Jayesh Bharat Devre
TE-A Roll No: 36

Camlin Page
Date / /

Experiment No. 1.

Aim- To study Raspberry Pi, Beagle board, Arduino & other micro-controller.

Theory - Study of Raspberry pi 3

It is a series of small single board computers developed in United Kingdom by Raspberry pi foundation to promote the teaching of basic computer science in schools and in developing countries.

A Raspberry pi zero with smaller size & reduced i/p and o/p & general input output capabilities was released in Nov. 2015 for US $5 on 28 Feb 2017 the Raspberry Pi zero w was launched. which is like Raspberry Pi zero with WiFi & Bluetooth for US $10 Processor speed ranges from 700 MHz to 1.2 GHz for Pi 3 onboard memory ranges from 256 MB to 1GB RAM. The boards have 1 to 4 USB ports for video o/p HDMI and composite video are supported with std 3.5 mm phone jacks for audio output.

Study of Beagle board -

The beagle board is low power single board computer produced by texas instruments in association with Digi key & Network element 14. The Beagle board also designed from open source software development in mind. The board was developed by small team of Engineers as an educational board that could be used in colleges. The board was designed using cadence or CAD for schematics & cadence Allegro for PCB manufacturing No. simulation software is used.

Raspberry Pi 3 Model B

Fig.1. Raspberry Pi3 Kit

Fig.2.Beagle Bone Kit

Roll No. 36

Study of Arduino -

The Arduino project started at the ~~International~~ Interaction design Institute IVREA (IDII) at that time students used Basic stamp micro-controller at a cost of $100. a considerable experce for many students.

The Initial Arduino core team consisted of Massimo Banzi, David Cuartilles. In October 2016 febrico musto, Arduino's former CEO, secured a 50% ownership of company, In April 2017 wired reported that musto has fabricated this academic record on his company's website Personal linked in accounts (even on Italian business documents. musto was until recently listed a holding a PHD from MIT. In some cases his bios has claimed an MBA from New york university. wired reported that Neither university has any record of Mustos attendence & musto later admitted in interview that he Never earned those degrees

ARM recognized independence as a core value of Arduino without any lock with ARM architecture. arduino intends to continue to work with all technology vendors & architectures.

Conclusion- Thus we have studied the history of Raspberry Pi, beagle board & Arduino.

Arduino Kit Fig.3.

Name- Jayesh Bharat Devre
TE(A) Roll No: 36

Camlin Page
Date / /

Experiment No. 2

Aim- Study of different operating system for Raspberry Pi understanding the process of os installation on Raspberry Pi.

No matter how good & powerful the hardware of Raspberry Pi is, without an operating system it is just piece of silicon, fiberglass & few other semi-conductor materials. There are several different operating systems for Raspberry Pi's.

(1) Raspbian- It is most popular linux based operating system for Raspberry Pi, It is an open source operating system based on debian which has been modified specifically for Raspberry pies. Raspberian is default, free & open source operating system & often comes with Raspberry pi kit. It is an official operating system of Raspberry pi foundation

(2) Pidora -
After waiting for long Raspberry pi users are finally getting an optimized version of fedora, the Pidora, to Replace the current Raspbian os. The News caused excitement among Raspberry pi community, who are finally getting the opportunity to explore fedora Remix for Pi ended up as failure. However the Seced center for development of open technology the authority group behind pidora, is confident that the Raspberry pi community would love newly optimized os

(3) Arch linux -

It is an excellent choice for many reasons. one of the greatest advantage is its simplicity in approach and attitude. Arch gives you the ability to build your system from ground up, including only the software you actually need. Arch has now finished its transition to system D from init scripts

(4) OSMC -

OSMC (open source media centre) is a free & open source media player based on linux. founded founded in 2014, osmc let you play back media from your local network, attached storage and internet. osmc is leading media centers in terms of feature set & community and is based on kodi project.

(5) RetroPIE -

It allows you to turn your Raspberry PI into a retro gaming machine It platform developed on the base of Raspbian Emulation station, Retro pie enable you to play your favorite arcade, home-console & classic pc games with minimum setup.

(6) RTSC OS- It is designed by Acorn computers ltd cambridge England. It is first release in 1987 It specifically designed to run on ARM chipset. It is fast compact & efficient

Camlin Page
Date / /

⑦ Firefox os-

Firefox os which is more associated with being a linux kernel based open source operating system primarily designed for smartphones & tablets computers. It was primarily designed as a community based alternative system utilizing open standards & HTML5 applications.

⑧ KALI linux-

It is debian based secutity auditing linux distribution. It is specially designed for digital forensics & penetration testing It is maintained & funded by offensive Security limited.

Conclusion- Thus we have studied Installation for various os in Raspberry PI.

Date

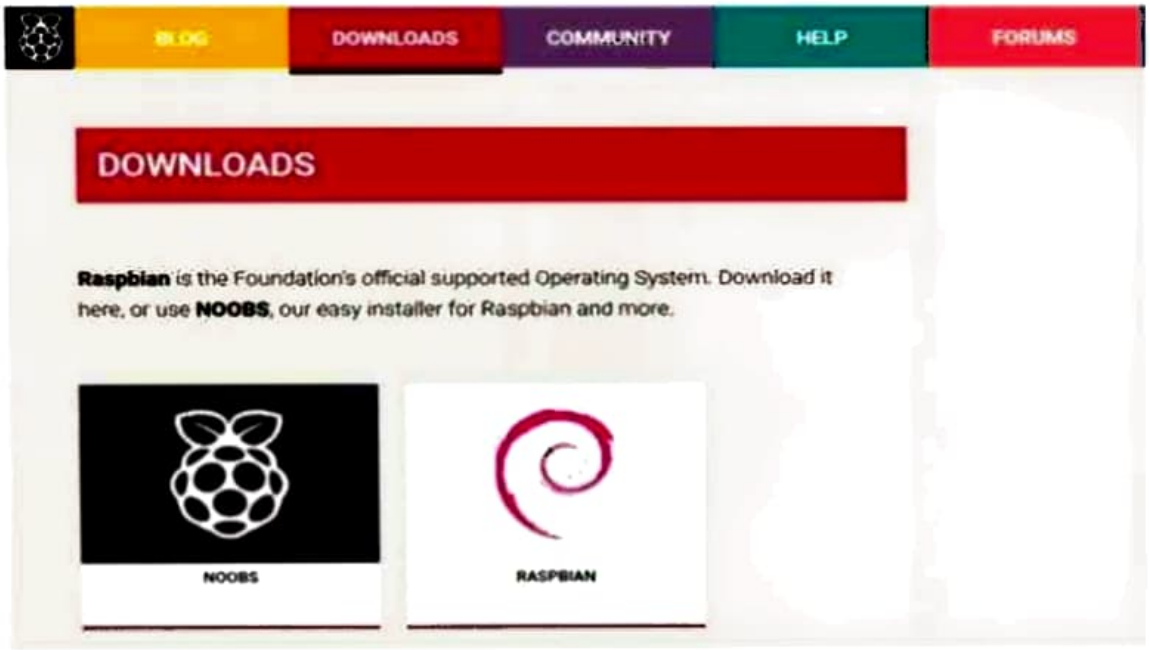
Installing OS for Raspberry-Pi-3

Aim: To understand the OS installation for Raspberry-Pi 3

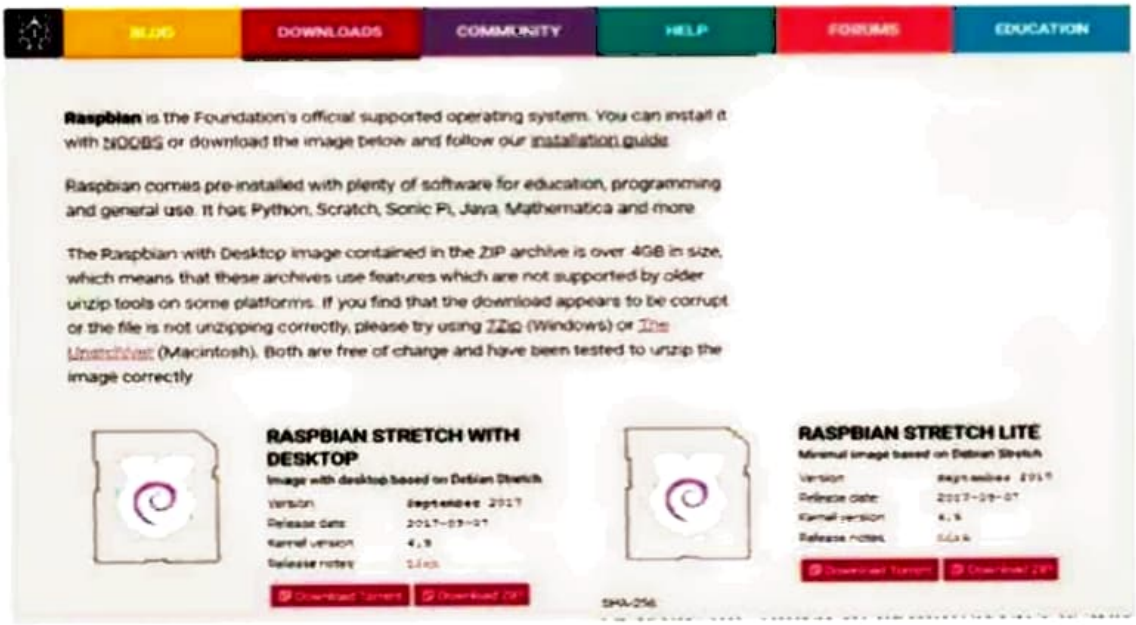
Process of OS installation on Raspberry Pi Board

1. Open the website: www.raspberrypi.org
2. Click on the "Downloads" tab

3.



4. **Click on the "RASPBIAN" option.**



5. We require "RASPBIAN STRETCH WITH DESKTOP", under this heading, click on "Download Torrent" option.
6. A "Torrent file" is downloaded.
7. But the actual OS is present in the ZIP file of this torrent.
8. So using this "Torrent file" and the "Bit Torrent" software, we download the ZIP file of the Raspbian OS.
9. So download the "Bit Torrent" software & install it.
10. Now open the "Bit Torrent" software.
11. Click on the option "+" and under this click on "Add Torrent".

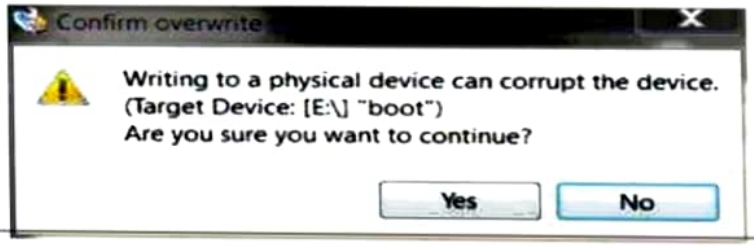
12. Here select the path of downloaded "Torrent file".

13. After selecting the torrent file, following window appears. Here click on OK

15. Now we have to write this disk image on SD card.
16. To write the OS on SD card, we require the software "win32 disk imager". So download thissoftware and install it.
17. After completion of the installation, the following window appears.

18. Open the unzipped file in the "Image file" option byselecting the path from the Blue icon. The selected path is shown in the below image.
19. Now plug-in the SD card reader having SD card inside it, in the USB port of your PC.
20. Ensure that your SD card reader is having the same drive which is shown in the Device option (near the blue icon)

21. After ensuring that the "Image file path" and the "Device" are selected correctly, now click'Write' button to write the image on the SD card.
22. After this the following window appears.



23. Here click 'Yes' and Confirm the overwrite
24. Image file will be written on SD card.
25. After the procedure is completed, it gives "Write Successful" message.
26. Congratulations! Your SD card is ready with your OS to work in the Raspberry-Pi-3 board.
27. Insert this SD card in Raspberry pi3.

28. Do the necessary connections and make the power ON. Your Raspberry-Pi starts and theDesktop of the OS is shown on the screen. Now Raspbery-Pi is ready to work on.

Name- Jayesh Bharat Devre
Roll No. 36 TEA

Camlin Page
Date / /

Experiment No. 3

Aim-

Study of connectivity & configuration of Raspberry Pi / Beagle board circuit with basic peripherals, LED's understanding GPIO & its use in program.

Theory :

Connectivity & configuration of Raspberry Pi Guides to configure Raspberry Pi

① raspi-config-

The raspberry pi configuration in Raspbian allowing you to easily enable features such as camera... & change your specific setting such as Keyboard layout.

② config.txt -

The Raspberry Pi configuration file.

③ wireless -

configuration to a pi as a wireless access point using Raspberry Pi 3 & Pi zero w's In built wireless connectivity or USB wireless Dongle.

④ Audio config-

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

⑤ Camera config-

Installing & setting up Raspberry Pi camera board.

⑥ External storage config-

mounting and setting up external storage on Raspberry Pi:

(7) localisation-
Setting up your pi to work in local language.

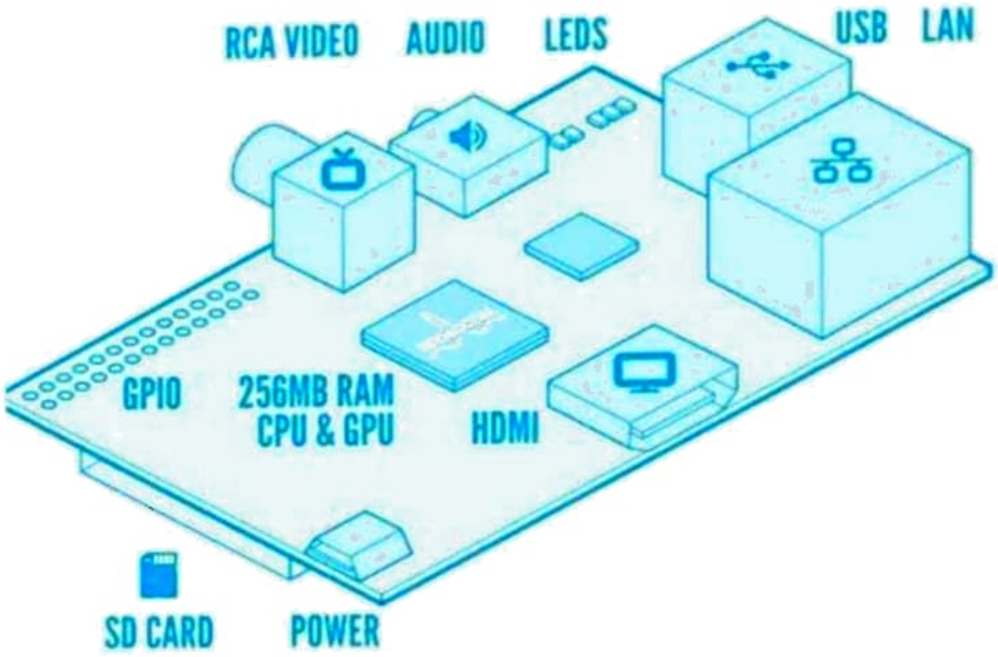
(8) Default Pin Config-
changing default Pin states.

(9) Device tree config-
Device trees, overlays & Parameters.

(10) Kernel command line-
linux kernel accepts a command line of parameters during boot.

(11) UART Configuration-
(12) Screen saver
(13) Wireless access point.

Connectivity of raspberry pi-
Connectivity is truly superb for such tiny device. especially on b version of Raspberry pi There are 2 USB 2.0 ports that can be used to hookup peripherals or adapters & this can be further expanded with power hub. Its worth nothing that both ports already share the bandwidth of singe channel to system bus.


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

GPO mode-

This option specifies that you are referring to pins by number of Pin the plug. and in the middle of diagrams below.

The GPIO.BCM option means that you are refering to the pins by the broad com soc channel number. this are the numbers after "GPIO" in green rectangles arround outside of bllow diagrams

① unfortunately the BCM numbers changed between versions of Pi1 model B.

② The Raspberry Pizero, Pi2B & Pi3B uses the Same numbering as B+

Resistor-

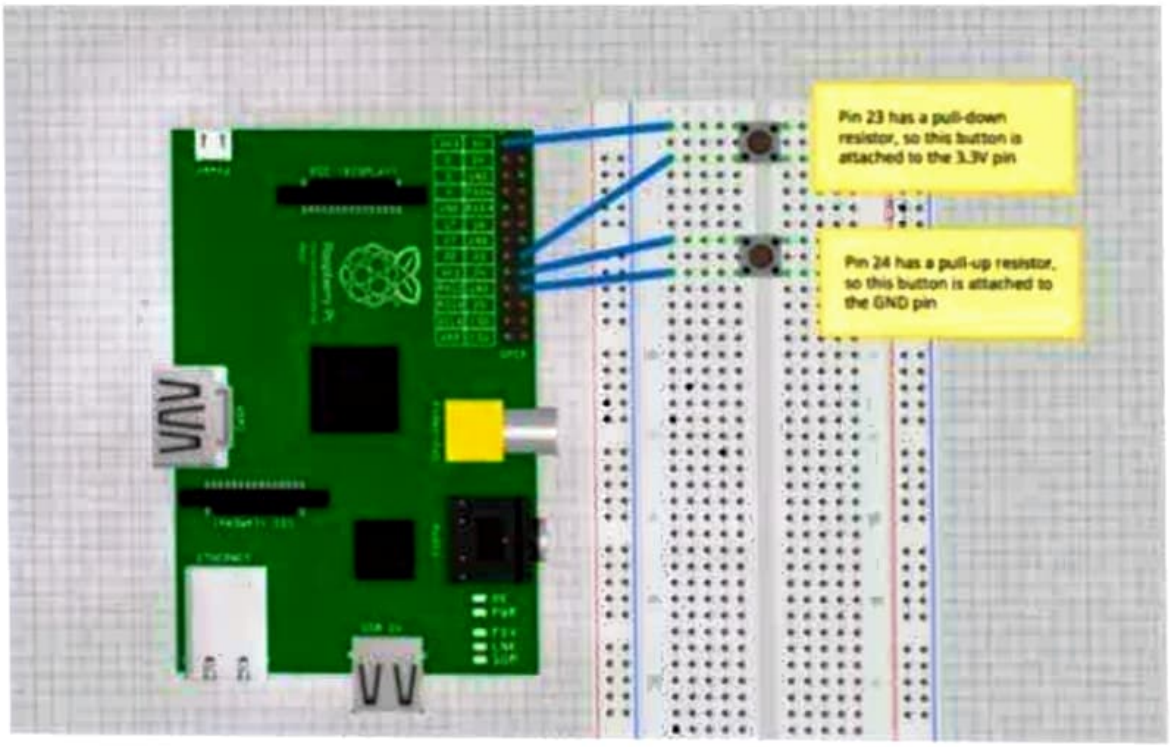
you must always use resistors to connect LED's upto the GPIO pins of the Raspberry Pi can only supply a small current. The LED's will want to Draw more & if allowed they will burn out the Raspberry Pi. Therefore putting resistors in the circuit will ensure that only this small current will flow & Pi will not be damaged.

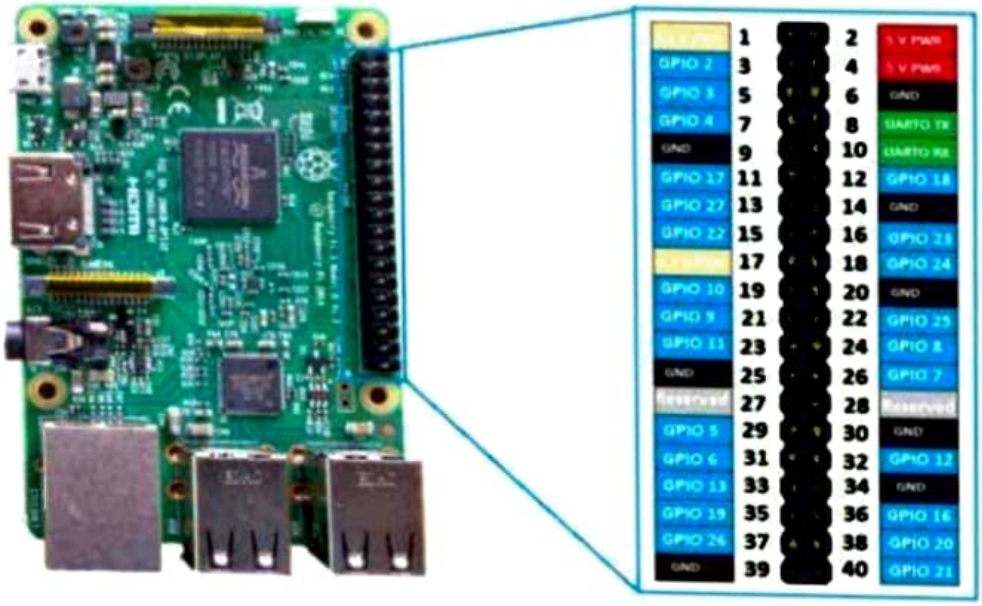
Jumper wires-

This 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.


Pin 23 has a pull-down resistor, so this button is attached to the 3.3V pin
Pin 24 has a pull-up resistor, so this button is attached to the GND pin

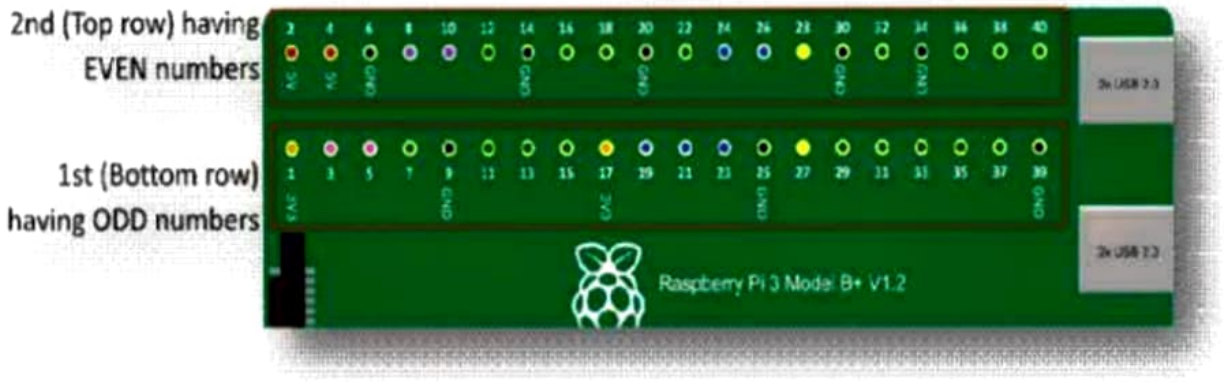

GPIO 2
GPIO 3
GPIO 4
GND
GPIO 17
GPIO 27
GPIO 22
GPIO 10
GPIO 9
GPIO 11
GND
Reserved
GPIO 5
GPIO 6
GPIO 13
GPIO 19
GPIO 26
GND
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
GND
GPIO 18
GND
GPIO 23
GPIO 24
GND
GPIO 25
GPIO 8
GPIO 7
GND
GPIO 12
GND
GPIO 16
GPIO 20
GPIO 21

Camlin Page
Date / /

- The end with the piece of plastic with a hole in it. it will go onto the Raspberry Pi's GPIO Pins.

Conclusion-

Thus we have studied about connectivity and Configuration of Raspberry Pi & also use GPIO.


2nd (Top row) having EVEN numbers
1st (Bottom row) having ODD numbers
Raspberry Pi 3 Model B+ V1.2

Name- Joyesh Devre
Roll No. 36-A

Camlin Page
Date / /

Experiment No. 4

Aim- understanding the connectivity of Raspberry Pi / Beagle board circuit with IR sensor. write an application to detect obstacle & notify user using LED's

Theory:
Infrared sensor works by emitting infrared signal / radiation & receiving of the signal when signal bounces back from any obstacle In other words the IR sensor works by continuously sending signal & continuously receiving signal comback by bouncing on any obstacle in the way.

Component IR sensor-
① Emitter- This component continuously emits infrared signal.

② Receiver - It waits for the signal which is bounced back by obstacle.

③ Indicator - on board LED to signal if obstacle is deducted by sensor.

④ output - Could be used as input for further processing of signal.

⑤ Ground - Ground / Negative point of circuit

⑥ Voltage - Input 3.3 V

IR Sensor Fig.1

Camlin Page
Date / /

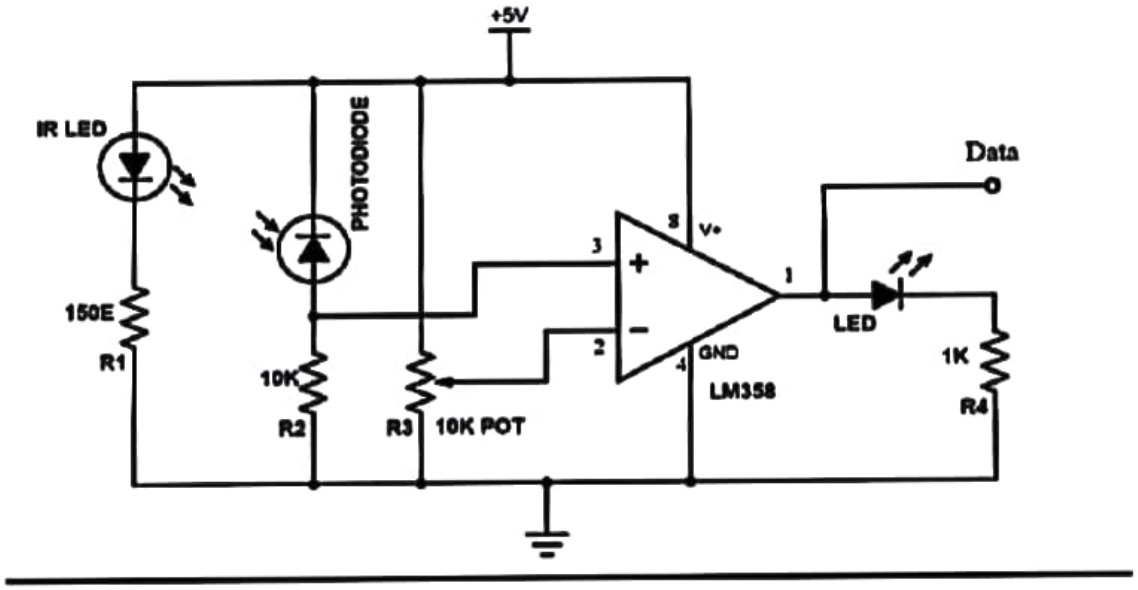
Objective-
We will creating a circuit using following components to detect obstacle.

1. Raspberry Pi 3
2. IR Sensor
3. 1 LED
4. 1 Resistor (330 Ω)
5. Few jumper cables
6. 1 Breadboard.

Circuit- to detect obstacles-
We will be creating a circuit which will turn on LED when obstacle occures. And as soon as obstacle removed from the way the LED will turn off.

Part 1: Connecting IR Sensor.
IR sensor has 3 pins, viz VCC, GND & OUT- We will use GPIO 17 ~~confused with~~ from receiving input from sensor.

1. Connect GPIO 17 from Raspberry pi to breadboard
2. Connect out pin of sensor with bread board.
3. Connect GND with Negative line on left side of breadboard.
4. Connect GND of IR sensor to breadboard.
5. Connect GND from step 3 to breadboard.
6. Connect VCC of IR sensor to breadboard.
7. Connect 3V3 to positive line of left side of breadboard.
8. Connect 3V3 to breadboard.



Circuit diagram of IR Sensor Fig.2

Objective

We will be creating a circuit using following components to detect obstacle.

1. Raspberry Pi 3
2. IR (Infrared) Sensor
3. 1 LED

Camlin Page
Date / /

Part 2:

Connecting LED

objective is to turn on LED when obstacle is detected.

① Connect GPIO 4 from board to breadboard.
② Connect positive point of LED to breadboard
③ Connect Negative point to the breadboard
④ Use resistor (330 Ω) to connect negative to Negative point of LED.

Now we are ready to send signal based on input received from IR sensor to turn on/off LED.

Part 3: Code to connect IR Sensor I/p with LED status.

```
from GPO zero import LED
form signal import Pause
import Rpi GPO as GPO
import time
GPo.set mode (GPIO. BCM)
LED_PIN. = 27
IR_PIN = 17
indicator = (LED PIN)
GPIO.setup (IR_PIN, GPIO.IN).
count = 1
while True:
    got Something = GPIO.input (IR_PIN)
    if got_something
        indicator.on()
    print "{:>3} Got. Something". format(count)

else
```

Camlin Page
Date / /

```
Indicator.OFF()
Print("{:>3} Nothing detected".format(count))
        count += 1
time.sleep(0.2)
```

Part4 : Executing the code

1] open terminal (on Pi itself or login through SSH login)

2] Navigate to the directory where the above code is saved.

3] Type sudo python3 ir-obstacle.py & press <enter>

on terminal it will start printing the status based on conditions.

Conclusions:

Thus we done connectivity of Raspberry Pi / Beagle board circuit with IR sensor, write an application to detect obstacle & notify user using LED's.

Name- Jayesh Devre
Roll No. 36-A

Carnlin Page
Date / /

Experiment 5

Aim- understanding & connectivity of Raspberry Pi /Beagle board with camera. write an application to capture & store the image.

Theory -

Raspberry pi camera module V2 replaced the original camera module in April 2016. The V2 Camera module has a Sony IMX219-8 megapixel sensor. The camera module can be used to take high definition video as well as stills photograph its easy to use for beginners, but has plenty to offer advanced user's if you are looking to expand your knowledge.

Pi camera:

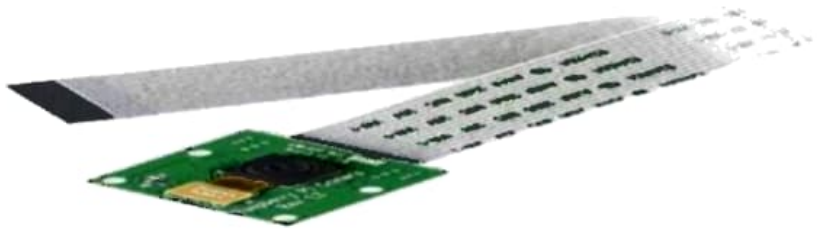
Camera preview:

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
camera.start_Preview()
sleep(10)
camera.stop-Preview()
```

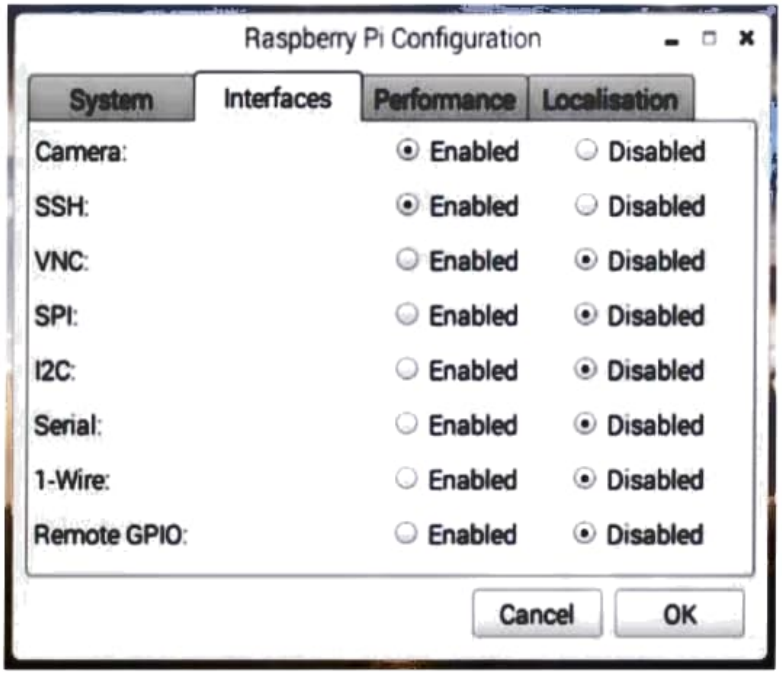
Rotating the Camera:

```
Camera.rotation = 180
Camera.start_preview()
sleep(10)
```

Pi Camera



Open Raspberry Pi Configuration and Enable the Camera



camera.stop_preview()

Storing the Image

```
from picamera import PiCamera
from time import sleep
Camera = PiCamera()
Camera.start Preview()
Sleep(10)
Camera.capture('/home/Pi/Desktop/image1.jpg')
Camera.stop_preview()
```

Recording the video

```
from pi camera import Pi camera
from time import sleep
Camera = Pi Camera()
Camera.start_preview()
Camera.start_recording('/home/pi/video.h264')
Sleep(10)
Camera.stop_recording()
camera.stop_preview()
```

Converting & playing video.

The video format need to get converted to MP4.
Sb install gpac.
sudo-apt-get install gpac.
Now convert the video to MP4.
MP4 Box -fps 30 -add video.h264 Video.mp4

Conclusion- Thus we have studied Picamera & also stored the images & videos using Pi camera

Name - Jayesh Bharat Devre
Roll No. 36-A

Camlin Page
Date / /

Experiment No. 6

Aim - understanding & connectivity of Raspberry pi with a zigbee module. write a Network application for communication between two devices using zigbee.

Theory -

Zigbee is a communication device used for data transfer between controllers, computers systems really anything with serial port. As it works with low power consumption the transmission distances is limited to 10-100 meters line of sight zigbee devices can transmit data over long distances by passing data to a mesh network of intermediate devices to reach more distant ones. The technology defined by the zigbee specification is an less expensive than networks

Python script to perform zigbee communication -

```
import serial
# Enable USB communication
ser = serial.Serial('/dev/ttyUSB0', 9600, TIMEOUT=1)
while True:
    ser.write("Hello User \r\n") # write a data.
    incoming = ser.readline().strip()
    print 'Received data:' + incoming
```

Conclusion -

Thus we have done zigbee communication between two Raspberry Pi devices.

Interfacing of Zigbee

Name- Tajesh Devre
Roll No- 36-A-TE

Camlin Page
Date / /

Experiment No. 7

Aim- write an application using Raspberry Pi / Beagle board to control the operation of stepper motor.

Theory-

· Stepper motor-

In stepper motor, as the name itself says the rotation of shaft is in step form. There are different types of stepper motor. and here we will be using most popular one that is unipolar stepper motor. Unlike DC motor, we can rotate stepper motor to any particular angle by giving it proper instructions.

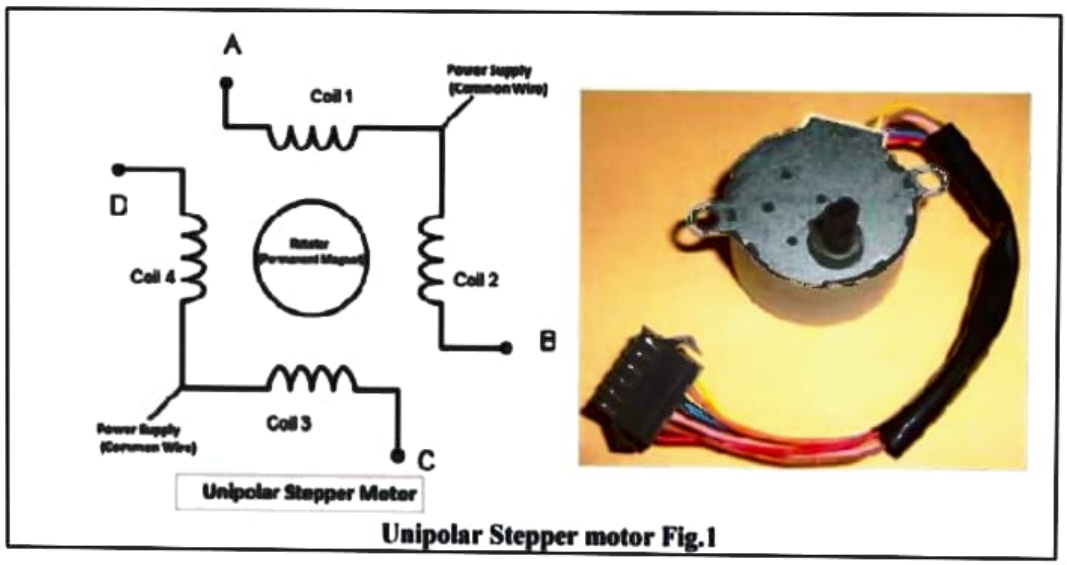
To rotate this 4 stage stepper motor we will deliver power pulses by using stepper motor driving circuit. The driver circuit takes logic triggers from P.I. If we control the logic triggers we control the power pulses & hence control the speed of stepper motor.

There are 40 GPIO output pins in Raspberry Pi 2 But out of 40 only 26 GPIO pins can be programmed Some of this pins performs special functions with Special GPIO put aside, we have only 17 GPIO remaining. And the sum of currents from all GPIO pins cannot exceed 50 mA.

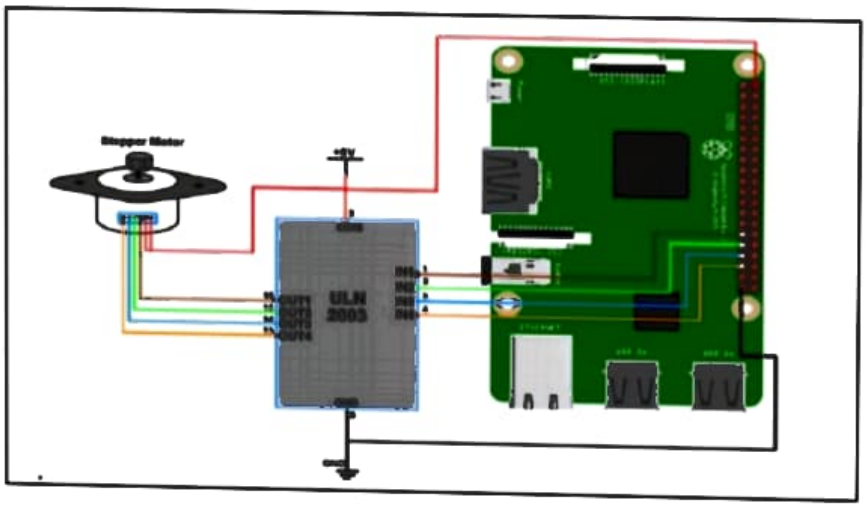
Sample program-

Stepper motor interfacing with Raspberry pi

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



Unipolar Stepper motor Fig.1



```
# asign GPIO pins from motor
motor_channel = (29, 31, 33, 35)
GPIO.setwarnings (FALSE)
GPIO.setmode (GPIO.BOARD)
# For defining more than 1 GPIO channel as ip/op
  use GPIO.setup (motor_channel, GPIO.OUT)
motor_direction = input ('Select motor direction a= anticl' c=clkwise )
  while True:
try:
  if (motordirection == 'c'):
  print ('motor running clockwise)
GPIO.OUTPUT (motor_channel, (GPIO.HIGH, GPIO.LOW, GPIO.LOW,
sleep (0.02)                              GPIO.HIGH))
else if (motor direction == 'a'):
  print ('motor running anticlockwise)
GPIO.OUTPUT (motor_chamel (GPIO.HIGH, GPIO.LOW
sleep (0.02)                    GPIO.LOW, GPIO.HIGH))

# Press ctrl+c for keyboard intrupt
  except keyboard intrupt:
# query for setting motor direction or exit
motor_direction = input ('select motor direction a=anticlockw
                                  (=clockwise or q=exit
# check for exit
  if (motor_direction == 'q')
  print ('motor stopped')
  Sys.exit (0).
```

Conclusion - Thus we have implemented application of stepper motor using python with Raspberry pi.

Name - Jayesh Devre
Roll No - 36-TE-A

Camlin Page
Date / /

Experiment No. 8

Aim - Write an application using Raspberry Pi / Beagle board to control the operation of hardware simulated traffic signal.

Theory - Attaching the traffic lights

The low voltage labs traffic light connect to the Pi using four pins. One of these needs to be ground, the other being actual GPIO pins used to control each of individual LED's.

Before powering up the Pi, attach the traffic lights so that the pins connect to the GPIO pins highlighted in red.

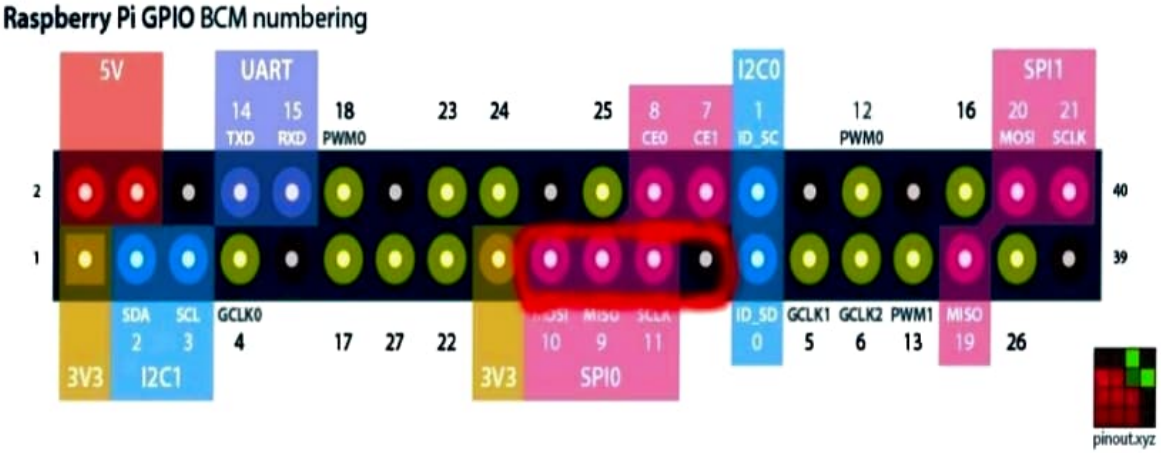
Programming the traffic lights

First you need to install a couple of extra software packages needed to allow you to download my sample code, and to give python access to GPIO Pins on Pi. Each of following at command line

```
Sudo apt-get install python-dev python-rpi.gpio
                                        .git
```

How it works -

The code for this is very simple. It starts by importing the RPI.GPIO library, plus time which gives us a timed wait function, signal that allows us to trap the signal sent when the user tries to quit the program & sys so we can send an appropriate exit signal back


Raspberry Pi GPIO BCM numbering
5V
UART
14 15
TXD RXD
18
PWM0
23
24
25
8 7
CE0 CE1
I2C0
1
ID_SC
12
PWM0
16
SPI1
20 21
MOSI SCLK
2
1
40
39
SDA SCL
2 3
I2C1
GCLK0
4
17
27
22
3V3
10 9 11
SPI0
ID_SD
0
GCLK1 GCLK2 PWM1
5 6 13
MISO
19
26
pinout.xyz

Camlin Page
Date / /

to the operating system before terminating

```
import RPI.GPIO as GPIO
    import time
    import signal
    import sys
```

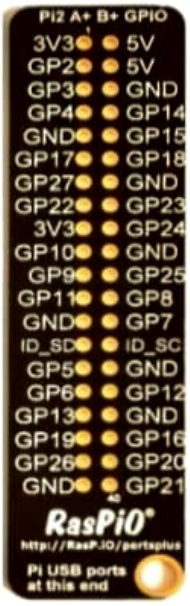
Next we put the GPIO library into "BCM" or "Broadcom" mode and sets pins 9, 10 and 11 to be used as outputs.

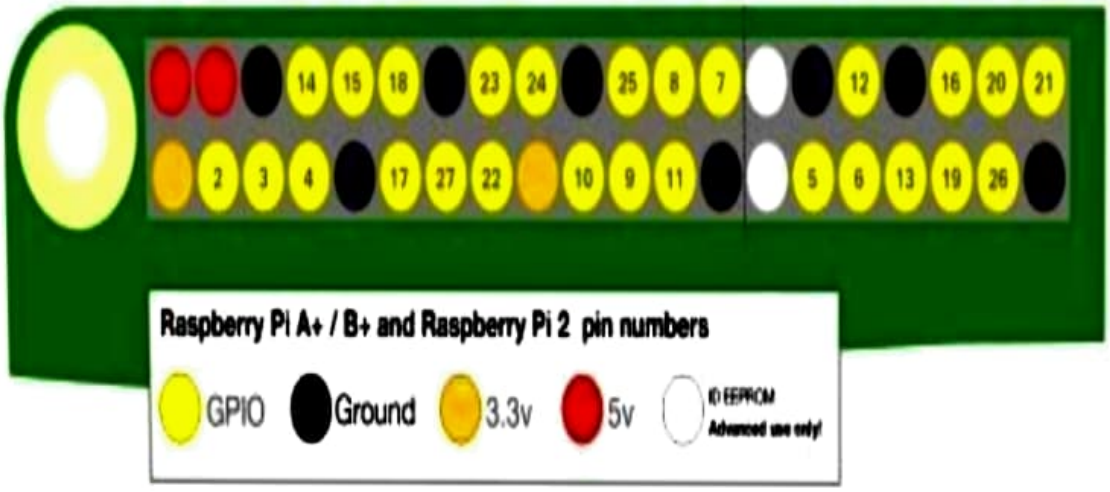
```
# setup.
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
```

The main part of program will run in an infinite loop until the user exists by stopping python with ctrl c

```
# Turnoff all lights when user ends demo.
def allLightsoff(signal, frame):
        GPIO.output(9, False)
        GPIO.output(10, False)
        GPIO.output(11, False)
        GPIO.cleanup()
        Sys.exit(0)
signal.signal(signal.SIGINT, allLightsoff)
```

The main body of code when consists of an infinite while loop that turns on red light (pin9) waits, turns on another light(pin10) waits, then cycles through the rest of traffic light pattern by turning the appropriate LED's on & off

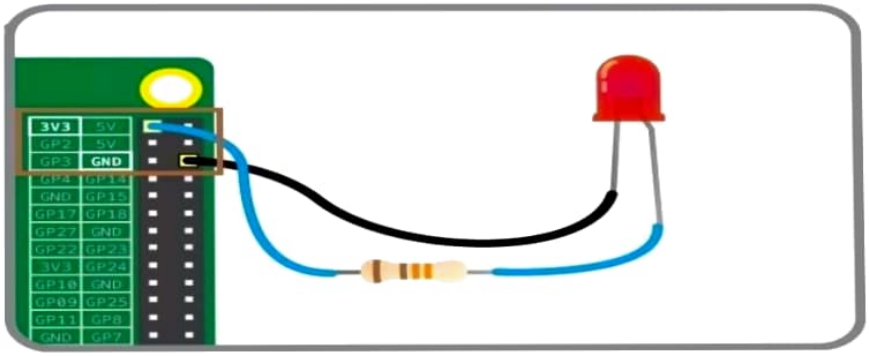

Pi2 A+ B+ GPIO
3V3 5V
GP2 5V
GP3 GND
GP4 GP14
GND GP15
GP17 GP18
GP27 GND
GP22 GP23
3V3 GP24
GP10 GND
GP9 GP25
GP11 GP8
GND GP7
ID_SD ID_SC
GP5 GND
GP6 GP12
GP13 GND
GP19 GP16
GP26 GP20
GND GP21
RasPiO
http://RasPiO/portsplus
Pi USB ports at this end


14 15 18 23 24 25 8 7 12 16 20 21
2 3 4 17 27 22 10 9 11 5 6 13 19 26
Raspberry Pi A+ / B+ and Raspberry Pi 2 pin numbers
GPIO
Ground
3.3v
5v
ID EEPROM
Advanced use only!

When control-c is passed an interrupt SIGINT is sent. This is handled by the all lights off function that switches all lights off. tidies up the GPIO library stuff and exists cleanly back to operating system

Conclusion-

Thus we have implemented the application for traffic signals using Raspberry pi


3V3
5V
GP2
5V
GP3
GND
GP4
GP14
GND
GP15
GP17
GP18
GP27
GND
GP22
GP23
3V3
GP24
GP10
GND
GP09
GP25
GP11
GP8
GND
GP7


[pi@raspberrypi: ~]
Programming
Office
Internet
Games
Accessories
Help
Preferences
Run...
Shutdown...
BlueJ Java IDE
Geany Programmer's Editor
Greenfoot Java IDE
Mathematica
Node-RED
Python 2 (IDLE)
Python 3 (IDLE)
Scratch
Scratch 2
Sense HAT Emulator
Sonic Pi
Thonny Python IDE
Wolfram


Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.3 (default, Jul 6 2012, 13:39:51)
Untitled
File Edit Format Run Options Windows Help

Name - Tajesh Devre
Roll No: 36-TE-A

Camlin Page
Date / /

Experiment No. 9

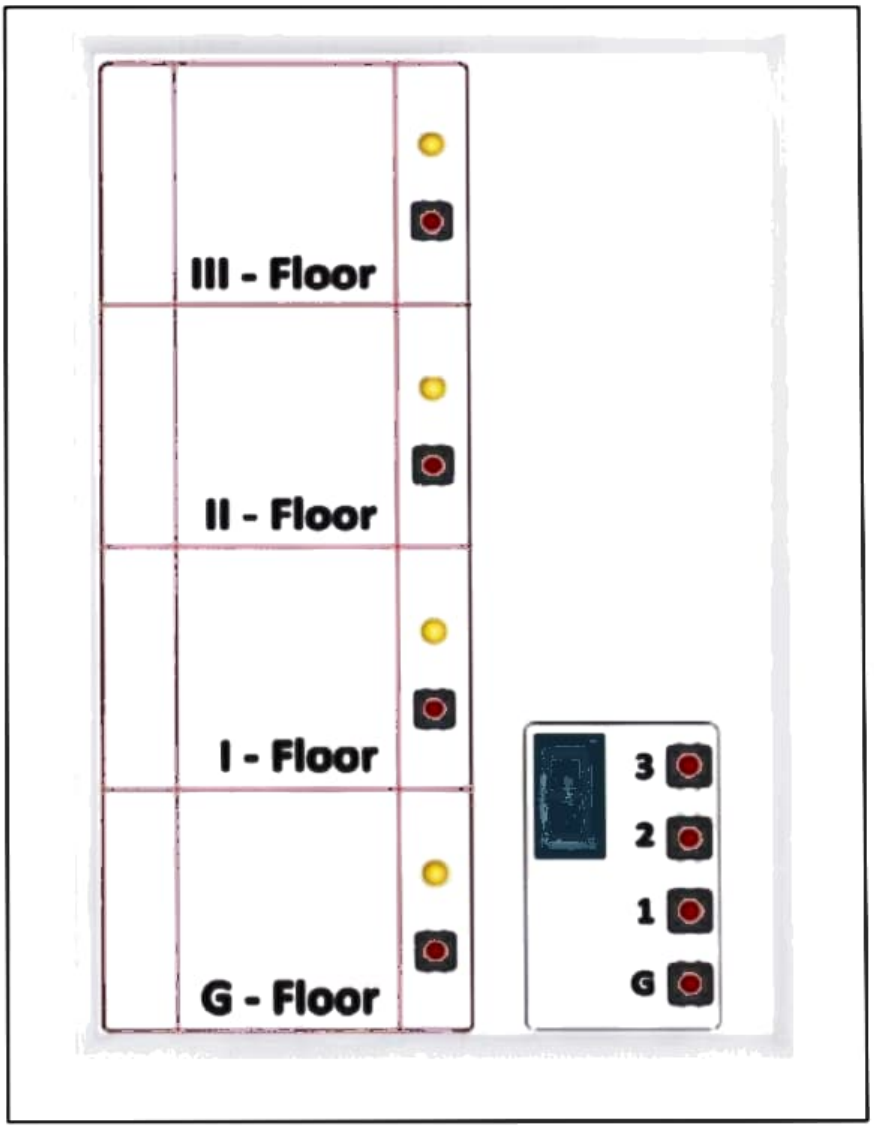
Aim - write an application using Raspberry pi to control the operation of hardware simulated lift elevator.

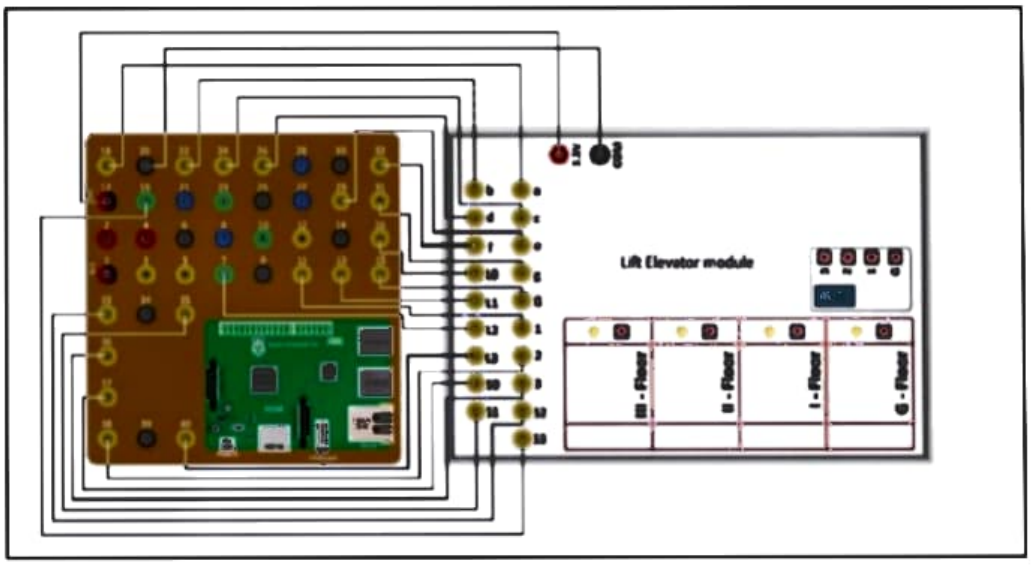
Aim / objectives - ① To understand the working principal of lift elevator.
② To interface the lift elevator module with Raspberry pi model
③ To program the Raspberry pi model to control operation of lift-elevator module

Software -
Raspbian OS (IDLE)

Hardware modules -
1. Raspberry pi Board module
2. Push buttons (qty. 8)
3. Raspbian os (IDLE).
4. leds (qty. 4)
5. Monitor.

Theory -
lift elevator module has two parts:
1. moving part inside the lift and.
2. Stationary part outside the lift at each floor to call lift.
3. In this simulation module, we have considered four floors of building.
4. The moving part also contains a Seven Segment display to indicate the current floor number when lift is moving.


III - Floor
II - Floor
I - Floor
G - Floor
3
2
1
G


Lift Elevator module
III - Floor
II - Floor
I - Floor
G - Floor

5. By pressing one of these buttons, the user indicates destination floor.
6. At each floor, the stationary part contains a buttons for calling the lift.
7. In real life as soon as the entering users get finished the lift door is closed and the lift starts moving toward destination.
8. In our module, this situation is indicated by "LED OFF" status. So the LED OFF status. indicates that the lift is moving towards the destination floor.

Safety precautions -
1. First make all the connections as per steps given below
2. Power supply.

Steps for assembling circuit:
1. Connect all pins of lift elevator module to pins of Raspberry pi module as shown in fig.

Procedure - 2. write program as per algorithm given
3. Save program.
4. Run code using Run module.

Algorithm -
1. Import GPIO and time libraries
2. Set GPIO mode as per board
3. Declare 4 push button pins of stationary part.
4. Declare 4 LED pins at each floor for detection of door close & open.

5. Sets the push button pins as input.
6. Set the seven segment display pins display in variables.
7. In loop if Button one is pressed then lift at floor 1 and LED at floor 1 get ON for 5 second gets off.
8. The seven segment displays the floor number of destination.

Conclusion -

Thus we have studied working principle of lift elevator to interface the elevator module with Raspberry pi & code of these.

```
#Interfacing Lift Elevator module with Raspberry-Pi-3
import RPi.GPIO as GPIO
import time

FloorButton0 = 37
FloorButton1 = 35
FloorButton2 = 33
FloorButton3 = 19

LiftButton0 = 15
LiftButton1 = 11
LiftButton2 = 38
LiftButton3 = 36

#GPIO setup for the LEDs
FloorLed0 = 16
FloorLed1 = 13
FloorLed2 = 7
FloorLed3= 40

#GPIO setup for the Seven Segment Display
segAPin=18
segBPin=22
segCPin=24
segDPin=26
segEPin=29
segFPin=32
segGPin=31

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

GPIO.setup(FloorButton0, GPIO.IN)
GPIO.setup(FloorButton1, GPIO.IN)
GPIO.setup(FloorButton2, GPIO.IN)
GPIO.setup(FloorButton3, GPIO.IN)

GPIO.setup(LiftButton0, GPIO.IN)
GPIO.setup(LiftButton1, GPIO.IN)
GPIO.setup(LiftButton2, GPIO.IN)
GPIO.setup(LiftButton3, GPIO.IN)

GPIO.setup(FloorLed0, GPIO.OUT) #Floor 1
```

```
GPIO.setup(FloorLed1, GPIO.OUT) #Floor 2

GPIO.setup(FloorLed2, GPIO.OUT) #Floor 3

GPIO.setup(FloorLed3, GPIO.OUT) #Floor 4

GPIO.setup(segAPin, GPIO.OUT)


GPIO.setup(segBPin, GPIO.OUT)
GPIO.setup(segCPin, GPIO.OUT)
GPIO.setup(segDPin, GPIO.OUT)
GPIO.setup(segEPin, GPIO.OUT)
GPIO.setup(segFPin, GPIO.OUT)
GPIO.setup(segGPin, GPIO.OUT)

digitclr=[0,0,0,0,0,0,0]
digit0=[1,1,1,1,1,1,0]
digit1=[0,1,1,0,0,0,0]
digit2=[1,1,0,1,1,0,1]
digit3=[1,1,1,1,0,0,1]

gpin=[18,22,24,26,29,32,31]
#routine to clear and then write to display
def digdisp(digit):
for x in range (0,7):
GPIO.output(gpin[x], digitclr[x])

for x in range (0,7):
GPIO.output(gpin[x], digit[x])

while True:

if (GPIO.input(FloorButton0)== True) :
GPIO.output(FloorLed0,1)
print"0"

digdisp(digit0)
time.sleep(1)
GPIO.output(FloorLed0,0)
time.sleep(3)

while True:

if(GPIO.input(LiftButton1)== True):
print'floor ONE'
```

```
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(2)
break

elif (GPIO.input(LiftButton2)== True):
print'floor TWO'
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(2)
break

elif (GPIO.input(LiftButton3)== True):
print'floor THREE'
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(1)
digdisp(digit3)
time.sleep(2)
break

elif (GPIO.input(FloorButton1) == True):

GPIO.output(FloorLed1, 1)
print"1"
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
time.sleep(4)

GPIO.output(FloorLed1, 0)

while True:

if(GPIO.input(LiftButton0)== True):
print 'floor ZERO'
digdisp(digit0)
time.sleep(2)
```

```
break

elif (GPIO.input(LiftButton2)== True):
print'floor TWO'
digdisp(digit2)
time.sleep(2)
break

elif (GPIO.input(LiftButton3)== True):
print'floor THREE'
digdisp(digit2)
time.sleep(1)
digdisp(digit3)
time.sleep(2)
break

elif (GPIO.input(FloorButton2) == True):

GPIO.output(FloorLed2, 1)

print"2"

digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(1)
time.sleep(5)
GPIO.output(FloorLed2, 0)

while True:

if(GPIO.input(LiftButton0)== True):
print 'floor ZERO'
digdisp(digit1)
time.sleep(1)
digdisp(digit0)
time.sleep(2)
break

elif (GPIO.input(LiftButton1)== True):
print 'floor ONE'
digdisp(digit1)
time.sleep(2)
break
```

```
elif (GPIO.input(LiftButton3)== True):
print'floor THREE'
digdisp(digit3)
time.sleep(2)
break

elif (GPIO.input(FloorButton3) == True):

GPIO.output(FloorLed3, 1)

print"3"
digdisp(digit0)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit2)
time.sleep(1)
digdisp(digit3)
time.sleep(6)

GPIO.output(FloorLed3, 0)

while True:

if (GPIO.input(LiftButton0)== True):
print 'floor ZERO'
digdisp(digit2)
time.sleep(1)
digdisp(digit1)
time.sleep(1)
digdisp(digit0)
time.sleep(2)
break

elif (GPIO.input(LiftButton1)== True):
print 'Floor ONE'
digdisp(digit2)
time.sleep(1)
digdisp(digit1)
time.sleep(2)
break

elif (GPIO.input(LiftButton2)== True):
print'floor TWO'
digdisp(digit2)
time.sleep(2)
```

```
break

else:
####    time.sleep(3)
digdisp(digit0)
GPIO.output(FloorLed0, 0)
GPIO.output(FloorLed1, 0)
GPIO.output(FloorLed2, 0)
GPIO.output(FloorLed3, 0)

else:
####    time.sleep(3)
digdisp(digit0)
GPIO.output(FloorLed1, 0)
GPIO.output(FloorLed2, 0)
GPIO.output(FloorLed3, 0)
GPIO.output(FloorLed0, 0)
```

Name - Jayesh Devre
Roll No: 36-TE-A

Camlin Page
Date / /

Experiment No. 10

Aim - Create a small dashboard application to be deployed on cloud. Different publisher devices can publish their information and interested application can subscribe.

Theory -

IOT Platforms -

The IOT platforms are suites of components those help to setup & manage the internet connected device A person can remotely collect data, monitor & manage all internet connected devices from sigle system.

IOT cloud platforms -

1. Kaa IOT platform
2. Sitewhere: open platform for IOT
3. Thingspeak: An open IOT platform with MATLAB analytics.
4. Device hive - IOT made easy
5. Zetta: API - first internet of things platform.

Kaa - features -

1. Manage an unlimited number of connected devices.
2. Setup cross device interoperatibility
3. Perform A/B serive testing
4. Perform real-time device monitoring
5. Collect & analyze sensor data.
6. Analyze user behaviour deliver target notifications
7. Create cloud services for smart products.

Camlin Page
Date / /

Sitewhere features-
1. Run any number of IoT applications on a sigle site where instance.
2. Spring delievers the core configuration Network.
3. Connect devices with MQTT, AMQP, STOMP & other protocols.
4. Add devices through self registration, REST ~~devices~~ Services, or in batches.
5. Default database storage is Mongodb
6. Eclipse Californium for CoAP messaging.
7. Influx DB for event data storage.
8. HBase for non-relational datastore.

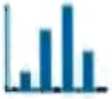
DeviceHive - features-
1. Directly integrate with Alexa
2. Visulization dashboard of your device.
3. Customize DeviceHive behaviour by running your custom javascript code.
4. Connect any device via REST API, Websockets or MQT
5. It comes with Apache Spark & spark streaming support.

Thingspeak - features-
1. Collect data in private channels
2. Share data with public channels
3. RESTful and MQTT APIS
4. MATLAB analytics & visulization
5. Alerts
6. Event Scheduling
7. App integrations.
8. worldwide community

Analytics

MATLAB Analysis

Explore and transform data.



MATLAB Visualizations

Visualize data in MATLAB plots.

Plugins

Display data in gauges, charts, or custom plugins.

Actions

ThingTweet

Connect a device to Twitter® and send alerts.

TweetControl

Listen to the Twitterverse and react in real time.

TimeControl

Automatically perform actions at predetermined times with ThingSpeak apps.

React

React when channel data meets certain conditions.

TalkBack

Queue up commands for your device.

ThingHTTP

Simplify device communication with web services and APIs.

Zetta features-

1. Built around Node.js, REST, Websockets & flow based "reactive programming".
2. supports wide range of hacker boards
3. Zetta allows you to assemble Smartphone apps device apps & cloud apps.

Conclusion -

Thus we have designed small application using Thingspeak.

Name - Jayesh Devre
Roll No: 36-A-TE

Experiment No. 11

Aim- Create a simple web interface for Raspberry Pi / Beagle board to control the connected LEDs remotely through interface.

Theory:

WiringPi -

WiringPi is a Pin based GPIO access library written in C for the BCM2835 used in the Raspberry Pi. It is released under GNU LGPLv3 licence and is usable from C, C++ & RTB as well as many other languages with suitable wrappers.

Install WiringPi -

WiringPi is not included with Raspbian So to begin, you'll need to download & install it

GPIO Command line utility -

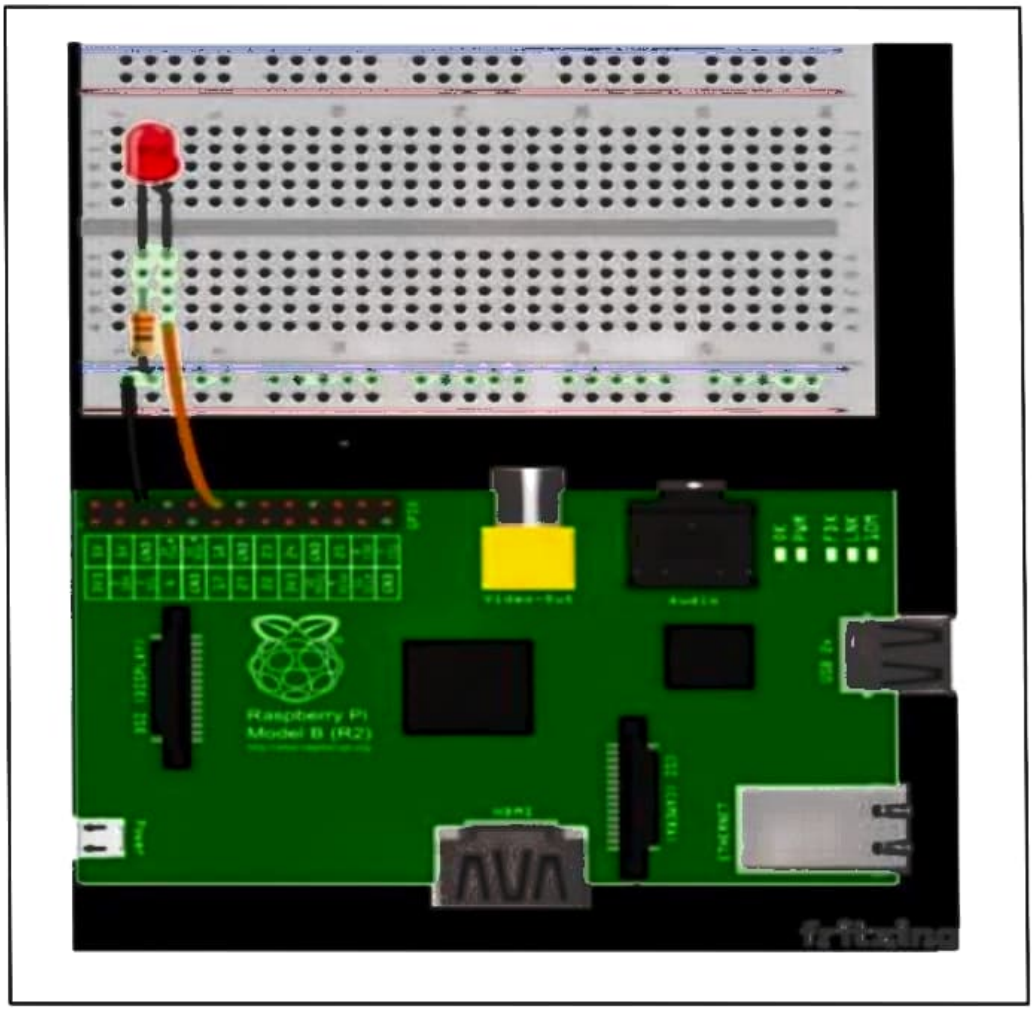
Task: Connect the LED GND to short pin GPIO18 to long pin.

GPIO Command line utility.

1. Glow the LED by value.
   gpio write 11
2. Off the LED by
   gpio write 10

Web interface to LED:

1. Create the frontage page using HTML which contains two buttons to put the LED


Raspberry Pi
Model B (R2)
fritzing

Camlin Page
Date / /

in ON or OFF state.

2. Control the data input from buttons using PHP page.

Conclusion-

Thus, we have created simple web interface for Rasphberry-Pi / Beagle board to control the Connected LED's remotely through Interface.

Name - Jayesh Derre

Roll No: 36-TE-A

Camlin Page
Date / /

Experiment No. 12

Aim - Develop a Real time application like Smart home with following Requirements: when user enters into house the required appliances like fan, light should be switched on. Appliances should also get controlled remotely by a suitable web interface. The objective of this application is student should construct Complete Smart-application in group

Theory:

Basics - Send emails using Python -

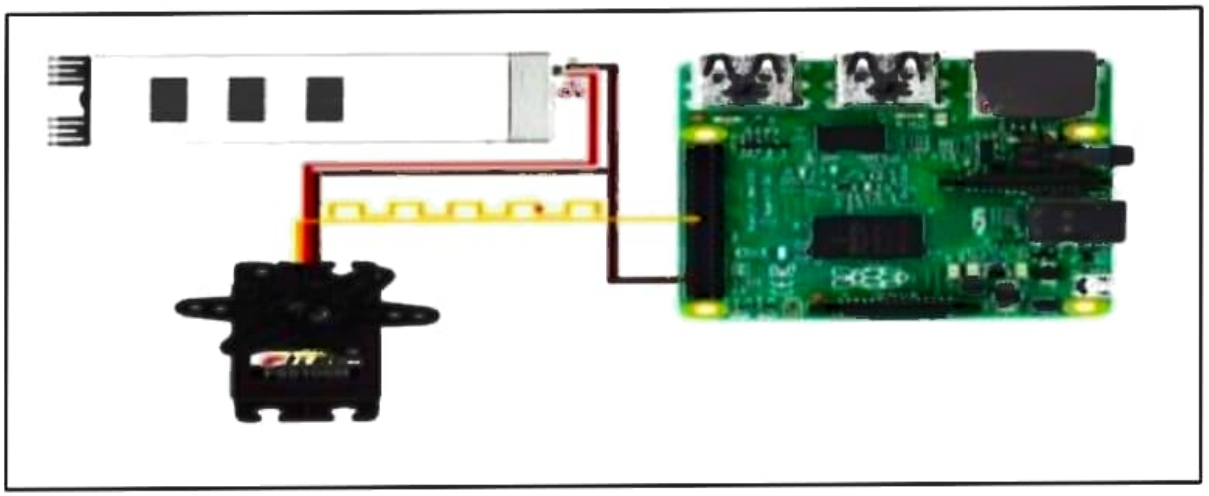
1. The Smtplib module of python is basically all you need to send simple emails, without any subject line.
2. But for real emails, you do need a subject line & lots of information.

How to send emails?

1. Setup the SMTP Server and log into your account
2. Add your message body.
3. Send the message using the SMTP server object.

The Smtplib -

1. The Smtplib module defines an SMTP client session object that can be used to send mail to any internet machine with an SMTP.
2. SMTP stands for simple mail Transfer protocol The SMTPlib modules is useful for communicating



with mail servers to send mail.
3. Sending mail is done with python's Smtplib using an SMTP server.
4. Actual ~~server~~ usage varies depending on complexity of email & settings of email server, the instructions here are based on sending email through Gmail.

Steps -
- create the lock/ unlock application to control the servo motor lock. Change its owner and group
- write application to read the image & send it as email attachment to user.
- write application using HTML-PHP to control the servo motor lock.

Conclusion -
Thus we have developed short ~~short~~ application for smart home system