comes with many preinstalled programming environments. Here we will be using Python for coding. 9 EDKITS ELECTRONICS

To open Python, click on the application Menu, navigate to Programming, then click on Python 3 (IDLE) an Integrated Development Environment for Python 3. After opening the IDE, go to files and open new file to start your code.

```
Code: import
RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BCM)
```

```
# init list with pin numbers
```

```
pinList = [5, 6, 13, 19, 26]
```



```
# loop through pins and set mode and state to 'high'
```

```
for i in pinList:
```

```
    GPIO.setup(i, GPIO.OUT)
```

```
    GPIO.output(i, GPIO.LOW)
```

```
# time to sleep between operations in the main loop
```

```
SleepTimeL = 2
```

```
# main loop
```

```
try:
```

```
    GPIO.output(5,
```

```
    GPIO.HIGH) print
```

```
    "ONE"
```

```
    time.sleep(SleepTimeL);
```

```
    GPIO.output(6,
```

```
    GPIO.HIGH) print
```

```
    "TWO"
```

```
    time.sleep(SleepTimeL);
```

```
    GPIO.output(13,
```

```
    GPIO.HIGH) print
```

```
    "THREE"
```

```
    time.sleep(SleepTimeL);
```

```
    GPIO.output(19,
```

```
    GPIO.HIGH) print
```

```
    "FOUR"
```

```
    time.sleep(SleepTimeL);
```

```
    GPIO.output(26,
```

```
    GPIO.HIGH) print
```

```
    "FIVE"
```

```
    time.sleep(SleepTimeL);
```

```
    GPIO.cleanup() print
```

```
    "Good bye!"
```

```
# End program cleanly with keyboard
```

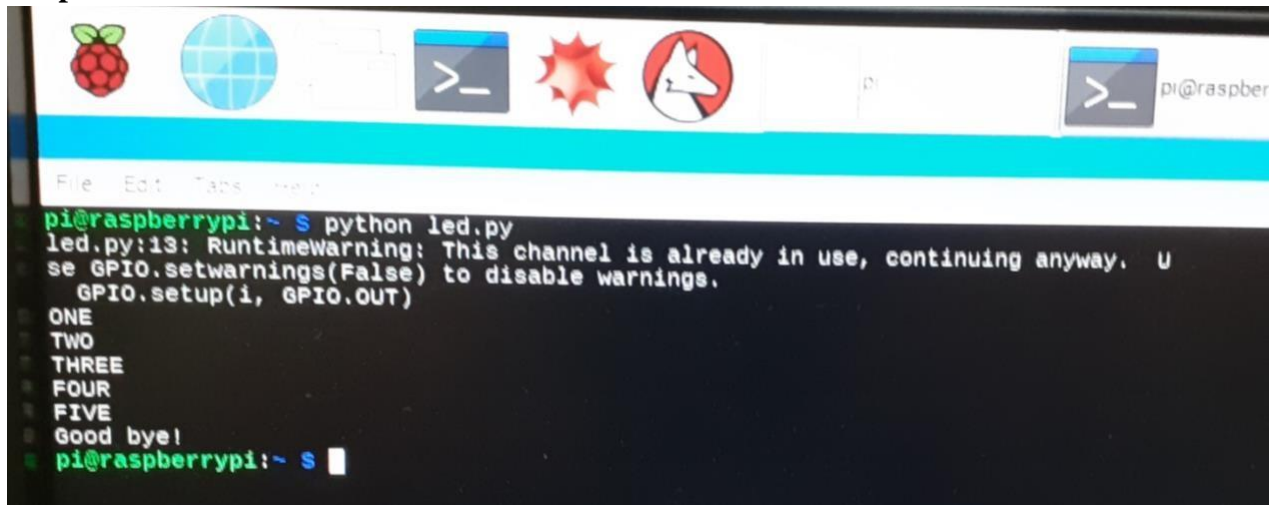
```
except KeyboardInterrupt:
```

```
    print " Quit"
```

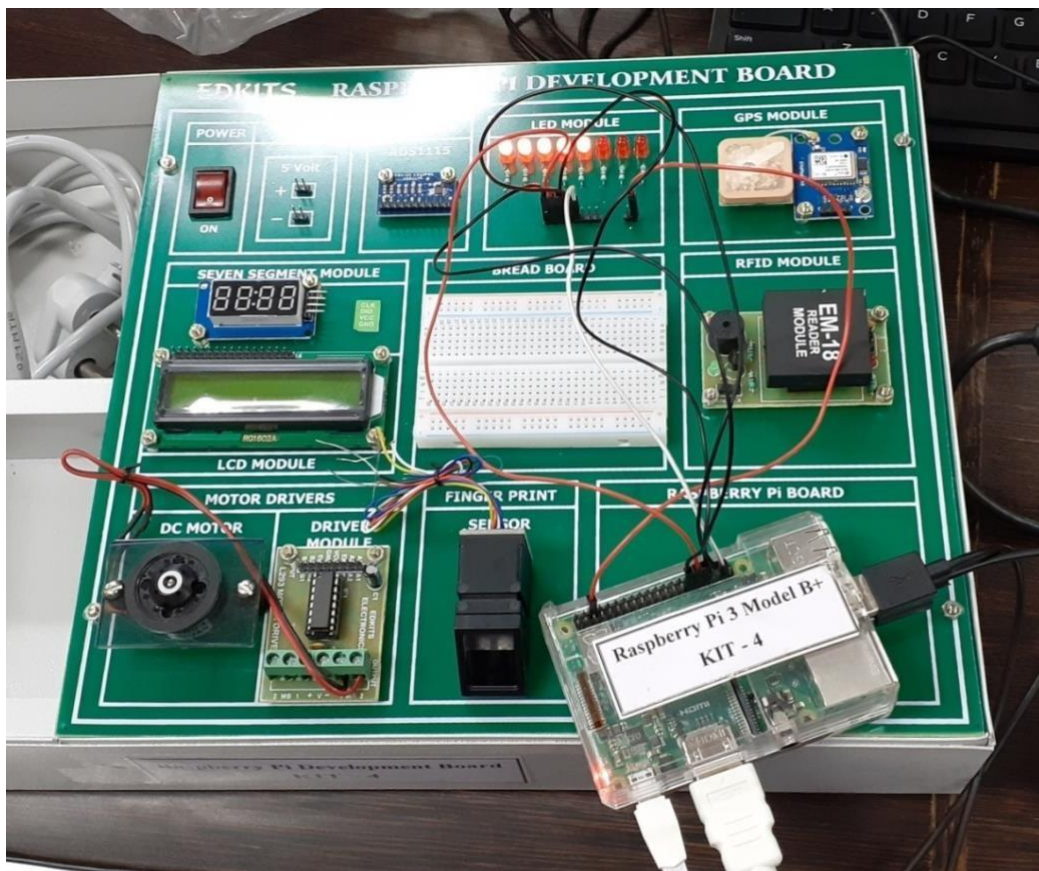
```
    # Reset GPIO settings
```

```
    GPIO.cleanup()
```

Output:



```
pi@raspberrypi:~$ python led.py
led.py:13: RuntimeWarning: This channel is already in use, continuing anyway. Use
GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(1, GPIO.OUT)
ONE
TWO
THREE
FOUR
FIVE
Good bye!
pi@raspberrypi:~$
```



Conclusion:

This practical is basically about the displaying different pattern of the led. In which led were connected to the raspberry pi using wires and after running the code we display different pattern of the led.

Practical No. 3

Displaying Time over 4-digit 7 segment display using Raspberry Pi

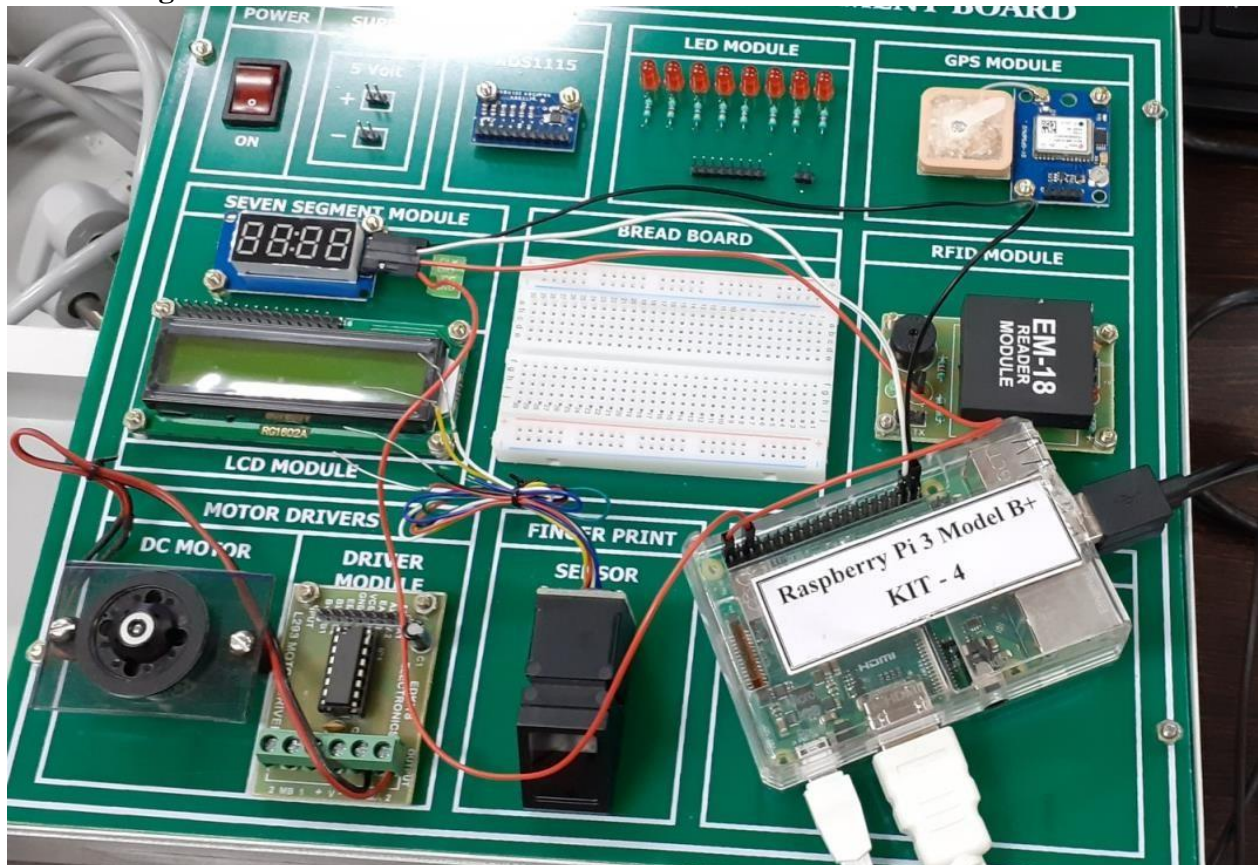
Aim: To display time over 4-digit 7 segment display using Raspberry Pi

Hardware Guide:

For completing this lesson, you will require the following things along with your initial raspberry pi setup

1. TM1637 4-digit seven segment Display board
2. Connecting wires

Circuit Diagram:



Software Guide:

1. Now to download libraries, open Web Browser on your Raspberry Pi and log on to the following link: <https://github.com/timwaizenegger/raspberrypi-examples/tree/master/actor-led->

7segment4numbers .Click on the actor-led-7segment-4numbers.zip folder and Now click on Download Button to download the file.

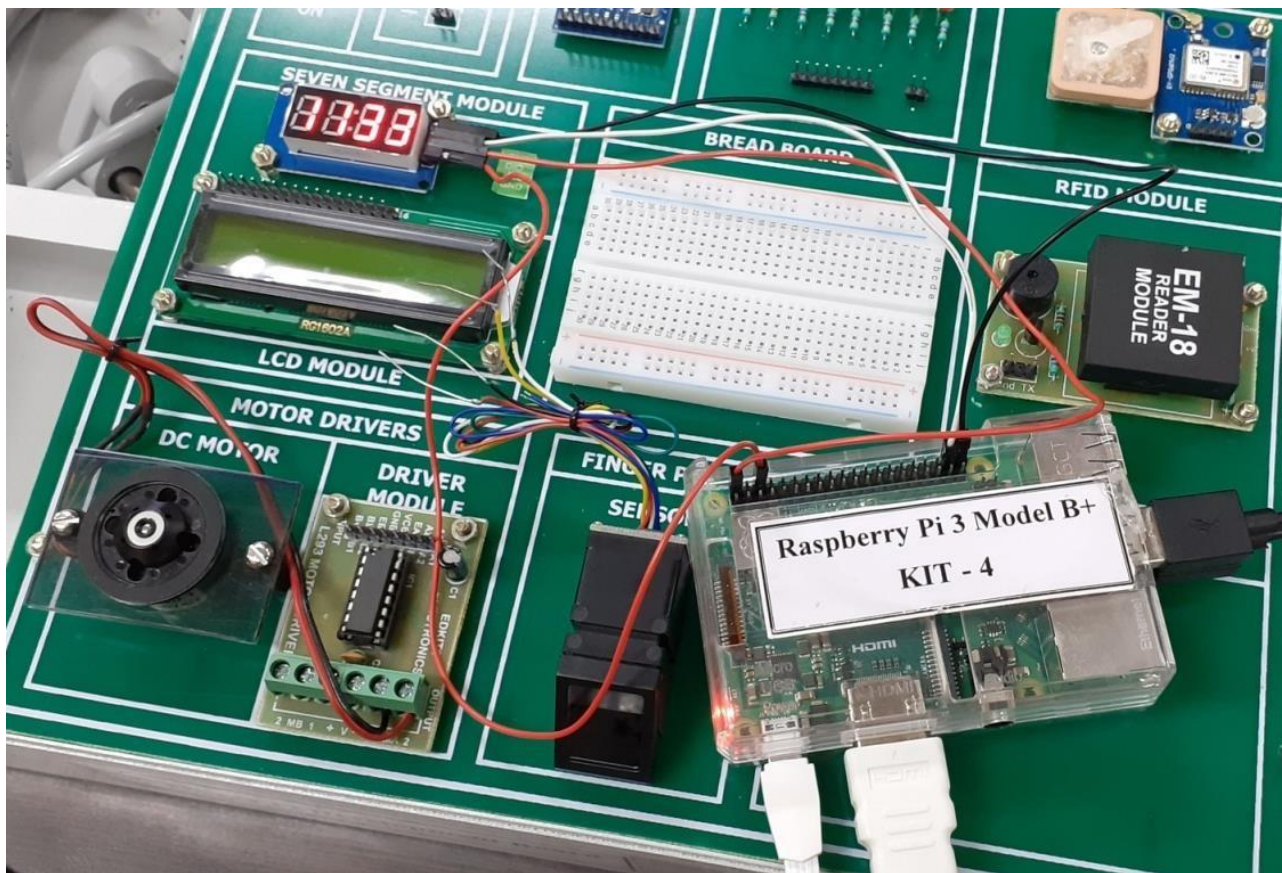
2. Now on your rpi move to /home/pi/Downloads/ location to find the zip file downloaded.
 3. Unzip the file and try to execute the different example codes present in that folder in Python 2 Idle.
 4. Now open Python 2 Idle, create a new file, write the code given below and save it in the same folder
- i.e. actor-led-7segment-4numbers since the code below is depended on tm1637.py file which is present in the same folder.

Code: Clock.py

```
from time import
sleep import
tm1637 try:
import thread
except
ImportError:
import _thread as
thread
# Initialize the clock (GND, VCC=3.3V, Example Pins are DIO-20 and
CLK21) Display = tm1637.TM1637(CLK=21, DIO=20, brightness=1.0)
try:
    print "Starting clock in the background (press CTRL + C to
stop):"
    Display.StartClock(military_time=False)
    print 'Continue Python script and tweak Display!'
    sleep(5)
    Display.ShowDoublepoint(False) # show leading zero -
    false sleep(5) loops = 3 while loops > 0:
    for i in range(0, 10):
        Display.SetBrightness(i / 10.0)
        sleep(0.5)
        loops -= 1
    Display.StopClock()
thread.interrupt_main() except KeyboardInterrupt:
    print "Properly closing the clock and open GPIO pins"
    Display.cleanup() Output:
```



```
File Edit Tabs Help
pi@raspberrypi:~$ python clock1.py
Starting clock in the background (press CTRL + C to stop):
Continue Python script and tweak Display!
Attempting to stop live clock
Properly closing the clock and open GPIO pins
Attempting to stop live clock
pi@raspberrypi:~$
```



Conclusion:

This practical was all about the connection of 4 digit seven segment led for displaying the time. For this one should have a proper connection of the kit and the led then run the python code.

Practical No. 4

Capturing Images with Raspberry Pi and Pi Camera

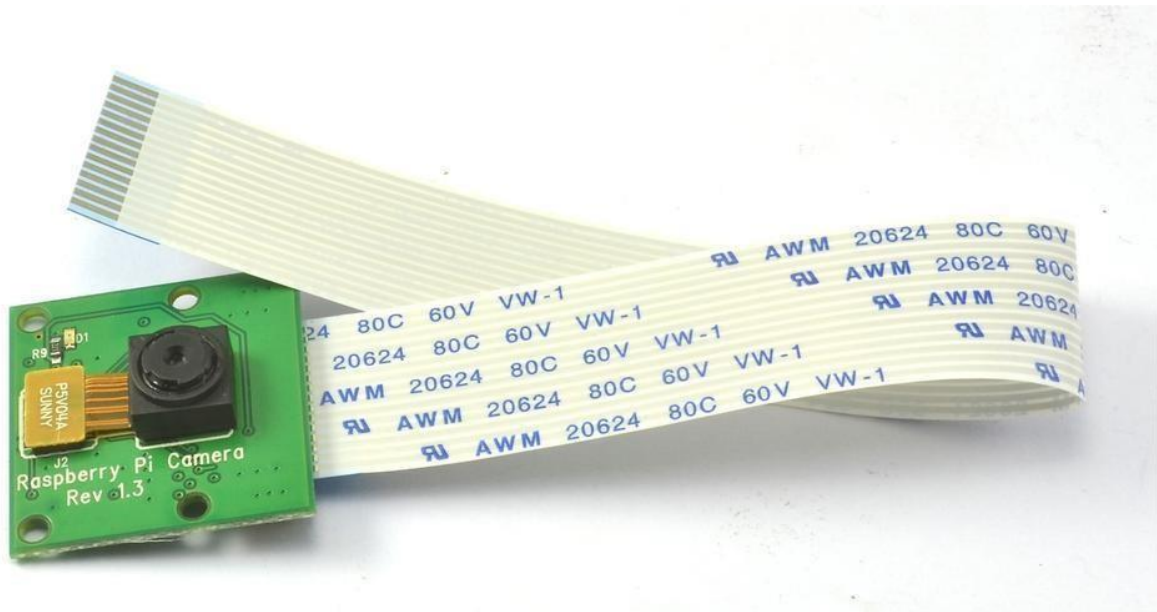
Aim: To capture images with Raspberry Pi and Pi camera

Hardware Guide:

For completing this lesson, you will require the Camera Module along with your initial raspberry pi setup.

Camera Module:

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.



36 EDKITS ELECTRONICS

Capturing Images with Raspberry Pi and Pi Camera

The Camera Module is a great accessory for the Raspberry Pi, allowing users to take still pictures and record video in full HD.

Hardware Guide:

For completing this lesson, you will require the Camera Module along with your initial raspberry pi setup.

Camera Module:

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.

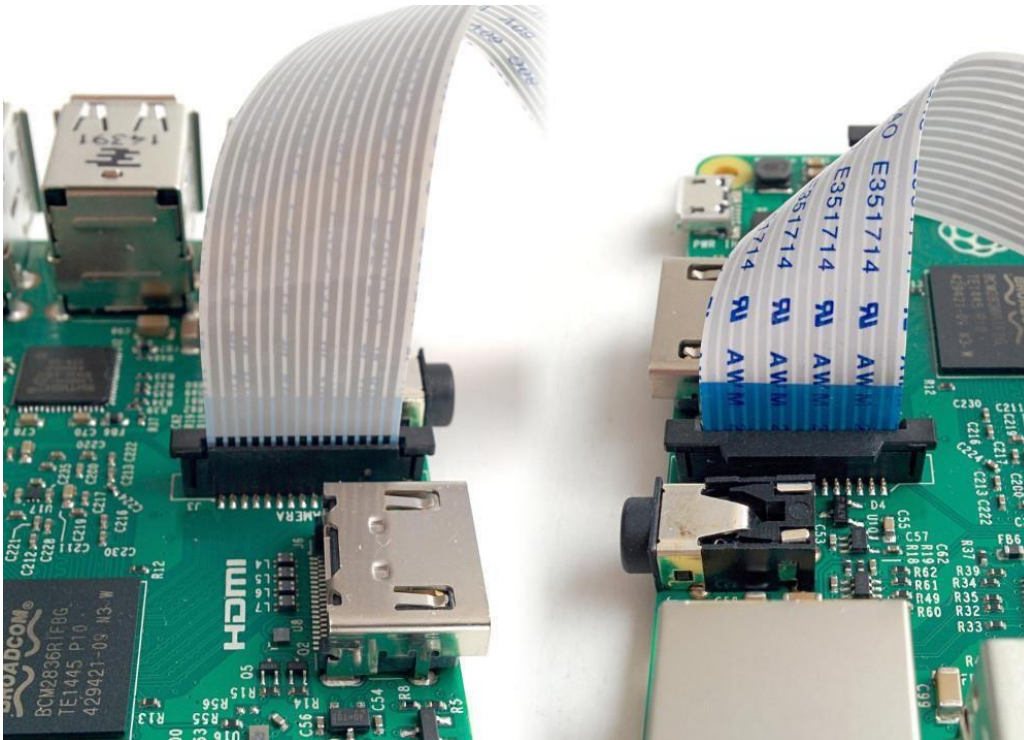
The Raspberry Pi Camera Board Features:

1. Fully Compatible with Both the Model A and Model B Raspberry Pi
2. 5MP Omnivision 5647 Camera Module
3. Still Picture Resolution: 2592 x 1944
4. Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
5. 15-pin MIPI Camera Serial Interface – Plugs Directly into the Raspberry Pi Board
6. Size: 20 x 25 x 9mm
7. Weight 3g
8. Fully Compatible with many Raspberry Pi cases

Connect the Camera Module:

First of all, with the Pi switched off, you'll need to connect the Camera Module to the Raspberry Pi's camera port, then start up the Pi and ensure the software is enabled.

1. Locate the camera port and connect the camera:



2. Start up the Pi.
3. Open the Raspberry Pi Configuration Tool from the main menu.
4. Ensure the camera software is enabled. If it's not enabled, enable it and reboot your Pi to begin.

Software Guide:

Now your camera is connected and the software is enabled, you can get started by capturing an image. You can capture an image by just typing a single line command. Open terminal window and type the command as follows:

```
$ sudo raspistill -o /home/pi/Desktop/image.jpg
```

This command will capture an image and store it at the specified location (here the location specified is

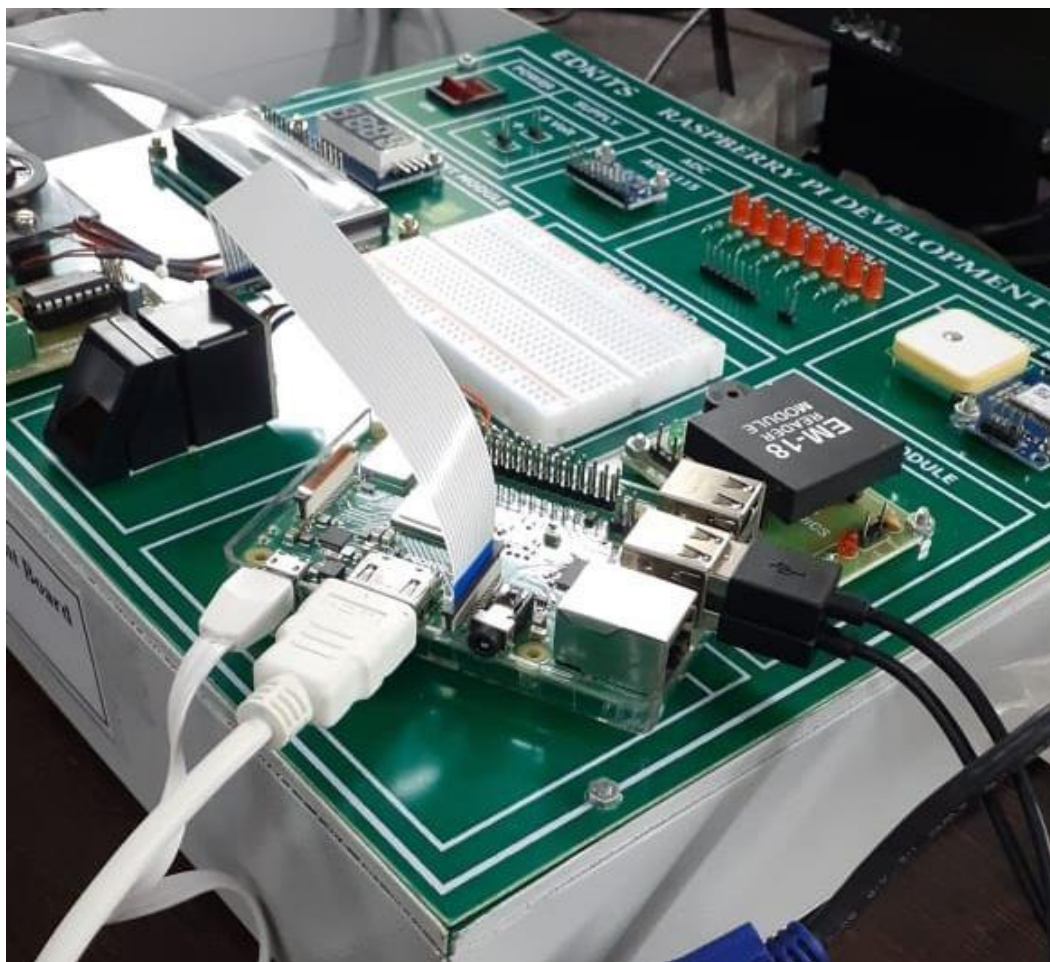
/home/pi/Desktop) with the specified name (here the name is 'image.jpg').

Code:**Visitor.py**

```
import picamera
from time import
sleep
```

```
#create object for PiCamera class
camera = picamera.PiCamera()
#set resolution
camera.resolution = (1024,
768)
camera.brightness = 60
camera.start_preview()
#add text on image
camera.annotate_text = 'Hi Pi
User'
sleep(5) #store image
camera.capture('image1.jpeg')
camera.stop_preview()
```

Output:



```
pi@raspberrypi:~$ cd Practical/
pi@raspberrypi:~/Practical$ ls
Clock FingerPrint Led osci Relay RFID Telegram Visitor Monitoring
pi@raspberrypi:~/Practical$ cd Visitor Monitoring/
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python video.py
()
video recording stopped
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
demo.h264 image1.jpeg video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ omxplayer demo.h264
Video codec omx-h264 width 640 height 480 profile 100 fps 25.000000
Subtitle count: 0, state: off, index: 1, delay: 0
V:PortSettingsChanged: 640x480@25.00 interlace:0 deinterlace:0 anaglyph:0 par:1.
00 display:0 layer:0 alpha:255 aspectMode:0
Have a nice day ;)
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
demo.h264 image1.jpeg video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ cat visitor.py
import picamera
from time import sleep

#create object for PiCamera class
camera = picamera.PiCamera()
#set resolution
camera.resolution = (1024, 768)
camera.brightness = 60
camera.start_preview()
#add text on image
camera.annotate_text = 'Hi Pi User'
sleep(5)
#store image
camera.capture('image1.jpeg')
camera.stop_preview()
pi@raspberrypi:~/Practical/Visitor Monitoring$ cat video.p
y
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.resolution = (640, 480)

print()
#start recording using pi camera
camera.start_recording("demo.h264")
#wait for video to record
camera.wait_recording(20)
#stop recording
camera.stop_recording()
camera.close()
pi@raspberrypi:~/Practical/Visitor Monitoring$
```

Conclusion:

In this practical session I have understood that how to connect the camera module with the raspberry pi kit then by running the code we can start the camera and take pictures.

Practical No. 5

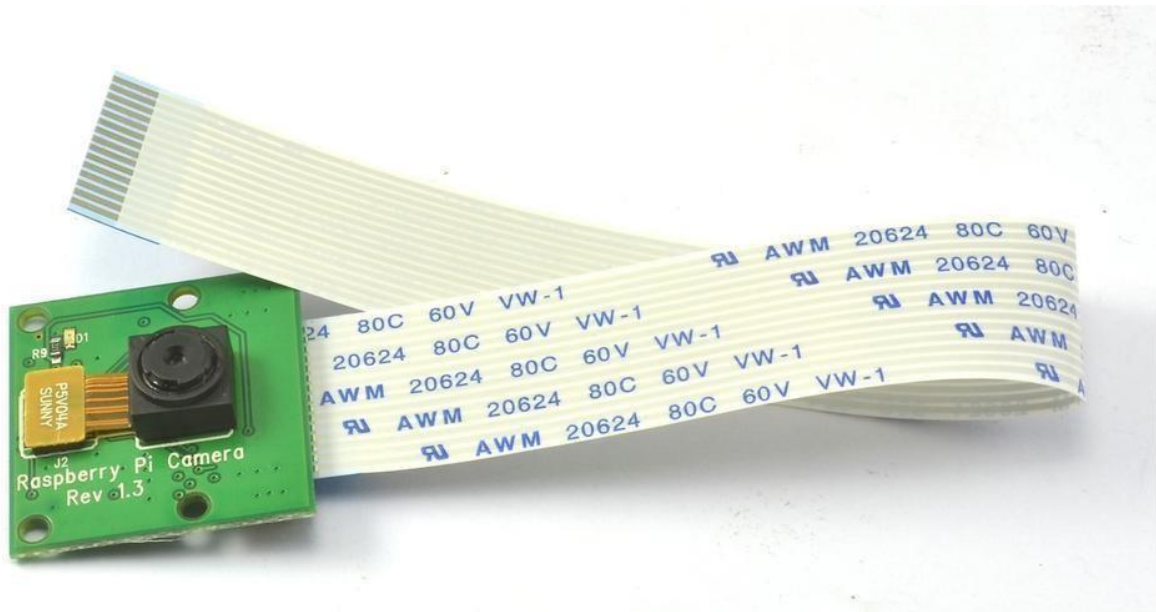
Capturing video with Raspberry Pi and Pi Camera

Aim: To capture video with Raspberry Pi and Pi camera

Hardware Guide:

For completing this lesson, you will require the Camera Module along with your initial raspberry pi setup.

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.



36 EDKITS ELECTRONICS

Capturing Video with Raspberry Pi and Pi Camera

The Camera Module is a great accessory for the Raspberry Pi, allowing users to take still pictures and record video in full HD.

For completing this lesson, you will require the Camera Module along with your initial raspberry pi setup.

Camera Module:

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system.

The Raspberry Pi Camera Board Features:

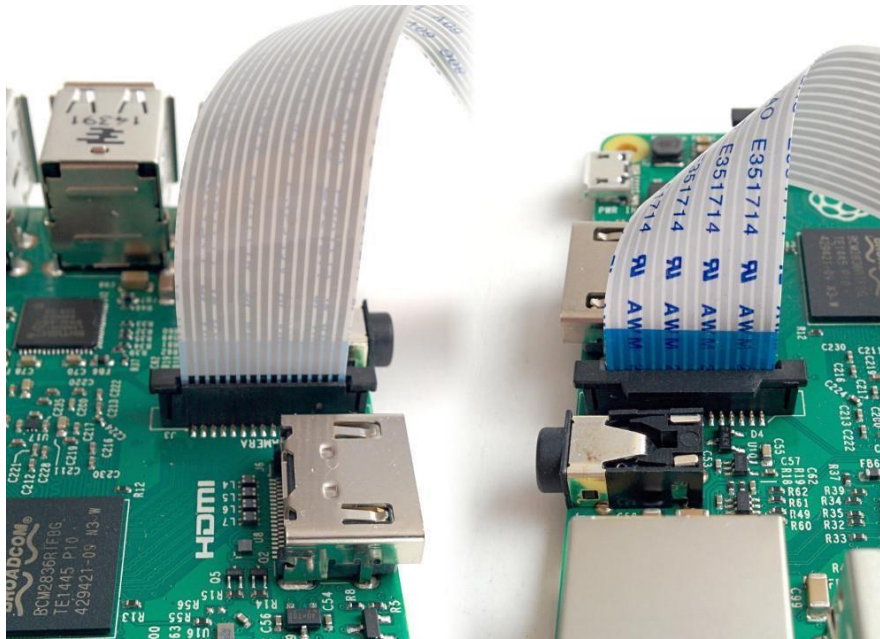
1. Fully Compatible with Both the Model A and Model B Raspberry Pi
2. 5MP Omnivision 5647 Camera Module
3. Still Picture Resolution: 2592 x 1944

4. Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
5. 15-pin MIPI Camera Serial Interface – Plugs Directly into the Raspberry Pi Board
6. Size: 20 x 25 x 9mm
7. Weight 3g
8. Fully Compatible with many Raspberry Pi cases

Connect the Camera Module:

First of all, with the Pi switched off, you'll need to connect the Camera Module to the Raspberry Pi's camera port, then start up the Pi and ensure the software is enabled.

1. Locate the camera port and connect the camera:



2. Start up the Pi.
3. Open the Raspberry Pi Configuration Tool from the main menu.
4. Ensure the camera software is enabled. If it's not enabled, enable it and reboot your Pi to begin.

Software Guide:

Now your camera is connected and the software is enabled, you can get started by capturing an image. You can capture an image by just typing a single line command. Open terminal window and type the command as follows:

```
$ sudo raspistill -o /home/pi/Desktop/image.jpg
```

This command will capture an image and store it at the specified location (here the location specified is /home/pi/Desktop) with the specified name (here the name is 'image.jpg').

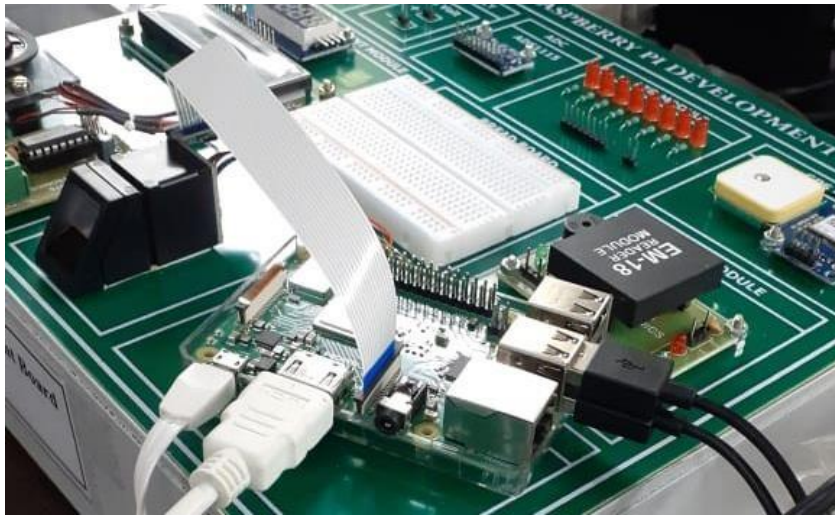
Code: **Video.py** import
picamera from time
import sleep camera =
picamera.PiCamera()


```

camera.resolution = (640,
480) print()
#start recording using pi camera
camera.start_recording("demo.h264")
#wait for video to record
camera.wait_recording(50)
#stop recording
camera.stop_recording()
camera.close() print("video
recording stopped")

```

Output:



```

pi@raspberrypi:~$ ls
Adafruit_Python_ADS1X15  led.py          Practical.zip  rhincient drives
b4camera.py              MagPi          Public        video.h264
Desktop                  Music          pyfingerprint Video5
Documents                Pictures       python games  wasiq1.py
Downloads                Practical      Templates     wasiq2.py
pi@raspberrypi:~$ cd Practical/
pi@raspberrypi:~/Practical$ ls
clock fingerprint led osci Relay RFID Telegram Visitor Monitoring
pi@raspberrypi:~/Practical$ cd Visitor\ Monitoring\
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python video.py
()
video recording stopped
pi@raspberrypi:~/Practical/Visitor Monitoring$ ls
demo.h264 image1.jpeg video.py visitor.py
pi@raspberrypi:~/Practical/Visitor Monitoring$ python visitor.py
Video codec omx-h264 width 640 height 480 profile 100 fps 25.000000
Subtitle count: 0, state: off, index: 1, delay: 0
V:PortSettingsChanged: 640x480@25.00 interlace:0 deinterlace:0 anaglyph:0 par:1.
00 display:0 layer:0 alpha:255 aspectMode:0
have a nice day ;)
pi@raspberrypi:~/Practical/Visitor Monitoring$ cat video.py
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.resolution = (640, 480)

print()
#start recording using pi camera
camera.start_recording("demo.h264")
#wait for video to record
camera.wait_recording(20)
#stop recording
camera.stop_recording()
camera.close()
pi@raspberrypi:~/Practical/Visitor Monitoring$

```

Conclusion:

In this practical I have learned to use the camera module to connect it with raspberry pi. After connection we have to run the code and execute it to take video.

Practical No. 6 GPS Module Interfacing with Raspberry Pi

Aim: To study GPS module interfacing with Raspberry Pi

Hardware Guide:

For completing this lesson, you will require the following things along with your initial raspberry pi setup

1. GPS module
2. USB to TTL converter
3. Connecting wires

Wiring up the Circuit:

1. Connect the VCC Pin of GPS Module to 3.3V Pin of USB to TTL converter
2. Connect the GND Pin of GPS Module to GND Pin of USB to TTL converter
3. Connect the Tx Pin of GPS Module to Rx Pin of USB to TTL converter
4. Connect the Rx Pin of GPS Module to Tx Pin of USB to TTL converter. 5. Lastly connect the USB to TTL converter to USB port of Raspberry Pi.

Software Guide:

Open Terminal Window and type the following command to know to which USB port the GPS module is attached: `ls /dev/ttyUSB*`

We can find whether our GPS module is working properly and the connections are correct by typing the following command: `sudo cat /dev/ttyUSB*`

(Here replace * with the port number to which GPS module is attached. You should be seeing a lot of text pass by. That means it works. Type Ctrl + c to return.)

Use 'gpsd':

You can always just read that raw data, but its much nicer if you can have some Linux software prettify it. We'll try out gpsd which is a GPS-handling Daemon (background-helper)

The first step is installing some software on your Raspberry Pi that understands the serial data that your GPS module is providing via `/dev/ttyUSB0`.

Thankfully other people have already done all the hard work for you of properly parsing the raw GPS data, and we can use (amongst other options) a nice little package named 'gpsd', which essentially acts as a layer between your applications and the actual GPS hardware, gracefully handling parsing errors, and providing a common, well-defined interfaces to any GPS module.

To install gpsd, make sure your Pi has an Internet connection and run the following commands from the console:

1. `sudo apt-get update`
2. `sudo apt-get install gpsd gpsd-clients python-gps` And install the software as it prompts you to do.

Raspbian systemd service fix:

Note if you're using the Raspbian Jessie or later release you'll need to disable a systemd service that gpsd installs. This service has systemd listen on a local socket and run gpsd when clients connect to it, however it will also interfere with other gpsd instances that are manually run (like in this guide). You will need to disable the gpsd systemd service by running the following commands:

1. `sudo systemctl stop gpsd.socket`
2. `sudo systemctl disable gpsd.socket`

Should you ever want to enable the default gpsd systemd service you can run these commands to restore it (but remember the rest of the steps in this guide won't work!):

1. `sudo systemctl enable gpsd.socket`
2. `sudo systemctl start gpsd.socket`

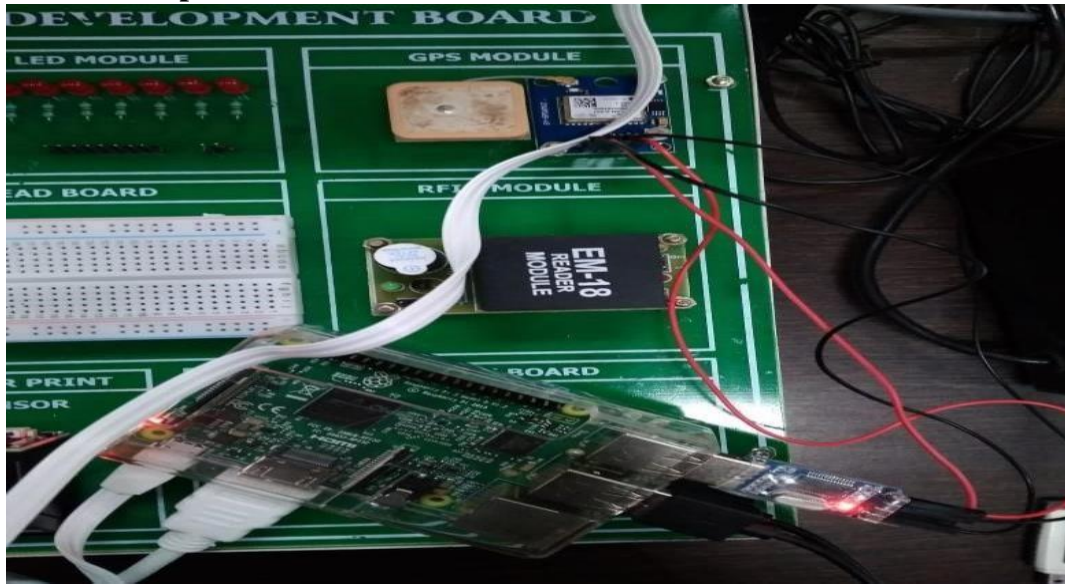
Try out 'gpsd'

After installing gpsd and disabling the gpsd systemd service as mentioned above you're ready to start using gpsd yourself.

Start gpsd and direct it to use USB. Simply entering the following command(Here we are assuming that GPS module is connected to USB0):

1. `sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock`
... which will point the gps daemon to our GPS device on the /dev/ttyAMA0 console Try running **gpsmon** to get a live-streaming update of GPS data!

Code and output:



```
$GPTXT,01,01,01,NMEA unknown msg*58
$GPRMC,,V,,,,,,,,,N*53
$GPVTG,,,,,,,,,N*30
$GPGGA,,,,,0,00,99.99,,,,,*48
$GPGSA,A,1,,,,,,,,,99.99,99.99,99.99*30
$GPGSV,1,1,00*79
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
$GPTXT,01,01,01,NMEA unknown msg*58
```

```
pi@raspberrypi:~ $ ls /dev/ttyUSB0
/dev/ttyUSB0
pi@raspberrypi:~ $ sudo cat /dev/ttyUSB0
```

Conclusion:

Learned to connect GPS module with the raspberry pi kit and then run the code that will display output as the location of the current place with timestamp and many other things.

Practical No. 7

RFID Module Interfacing with Raspberry Pi

Aim: Interfacing Raspberry Pi with RFID.

Hardware Guide:

For completing this lesson, you will require the following things along with your initial raspberry pi setup

1. RFID module
2. USB to TTL converter
3. Connecting wires

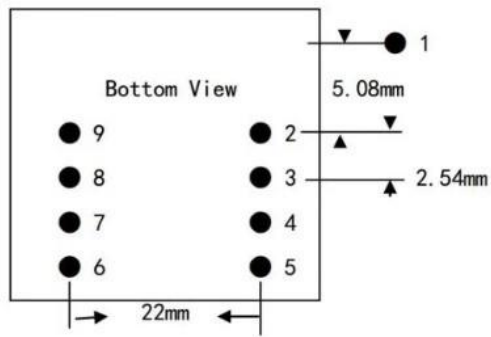
RFID Module:

RFID (Radio Frequency Identification) uses electromagnetic fields to read, monitor and transfer data from tags attached to different objects. It is not necessary that the cards are to be in visibility of the reader, it can be embedded in the tracked object. The tags can be actively powered from a power source or can be passively powered from the incoming electromagnetic fields.

EM-18 RFID reader module is one of the commonly used reader and can read any 125KHz tags. It features low cost, low power consumption, small form factor and easy to use. It provides both UART and Wiegand26 output formats. It can be directly interfaced with microcontrollers using UART and with PC using an RS232 converter.

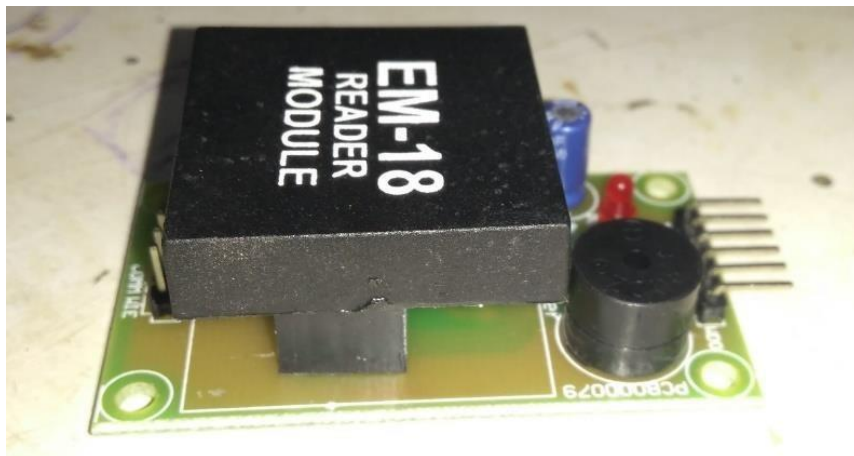
The module radiates 125KHz through its coils and when a 125KHz passive RFID tag is brought into this field it will get energized from this field. These passive RFID tags mostly consist of CMOS IC EM4102 which can get enough power for its working from the field generated by the reader.

Pinout of the RFID module is given as follows:



EM-18 RFID Reader Module – Bottom View

Pin No.	Name	Function
1	VCC	5V
2	GND	Ground
3	BEEP	BEEP and LED
4	ANT	No Use
5	ANT	No Use
6	SEL	HIGH selects RS232, LOW selects WEIGAND
7	TX	UART TX, When RS232 is Selected
8	D1	WIEGAND Data 1
9	D0	WIEGAND Data 0



Wiring up your Circuit:

1. Connect TX pin of Module to Rx Pin of USB to TTL converter
2. Connect the GND Pin of Module to GND Pin of USB to TTL converter
3. Connect the positive of 5V external supply to VCC pin of module.
4. Connect the negative/GND of 5V external supply to GND of Module
5. Finally connect the USB to TTL converter to USB of raspberry Pi

6. Connect the green LED to Pin37 of raspberry Pi
7. Connect the red LED to Pin35 of raspberry pi
8. And connect the buzzer to Pin33 of raspberry Pi.

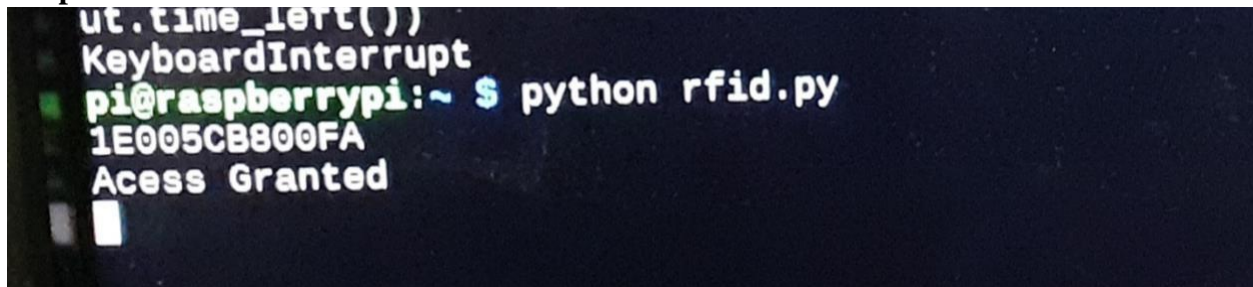
Software Guide:

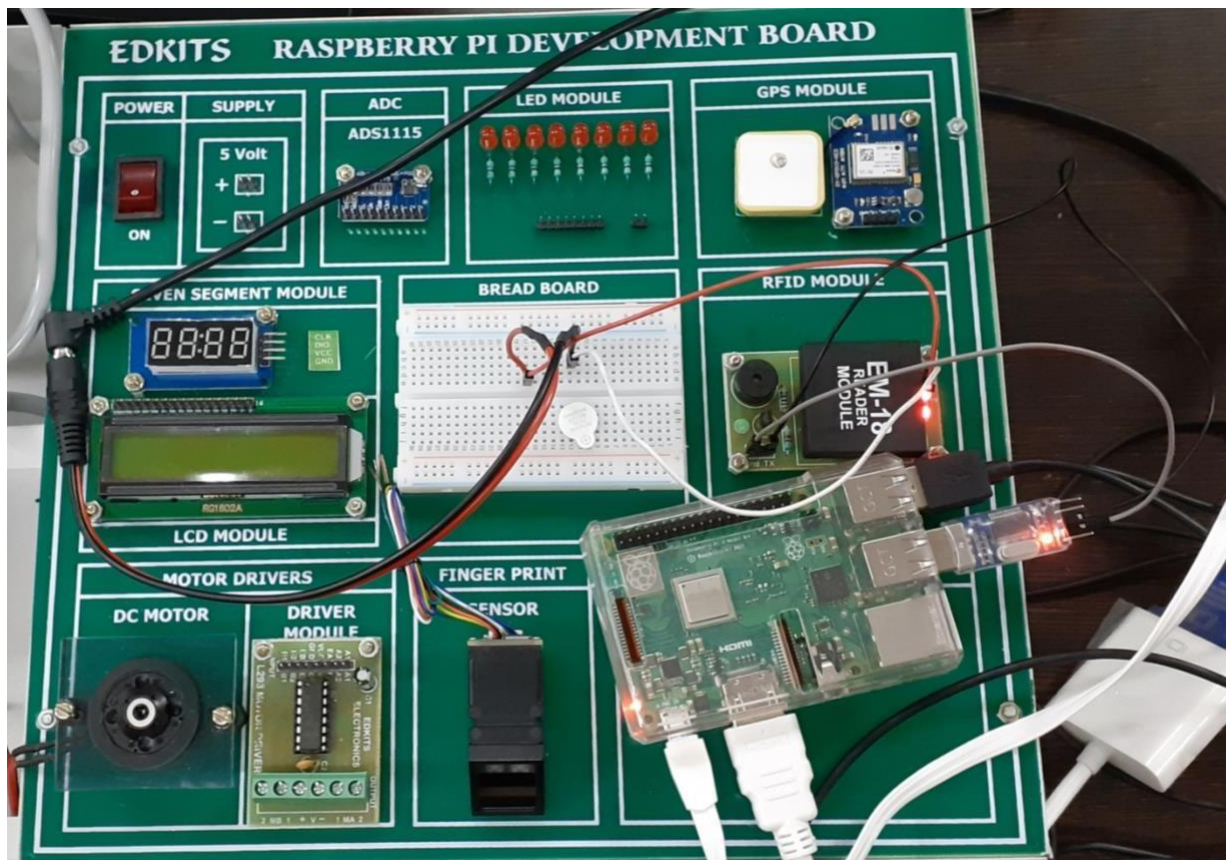
After connecting the Module to raspberry pi via USB to TTL converter, check for the port number to which it is being connected by following command: `ls /dev/ttyUSB*` Now open Python 2 Idle and write the following code and test the working of

Code:

```
import time
import
serial def
read_rfid():
    ser=serial.Serial("/dev/ttyUSB0") #open named port
ser.baudrate=9600 #set baudrate to 9600
    data=ser.read(12) # read 12 character from serial port to
data    ser.close() #close data    return data # return data
try:    while True:        id=read_rfid() #function call
print(id) # print Rfid        if id=="1E005CB800FA":
print("Access Granted")        else:
        print("Access
Denied") finally:
print("bye")
```

Output:





Conclusion:

Understood to set the hardware like connection of the RFID module and the raspberry pi. Then run the code and move card above the RFID module if the card is acceptable then it will display message as access granted otherwise it will display access denied.