RASPBERRY PI BASED GPS TRACKING SYSTEM AND FACE RECOGNITION SYSTEM

B.Tech Major Project Thesis

Submitted in partial fulfillment for the award of the Degree of Bachelor of Technology in Electrical and Electronics Engineering

By

V.RUTHVIK	**(B100469EE)**
M.NAVEEN KUMAR	**(B100647EE)**
S.GOPAL	**(B090814EE)**
NEERAJ KARNANI	**(B080674EE)**

Under the guidance of
Dr G.JAGADANAND

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY CALICUT
NIT Campus P.O., Calicut - 673601, India

CERTIFICATE

This is to certify that the report entitled "GPS TRACKING SYSTEM AND ATTENDANCE REGISTERING SYSTEM USING RASPBERRY-PI" is a bona fide record of the project was done by ***V.RUTHVIK*** *(B100469EE),* ***M.NAVEEN KUMAR*** *(B100647EE),* ***S.GOPAL*** *(B090814EE) and* ***NEERAJ KARNANI*** *(B080674EE) under my supervision and guidance, in partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Electrical & Electronic Engineering from National Institute of Technology Calicut for the year 2014.*

Dr. G.JAGADANAND(Guide)

Assistant Professor

Dept. of Electrical Engineering

Dr. SUSY THOMAS

Professor & Head

Dept. of Electrical Engineering

Place:

Date:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, **Dr. G.Jagadanand**, Assistant Professor, Department of Electrical Engineering, National Institute of Technology Calicut, for allowing us to undertake this project and for his valuable suggestions, guidance and support during the course of work.

We also want to thank **Dr.Elizabeth P Cherian**, Associate Professor, Coordinator (main projects- EED) and **Dr. Susy Thomas**, HOD-EED for giving us a chance to carry out this project.

V.Ruthvik (B100469EE)

M.Naveen Kumar (B100647EE)

S.Gopal (B090814EE)

Neeraj Karnani (B080674EE)

ABSTRACT

The Raspberry Pi is a credit-card-sized single-board ARM Processor based computer. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, which can be used for interfacing external peripherals. Raspberry Pi based GPS tracking and face recognition system deals with interfacing various peripherals like GPS module, compass, DC motors, Raspberry Pi camera. A GPS Module will be interfaced with Raspberry Pi to make a GPS tracking system. This system can be used to know location of any vehicle or person across the world. Compass and DC motors will be interfaced to move the robot to desired location. A Raspberry pi camera will be interfaced to the Raspberry pi to make a facial recognition system using eigenfaces algorithm.

CONTENTS

List of Figures

List of Abbreviations

UART- Universal Asynchronous Receiver/Transmitter

GPIO-General-purpose input/output

LED- Light-emitting diode

OS-Operating system

GPS -Global Positioning System

OpenCV- Open Source Computer Vision

NMEA- National Marine Electronics Association

I^2C- Inter-Integrated Circuit

Chapter 1

RASPBERRY PI

1.1 Introduction

The Raspberry Pi is a credit-card-sized single-board ARM Processor based computer. The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up to 1 GHz, without affecting the warranty), VideoCore IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

The Raspberry Pi Foundation provides Debian and Arch Linux ARM distributions for download. Python as the main programming language with support for C and Perl. In this project Raspberry Pi Model B is used and all codes will be written in C/Python.

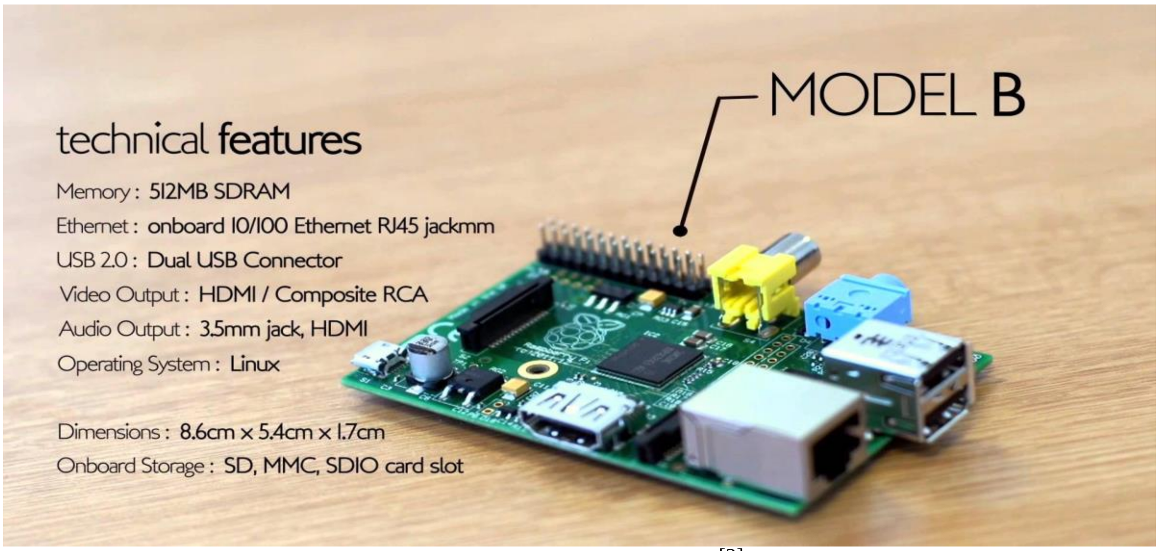

MODEL B
technical features
Memory : 512MB SDRAM
Ethernet : onboard 10/100 Ethernet RJ45 jackmm
USB 2.0 : Dual USB Connector
Video Output : HDMI / Composite RCA
Audio Output : 3.5mm jack, HDMI
Operating System : Linux
Dimensions : 8.6cm x 5.4cm x 1.7cm
Onboard Storage : SD, MMC, SDIO card slot

Figure-1.1 Raspberry pi[2]

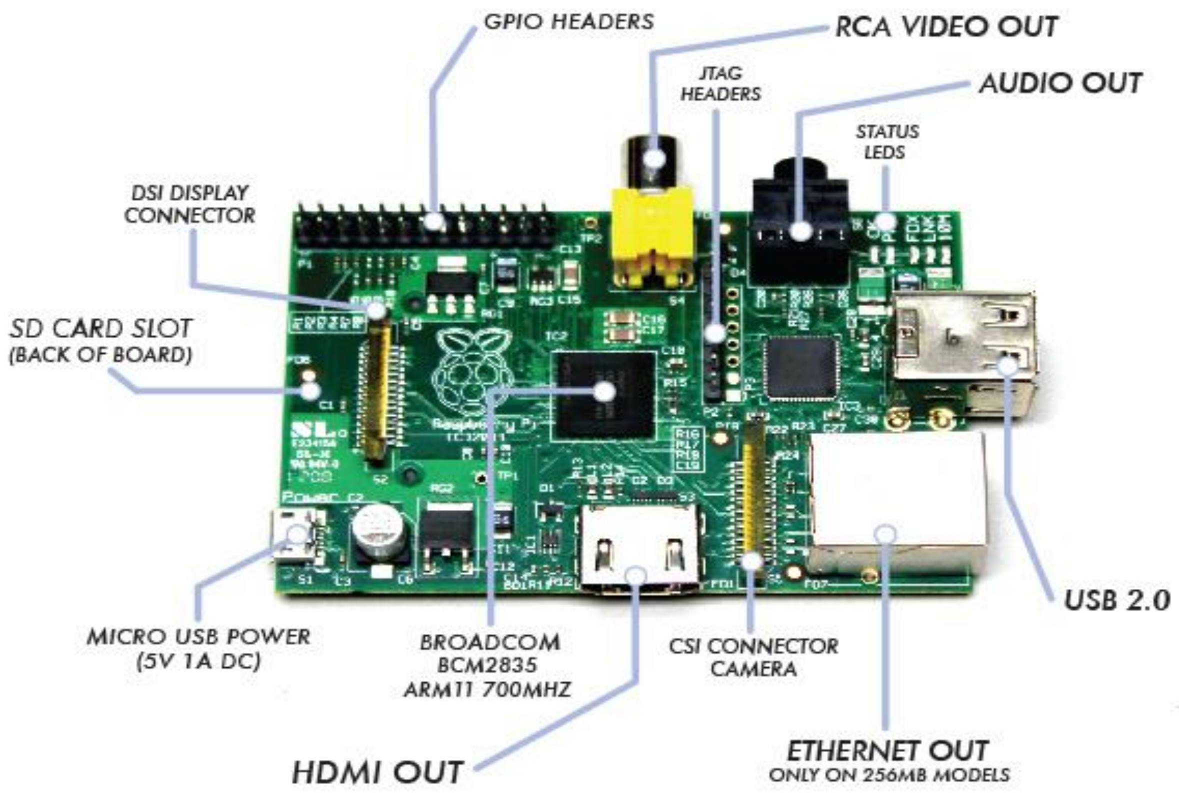
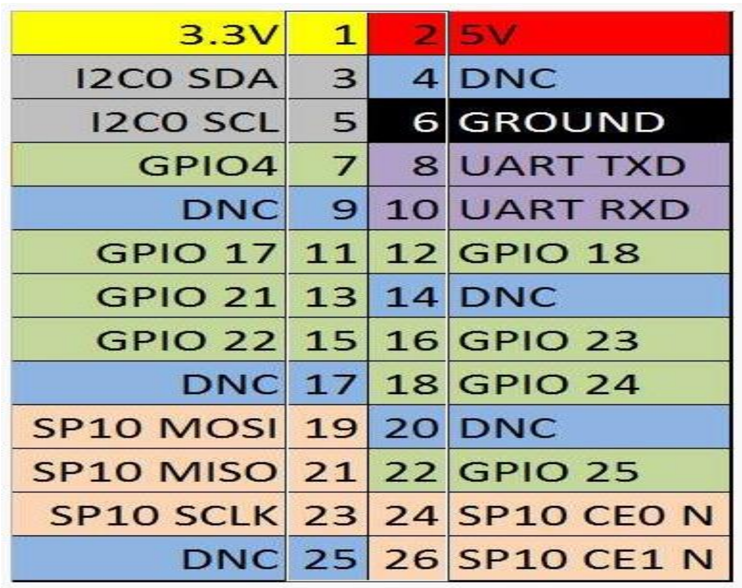

GPIO HEADERS
RCA VIDEO OUT
JTAG HEADERS
AUDIO OUT
STATUS LEDS
DSI DISPLAY CONNECTOR
SD CARD SLOT (BACK OF BOARD)
USB 2.0
MICRO USB POWER (5V 1A DC)
BROADCOM BCM2835 ARM11 700MHZ
CSI CONNECTOR CAMERA
ETHERNET OUT ONLY ON 256MB MODELS
HDMI OUT

Figure-1.2- Internal parts of Raspberry Pi



3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Figure-1.3- Raspberry Pi GPIO Diagram

1.2 Installing OS in Raspberry Pi

1.2.1 Steps to install OS in Raspberry pi

Things needed:

- Memory card(at least 4GB)

Figure-1.4 SD Memory Card

- Any of the below raspberry pi compatible OS(all are open source):

  (www.raspberrypi.org/downloads)

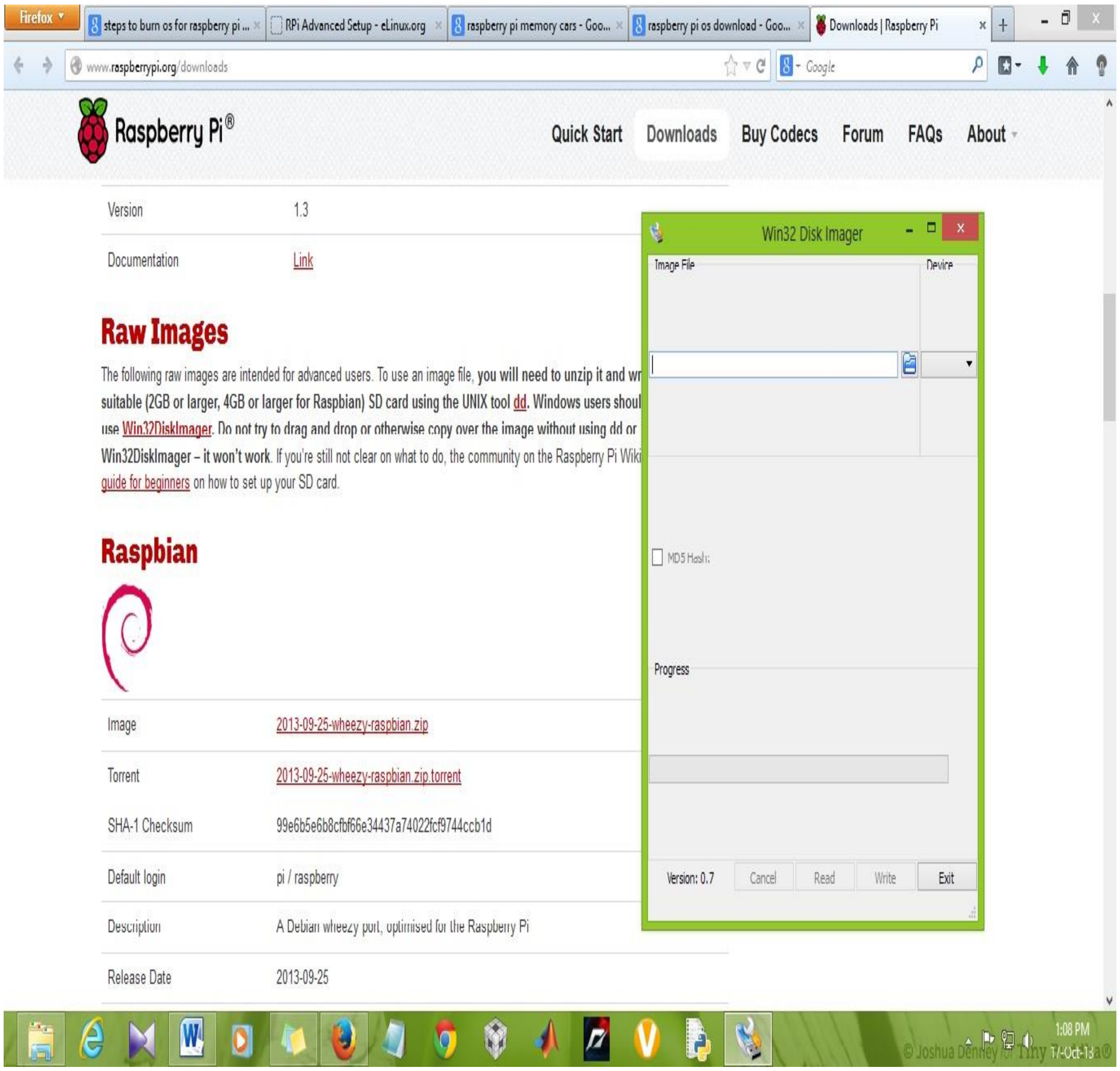
  (a) Wheezy Raspian(recommended for beginners)
  (b) NOOBS
  (c) Pipa OS
  (d) RISC OS
  (e) Arch Linux
- Software needed to install the OS (for windows version)

  (a) Win32DiskImager (available for free download)

1.2.2 Steps for setting up Raspberry-Pi

After downloading the zip file of OS extract it and then burn it into a Memory card.

A screen shot of web page for downloads and win32disk imager software is shown.


Firefox
steps to burn os for raspberry pi ...
RPi Advanced Setup - eLinux.org
raspberry pi memory cars - Goo...
raspberry pi os download - Goo...
Downloads | Raspberry Pi
www.raspberrypi.org/downloads
Google
Raspberry Pi®
Quick Start
Downloads
Buy Codecs
Forum
FAQs
About
Win32 Disk Imager
Image File
Device
MD5 Hash:
Progress
Version: 0.7
Cancel
Read
Write
Exit
1:08 PM
© Joshua Denney for Tiny Tutorial®

Version	1.3
Documentation	Link

Raw Images

The following raw images are intended for advanced users. To use an image file, **you will need to unzip it and wr** **suitable (2GB or larger, 4GB or larger for Raspbian) SD card using the UNIX tool dd. Windows users shoul** **use Win32DiskImager. Do not try to drag and drop or otherwise copy over the image without using dd or** **Win32DiskImager – it won't work**. If you're still not clear on what to do, the community on the Raspberry Pi Wiki guide for beginners on how to set up your SD card.

Raspbian

Image	2013-09-25-wheezy-raspbian.zip
Torrent	2013-09-25-wheezy-raspbian.zip.torrent
SHA-1 Checksum	99e6b5e6b8cfbf66e34437a74022fcf9744ccb1d
Default login	pi / raspberry
Description	A Debian wheezy port, optimised for the Raspberry Pi
Release Date	2013-09-25

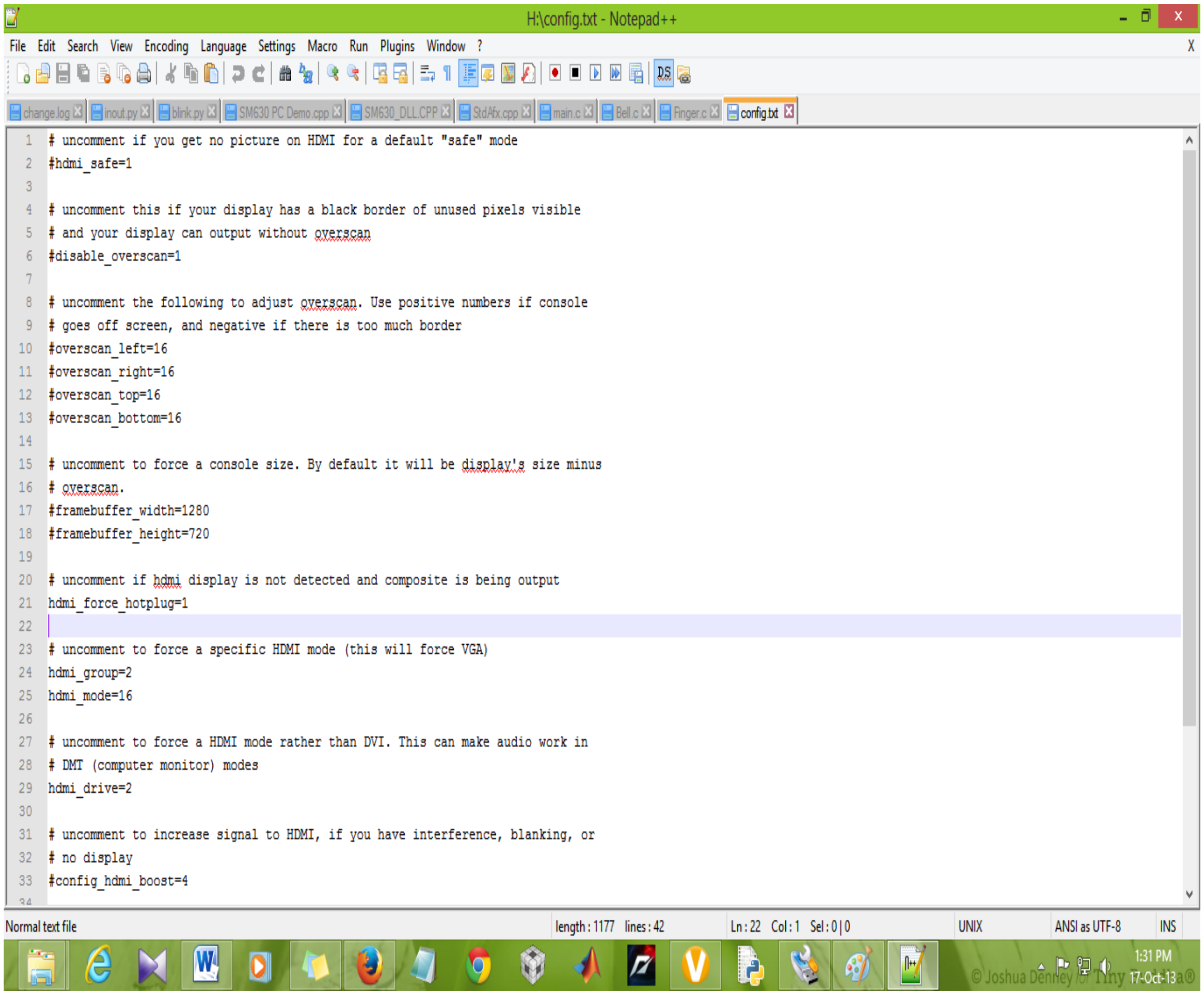
Figure-1.5 Screenshot of website for Raspberry-Pi OS download

(1) If we use a monitor with HDMI input this step can be skipped, if we use HDMI to VGA converter we have to go through this step.

Open the config file in Memory card where the OS was installed with any text editor and make the changes suggested here.

```
hdmi_force_hotplug=1
hdmi_group=2
hdmi_mode=16
hdmi_drive=2
```

A screenshot is shown below.



```
# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280
#framebuffer_height=720

# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (this will force VGA)
hdmi_group=2
hdmi_mode=16

# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
hdmi_drive=2

# uncomment to increase signal to HDMI, if you have interference, blanking, or
# no display
#config_hdmi_boost=4
```

Figure-1.6 Editing config for enabling HDMI to VGA converter

1) Next plugin the SD card into Raspberry Pi and power up Raspberry Pi (note that minimum current rating for power supply is 700mA). After booting up the Raspberry Pi the screen should look like in figure-1.7. For changing any settings use the command "*sudo raspi-config*".

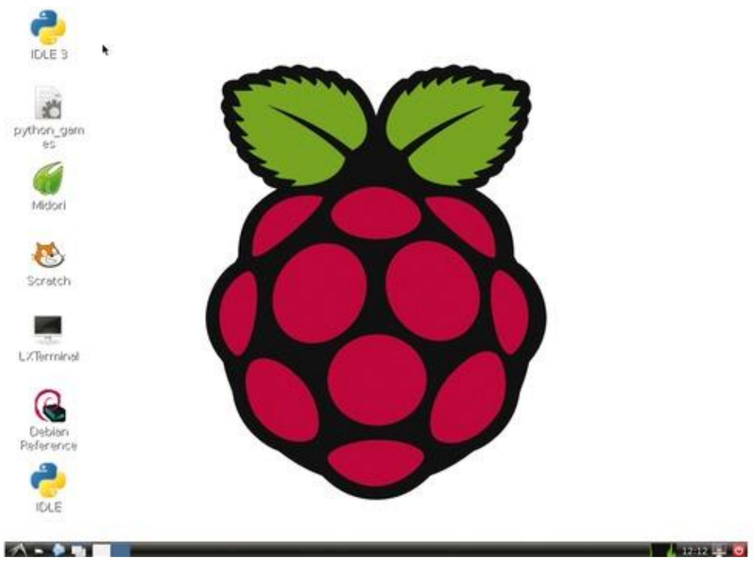
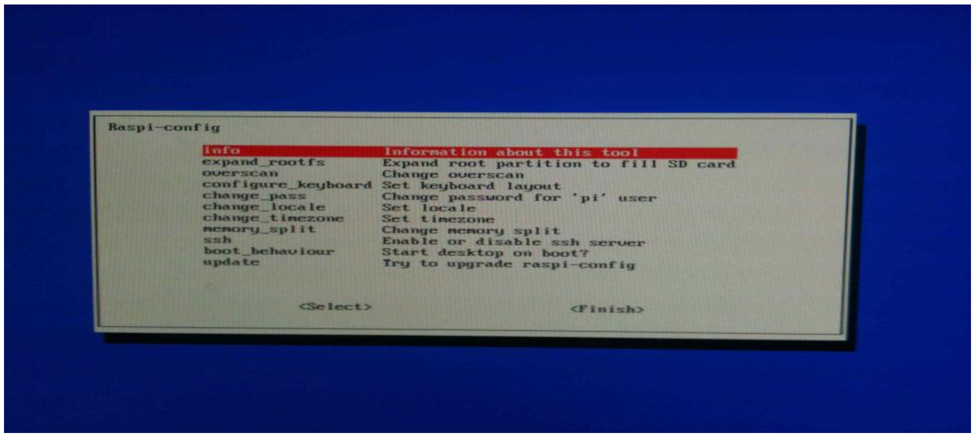


Figure-1.7 Screenshot of Raspberry Pi with Raspbian OS in it



```
Raspi-config

        info                 Information about this tool
        expand_rootfs        Expand root partition to fill SD card
        overscan             Change overscan
        configure_keyboard   Set keyboard layout
        change_pass          Change password for 'pi' user
        change_locale        Set locale
        change_timezone      Set timezone
        memory_split         Change memory split
        ssh                  Enable or disable ssh server
        boot_behaviour       Start desktop on boot?
        update               Try to upgrade raspi-config

             <Select>                      <Finish>
```

Figure-1.8 Screenshot after Raspberry-Pi has been powered up

(2) Of the above settings configure_keyboard and boot_behaviour are of main Flconcern, change the keyboard layout to that of yours and change boot_behaviour to start desktop on boot.
(3) Next step is connecting the PI to internet.

Commands needed to reconfigure the IP settings

- cat /etc/network/interfaces
  when this command is executed on LXterminal, text below is shown
  iface lo inet loopback
  iface eth0 inet dhcp

This file currently reads 'iface' meaning interface followed by 'eth0' which is you're the built in network connection on your Raspberry Pi. Next is 'dhcp' which means that we want to use DHCP to obtain an IP address. The above file is edited so that a static connection is established.

```
sudo nano /etc/network/interfaces
```

This will allow you to edit the file using nano. Change the line that reads

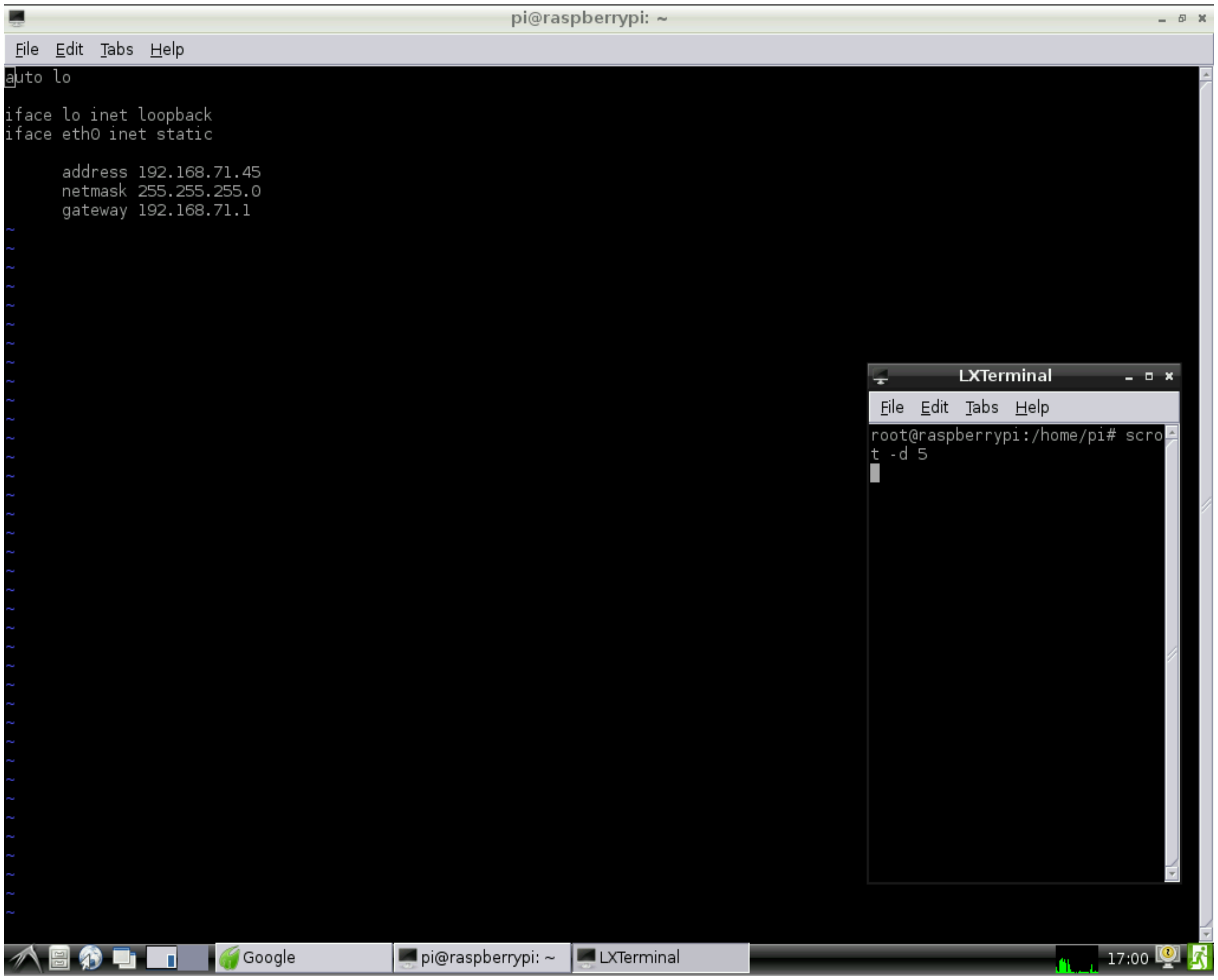
“iface eth0 inet dhcp to iface eth0 inet static.

Change the network settings as per your requirements.

- address 192.168.100.1
- netmask 255.255.255.0
- network 192.168.100.0
- broadcast 192.168.100.255
- gateway 192.168.100.254

After editing reboot Pi using command sudo reboot.

After above settings network settings should look like figure-1.9.



```
auto lo

iface lo inet loopback
iface eth0 inet static

      address 192.168.71.45
      netmask 255.255.255.0
      gateway 192.168.71.1
```

```
root@raspberrypi:/home/pi# scrot -d 5
```

Figure-1.9 Network settings of Raspberry-Pi

1.3 LED blinking

An LED has been connected to GPIO pin 7(as shown in figure-1.10) and the Raspberry pi has been programmed to blink the LED. Open IDLE(Python 2.7 Interpreter) to write code.Python code for this program has been provided in Appendix-I.

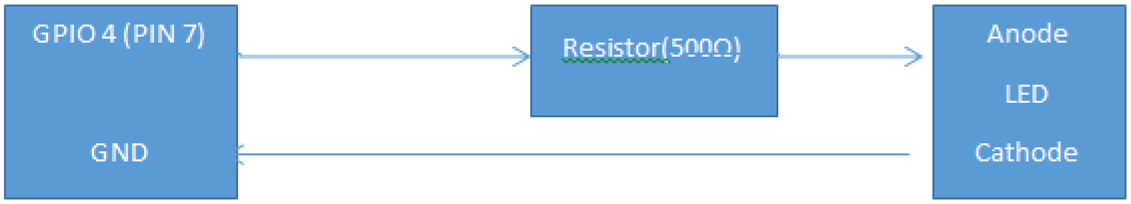
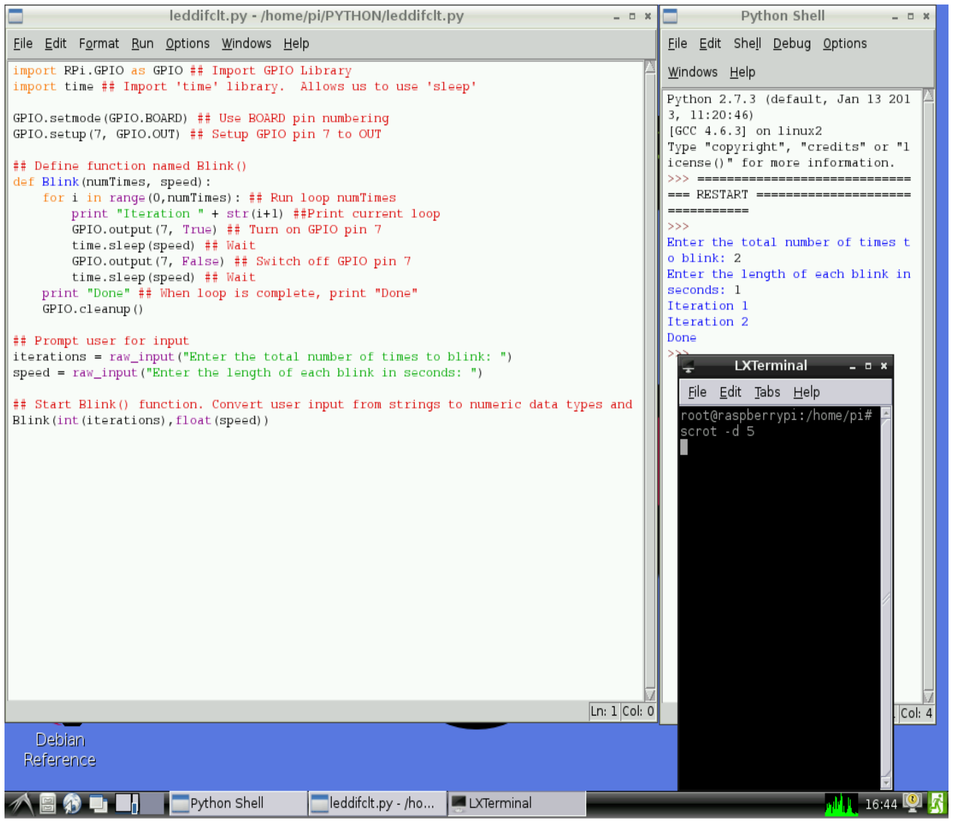


Figure-1.10 Connection diagram for LED blinking


leddifclt.py - /home/pi/PYTHON/leddifclt.py
File Edit Format Run Options Windows Help
Python Shell
File Edit Shell Debug Options Windows Help
LXTerminal
File Edit Tabs Help
Ln: 1 Col: 0
Col: 4
Debian Reference
Python Shell
leddifclt.py - /ho...
LXTerminal
16:44

```python
import RPi.GPIO as GPIO ## Import GPIO Library
import time ## Import 'time' library.  Allows us to use 'sleep'

GPIO.setmode(GPIO.BOARD) ## Use BOARD pin numbering
GPIO.setup(7, GPIO.OUT) ## Setup GPIO pin 7 to OUT

## Define function named Blink()
def Blink(numTimes, speed):
    for i in range(0,numTimes): ## Run loop numTimes
        print "Iteration " + str(i+1) ##Print current loop
        GPIO.output(7, True) ## Turn on GPIO pin 7
        time.sleep(speed) ## Wait
        GPIO.output(7, False) ## Switch off GPIO pin 7
        time.sleep(speed) ## Wait
    print "Done" ## When loop is complete, print "Done"
    GPIO.cleanup()

## Prompt user for input
iterations = raw_input("Enter the total number of times to blink: ")
speed = raw_input("Enter the length of each blink in seconds: ")

## Start Blink() function. Convert user input from strings to numeric data types and
Blink(int(iterations),float(speed))
```

```
Python 2.7.3 (default, Jan 13 201
3, 11:20:46)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "l
icense()" for more information.
>>> ==============================
=== RESTART ======================
==========
>>>
Enter the total number of times t
o blink: 2
Enter the length of each blink in
seconds: 1
Iteration 1
Iteration 2
Done
>>>
```

```
root@raspberrypi:/home/pi#
scrot -d 5
```

Figure-1.11 Programming Raspberry pi for LED blinking.

Run the code to know if there are any errors. Open the terminal and change present working directory to that of code’s by using command “*cd X*”, where X is the name of the directory. Run the command *“sudo python filename.py”*.

CHAPTER 2
PERIPHERAL INTERFACING

2.1 GPS module

2.1.1 Introduction

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.
A GPS receiver calculates its position by precisely timing the signals sent by GPS satellites high above the Earth. Each satellite continually transmits messages that include

1. The time the message was transmitted
2. Satellite position at time of message transmission

The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the speed of light.
The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver. UART protocol will be used to connect GPS module to the Raspberry Pi.

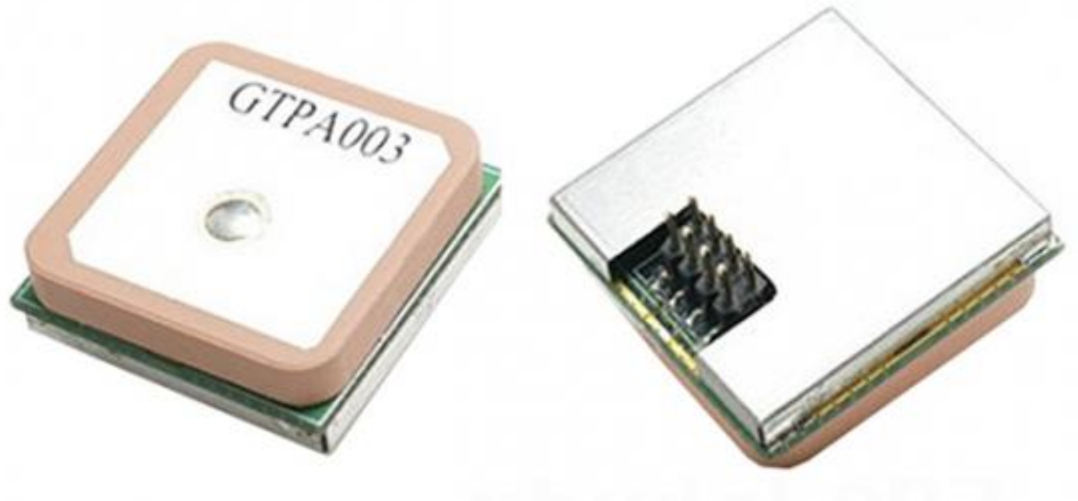


Figure-2.1 GPS Module[10]

2.1.2 UART

A universal asynchronous receiver/transmitter, abbreviated as UART, is a piece of computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms.

A UART usually contains the following components:

- A clock generator, usually a multiple of the bit rate to allow sampling in the middle of a bit period.
- Input and output shift registers
- Transmit/receive control
- Read/write control logic
- Transmit/receive buffers (optional)
- Parallel data bus buffer (optional)
- First-in, first-out (FIFO) buffer memory (optional)

The Universal Asynchronous Receiver/Transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

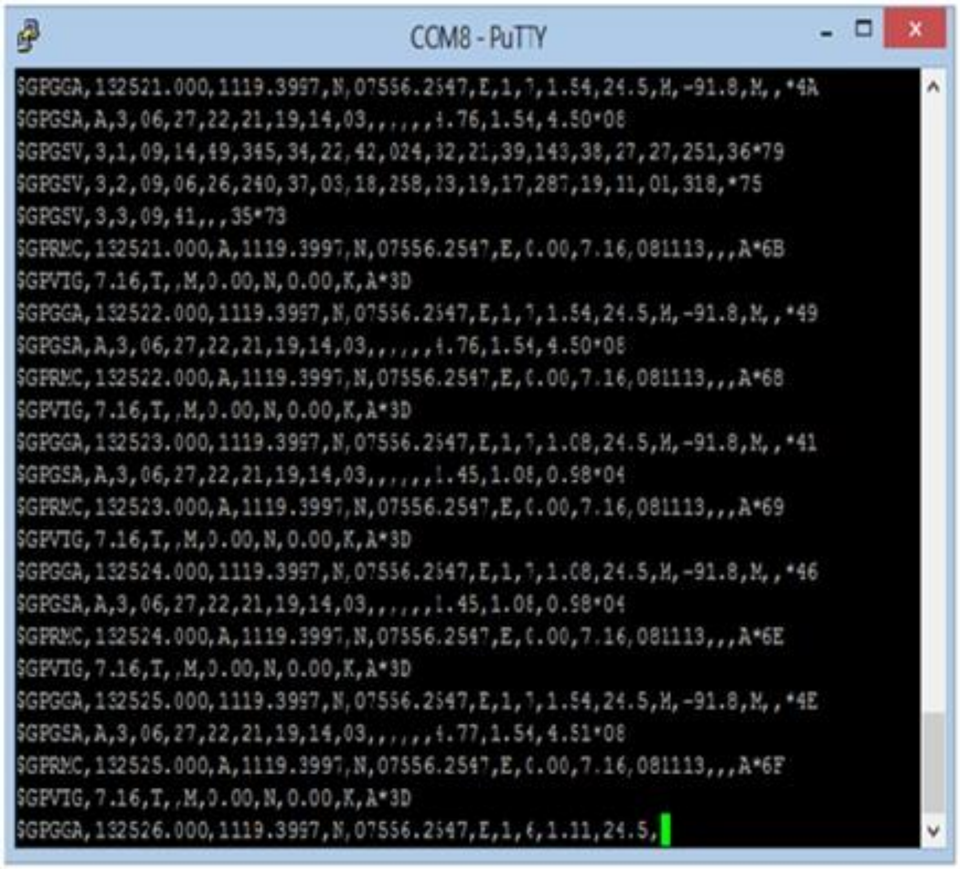
Using the Raspberry pi's serial port:

To use Raspberry pi's UART as serial communication, serial console on it has to disabled.

- Create a backup of the /boot/cmdline.txt file, which contains the kernel parameters by running the command "*sudo cp /boot/cmdline.txt /boot/cmdline_backup.txt*".
- To edit the file run "*sudo vi /boot/cmdline.txt*".
- Originally it contains "*dwc_otg.lpm_enable=0 rpitestmode=1 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait"* change it to *"dwc_otg.lpm_enable=0 rpitestmode=1 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait".*
- Run the command "*sudo vi /etc/inittab"* and comment out the following line:
  "2:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100".
- Install minicom in Raspberry pi by running the command "*sudo apt-get install minicom".* And run it by "`minicom -b x -o -D /dev/ttyAMA0". Where 'x' is the baud rate ex:9600.`

2.1.3 Testing GPS module

The GPS module has been tested using PUTTY software. PUTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet and rlogin. Figure-2.2 shows PUTTY output.



```
$GPGGA,132521.000,1119.3997,N,07556.2547,E,1,7,1.54,24.5,M,-91.8,M,,*4A
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,4.76,1.54,4.50*08
$GPGSV,3,1,09,14,49,345,34,22,42,024,32,21,39,143,38,27,27,251,36*79
$GPGSV,3,2,09,06,26,240,37,03,18,258,23,19,17,287,19,11,01,318,*75
$GPGSV,3,3,09,41,,,35*73
$GPRMC,132521.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*6B
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132522.000,1119.3997,N,07556.2547,E,1,7,1.54,24.5,M,-91.8,M,,*49
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,4.76,1.54,4.50*08
$GPRMC,132522.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*68
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132523.000,1119.3997,N,07556.2547,E,1,7,1.08,24.5,M,-91.8,M,,*41
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,1.45,1.08,0.98*04
$GPRMC,132523.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*69
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132524.000,1119.3997,N,07556.2547,E,1,7,1.08,24.5,M,-91.8,M,,*46
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,1.45,1.08,0.98*04
$GPRMC,132524.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*6E
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132525.000,1119.3997,N,07556.2547,E,1,7,1.54,24.5,M,-91.8,M,,*4E
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,4.77,1.54,4.51*08
$GPRMC,132525.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*6F
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132526.000,1119.3997,N,07556.2547,E,1,6,1.11,24.5,
```

Figure-2.2 Testing GPS module in windows

2.1.4 Install GPSD Daemon in Raspberry Pi

- Run “*sudo apt-get install gpsd gpsd-clients python-gps*” to install GPSD Daemon.
- Make the connections as shown in figure-2.3.

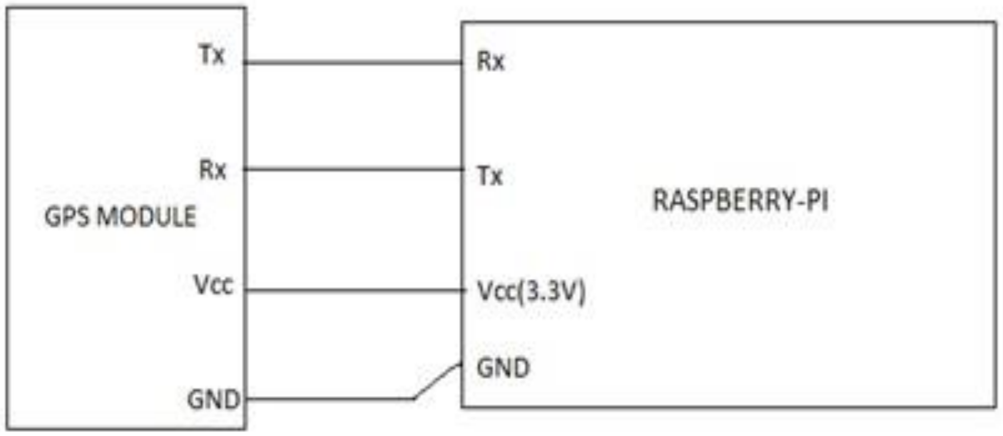
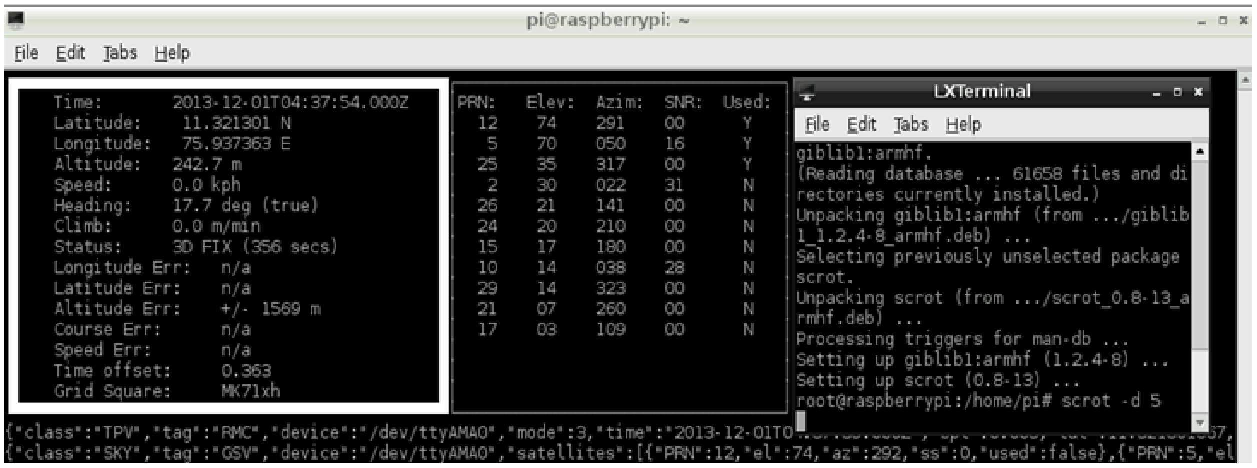


Figure-2.3 Connections between Raspberry Pi and GPS module

- Run the command "*sudo gpsd /dev/ttyAMA0 -F /var/run/gpsd.sock"* to point the gps daemon to our GPS device on the serial port to TTY adapter cable.
- After executing the command "*cgps –s*" the output should look like below:

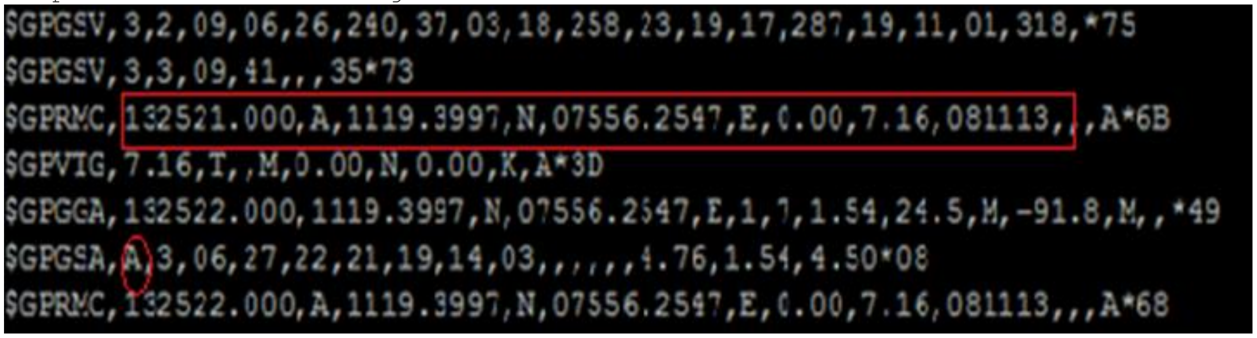


```
    Time:        2013-12-01T04:37:54.000Z      PRN:   Elev:  Azim:  SNR:  Used:
    Latitude:     11.321301 N                    12     74    291    00      Y
    Longitude:    75.937363 E                     5     70    050    16      Y
    Altitude:    242.7 m                         25     35    317    00      Y
    Speed:       0.0 kph                          2     30    022    31      N
    Heading:     17.7 deg (true)                 26     21    141    00      N
    Climb:       0.0 m/min                       24     20    210    00      N
    Status:      3D FIX (356 secs)               15     17    180    00      N
    Longitude Err:   n/a                         10     14    038    28      N
    Latitude Err:    n/a                         29     14    323    00      N
    Altitude Err:    +/- 1569 m                  21     07    260    00      N
    Course Err:      n/a                         17     03    109    00      N
    Speed Err:       n/a
    Time offset:     0.363
    Grid Square:     MK71xh
{"class":"TPV","tag":"RMC","device":"/dev/ttyAMA0","mode":3,"time":"2013-12-01T0
{"class":"SKY","tag":"GSV","device":"/dev/ttyAMA0","satellites":[{"PRN":12,"el":74,"az":292,"ss":0,"used":false},{"PRN":5,"el
```

```
giblib1:armhf.
(Reading database ... 61658 files and di
rectories currently installed.)
Unpacking giblib1:armhf (from .../giblib
1_1.2.4-8_armhf.deb) ...
Selecting previously unselected package
scrot.
Unpacking scrot (from .../scrot_0.8-13_a
rmhf.deb) ...
Processing triggers for man-db ...
Setting up giblib1:armhf (1.2.4-8) ...
Setting up scrot (0.8-13) ...
root@raspberrypi:/home/pi# scrot -d 5
```

Figure-2.4 GPS output using GPSD daemon

- For simple testing of serial port GPS 'minicom' can be used by running the command "*`minicom -b x -o -D /dev/ttyAMA0`*". `Where 'x' is the baud rate of GPS ex:9600.`
- `Output will look like figure-2.5.`



```
$GPGSV,3,2,09,06,26,240,37,03,18,258,23,19,17,287,19,11,01,318,*75
$GPGSV,3,3,09,41,,,35*73
$GPRMC,132521.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*6B
$GPVTG,7.16,T,,M,0.00,N,0.00,K,A*3D
$GPGGA,132522.000,1119.3997,N,07556.2547,E,1,7,1.54,24.5,M,-91.8,M,,*49
$GPGSA,A,3,06,27,22,21,19,14,03,,,,,,4.76,1.54,4.50*08
$GPRMC,132522.000,A,1119.3997,N,07556.2547,E,0.00,7.16,081113,,,A*68
```

Figure-2.5 Output of minicom showing coordinates of location

Highlighted letter '**A**' in the above figure indicates that the GPS signal is strong. If the signal is weak letter '**V**' will be displayed. For GPS module to operate properly clear sky should be visible it may not work properly indoors.

Highlighted rectangular box shows the coordinates of the location (NITC Chemical Engineering Department) and date recorded (08-11-2013). Complete data shown in the figure-2.5 may not be useful to have higher level access or to fetch required data a python code is written (refer appendix-II) whose output will be shown as below screenshot.

2.1.5 GPS data parsing using python:

Only the latitude and longitude are required constituents from GPS data. Both longitude and latitude can be extracted from the *$GPGGA* line. Algorithm to parse GPS data is shown in figure-2.6. Python code for extracting longitude and latitude is given in Appendix-I

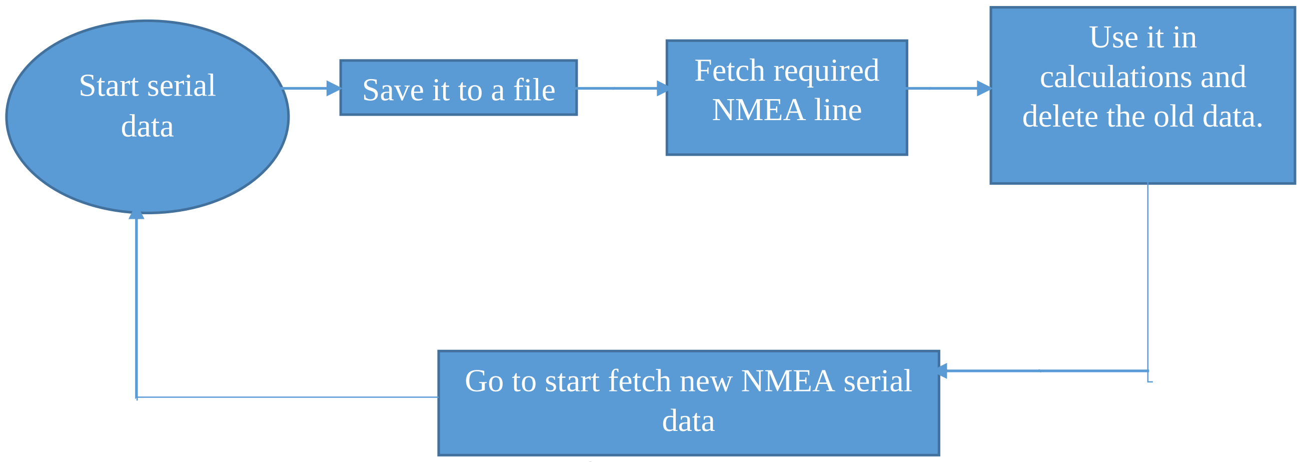

Start serial data
Save it to a file
Fetch required NMEA line
Use it in calculations and delete the old data.
Go to start fetch new NMEA serial data

Figure-2.6 Algorithm for extracting required data

2.2 Compass

2.2.1 Introduction

A compass is a navigational instrument that shows directions in a frame of reference that is stationary relative to the surface of the Earth. A traditional compass is shown in figure-2.7.

Figure-2.7 Dry magnetic portable compass

3-Axis Digital Compass IC HMC5883L[2]:

The Honeywell HMC5883L (figure-2.8) is a surface-mount, multi-chip module designed for-low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry. The HMC5883L includes state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I^{2}C serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9mm surface mount 16-pin leadless chip carrier (LCC). I^{2}C protocol is used to interface compass with Raspberry-Pi.

Figure-2.8 IC HMC5883L

2.2.2 I^{2}C Protocol

Control of this device is carried out via the I^{2}C bus. This device will be connected to this bus as a slave device under the control of a master device, such as the processor.

I^{2}C (Inter-Integrated Circuit) is a multi-master serial single-ended computer bus, invented by the Philips semiconductor division, today NXP Semiconductors, and used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other digital electronic devices. I^{2}C uses only two bidirectional open-drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V although systems with other voltages are permitted. The I^{2}C reference design has a 7-bit or a 10-bit (depending on the device used) address space. Common I^{2}C bus speeds are the 100 kbps standard mode and the 10 kbps low-speed mode, but arbitrarily low clock frequencies are also allowed.

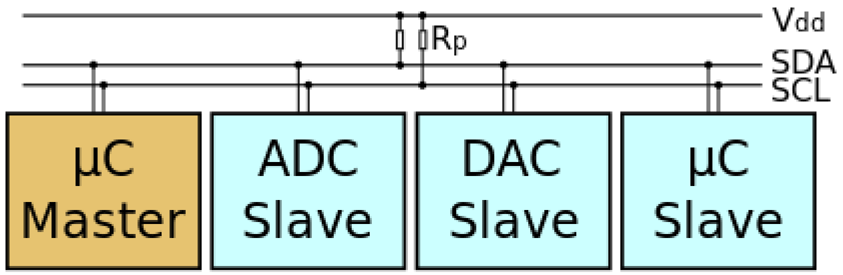


Figure-2.9 Sample schematic with one master, 3 slave nodes & pull-up resistors

The bus has two roles for nodes: master and slave

- Master node — node that generates the clock and initiates communication with slaves
- Slave node — node that receives the clock and responds when addressed by the master

The bus is a multi-master bus which means any number of master nodes can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent).

There are four potential modes of operation for a given bus device, although most devices only use a single role and its two modes:

- master transmit — master node is sending data to a slave
- master receive — master node is receiving data from a slave
- slave transmit — slave node is sending data to the master
- slave receive — slave node is receiving data from the master

Message protocols:

I²C defines basic types of messages, each of which begins with a START and ends with a STOP:

- Single message where a master writes data to a slave;
- Single message where a master reads data from a slave;
- Combined messages, where a master issues at least two reads and/or writes to one or more slaves.

Enabling the I²C Port:

The I²C ports need to be enabled in Raspbian before they can be used.

- Edit the modules file by executing "*sudo nano /etc/modules*".
- Add the following lines to the file:
  i2c-bcm2708
  i2c-dev
- Exit and save the file.
- Now edit the modules blacklist file by executing *"sudo nano /etc/modprobe.d/raspi-blacklist.conf"*
- Add a '#' character to this line so it commented out:

  "#blacklist i2c-bcm2708"

- Exit and save the file.
- Finally install the I²C utilities by executing "*sudo apt-get install python-smbus i2c-tools".*
- Enter *"sudo reboot"* to restart the pi and now the I²C pins will be available to use.
- To install smbus python I²C libraries run the command "*sudo apt-get install python-smbus".*

2.2.3 Basic device operation

Anisotropic Magneto-Resistive Sensors:

The Honeywell HMC5883L magnetoresistive sensor circuit is a trio of sensors and application specific support circuits to measure magnetic fields. With power supply applied, the sensor converts any incident magnetic field in the sensitive axis directions to a differential voltage output. The magnetoresistive sensors are made of a nickel-iron (Permalloy) thin-film and patterned as a resistive strip element. In the presence of a magnetic field, a change in the bridge resistive elements causes a corresponding change in voltage across the bridge outputs.

These resistive elements are aligned together to have a common sensitive axis (indicated by arrows in the pinout diagram) that will provide positive voltage change with magnetic fields increasing in the sensitive direction. Because the output is only proportional to the magnetic field component along its axis, additional sensor bridges are placed at orthogonal directions to permit accurate measurement of magnetic field in any orientation.

Registers:

This device is controlled and configured via a number of on-chip registers, which are described in this section. In the following descriptions, set implies a logic 1, and reset or clear implies a logic 0, unless stated otherwise.

Register Access:

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

To minimize the communication between the master and this device, the address pointer updated automatically without master intervention. The register pointer will be incremented by 1 automatically after the current register has been read successfully.

The address pointer value itself cannot be read via the I^2C bus.

Any attempt to read an invalid address location returns 0's, and any write to an invalid address location or an undefined bit within a valid address location is ignored by this device.

To move the address pointer to a random register location, first issue a "write" to that register location with no data byte following the commend. For example, to move the address pointer to register 10, send 0x3C 0x0A. Connections for interfacing compass with Raspberry Pi is shown in figure-2.10.

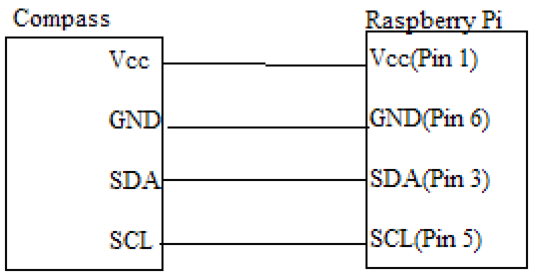


Figure-2.10 Connection diagram for interfacing compass

Python code for interfacing compass with Raspberry-Pi is given in appendix-I.

2.3 GPIO and motors

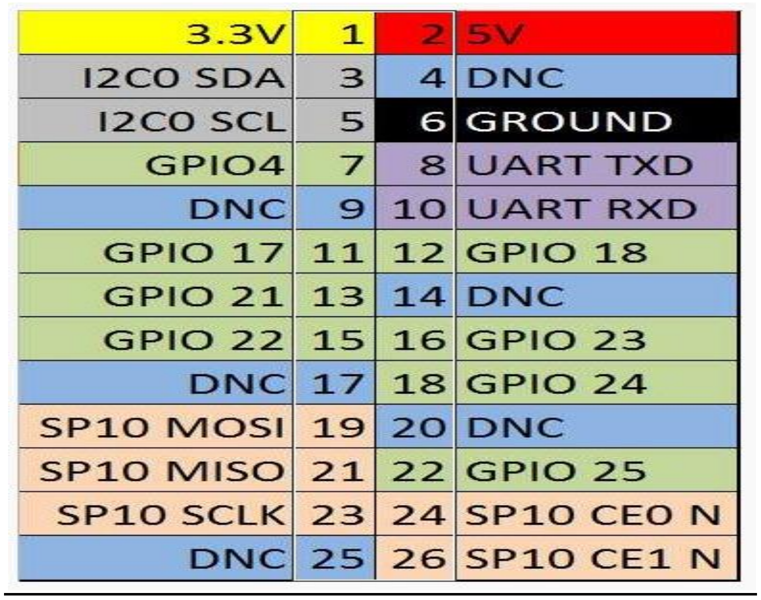
Two 12V DC motors (figure-2.11) interfaced with the L298 Driver (figure-2.12) are used to move the robot to desired location. 300RPM 12V DC geared motors for robotics applications. Very easy to use and available in standard size. Nut and threads on shaft to easily connect and internal threaded shaft for easily connecting it to wheel.

The L298 Driver is a high voltage, high current dual full bridge driver designed to accept standard TTL logic levels and drive inductive loads such relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. Datasheets of L298 Driver is given in appendix. Python code to rotate the motors for certain time is given in appendix-III. Connection diagram for interfacing motor driver is shown in figure-2.14.

Figure-2.11 12V DC motor

Figure-2.12 L298 Driver

- “*sudo apt-get install RPi.GPIO*”.
- *“sudo apt*-get install *python-dev”*.



3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Figure-2.13 Raspberry Pi GPIO Diagram

Algorithm to rotate the motors for certain time:

- Configure Raspberry pi’s 7, 11,13,15,16&18 as output ports.
- Pins 13,15,16&18 will be connected to the motors as inputs and pins 7&11 logic enable for motors.
- Ask for number of seconds to rotate in forward and backward directions.
- Clear the GPIO ports.

Python code to rotate the motors in back and forth directions is given in appendix-I.

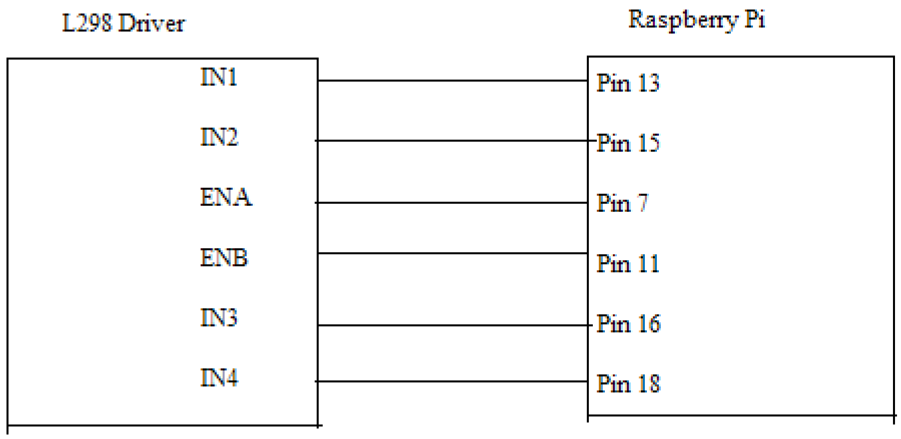


Figure-2.14 Connection diagram for interfacing motor driver

Sample Application:

The robot can be turned towards desired direction by rotating the motors and checking the compass reading.

Algorithm:

1. Check the output of the compass and calculate the bearing (azimuth).
2. If the bearing is not zero rotate the motors by using PWM.
3. Repeat the above steps until the bearing is in between 358° and 2°.

Python code to rotate the robot towards the North is given in appendix-I.

2.4 Raspberry-Pi Camera

2.4.1 Introduction

The camera module (figure-2.16) is capable of 1080p video and still images and connects directly to Raspberry Pi. The dimensions of the camera board are around 25mm x 20mm x 9mm and weighing in at just over 3g. The sensor has a native resolution of 5 megapixel, and has a fixed focus lens onboard. In terms of still images, the camera is capable of 2592 x 1944 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. This module is only capable of taking pictures and video, not sound.

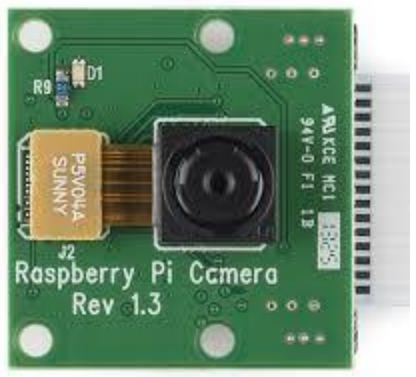
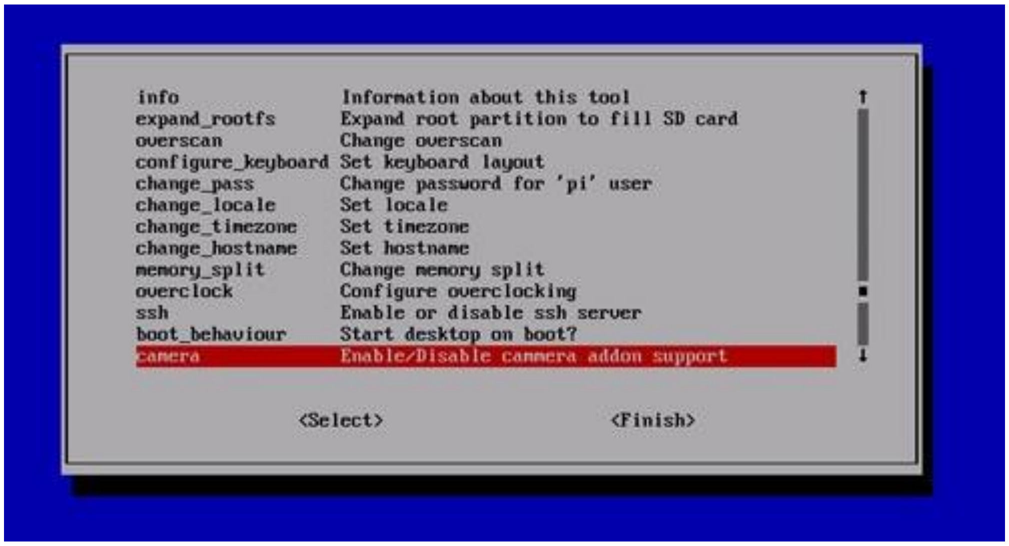


Figure-2.16 Raspberry Pi Camera[11]

Steps to install Raspberry pi camera:

- As the camera can be damaged by static electricity, one has to get discharged by touching an earthed object (e.g. a radiator or PC Chassis) before removing the camera from its grey anti-static bag.
- Install the Raspberry Pi Camera module by inserting the cable into the Raspberry Pi. The cable slots into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port.
- Boot up the Raspberry Pi.

- From the prompt, run "sudo raspi-config". If the "camera" option is not listed, we will need to run a few commands to update your Raspberry Pi. Run "sudo apt-get update" and "sudo apt-get upgrade".
- After running "sudo raspi-config" again we would now see the "camera" option. The screen-shot looks like figure-2.17.



```
info                Information about this tool
expand_rootfs       Expand root partition to fill SD card
overscan            Change overscan
configure_keyboard  Set keyboard layout
change_pass         Change password for 'pi' user
change_locale       Set locale
change_timezone     Set timezone
change_hostname     Set hostname
memory_split        Change memory split
overclock           Configure overclocking
ssh                 Enable or disable ssh server
boot_behaviour      Start desktop on boot?
camera              Enable/Disable cammera addon support

              <Select>                 <Finish>
```

Figure-2.17 Screen-shot after installing Raspberry pi camera

- Navigate to the "camera" option, and enable it. Select “Finish” and reboot the Raspberry Pi.
- Execute “*sudo apt-get install python-picamera*” to program the raspberry pi camera.

Taking a photo with Raspberry pi camera:

- "raspistill" is a command line application that allows us to capture images with your camera module.
- To capture an image in jpeg format, type "raspistill -o image.jpg" at the prompt, where "image" is the name of the image.

Record a video with Raspberry pi camera:

- "raspivid" is a command line application that allows us to capture video with the camera module.
- To capture a 10 second video with the Raspberry Pi camera module, run "raspivid -o video.h264 -t 10000" at the prompt, where "video" is the name of the video and "10000" is the duration of the video to be captured in milliseconds.

Chapter-3

FACE RECOGNITION

Some facial recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distills an image into values and compares the values with templates to eliminate variances.

Popular recognition algorithms include Principal Component Analysis using eigenfaces, Linear Discriminate Analysis, Elastic Bunch Graph Matching using the Fisherface algorithm, the Hidden Markov model, the Multilinear Subspace Learning using tensor representation, and the neuronal motivated dynamic link matching.

Notable software with face recognition ability include:

- digiKam (KDE)
- iPhoto (Apple)
- OpenCV (Open Source)
- Photoshop Elements (Adobe Systems)
- Picasa (Google)
- Picture Motion Browser (Sony)
- Windows Live Photo Gallery (Microsoft)
- DeepFace research project with product "Tag Suggestions" (Facebook)

Advantages and disadvantages:

Among the different biometric techniques, facial recognition may not be the most reliable and efficient. However, one key advantage is that it does not require the cooperation of the test subject to work. Properly designed systems installed in airports, multiplexes, and other public places can identify individuals among the crowd, without passers-by even being aware of the system. Other biometrics like fingerprints, iris scans, and speech recognition cannot perform this kind of mass identification. However, questions have been raised on the effectiveness of facial recognition software in cases of railway and airport security.

Chapter-4

OPENCV

4.1 Introduction

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center in Nizhny Novgorod, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are now full interfaces in Python, Java and MATLAB/OCTAVE (as of version 2.5). The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Ch, Ruby have been developed to encourage adoption by a wider audience.

OpenCV runs on Windows, Android, Maemo, FreeBSD, OpenBSD, iOS, BlackBerry 10, Linux and OS X. The user can get official releases from SourceForge, or take the current snapshot under SVN from there. OpenCV uses CMake.

4.2 OpenCV installation

- Execute the commands
  “sudo apt-get update”,
  ” sudo apt-get install build-essential cmake pkg-config python-dev libgtk2.0-dev libgtk2.0 zlib1g-dev libpng-dev libjpeg-dev libtiff-dev libjasper-dev libavcodec-dev swig”
- Answer yes to any questions about proceeding and wait for the libraries and dependencies to be installed.
- Download and unpack the OpenCV source code by executing the following commands: *“Wgethttp://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.7/opencv-2.4.7.tar.gz”*

  “tar zxvf opencv-2.4.7.tar.gz”

- Now change to the directory with the OpenCV source and execute the following cmake command to build the makefile for the project. Note that some of the parameters passed in to the cmake command will disable compiling performance tests and GPU accelerated algorithms in OpenCV.
  “*cd opencv-2.4.7”*

  “cmake-DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local -DBUILD_PERF_TESTS=OFF-DBUILD_opencv_gpu=OFF-DBUILD_opencv_ocl=OFF”

- Compile the project by executing: *make*
- Once compilation is complete install the compiled OpenCV libraries by executing:

  “sudo make install”

- The code for this project is written in python and has a few dependencies that must be installed. Once connected to your Raspberry Pi in a terminal session, execute the following commands:

  "sudo apt-get install python-pip"

  "sudo apt-get install python-dev"

  "sudo pip install picamera"

  "sudo pip install rpio"

4.3 Eigenfaces

Eigenfaces[9] is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

Algorithmic Description:

It is very important that the face images must be centered and of the same size. Suppose Γ is an N^2X1 vector, corresponding to an NXN face image I. The idea is to represent Γ (Φ= Γ - mean face) into a low-dimensional space:

$$\Phi\hat{} - mean = w_1u_1 + w_2u_2 + \ldots w_K u_K \ (K<<N^2) \quad (4.3.1)$$

Computation of the eigenfaces:

Step 1: obtain face images $I_1, I_2, ..., I_M$ (training faces)

Step 2: represent every image I_i as a vector Γ_i

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i \quad (4.3.2)$$

Step 4: subtract the mean face:

$$\Phi_i = \Gamma_i - \Psi \quad (4.3.3)$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi^T{}_n = AA^T \quad (N^2XN^2 \text{ matrix}) \quad (4.3.4)$$

Where $A = [\Phi_1, \Phi_2 \ldots \Phi_M]$ $(N^2XM$ matrix)

Step 6: compute the eigenvectors u_i of AA^T.

But computation of eigenvectors for the matrix AA^T is not practical as the size of the matrix will be very large.

Step 6.1: consider the matrix $A^T A$ (MXM matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A\, v_i = \mu_i\, v_i \tag{4.3.5}$$

The relationship between u_i and v_i:

$$A^T A\, v_i = \mu_i\, v_i \Longrightarrow AA^T A\, v_i = \mu_i\, Av_i$$

$$\Longrightarrow C\, Av_i = \mu_i\, Av_i \text{ or } C\, u_i = \mu_i\, v_i \text{ where } u_i = A\, v_i \tag{4.3.6}$$

Note: $A^T A$ can have up to N^2 eigenvalues and eigenvectors. AA^T can have up to M eigenvalues and eigenvectors. The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M largest eigenvalues of $A A^T$ (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = A\, v_i$

(Normalize u_i such that $||u_i|| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

Representing faces onto this basis:

Each face (minus the mean) Φ_i in the training set can be represented as a linear combination of the best K eigenvectors:

$$\Phi_i\text{- mean} = \sum_{j=1}^{K} w_j u_j,\ (w_j = u^T{}_j \Phi_i) \tag{4.3.7}$$

(Where u_j- eigenfaces)

Each normalized training face Φ_i is represented in this basis by a vector:

$$\Omega_i = \begin{bmatrix} w^i{}_1 \\ w^i{}_2 \\ . \\ \vdots \\ \\ w^i{}_K \end{bmatrix},\ i=1,2.....M \tag{4.3.8}$$

Face Recognition Using Eigenfaces:

Given an unknown face image Γ (centered and of the same size like the training faces) follow these steps:

Step 1: Normalize Γ: $\Phi = \Gamma - \Psi$

Step 2: Project on the eigenspace

$$\Phi' = \sum_{j=1}^{K} w_j u_j \,, \text{ where } (wj = u^T{}_j \Phi_i) \qquad (4.3.9)$$

Step 3: Represent Φ' as: $\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \vdots \\ \\ w_K \end{bmatrix}$ (4.3.10)

Step 4: Find $e_r = \min_l \|\Omega - \Omega^l\|$ (4.3.11)

Step 5: If $e_r < T_r$, then G is recognized as face *l* from the training set. Where T_r is the threshold error.

The distance e_r is called distance within the face space (difs) and T_r is the threshold error.

We can use the Mahalanobis distance to compute e_r.

$$||\Omega - \Omega_l|| = \sum_{j=1}^{K} \frac{1}{\lambda_i} (w_i - w^k{}_i)^2 \qquad (4.3.12)$$

Yale Face database: A The AT&T Face database is good for initial tests, but it's a fairly easy database.database consists of 15 people each with 11 grayscale images sized 320 X 243 pixel. There are changes in the light conditions (center light, left light, right light), facial expressions (happy, normal, sad, sleepy, surprised, wink) and glasses (glasses, no-glasses).

4.4 Haar feature-based cascade classifier for object detection

First, a classifier (namely a cascade of boosted classifiers working with haar-like features) is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a “1” if the region is likely to show the object (i.e., face/car), and “0” otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily “resized” in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales.

The word “cascade” in the classifier name means that the resultant classifier consists of several simpler classifiers (stages) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The word “boosted” means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using one of four different boosting techniques (weighted voting). Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported. The basic classifiers are decision-tree classifiers with at least 2 leaves. Haar-like features are the input to the basic classifiers, and are calculated as described below. The current algorithm uses the following Haar-like features:

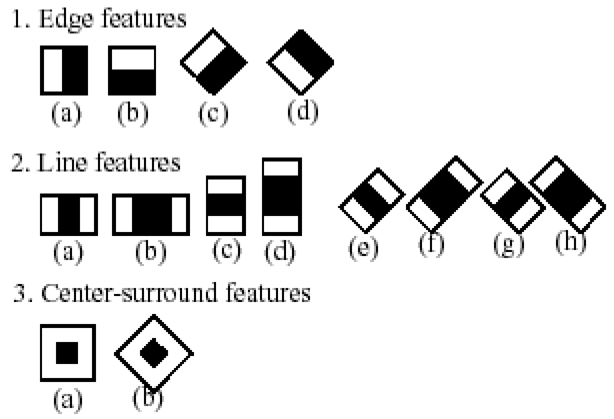


Figure-4.1 Various haar like features

- Haar features are similar to convolution kernels which are used to detect the presence of that feature in the given image.
- Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle (figure-4.2)

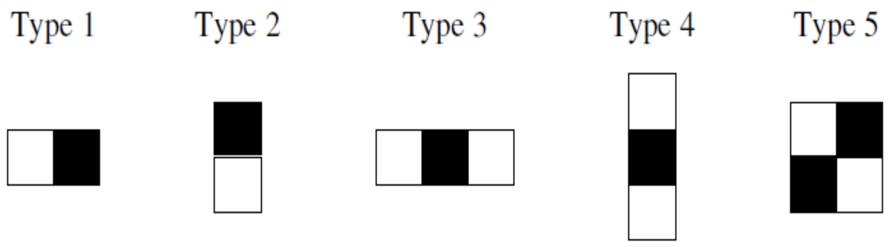


Figure-4.2 Haar features used in viola Jones

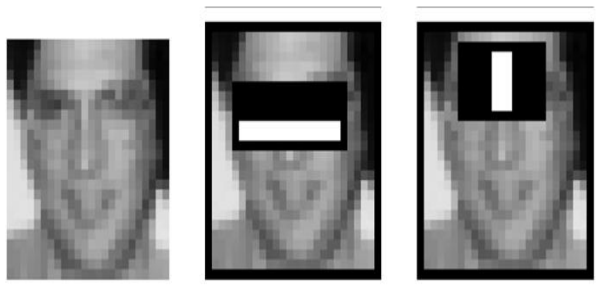
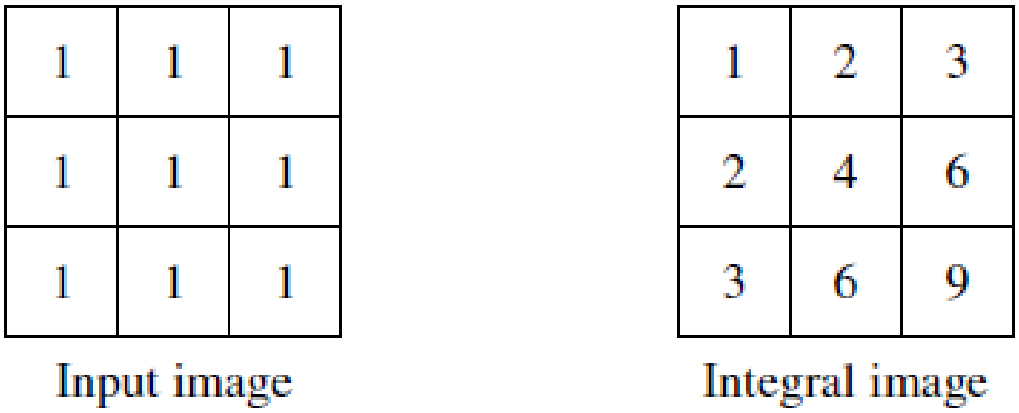


Figure-4.3 Applying haar features on a given image

Viola jones algorithm[5] uses a 24x24 window as the base window size to start evaluating these features in any given image. There will be about 160,000+ features in this window if all possible parameters of the haar features like position, scale and type calculating are considered. Since it is clear that huge number of these rectangular haar features have to be evaluated each time Viola Jones have come up with a neat technique to reduce the computation rather than summing up all pixel values under the black and white rectangles every time. Concept of integral image is introduced to find the sum of all pixels under a rectangle with just 4 corner values of the integral image. In an integral image (figure-4.4) the value at pixel (x,y) is the sum of pixels above and to the left of (x,y).



1	1	1
1	1	1
1	1	1

1	2	3
2	4	6
3	6	9

Figure-4.4 Calculation of Integral images

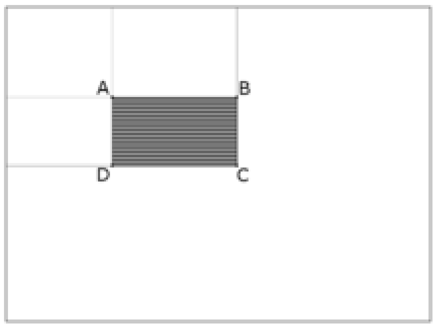


Figure-4.5 Finding the sum of a rectangular area

$$\sum_{\substack{x_0 < x \leq x_1 \\ y_0 < y \leq y_1}} i(x,y) = I(C) + I(A) - I(B) - I(D) \qquad (4.4.1)$$

Integral image allows for the calculation of sum of all pixels inside any given rectangle using only four values at the corners of the rectangle.

Sum of all pixels in

D = 1+4-(2+3)

= A+(A+B+C+D)-(A+C+A+B)

= D

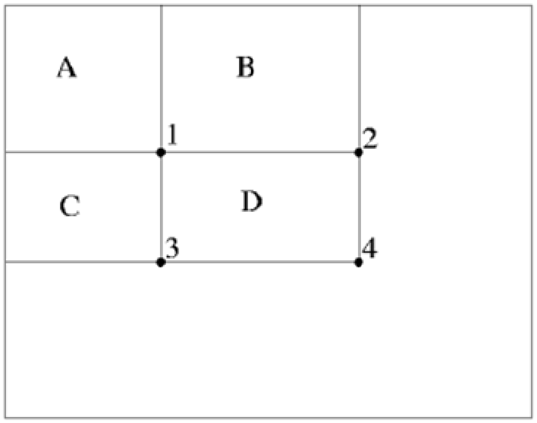


Figure-4.6 Calculating Integral Images

As stated previously there can be approximately 160,000 + feature values within a detector at 24x24 base resolution which need to be calculated. But it is to be understood that only few set of features will be useful among all these features to identify a face as in figure-4.7.

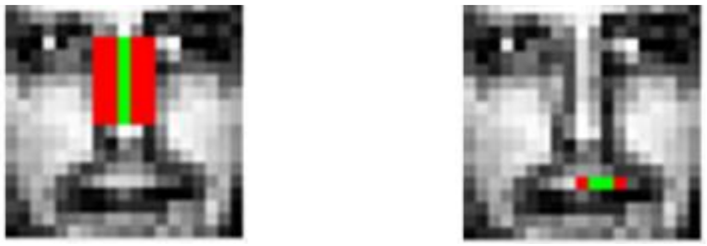


Figure-4.7 Relevant and irrelevant features

Adaboost Algorithm:

Adaboost is a machine learning algorithm which helps in finding only the best features among all these 160,000+ features. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window has a face or not. Each of the selected features are considered okay to be included if they can atleast perform better than random guessing (detects more than half the cases).

These features are also called as weak classifiers. Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \cdots \qquad (4.4.2)$$

F(x) - Strong classifier; f_1(x) - weak classifier

Algorithm:

- AdaBoost starts with a uniform distribution of "weights" over training examples.
- Select the classifier with the lowest weighted error (i.e. a "weak" classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

$$h(x)_{strong} = \begin{cases} 1 & \alpha_1 h_1(x) + \cdots + \alpha_n h_n(x) \geq \frac{1}{2}(\alpha_1 + \cdots + \alpha_n) \\ 0 & otherwise \end{cases} \tag{4.4.3}$$

Cascading:

The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size. Even if an image should contain one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces). So the algorithm should concentrate on discarding non-faces quickly and spend more on time on probable face regions. Hence a single strong classifier formed out of linear combination of all best features is not a good to evaluate on each window because of computation cost. Therefore a cascade classifier is used which is composed of stages each containing a strong classifier. So all the features are grouped into several stages where each stage has certain number of features. The job of each stage is used to determine whether a given sub window is definitely not a face or may be a face. A given sub window is immediately discarded as not a face if it fails in any of the stage as shown in the figure-4.8.

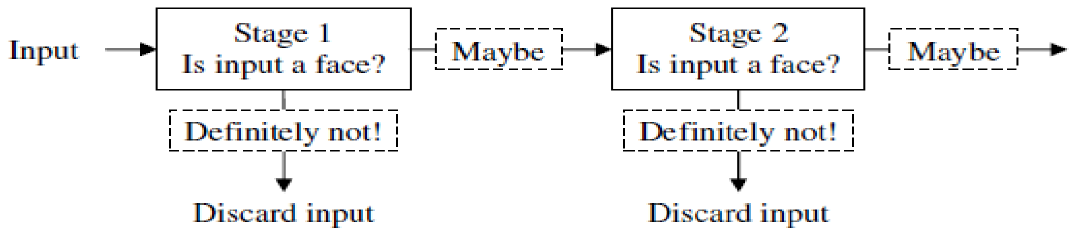


Figure-4.8 Cascading

To design a cascade we must choose:

- Number of stages in cascade (strong classifiers).
- Number of features of each strong classifier.

Threshold of each strong classifier

Lanczos resampling:

Lanczos resampling or Lanczos filter is a mathematical formula used to smoothly interpolate the value of a digital signal between its samples. It maps each sample of the given signal to a translated and scaled copy of the Lanczos kernel, which is a sinc function windowed by the central hump of a dilated sinc function. The sum of these translated and scaled kernels is then evaluated at the desired points.

Lanczos resampling is typically used to increase the sampling rate of a digital signal, or to shift it by a fraction of the sampling interval. It is often used also for multivariate interpolation, for example to resize or rotate a digital image. It has been considered the "best compromise" among several simple filters for this purpose.

The effect of each input sample on the interpolated values is defined by the filter's reconstruction kernel *L(x),* called the Lanczos kernel. It is the normalized sinc function *sinc(x),* windowed (multiplied) by the Lanczos window, or sinc window, which is the central lobe of a horizontally-stretched sinc function *sinc(x/a)* for $-a \le x \le a$.

$$L(x) = \begin{cases} sinc(x)\ sinc(x/a) & if\ -a < x < a \\ 0 & otherwise \end{cases} \tag{4.4.4}$$

Equivalently,

$$L(x) = \begin{cases} 1 & if\ x = 0 \\ \frac{a \sin(\pi x) \sin(\pi x/a)}{\pi^2 x^2} & if\ 0 < |x| < a \\ 0 & otherwise \end{cases} \tag{4.4.5}$$

The parameter 'a' is a positive integer, typically 2 or 3, which determines the size of the kernel. The Lanczos kernel has $2a - 1$ lobes, a positive one at the center and $a - 1$ alternating negative and positive lobes on each side.

Interpolation formula:

Given a one-dimensional signal with samples s_i, for integer values of *i*, the value *S(x)* interpolated at an arbitrary real argument *x* is obtained by the discrete convolution of those samples with the Lanczos kernel; namely,

$$S(x) = \sum_{i=[x]-a+1}^{[x]+a} s_i\, L(x - i) \tag{4.4.6}$$

where a is the filter size parameter and $[x]$ is the floor function. The bounds of this sum are such that the kernel is zero outside of them.

Multidimensional interpolation:

Lanczos filter's kernel in two dimensions is simply the product of two one-dimensional kernels:

$$L(x, y) = L(x)\ .\ L(y) \tag{4.4.7}$$

Given a two-dimensional signal s_{ij}defined at integer points (i, j) of the plane (e.g. intensities of pixels in a digital image), the reconstructed function is

$$S(x, y) = \sum_{i=[x]-a+1}^{[x]+a} \sum_{j=[y]-a+1}^{[y]+a} s_{ij} \, L(x-i) \quad L(y-j) \qquad (4.4.8)$$

When resampling a two-dimensional signal at regularly spaced points (x, y), one can save some computation by resampling the entire signal along a single axis, then resampling the resulting two-dimensional signal along the other axis. These formulas generalize to signals with three or more dimensions, in the obvious way.

Chapter- 5

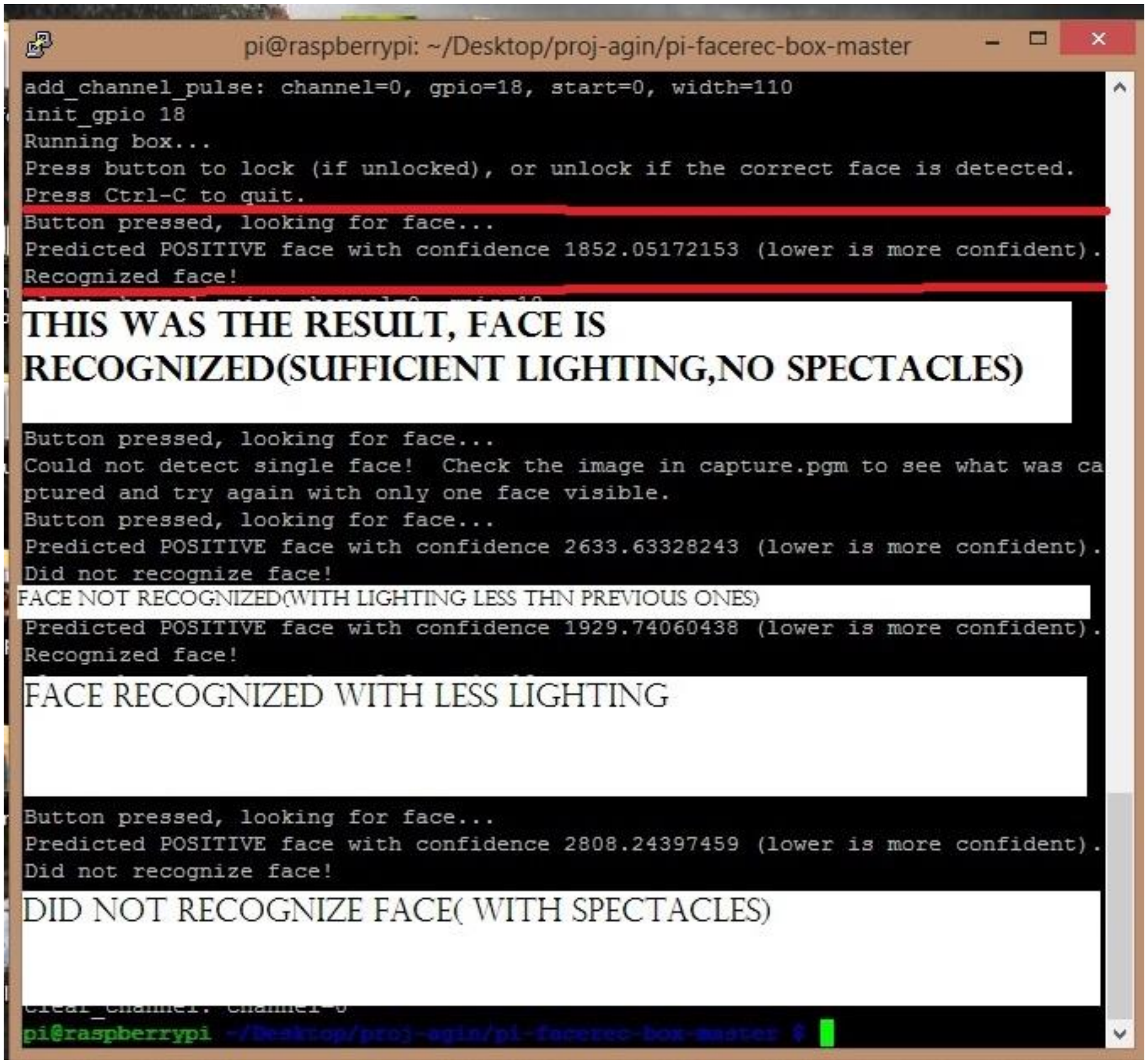
RESULTS AND DISCUSSION

Results:

GPS module has been interfaced with Raspberry pi and the co-ordinates of location (Chemical Department, NIT Calicut) has been parsed successfully.

Compass and motors have been successfully interfaced and simple application like automatically setting the rover to the North (bearing 0) is done.

Raspberry Pi camera has been successfully interfaced and face recognition is implemented, results are shown in figure-5.1.



```
add_channel_pulse: channel=0, gpio=18, start=0, width=110
init_gpio 18
Running box...
Press button to lock (if unlocked), or unlock if the correct face is detected.
Press Ctrl-C to quit.
Button pressed, looking for face...
Predicted POSITIVE face with confidence 1852.05172153 (lower is more confident).
Recognized face!
```

THIS WAS THE RESULT, FACE IS RECOGNIZED(SUFFICIENT LIGHTING,NO SPECTACLES)

```
Button pressed, looking for face...
Could not detect single face!  Check the image in capture.pgm to see what was ca
ptured and try again with only one face visible.
Button pressed, looking for face...
Predicted POSITIVE face with confidence 2633.63328243 (lower is more confident).
Did not recognize face!
```

FACE NOT RECOGNIZED(WITH LIGHTING LESS THN PREVIOUS ONES)

```
Predicted POSITIVE face with confidence 1929.74060438 (lower is more confident).
Recognized face!
```

FACE RECOGNIZED WITH LESS LIGHTING

```
Button pressed, looking for face...
Predicted POSITIVE face with confidence 2808.24397459 (lower is more confident).
Did not recognize face!
```

DID NOT RECOGNIZE FACE(WITH SPECTACLES)

```
clear_channel: channel=0
pi@raspberrypi ~/Desktop/proj-agin/pi-facerec-box-master $
```

Figure-5.1 Face recognition output

FUTURE SCOPE OF THE PROJECT:

The present set up can be used for full implementation of autonomous vehicle guided by vision, GPS and compass.

APPENDIX-I

Python code for LED blinking:

```
import RPi.GPIO as GPIO ## Import GPIO Library
import time ## Import 'time' library.  Allows us to use 'sleep'

GPIO.setmode(GPIO.BOARD) ## Use BOARD pin numbering
GPIO.setup(7, GPIO.OUT) ## Setup GPIO pin 7 to OUT

## Define function named Blink()
def Blink(numTimes, speed):
    for i in range(0,numTimes): ## Run loop numTimes
        print "Iteration " + str(i+1) ##Print current loop
        GPIO.output(7, True) ## Turn on GPIO pin 7
        time.sleep(speed) ## Wait
        GPIO.output(7, False) ## Switch off GPIO pin 7
        time.sleep(speed) ## Wait
    print "Done" ## When loop is complete, print "Done"
    GPIO.cleanup()

## Prompt user for input
iterations = raw_input("Enter the total number of times to blink: ")
speed = raw_input("Enter the length of each blink in seconds: ")

## Start Blink() function. Convert user input from strings to numeric data types and pass to
Blink() as parameters
Blink(int(iterations),float(speed))
```

Python code for extracting longitude and latitude:

```
import os
 import serial
 def get_present_gps():
     ser=serial.Serial('/dev/ttyAMA0',9600) #open serial communication
     ser.open()
                                              # open a file to write gps data
     f = open('/home/pi/Desktop/gps1', 'w')
     data=ser.read(512)                     # read 1024 bytes
     f.write(data)                          #write data into file
     f.flush()                               #flush from buffer into os buffer
                                             #ensure to write from os buffers(internal) into disk
     f = open('/home/pi/Desktop/gps1', 'r')    # fetch  the required file
```

```
    for line in f.read().split('\n'):          #GPS parsing
        if line.startswith('$GPGGA'):
            lat, _, lon= line.split(',')[2:5]
            try:
                lat=float(lat)
                lon=float(lon)

                print lat
                print lon
                a=[lat,lon]
                print a[0]
                print a
            except:
                pass
get_present_gps()
```

Python Code for interfacing compass:

```
import smbus
import time
import math
bus = smbus.SMBus(1)
address = 0x1e
def read_byte(adr):          #communicate with compass
    return bus.read_byte_data(address, adr)
def read_word(adr):
    high = bus.read_byte_data(address, adr)
    low = bus.read_byte_data(address, adr+1)
    val = (high<< 8) + low
    return val
def read_word_2c(adr):
    val = read_word(adr)
    if (val>= 0x8000):
        return -((65535 - val)+1)
    else:
        return val
def write_byte(adr,value):
    bus.write_byte_data(address, adr, value)
write_byte(0, 0b01110000)
write_byte(1, 0b00100000)
write_byte(2, 0b00000000)
scale = 0.92
x_offset = -10
y_offset = 10
x_out = (read_word_2c(3)- x_offset+2) * scale      #calculating x,y,z coordinates
y_out = (read_word_2c(7)- y_offset+2)* scale
z_out = read_word_2c(5) * scale
```

```
bearing = math.atan2(y_out, x_out)+.48              #0.48 is correction value
if(bearing < 0):
    bearing += 2* math.pi
print "Bearing:", math.degrees(bearing)
```

Python code to rotate the motors for certain time:

```
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)     #set 7,11,13,15,16,18 as o/p pins
gpio.setup(11,gpio.OUT)
gpio.setup(13,gpio.OUT)
gpio.setup(15,gpio.OUT)
gpio.setup(16,gpio.OUT)
gpio.setup(18,gpio.OUT)
gpio.output(7,True)                #set 7,11 as enables
gpio.output(11,True)
n=raw_input(" enter no of seconds")
try:
    while True:
        gpio.output(13,True)     #13,15 to control M1
        gpio.output(15,False)
        gpio.output(16,True)      #16,18 to control M2
        gpio.output(18,False)
        time.sleep(float(n))
        gpio.output(13,False)
        gpio.output(15,True)
        gpio.output(16,False)
        gpio.output(18,True)
        time.sleep(float(n))
except KeyboardInterrupt:
    pass
gpio.cleanup()
```

Python code to rotate the robot towards the North:

```
import RPi.GPIO as gpio
import smbus
import time
import math
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)        #set 7,11,13,15,16,18 as o/p pins
gpio.setup(11,gpio.OUT)
gpio.setup(13,gpio.OUT)
gpio.setup(15,gpio.OUT)
gpio.setup(16,gpio.OUT)
gpio.setup(18,gpio.OUT)
def get_bearing():
    bus=smbus.SMBus(1)
    address=0x1e
    def read_byte(adr):
        return bus.read_byte_data(address, adr)
    def read_word(adr):
        high = bus.read_byte_data(address, adr)
        low = bus.read_byte_data(address, adr+1)
        val = (high<< 8) +low
        return val
    def read_word_2c(adr):
        val = read_word(adr)
        if (val>=0x8000):
            return -((65535-val)+1)
        else:
            return val
    def write_byte(adr,value):
        bus.write_byte_data(address, adr, value)
    write_byte(0, 0b01110000)
    write_byte(1, 0b00100000)
    write_byte(2, 0b00000000)
    scale = 0.92
    x_offset = -10
    y_offset = 10
    x_out = (read_word_2c(3)- x_offset+2) * scale
    y_out = (read_word_2c(7)- y_offset+2)* scale
    z_out = read_word_2c(5) * scale
    bearing = math.atan2(y_out, x_out)+.45
    if (bearing<0):
        bearing +=2*math.pi
        return math.degrees(bearing)
    else:
        return math.degrees(bearing)
def rotate():
    a=get_bearing()
    if (a>=0 and a<=1):
        gpio.output(7,False)
        gpio.output(11,False)
        gpio.cleanup()
    elif(a>=358 and a<=360):
        gpio.output(7,False)
        gpio.output(11,False)
        gpio.cleanup()
    else:
        gpio.setup(18,False)
        gpio.setup(16,False)
        time.sleep(.2)
        gpio.output(7,False)
        gpio.output(11,True)
        p=gpio.PWM(11,80)
        p.start(1)
        time.sleep(.2)
        gpio.setup(13,False)
        gpio.setup(15,False)
        gpio.setup(16,True)
        gpio.setup(18,False)
        p.ChangeDutyCycle(50)
        rotate()
rotate()
```

Code for serial communication (receiving) with python interpreter:

```
import serial

scr=serial.Serial('/dev/ttyAMA0',9600)    #config serial port

scr.open()

scr.isOpen()

scr.inWaiting()

scr.read(scr.inWaiting())        #read all the waiting data

scr.close()
```

APPENDIX-II

Python code for configuring camera, GPIO for switch and face recognition:

```
# Raspberry Pi Face Recognition
# Copyright 2013 Tony DiCola
# Copyright 2014 vaila Ruthvik(ruthvik.nitc@gmail.com)
# Edit the values below to configure the training and usage of the
# face recognition box.
# Pi GPIO port which is connected to the lock servo signal line.
# Pulse width value (in microseconds) for the servo at the unlocked and locked
# position.  Center should be a value of 1500, max left a value of 1000, and
# max right a value of 2000.
# Pi GPIO port which is connected to the button.
BUTTON_PIN = 25
# Down and up values for the button.  The code expects to detect a down to up
# transition as an activation of the button.  Therefore a normally open button
# should be False (low) when down and True (high) when up.
BUTTON_DOWN = False  # Low signal
BUTTON_UP   = True   # High signal
# Threshold for the confidence of a recognized face before it's considered a
# positive match.  Confidence values below this threshold will be considered
# a positive match because the lower the confidence value, or distance, the
# more confident the algorithm is that the face was correctly detected.
# Start with a value of 3000, but you might need to tweak this value down if
# you're getting too many false positives (incorrectly recognized faces), or up
# if too many false negatives (undetected faces).
POSITIVE_THRESHOLD = 2000.0
# File to save and load face recognizer model.
TRAINING_FILE = 'training.xml'
# Directories which contain the positive and negative training image data.
```

```
POSITIVE_DIR = './training/positive'
NEGATIVE_DIR = './training/negative'
# Value for positive and negative labels passed to face recognition model.
# Can be any integer values, but must be unique from each other.
# You shouldn't have to change these values.
POSITIVE_LABEL = 1
NEGATIVE_LABEL = 2

# Size (in pixels) to resize images for training and prediction.
# Don't change this unless you also change the size of the training images.
FACE_WIDTH  = 92
FACE_HEIGHT = 112

# Face detection cascade classifier configuration.
# You don't need to modify this unless you know what you're doing.
# See: http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html
HAAR_FACES         = 'haarcascade_frontalface_alt.xml'
HAAR_SCALE_FACTOR  = 1.3
HAAR_MIN_NEIGHBORS = 4
HAAR_MIN_SIZE      = (30, 30)

# Filename to use when saving the most recently captured image for debugging.
DEBUG_IMAGE = 'capture.pgm'

def get_camera():
	# Camera to use for capturing images.
	# Use this code for capturing from the Pi camera:
	import picam
	return picam.OpenCVCapture()
```

Python code for hardware settings:

```
Copyright 2013 Tony DiCola
Copyright 2014 Vaila Ruthvik(ruthvik.nitc@gmail.com)
This code can be sujected to any kind of editions and distributions provided the names of
previous writers are provided as above.
"""
import time
import cv2
import RPIO
from RPIO import PWM
import picam
import config
import face
class GPIO(object):
        """Class to represent the state and encapsulate access to the hardware of
        the treasure box."""
        def __init__(self):
                # Initialize button.
                RPIO.setup(config.BUTTON_PIN, RPIO.IN)
                # Set initial button state.
                self.button_state = RPIO.input(config.BUTTON_PIN)
                self.is_locked = None
        def is_button_up(self):
                """Return True when the box button has transitioned from down to up (i.e.
                the button was pressed)."""
                old_state = self.button_state
                self.button_state = RPIO.input(config.BUTTON_PIN)
                # Check if transition from down to up
                if old_state == config.BUTTON_DOWN and self.button_state ==
config.BUTTON_UP:
```

```
                # Wait 20 milliseconds and measure again to debounce switch.
                time.sleep(20.0/1000.0)
                self.button_state = RPIO.input(config.BUTTON_PIN)
                if self.button_state == config.BUTTON_UP:
                    return True
            return False
```

Pi camera device capture class for OpenCV:

```
Copyright 2014 Vaila Ruthvik(ruthvik.nitc@gmail.com)
Copyright 2013 Tony DiCola
Courtesy OpenCv
Pi camera device capture class for OpenCV.  This class allows you to capture a
single image from the pi camera as an OpenCV image.
"""
import io
import time
import cv2
import numpy as np
import picamera
import config
class OpenCVCapture(object):
    def read(self):
        """Read a single frame from the camera and return the data as an OpenCV
        image (which is a numpy array).
        """
        # This code is based on the picamera example at:
        #   http://picamera.readthedocs.org/en/release-1.0/recipes1.html#capturing-to-an-
opencv-object
        # Capture a frame from the camera.
        data = io.BytesIO() # operating with binary data
        with picamera.PiCamera() as camera:
            camera.capture(data, format='jpeg') #capture an image from the camera and storing it in output
        data = np.fromstring(data.getvalue(), dtype=np.uint8) #uint8 is used in graphics (colors have generally non -ve values)
        # Decode the image data and return an OpenCV image.
        image = cv2.imdecode(data, 1)   #imdecode reads an image from buffer and 1 will force the data to be a clolor image.
        # Save captured image for debugging.
        cv2.imwrite(config.DEBUG_IMAGE, image) #saves an image to a specified file.
        # Return the captured image data.
        return image
```

Python code for functions to help with the detection and cropping of faces:

```
Face Detection Helper Functions
Copyright 2014 Vaila Ruthvik(ruthvik.nitc@gmail.com)
Courtesy OpenCv, Tony DiCola
Functions to help with the detection and cropping of faces.
"""
import cv2
import config
haar_faces = cv2.CascadeClassifier(config.HAAR_FACES)
def detect_single(image): # iamge is a matrix of type cv_8U containing an image where object are detected
    """Return bounds (x, y, width, height) of detected face in grayscale image.
      If no face or more than one face are detected, None is returned.
    """
    faces = haar_faces.detectMultiScale(image,
                            scaleFactor=config.HAAR_SCALE_FACTOR,
                            minNeighbors=config.HAAR_MIN_NEIGHBORS,
                            minSize=config.HAAR_MIN_SIZE,
                            flags=cv2.CASCADE_SCALE_IMAGE)
#HAAR_SCALE_FACTORis the parameter specifying how much image size is reduced at each.
    if len(faces) != 1:                          #HAAR_MIN_NEIGHBOURS says how many
neighbours each candidate rectangle should have to retain.
                return None                          #HAAR_MIN_SIZE says minimum possible
object size objects smaller than that are ignored.
    return faces[0]
def crop(image, x, y, w, h):
    """Crop box defined by x, y (upper left corner) and w, h (width and height)
    to an image with the same aspect ratio as the face training data.  Might
    return a smaller crop if the box is near the edge of the image.
    """
    crop_height = int((config.FACE_HEIGHT / float(config.FACE_WIDTH)) * w)
    midy = y + h/2
    y1 = max(0, midy-crop_height/2)
    y2 = min(image.shape[0]-1, midy+crop_height/2)
    return image[y1:y2, x:x+w]
def resize(image):
    """Resize a face image to the proper size for training and detection.
    """
    return cv2.resize(image,
                               (config.FACE_WIDTH, config.FACE_HEIGHT),
                               interpolation=cv2.INTER_LANCZOS4)
#cv2.INTER_LANCZOS4 selects smooth interpolation technique for resizing an image.
```

Python code for Positive Image Capture Script:

Copyright 2013 Tony DiCola

Copyright 2014 Ruthvik Vaila

This code can be sujected any kind of editions and distributions provided the names ofprevious writers are provided as above.

Run this script to capture positive images for training the face recognizer.

```
"""
import glob
import os
import sys
import select
import cv2
import hardware
import config
import face
# Prefix for positive training image filenames.
POSITIVE_FILE_PREFIX = 'positive_'
def is_letter_input(letter):
        # Utility function to check if a specific character is available on stdin.
        # Comparison is case insensitive.
        if select.select([sys.stdin,],[],[],0.0)[0]:
                input_char = sys.stdin.read(1)
                return input_char.lower() == letter.lower()
        return False
if __name__ == '__main__':
        camera = config.get_camera()
        box = hardware.GPIO()
        # Create the directory for positive training images if it doesn't exist.
        if not os.path.exists(config.POSITIVE_DIR):
                os.makedirs(config.POSITIVE_DIR)
        # Find the largest ID of existing positive images.
        # Start new images after this ID value.
        files = sorted(glob.glob(os.path.join(config.POSITIVE_DIR,
                POSITIVE_FILE_PREFIX + '[0-9][0-9][0-9].pgm')))
        count = 0
        if len(files) > 0:
```

```
        # Grab the count from the last filename.
        count = int(files[-1][-7:-4])+1
    print 'Capturing positive training images.'
    print 'Press button or type c (and press enter) to capture an image.'
    print 'Press Ctrl-C to quit.'
    while True:
        # Check if button was pressed or 'c' was received, then capture image.
        if box.is_button_up() or is_letter_input('c'):
            print 'Capturing image...'
            image = camera.read()
            # Convert image to grayscale.
            image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
            # Get coordinates of single face in captured image.
            result = face.detect_single(image)
            if result is None:
                print 'Could not detect single face!  Check the image in capture.pgm'
\
                      ' to see what was captured and try again with only one face
visible.'
                continue
            x, y, w, h = result
            # Crop image as close as possible to desired face aspect ratio.
            # Might be smaller if face is near edge of image.
            crop = face.crop(image, x, y, w, h)
            # Save image to file.
            filename = os.path.join(config.POSITIVE_DIR, POSITIVE_FILE_PREFIX
+ '%03d.pgm' % count)
            cv2.imwrite(filename, crop)
            print 'Found face and wrote training image', filename
            count += 1
```

REFERENCES

[1]. www.raspberrypi.org/raspbian

[2]. www.wikipedia.org/raspberrypi

[3]. https://www.nmea.org/content/about_the_nmea/missionvisval.asp

[4]. https://learn.adafruit.com/adafruit-hmc5883l-breakout-triple-axis-magnetometer-compass-sensor/overview

[5]. http:en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

[6]. https://github.com/ruthvik92/HMC5883L-GPS--DCMotor-PythonCodesForRaspberryPi

[7]. http://opencv.org/

[8]. http://raspberrypi.stackexchange.com/users/9043/ruthvik-vaila

[9]. M. Turk and A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol.3, no. 1, pp. 71-86, 1991.

[10]. http://www.rhydolabz.com/documents/gps_gsm/MiniGPS_UserManual_A01.pdf

[11]. http://www.raspberrypi.org/wp-content/uploads/2013/07/RaspiCam-Documentation.pdf