

MAJOR PROJECT REPORT

ON

ARTIFICIAL INTELLIGENT RESPONDER V2.0

Submitted in partial fulfillment of the requirement for the award of degree of
Bachelor of Technology

In

Electronics and Communication Engineering

Under the Supervision of:

Mr. Vaneet Singh

Mr. Parveen Kumar

Submitted By:

Gaganpreet Singh (01513202811)

Baljeet Singh (03013202811)



Department of Electronics and Communication Engineering
Guru Tegh Bahadur Institute of Technology
Affiliated to GGSIP University, New Delhi
(Session: 2011-2015)

DECLARATION

This is to certify that Report entitled “ARTIFICIAL INTELLIGENT RESPONDER v2.0” which is submitted by us in partial fulfillment of the requirement for the award of degree B.Tech. in Electronics and communication Engineering to Guru Tegh Bahadur Institute of Technology , Guru Gobind Singh Indraprastha University, Delhi comprises only our original work and due acknowledgement has been made in the text to all other material used.

Date:

Gaganpreet Singh (01513202811)

Baljeet Singh (03013202811)

CERTIFICATE

This is to certify that Report entitled “ARTIFICIAL INTELLIGENT RESPONDER v2.0” which is submitted by me in partial fulfillment of the requirement for the award of degree B.Tech. In Electronics and communication Engineering to Guru Tegh Bahadur Institute of Technology, GGSIP University, Delhi is a record of the candidates own work carried out by them under our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Prof. Vaneet Singh
(Project Guide & HOD ECE)

Mr. Parveen Kumar
(Project Guide)

Mrs. Prachi Dewan
(Project Coordinator)

Mrs. Manpreet Kaur
(Project Coordinator)

Mrs. Vidisha Khetarpal
(Project Coordinator)

Prof. Amrik Singh
(Overall Project Coordinator)

ACKNOWLEDGEMENT

I would like to express great gratitude towards our coordinators, Mrs. Manpreet Kaur, Mrs. Vidisha Khetarpal, Mrs. Prachi Dewan, Prof. Vaneet Singh, Mr. Parveen Kumar, Prof. Amrik Singh, Mr. Sahaj Singh and Mr. Mahendar who have given us suggestions, support and help. Our sincere thanks are also due to the rest of faculty of the college whose advice also helped us to accomplish the project in time or in any other aspects of our study at Guru Tegh Bahadur Institute of Technology.

ABSTRACT

The main aim to develop A.I.R (ARTIFICIAL INTELLIGENT RESPONDER) is to provide a Virtual Tutor (having some sort of intelligent capability) for children, where we can strive towards providing high quality education to children. There will be no need to worry about children education and homework. No need to turn on your PC/Laptop to search for solutions for any Query, Just ask to AIR and it will be always ready to search the answer over the internet and convert it to audio Speech within 5-6 seconds (with 512 kbps internet connection).In Updated Version of AIR (AIR v2.0) There is a Language Translation Feature Along With High Definition Touch Screen That's all in a Small Area (Compact Version/Fully Portable).

List of tables

| Serial Number | Table Name | Page Number |
|----------------------|---|--------------------|
| 1 | Five Status LEDs. | 5 |
| 2 | Some of the most important directories in Raspbian file system. | 18 |
| 3 | The Switches that can be used with chmod. | 20 |

List of figures

| <i>Figure Number</i> | <i>Figure Name</i> | <i>Page Number</i> |
|---------------------------------|--|-------------------------------|
| 1.1 | A map of the hardware interface of the Raspberry Pi | 3 |
| 1.2 | Some of the older boards came equipped with polyfuses | 4 |
| 1.3 | The Pins and headers on the Raspberry Pi | 5 |
| 1.4 | The basic peripherals: a microUSB power supply, cables, and SD card. | 7 |
| 1.5 | The Raspi-config tool menu | 9 |
| 2.1 | The graphical desktop | 14 |
| 2.2 | LXTerminal gives you access to the command line (or shell). | 16 |
| 2.3 | File permissions for owner, group, and everyone. | 19 |
| 2.4 | Python options on the Raspbian Desktop | 22 |
| 2.5 | IDLE Interactive Shell | 24 |
| 3.1 | Block Diagram of AIR | 29 |

Index

| <i>Topic</i> | <i>Page Number</i> |
|--------------------------------------|-------------------------------|
| DECLARATION | i |
| CERTIFICATE | ii |
| ACKNOWLEDGEMENT | iii |
| ABSTRACT | iv |
| 1. Chapter 1 : Introduction | 1 |
| 1.1 Raspberry Pi | 2 |
| 1.2 The Proper Peripherals | 6 |
| 1.3 Flash the SD Card & Boot up | 8 |
| 1.4 Configuring Pi | 8 |
| 1.5 Shutting Down | 11 |
| 2. Chapter 2 :Literature Review | 12 |
| 2.1. Getting Around Linux on the RPi | 13 |
| 2.1.1. Using the Command Line | 15 |
| 2.1.2. Files and the File system | 16 |
| 2.1.3. Sudo and Permissions | 19 |
| 2.1.4. Installing New Software | 20 |
| 2.2. Python on the Pi | 22 |
| 2.2.1. Hello Python | 23 |
| 2.2.2. Advantages of Python | 25 |

| | |
|--|----|
| 3. Artificial Intelligent Responder (A.I.R.) | 27 |
| 3.1.Introduction | 28 |
| 3.2.Block Diagram | 29 |
| 3.2.1. Automatic Noise Cut-off Mechanism | 30 |
| 3.2.2. Microphone | 30 |
| 3.2.3. Memory | 30 |
| 3.2.4. LCD display | 30 |
| 3.2.5. Software | 30 |
| 3.3. Speech recognition | 31 |
| 3.3.1. Factors on which Accuracy of SR depends | 32 |
| 3.4. Artificial Intelligence | 33 |
| 3.5. Extra Features Added in AIR v2.0 | 34 |
| 3.5.1. Language Translation | 34 |
| 3.5.2. Self-Power Backup | 35 |
| 3.5.3. Run Up to 10 hours | 35 |
| 3.5.4. Stability Increase | 35 |
| 3.5.5. Fully Portable | 35 |
| 3.5.6. A mini LAPTOP | 35 |
| References | 36 |
| Appendix | 38 |

Chapter 1

Introduction

1.1 Raspberry Pi

Let's start with a quick tour of the main Hardware used in the project: **The Raspberry Pi**.

We can think of the Raspberry Pi as a microcontroller development board like Arduino, or as a laptop replacement. In fact it is more like the exposed innards of a mobile device, with lots of maker-friendly headers for the various ports and functions. Figure 1-1 shows all the parts of the board, as described below.

- a) ***The Processor:*** At the heart of the Raspberry Pi is the same processor you would have found in the iPhone 3G and the Kindle 2, so you can think of the capabilities of the Raspberry Pi as comparable to those powerful little devices. This chip is a 32 bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. ARM chips come in a variety of architectures with different cores configured to provide different capabilities at different price points. The Model B has 512MB of RAM and the Model A has 256 MB. (The first batch of Model Bs had only 256MB of RAM.)

- b) ***The Secure Digital (SD) Card slot:*** You'll notice there's no hard drive on the Pi; everything is stored on an SD Card. One reason you'll want some sort of protective case sooner than later is that the solder joints on the SD socket may fail if the SD card is accidentally bent.

- c) ***The USB port:*** On the Model B there are two USB 2.0 ports, but only one on the Model A. Some of the early Raspberry Pi boards were limited in the amount of current that they could provide. Some USB devices can draw up 500mA. The original Pi board supported 100mA or so, but the newer revisions are up to the full USB 2.0 spec. One way to check your board is to see if you have two polyfuses limiting the current (see [Figure 1-2](#)). In any case, it is probably not a good idea to charge your cell phone with the Pi. You can use a powered external hub if you have a peripheral that needs more power.

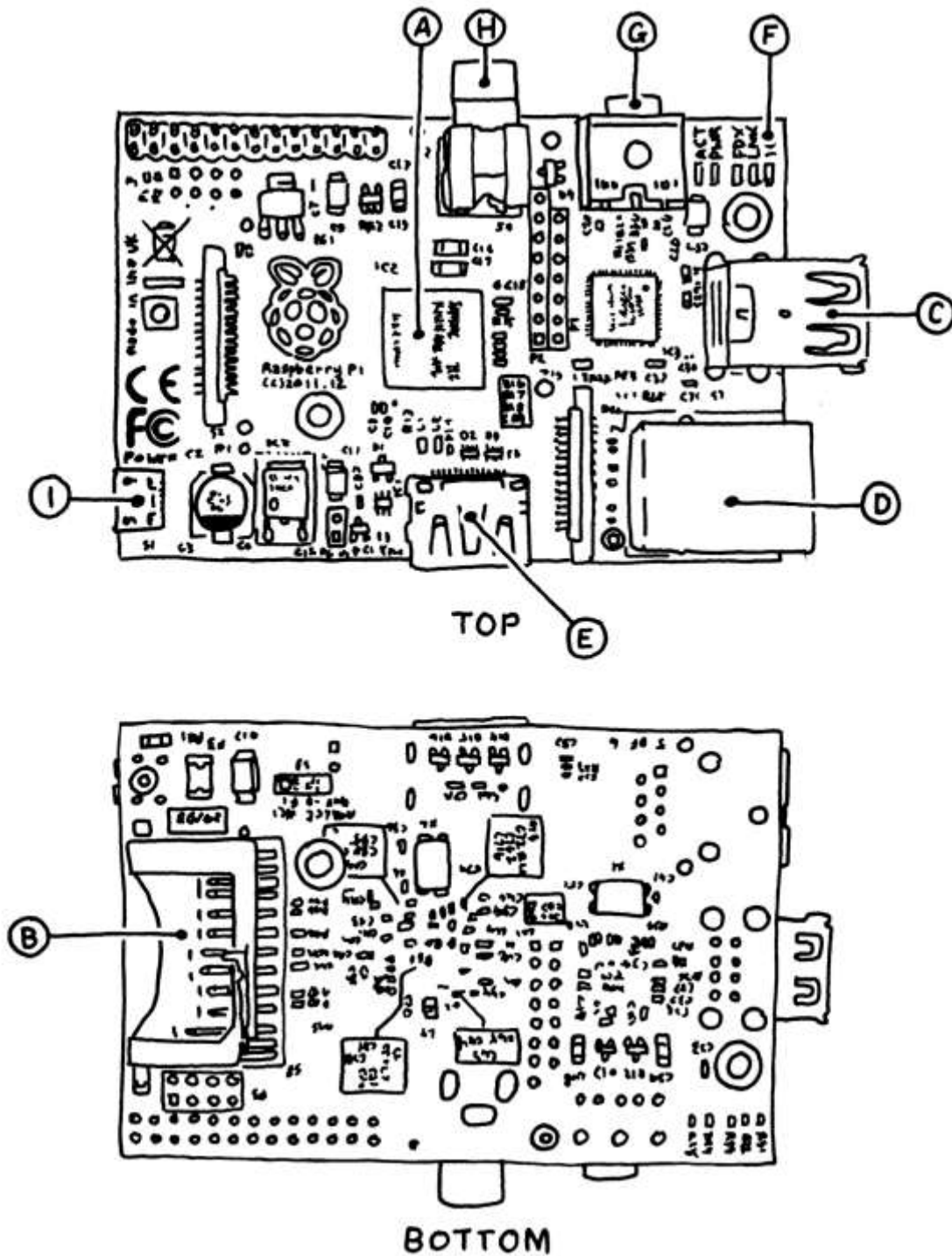


Figure 1-1. A map of the hardware interface of the Raspberry Pi

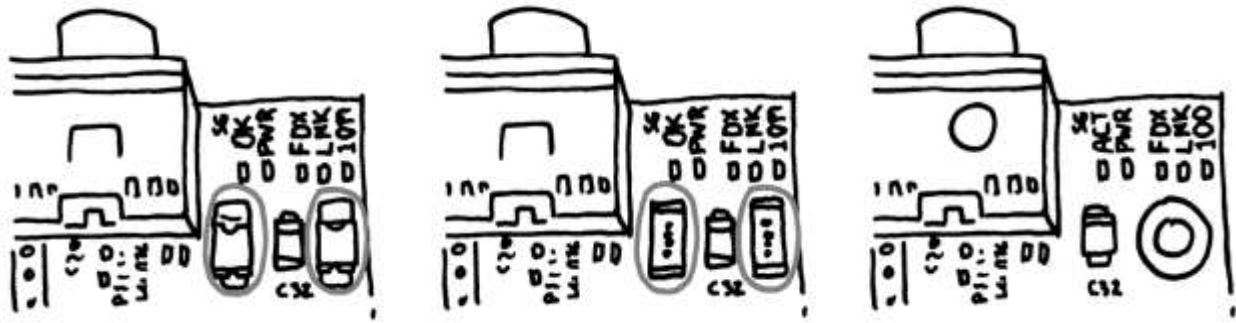


Figure 1-2. Some of the older boards came equipped with polyfuses (left) to protect the USB hub. Some boards have the polyfuses replaced with jumpers (center), and the latest revision of the Model B removed them and uses the space for a mounting hole (right).

- d) **Ethernet port.** The model B has a standard RJ45 Ethernet port. The Model A does not, but can be connected to a wired network by a USB Ethernet adapter (the port on the Model B is actually an onboard USB to Ethernet adapter). Wi-Fi connectivity via a USB dongle is another option.
- e) **HDMI connector.** The HDMI port provides digital video and audio output. 14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.
- f) **Status LEDs^{5 h}.** The Pi has five indicator LEDs that provide visual feedback (Table 1-1).
- g) **Analog Audio output.** This is a standard 3.5mm mini analog audio jack, intended to drive high impedance loads (like amplified speakers). Headphones or unpowered speakers won't sound very good; in fact, as of this writing the quality of the analog output is much less than the HDMI audio output you'd get by connecting to a TV over HDMI. Some of this has to do with the audio driver software, which is still evolving.
- h) **Composite video out.** This is a standard RCA-type jack that provides composite NTSC or PAL video signals. This video format is extremely low-resolution compared to HDMI. If you have a HDMI television or monitor, use it rather than a composite television.

Table 1-1. *The five status LEDs.*

| | | |
|-----|--------|---|
| ACT | Green | Lights when the SD card is accessed (marked OK on earlier boards) |
| PWR | Red | Hooked up to 3.3V power |
| FDX | Green | On if network adapter is full duplex |
| LNK | Green | Network activity light |
| 100 | Yellow | On if the network connection is 100Mbps (some early boards have a 10M misprint) |

- i) **Power input.** One of the first things you'll realize is that there is no power switch on the Pi. This microUSB connector is used to supply power (this isn't an additional USB port; it's only for power). MicroUSB was selected because the connector is cheap USB power supplies are easy to find.

Figure 1-3 shows all of the power and input/output (IO) pins on the Raspberry Pi, which are explained next.

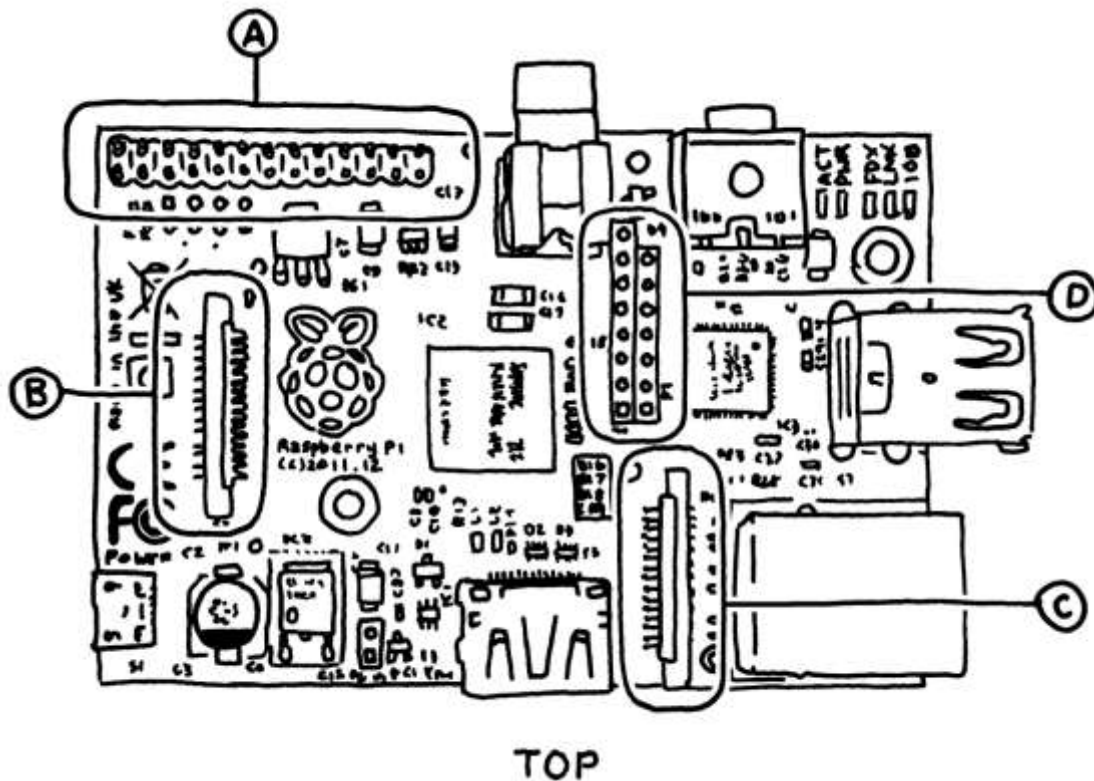


Figure 1-3. The Pins and headers on the Raspberry Pi

- a) ***General Purpose Input and Output (GPIO) and other pins.*** We use these pins to read buttons and switches and control actuators like LEDs, relays, or motors.
- b) ***The Display Serial Interface (DSI) connector.*** This connector accepts a 15 pin flat ribbon cable that can be used to communicate with a LCD or OLED display screen.
- c) ***The Camera Serial Interface (CSI) connector.*** This port allows a camera module to be connected directly to the board.
- d) ***P2 and P3 headers.*** These two rows of headers are the JTAG testing headers for the Broadcom chip (P2) and the LAN9512 networking chip (P3). Because of the proprietary nature of the Broad

1.2 The Proper Peripherals

Now that we knew where everything is on the board, we needed to know a few things about the proper peripherals (some are shown in Figure 1-4) to use with the Pi. There are a bunch of prepackaged starter kits that have wellvetted parts lists; there are a few caveats and gotchas when fitting out your Raspberry Pi. There's a definitive list of supported peripherals on the main wiki.

1. ***A power supply.*** This is the most important peripheral to get right; you should use a microUSB adapter that can provide 5V and at least 700mA of current (500mA for the Model A). A cell phone charger won't cut it, even if it has the correct connector. A typical cell phone charger only provides 400mA of current or less, but check the rating marked on the back. An underpowered Pi may still seem to work but will be flaky and may fail unpredictably.

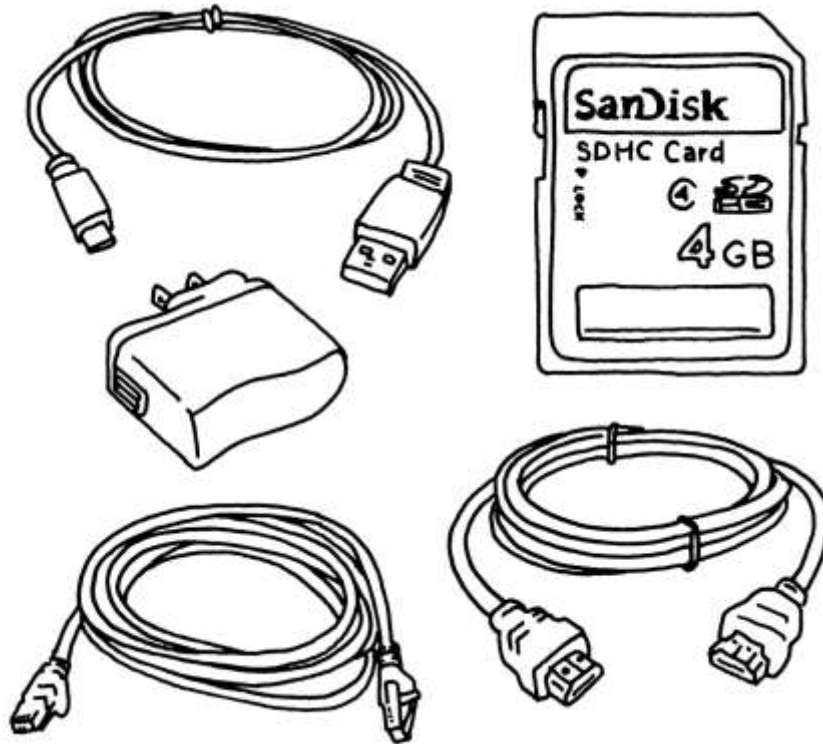


Figure 1-4. The basic peripherals: a microUSB power supply, cables, and SD card. We needed at least a 4GB Class 4 SD Card (MicroSD cards with an adapter are ok to use as well).

2. **An SD Card.** We needed at least 4GB, and it should be a Class 4 card. Class 4 cards are capable of transferring at least 4MB/sec. Some of the earlier Raspberry Pi boards had problems with Class 6 or higher cards, which are capable of faster speeds but are less stable. A microSD card in an adapter is perfectly usable as well.
3. **An HDMI cable.** If you're connecting to a monitor you'll need this, or an appropriate adapter for a DVI monitor
4. **Ethernet cable.** Need to connect to Internet? Just plug in the Ethernet cable into LAN and connect it to Pi.

1.3 Flash the SD Card & Boot up

Many vendors sell SD cards with the operating system pre-installed; for some people this may be the best way to get started. Even if it isn't the latest release you can easily upgrade once you get the Pi booted up and on the Internet.

Raspbian also has a network installer. To use this tool, you need to put the installer files on a SD Card (formatted as FAT32, which is typical for these cards) and then boot up the Pi with the card inserted. The catch is that you'll need to be connected to the Internet for this to work.

Follow these steps to boot up your Raspberry Pi for the first time:

1. Plug the SD card into the socket.
2. Plug in a USB keyboard and mouse. On the Model A, plug them into a powered hub, then plug the hub into the Pi.
3. Plug the HDMI output into your TV or monitor. Make sure your monitor is on.
4. Plug in the power supply. In general, try to make sure everything else is hooked up before connecting the power.

The very first time you boot up you'll be presented with a few the raspi-config tool (see Figure 1-8). There are a few key settings you'll need to tweak here; chances are good that your Raspberry Pi won't work exactly the way you want right out of the box. If you need to get back to this configuration tool at any time by typing the following at the command line:

sudo raspi-config

1.4 Configuring Pi

Next we'll walk through the steps and show which configuration options are essential and which you can come back to if you need them. When setting options in the tool, use the up and down arrows to move around the list, the space bar to select something, and tab to change fields or

move the cursor to the buttons at the bottom of the window. Let's go in the order of the menu options in the configuration tool:

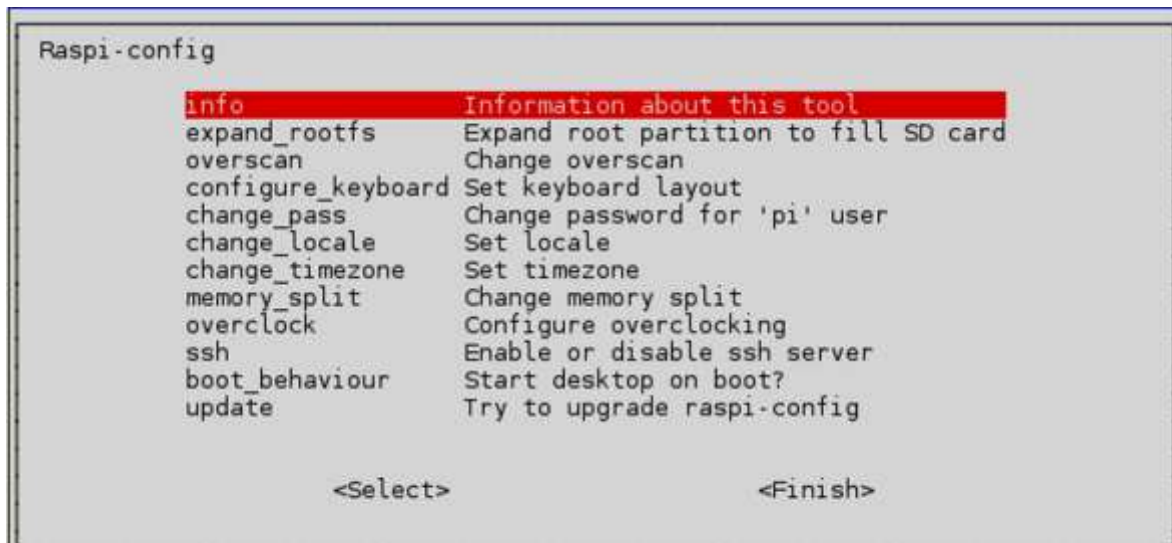


Figure 1-5. The Raspi-config tool menu

- 1. Expand rootfs** You should always choose this option; this will enlarge the filesystem to let you use the whole SD card.
- 2. Overscan** Leave the overscan option disabled at first. If you have a high definition monitor you may find that text runs off the side of the screen. To fix this enable the overscan and change the values to fit the image to the screen. The values indicate the amount of overscan so the display software can correct; use positive values if the image goes off the screen, negative if there are black borders around the edge of the display.
- 3. Keyboard** The default keyboard settings are for a generic keyboard in a UK-style layout. If you want the keys to do what they're labeled to do, you'll definitely want to select a keyboard type and mapping that corresponds to your setup. Luckily the keyboard list is very robust. Note that your locale settings can affect your keyboard settings as well.
- 4. Password** It's a good idea to change the default password from raspberry to something a little stronger.

5. **Change Locale** If you're outside the UK you should change your locale to reflect your language and character encoding preferences. The default setting is for UK English with a standard UTF-8 character encoding (en_GB.UTF-8). Select en_US.UTF-8 if you're in the US.
6. **Memory split** This option allows you to change the amount of memory used by the CPU and the GPU. Leave the default split for now.
7. **Overclock** You now have the option of running the processor at speeds higher than 700MHz with this option. For your first time booting, leave the default settings or try Medium or Modest. You may want to return to this later (Turbo mode can run at 1000MHz).
8. **SSH** This option turns on the Secure Shell (ssh) server, which will allow you to login to the Raspberry Pi remotely over a network. This is really handy, so you should turn it on.
9. **Desktop Behavior** This option lets you boot straight to the graphical desktop environment and is set to Yes by default. If you select No, you'll get the command line when you boot up and you'll have to login and start the graphical interface manually like this:

```
raspberrypi login: pi  
Password: raspberry  
pi@raspberrypi ~ $ startx
```

When you're in the graphical desktop, your command prompt will disappear.

You can open a terminal program to get a command prompt while you're in the graphical desktop. Click the desktop menu in the lower left, and then choose Accessories→LXTerminal.

10. **Update** Finally, if you're connected to the Internet you'll be able to update the conifg utility with this option. When you're done, select Finish and you'll be dumped back to the command line. Type:

```
pi@raspberrypi ~ $ sudo reboot
```

And your Pi will reboot with your new settings. If all goes well (and if you chose the option to boot straight to the graphical desktop environment) you should see the Openbox window manager running on the Lightweight X11 Desktop Environment (LXDE). You're off and running!

1.5 Shutting Down

There's no power button on the Raspberry Pi (although there is a header for a reset switch on newer boards). The proper way to shutdown is through the Logout menu on the graphical desktop; select Shutdown to halt the system. You can also shut down from the command line by typing:

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

Be sure to do a clean shutdown (and don't just pull the plug). In some cases you can corrupt the SD card if you turn off power without halting the system.

Chapter 2

Literature Review

2.1 Getting Around Linux on the RPi

Raspbian comes with the Lightweight X11 Desktop Environment (LXDE) graphical desktop environment installed. This is a trimmed-down desktop environment for the X Window System that has been powering the GUIs of Unix and Linux computers since the 80s. Some of the tools you see on the Desktop and in the menu are bundled with LXDE (the Leafpad text editor and the LXTerminal shell, for instance).

Running on top of LXDE is Openbox, a window manager that handles the look and feel of windows and menus. If you want to tweak the appearance of your desktop, go to the Openbox configuration tools (click the desktop menu in the lower left, then choose Other→Openbox Configuration Manager). Unlike OS X or Windows, it is relatively easy to completely customize your desktop environment or install alternate window managers. Some of the other distributions for Raspberry Pi have different environments tuned for applications like set top media boxes, phone systems or network firewalls.

The File Manager

If you prefer not to move files around using the command line (more on that in a moment), select the File Manager from the Accessories menu. You'll be able to browse the filesystem using icons and folders the way you're probably used to doing.

The Web Browser

The default web browser is Midori, designed to work well with limited resources. It's easy to forget how much work web browsers do these days. Because Raspbian is designed to be a very lightweight OS distribution, there are a number of features you may expect in a web browser that are not available. For example Flash and the Java plugin is not installed (so no YouTube), and Midori does not support HTML5 video. Later we'll look at how to install new software (like Java). Look for tools and menu items in the pull-down menu in the upper right corner of the

window (see Figure 2-2). There are a couple of other browser options, notably NetSurf and Dillo.

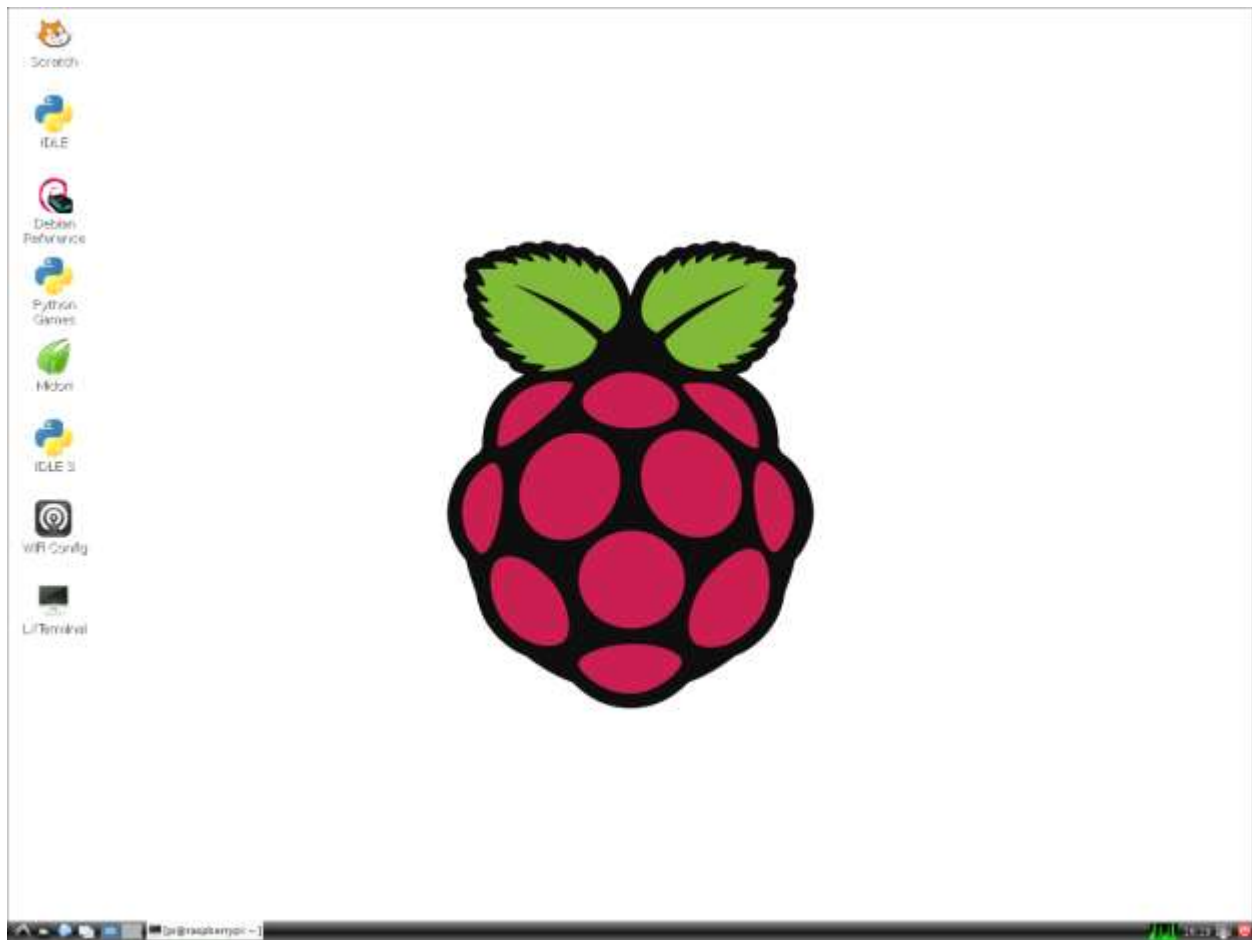


Figure 2-1. The graphical desktop

Video and Audio

Multimedia playback is handled by omxplayer, which is a bit experimental as of this writing. It is only available as a command line utility. Omxplayer is specially designed to work with the Graphics Processing Unit (GPU) on the processor; other free software like VLC and mPlayer won't work well with the GPU.

Text Editor

Leafpad is the default text editor, which is available under the main menu. You've also got Nano, which is an easy-to-learn bare bones text editor. Traditional Unix text editors like vim or emacs are not installed by default, but can be easily added

Copy and Paste

Copy and paste functions work between applications pretty well, although you may find some oddball programs that aren't consistent. If you have a middle button on your mouse you can select text by highlighting it as you normally would (click and drag with the left mouse button) and paste it by pressing the middle button while you have the mouse cursor over the destination window.

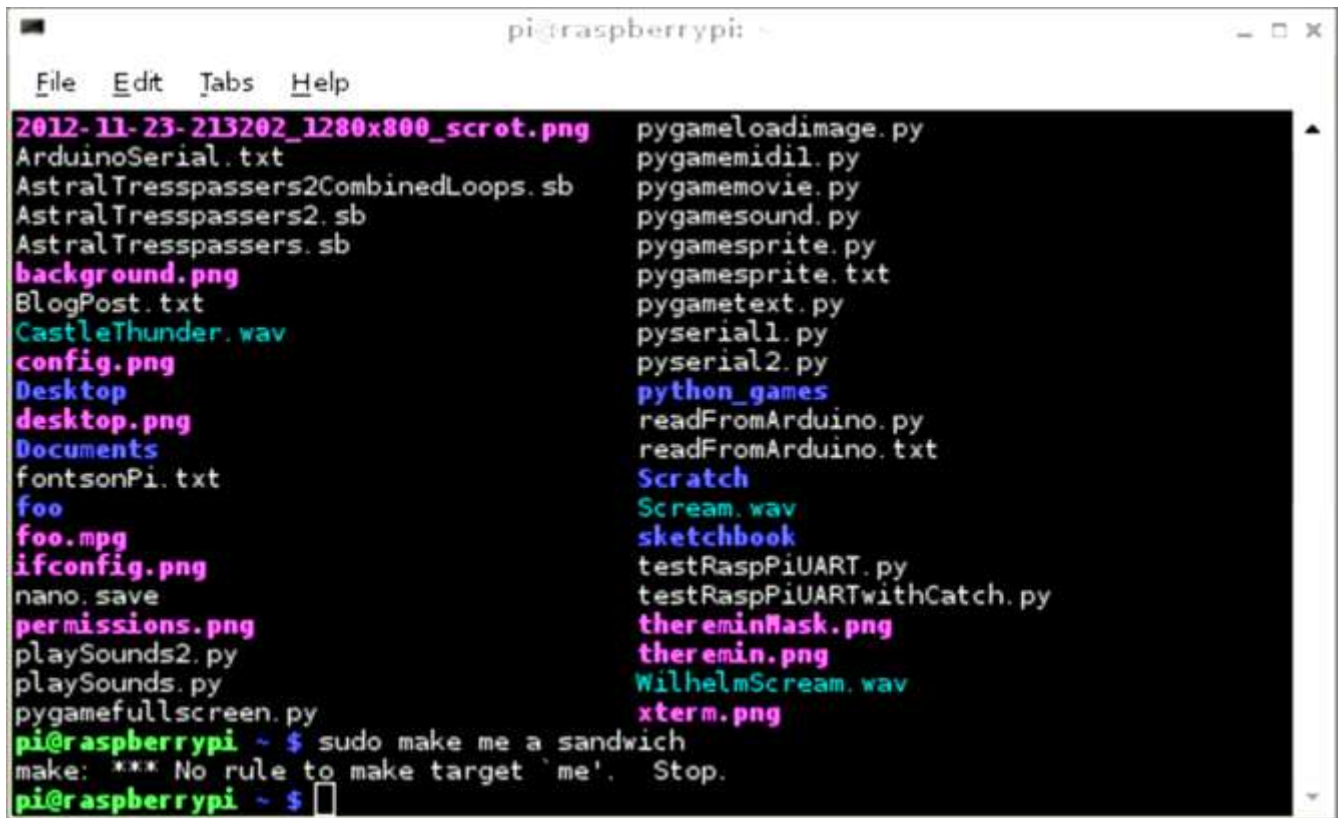
The Shell

A lot of tasks are going to require you to get to the command line and run commands there. The LXTerminal program provides access to the command line or shell. The default shell on Raspbian is the Bourne Again Shell (bash), which is very common on Linux systems. There's also an alternative called dash. You can change shells via the program menu, or with the `chsh` command.

2.1.1 Using the Command Line

If it helps, you can think of using the command line as playing a text adventure game, but with the files and the filesystem in place of Grues and mazes of twisty passages. If that metaphor doesn't help you, don't worry; all the commands and concepts in this section are standard Linux and are valuable to learn.

Before you start, open up the LXTerminal program (Figure 2-3). There are two tricks that make life much easier in the shell: autocomplete and command history. Often you will only need to type the first few characters of a command or filename, then hit `tab`. The shell will attempt to autocomplete the string based on the files in the current directory or programs in commonly used directories (the shell will search for executable programs in places like `/bin` or `/usr/bin/`). If you hit the up arrow on the command line you'll be able to step back through your command history, which is useful if you mistyped a character in a long string of commands.



```
pi@raspberrypi ~  
File Edit Tabs Help  
2012-11-23-213202_1280x800_screenshot.png  pygameloadimage.py  
ArduinoSerial.txt                          pygamemidil.py  
AstralTresspassers2CombinedLoops.sb         pygamemovie.py  
AstralTresspassers2.sb                     pygamesound.py  
AstralTresspassers.sb                      pygamesprite.py  
background.png                             pygamesprite.txt  
BlogPost.txt                              pygametext.py  
CastleThunder.wav                         pyserial1.py  
config.png                               pyserial2.py  
Desktop                                  python_games  
desktop.png                             readFromArduino.py  
Documents                               readFromArduino.txt  
fontsonPi.txt                           Scratch  
foo                                     Scream.wav  
foo.mpg                                sketchbook  
ifconfig.png                           testRaspPiUART.py  
nano.save                             testRaspPiUARTwithCatch.py  
permissions.png                       thereminMask.png  
playSounds2.py                       theremin.png  
playSounds.py                         WilhelmScream.wav  
pygamefullscreen.py                   xterm.png  
pi@raspberrypi ~ $ sudo make me a sandwich  
make: *** No rule to make target `me'. Stop.  
pi@raspberrypi ~ $
```

Figure 2-2 LXTerminal gives you access to the command line (or shell).

2.1.2 Files and the File system

Table 2-1 shows some of the important directories in the filesystem. Most of these follow the Linux standard where files should go; a couple are specific to the Raspberry Pi. The /sys directory is where you can access all of the hardware on the Raspberry Pi.

You'll see your current directory displayed before the command prompt. In Linux your home directory has a shorthand notation: the tilde (~). When you open the LXTerminal you'll be dropped into your home directory and your prompt will look like this:

```
pi@raspberrypi ~ $
```

Here's an explanation of that prompt:

pi@ raspberrypi ~ \$

- Your username, pi, followed by the at (@) symbol.
- The name of your computer (raspberrypi is the default host name).
- The current working directory of the shell. You always start out in your home directory (~).
- This is the shell prompt. Any text you type will appear to the right of it. Press Enter or Return to execute each command you type.

Use the cd (change directory) command to move around the filesystem. The following two commands have the same effect (changing to the home directory) for the pi user:

cd /home/pi/

cd ~

If the directory path starts with a forward slash it will be interpreted as an absolute path to the directory. Otherwise the directory will be considered relative to the current working directory. You can also use . and .. to refer to the current directory and the current directory's parent. For example, to move up to the top of the filesystem:

pi@raspberrypi ~ \$ cd ..

pi@raspberrypi /home \$ cd ..

You could also get there with the absolute path /:

pi@raspberrypi ~ \$ cd /

Once you've changed to a directory, use the ls command to list the files there.

Table 2.1: Some of the most important directories in Raspbian file system.

| Directory | Description |
|--------------|--|
| / | |
| /bin | Programs and commands that all users can run |
| /boot | All the files needed at boot time |
| /dev | Special files that represent the devices on your system |
| /etc | Configuration files |
| /etc/init.d | Scripts to start up services |
| /etc/X11 | X11 configuration files |
| /home | User home directories |
| /home/pi | Home directory for pi user |
| /lib | Kernel modules/drivers |
| /media | Mount points for removable media |
| /proc | A virtual directory with information about running processes and the OS |
| /sbin | Programs for system maintenance |
| /sys | A special directory on the Raspberry Pi that represents the hardware devices |
| /tmp | Space for programs to create temporary files |
| /usr | Programs and data usable by all users |
| /usr/bin | Most of the programs in the operating system reside here |
| /usr/games | Yes, games |
| /usr/lib | Libraries to support common programs |
| /usr/local | Software that may be specific to this machine goes here |
| /usr/sbin | More system administration programs |
| /usr/share | Things that are shared between applications like icons or fonts |
| /usr/src | Linux is open source; here's the source! |
| /var | System logs and spool files |
| /var/backups | Backup copies of all the most vital system files |
| /var/cache | Any program that caches data (like apt-get or a web browser) stores it here. |
| /var/log | All of the system logs and individual service logs |
| /var/mail | All user email is stored here, if you're set up to handle email |
| /var/spool | Data waiting to be processed (e.g. incoming email, print jobs) |

2.1.3 Sudo and Permissions

Linux is a multiuser operating system; the general rule is that everyone owns their own files and can create, modify, and delete them within their own space on the filesystem. The root (or super) user can change any file in the filesystem, which is why it is good practice to not log in as root on a day-to-day basis.

There are some tools like `sudo` that allow users to act like super users for performing tasks like installing software without the dangers (and responsibilities) of being logged in as root. You'll be using `sudo` a lot when interacting with hardware directly, or when changing system-wide configurations.

Each file belongs to a particular user and a particular group. Use the `chown` and `chgrp` commands to change the owner or group of a file. Note that you must be root to use either of these two commands:

```
pi@raspberrypi ~ $ sudo chown pi garply.txt
pi@raspberrypi ~ $ sudo chgrp staff plugh.txt
```

Each file also has a set of permissions that show whether a file can be read, written or executed. These permissions can be set for the owner of the file, the group, or for everyone (see Figure 2-5)

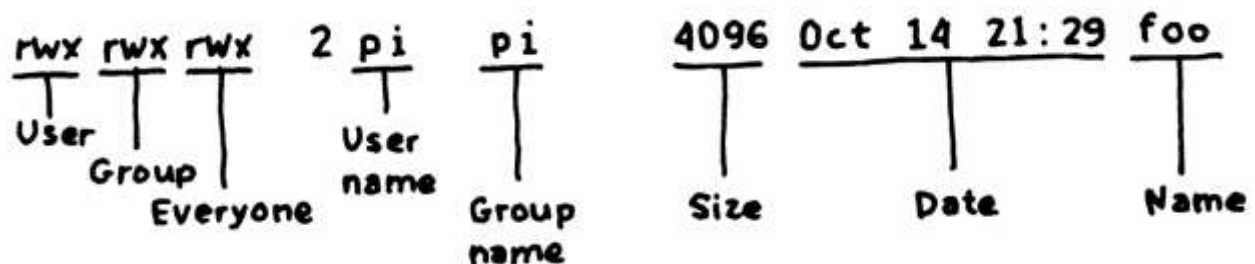


Figure 2-3. File permissions for owner, group, and everyone.

You set the individual permissions with the `chmod` command. The switches for `chmod` are summarized in Table 2-2.

Table 2-2. *The switches that can be used with `chmod`.*

| | |
|---|-------------------------|
| u | user |
| g | group |
| o | others not in the group |
| a | all/everyone |
| r | read permission |
| w | write permission |
| x | execute permission |
| + | add permission |
| - | remove permission |

Here are a few examples of how you can combine these switches :

- a) ***`chmod u+rx,g-rwx,o-rwx wibble.txt`***
- b) ***`chmod g+wx wobble.txt`***
- c) ***`chmod -rw,+r wubble.txt`***

- a) Allow only the user to read, write, execute
- b) Add permission to read and execute to entire group
- c) Make read only for everyone

The only thing protecting your user space and files from other people is your password, so you better chose a strong one. Use the ***passwd*** command to change it, especially if you're putting your Pi on a network.

2.1.4 Installing New Software

One of the areas that Linux completely trounces other operating systems is in software package management. Package managers handle the downloading and installation of software, and they automatically handle downloading and installing dependencies. Keeping with the modular approach, many software packages on Linux depend on other pieces of software. The package manager keeps it all straight, and the package managers on Linux are remarkably robust.

Raspbian comes with a pretty minimal set of software, so you will soon want to start downloading and installing new programs. The examples in this book will all use the command line for this, since it is the most flexible and quickest way of installing software.

The program `apt-get` with the `-install` switch is used to download software. `apt-get` will even download all of the other software that your package requires so you don't have to go hunting around for dependencies. Software has to be installed with superuser permissions, so always use `sudo` (this command installs the Emacs text editor):

```
pi@raspberrypi ~ $ sudo apt-get install emacs
```

2.2 Python on the Pi

Python is an interpreted language, which means that you can write a program or script and execute it directly rather than compiling it into machine code. Interpreted languages are a bit quicker to program with, and you get a few side benefits. For example, in Python you don't have to explicitly tell the computer whether a variable is a number, a list, or a string; the interpreter figures out the data types when you execute the script.

The Python interpreter can be run in two ways: as an interactive shell to execute individual commands, or as a command line program to execute standalone scripts. The integrated development environment (IDE) bundled with Python and the Raspberry Pi is called IDLE (see Figure 3-1).

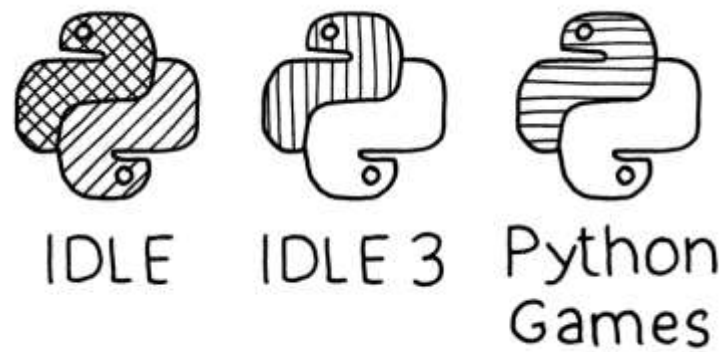


Figure2.4. Python options on the Raspbian Desktop: The IDLE integrated development environment for Python 2.0 (left), IDLE for Python 3.0 (middle), and a collection of sample games implemented in Pygame from invent-withpython.com (right).

2.2.1 Hello Python

The best way to start learning Python is to jump right in. Although you can use any text editor to start scripting, we'll start out using the IDE. Open up the IDLE 3 application. To run IDLE, double-click the IDLE 3 icon on the desktop, or click the desktop menu in the lower left, and choose Programming→ IDLE 3.

IDLE can take several seconds to start up, but when it appears, you'll see a window with the interactive shell. The triple chevron (`>>>`) is the interactive prompt; when you see the prompt, it means the interpreter is waiting for your commands. At the prompt type:

```
>>> print("Gaganpreet")
```

And hit Enter or Return. Python executes that statement and you'll see the result in the shell window. Note that the `print()` command is one of the things that changed in Python 3.0; if you get a syntax error, check to make sure you're running the 3.0 version of IDLE.

You can use the shell as a kind of calculator to test out statements or calculations:

```
>>> 3+4+5
12
```

Think of the statements executed in the interactive shell as a program that you're running one line at a time. You can set up variables or import modules:

```
>>> import math
>>> (1 + math.sqrt(5)) / 2
1.618033988749895
```

The import command makes all of Python's math functions available to your program. To set up a variable, use the assignment operator (=):

```
>>> import math
>>> radius = 200
>>> radius * 2 * math.pi
125.66370614359173
```

If you want to clear all variables and start in a fresh state, select Shell→Restart Shell to start over. You can also use the interactive shell to get information about how to use a particular statement, module, or other Python topics with the help() command:

```
help("print")
```

To get a listing of all of the topics available, try:

```
help("topics")
help("keywords")
help("modules")
```

The Python interpreter is good for testing statements or simple operations, but you will often want to run your Python script as you would a standalone application. To start a new Python program, select File→New Window, and IDLE will give you a script editing window (see Figure 3-2).

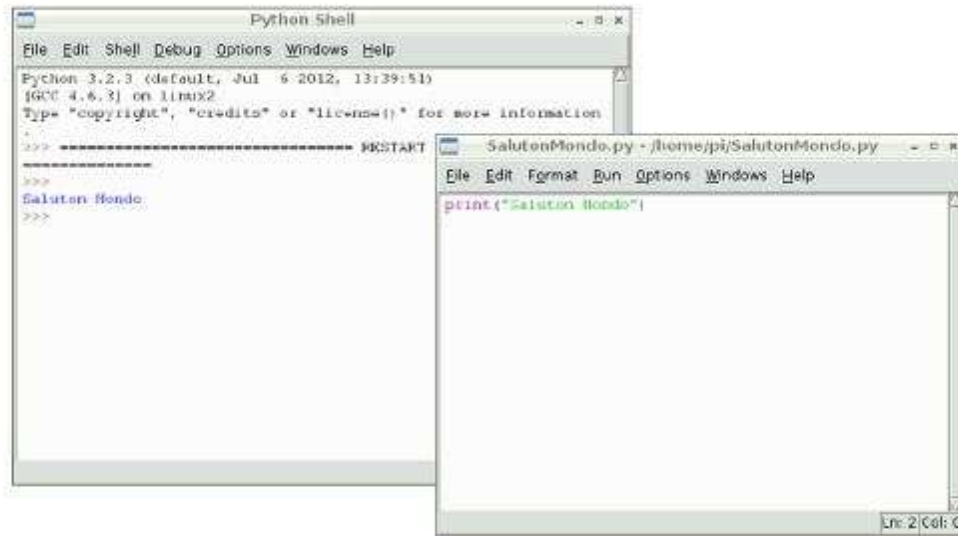


Figure 2.5 IDLE Interactive Shell

Try typing a line of code and selecting Run→Run Module. You’ll get a warning that “Source Must Be Saved OK To Save?”. Save your script in your home directory as `SalutonMondo.py` and you’ll see it execute in the shell.

2.2.2 Advantages of Python

Python is a high level, object-oriented and interpreted programming language for the web. It is also regarded as a strong server side scripting language. Python code resembles the pseudo code just like all the scripting languages. The elegant design and syntax rules of this programming language make it quite readable even among the multi programmer development teams. The Python language hardly provides a rich syntax, which is quite useful. The whole idea behind that is to keep you thinking regularly about the different business rules of your application and not to spend any time trying to figure out which command you should use.

The advantages of Python are:

1. **Readability** The syntax of Python is readable and clear. The way it is organized, it imposes some order to programmers. Beginners and experts can easily understand the code and

everyone can become quite productive in Python very rapidly. It is quite essential to mention that Python has lesser “dialects” than other popular languages, such as Perl.

It Is Straightforward to Get Support The community of Python always offers good amount of support to the Python users. As we are already aware of the fact that Python code is freely accessible for everyone and therefore millions of developers all over the world are functioning hard to find bugs and craft patches to fix all those bugs. Several individuals are crafting fresh enhancements to the language and sending them for approval.

2. ***Quick to Learn*** the Python programming language is quite simple to learn as its source code resembles the pseudo code. Here, you don’t need to train yourself too much. You can get pretty quick results without actually wasting too much of your time. As soon as you start learning the Python language, you can carry out helpful coding almost immediately. After you gain some practice, your productivity is going to enhance a great deal. You can even design an object- oriented and high level programming code. For small tasks, this is a great feature.
3. ***Quick to Code*** Python programming language offers quick feedback in a lot of ways. First of all, the programmer can easily skip several tasks that other languages want him to take. The cost of program maintenance comes down eventually. Python even permits a quick adaptation of the code. This language can simply be termed as a ready-to-run language, which just requires a simple code to be executed. Testing and playing around with your programming code becomes quite simple with Python. This language even offers a bottom up development style in which you can easily construct your applications by testing and importing crucial functions in the interpreter before you write the top-level code, which calls all the functions. Not many people know that the interpreter is easily extensible. It allows you to embed your favorite C code as a simple yet compiled extension module.
4. ***Reusability*** Python motivates the program reusability by carefully implementing packages and modules. A huge set of modules has already been innovated and is offered as the Standard Python Library, which is an essential part of the Python distribution. You can

simply share the functionality between different programs just by breaking them into several modules and reusing them as components of other specific programs.

5. ***Portability*** Python not just runs on multiple systems, but it even has a similar interface on different platforms. The design of this language isn't particularly attached to a certain operating system as it is written in portable ANSI C. It implies that you can easily write a Python program on a MAC operating system, test it on a Linux operating system, and upload it easily to a Windows system.
6. ***Object-oriented Programming*** Normally, scripting languages have different object-orientation support in the programming language. It works as a simple add-on. However, everything in this language is specially designed to be object oriented. You can get started with the programming process by making use of non-OO structures. Object oriented programming is not just simple, but quite beneficial as well.

Chapter 3

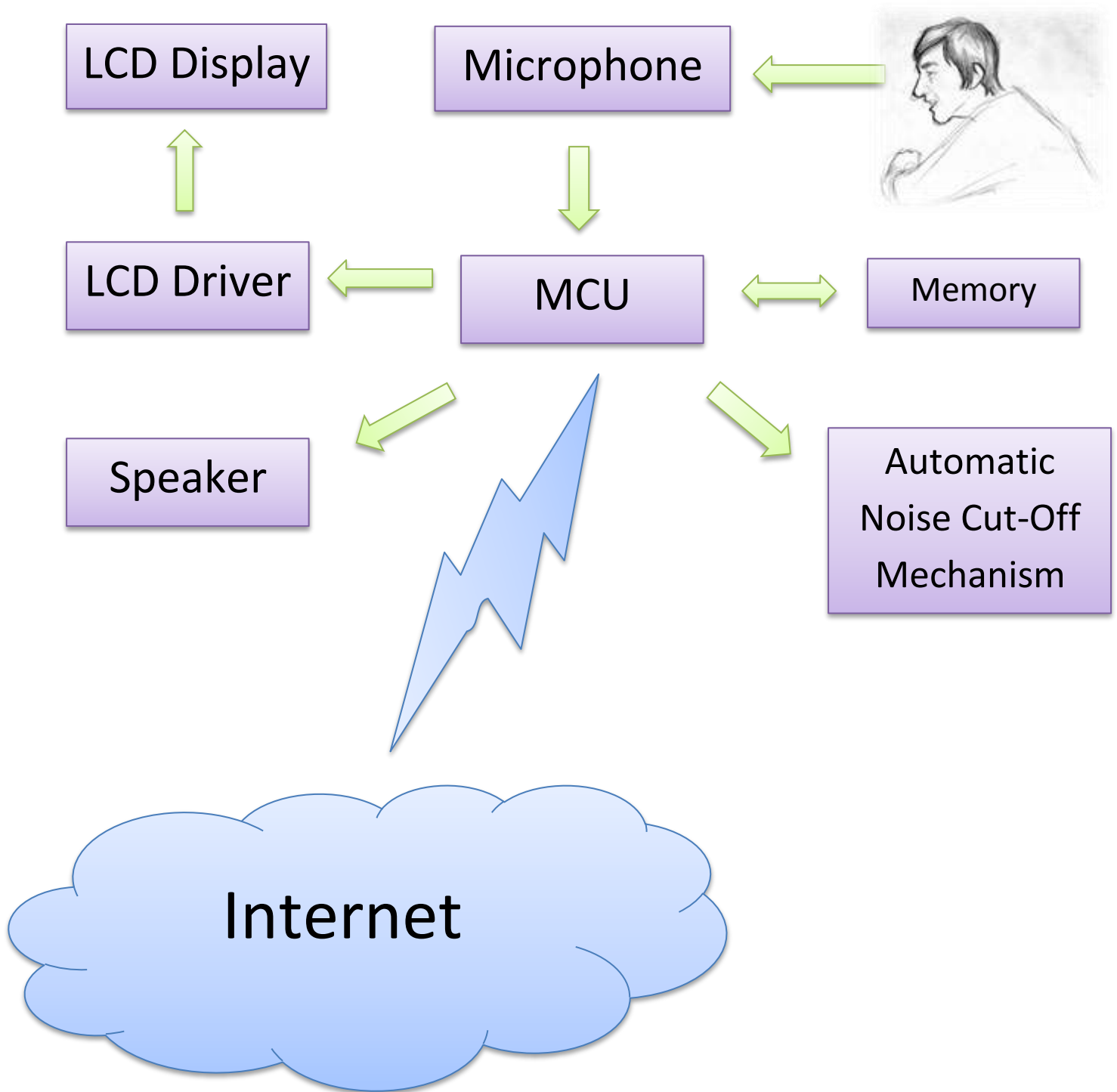
Artificial Intelligent Responder (A.I.R v2.0)

3.1 Introduction

The main aim to Develop AIR is to provide a Virtual Tutor (Artificial intelligent capability for children/Adult, where we can strive towards providing high quality education to children. Now there is no need to worry about the children education and homework. No need to turn on your PC/Laptop to search for solutions for any Query, Just ask to AIR and it'll be always ready to search the answer over the internet and convert it to audio Speech within 5-6 seconds (with 1 Mbps internet connection) Queries that have been tested successfully :-

1. Mathematics
2. Statistics & data analysis
3. Words & linguistic problem
4. Units and measurement
5. Dates and times
6. People and history
7. Money and Finance
8. Chemistry
9. Culture
10. Art
11. Engineering
12. Places and geography
13. Shopping
14. Sports
15. Games
16. Astronomy and Much More!!

3.2 Block Diagram



3.2.1 Automatic Noise Cut-off Mechanism

When the user stops speaking the query to AIR, it automatically detects that and stops listening .Also whenever user pauses in between , the mechanism detects the silent period and removes it from the Query Clip.

3.2.2 Microphone:

It is input to the MCU which input the voice from real world and send it to the MCU after converting into electrical signal. Children/adult can ask the question from this microphone

3.2.3 Memory:

It saves the Images, Greeting text, Question and its answer for the future reference

3.2.4 LCD display:

Display the result on LCD Display so that if the audio output is not audible or understandable. A person can use this

3.2.5 Software:

The CPU is interrupted after getting signal from sensor. After the several processing are done which is given below:-

1. Greet the person
2. Take the query from the person
3. Convert it to text
4. Scan the internet using this text with various API calling and algorithm
5. After getting result, Display it on the LCD as well as convert it to Audio for Audio Output
6. And waiting for next query if any.

3.3 Speech recognition

In computer science and electrical engineering, **speech recognition** (SR) is the translation of spoken words into text. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT).

Some SR systems use "speaker-independent speech recognition" while others use "training" where an individual speaker reads sections of text into the SR system. These systems analyze the person's specific voice and use it to fine-tune the recognition of that person's speech, resulting in more accurate transcription. Systems that do not use training are called "speaker-independent" systems. Systems that use training are called "speaker-dependent" systems.

Speech recognition applications include voice user interfaces such as voice dialling (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), domotic appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed Direct Voice Input).

The term *voice recognition* or *speaker identification* refers to identifying the speaker, rather than what they are saying. Recognizing the speaker can simplify the task of translating speech in systems that have been trained on a specific person's voice or it can be used to authenticate or verify the identity of a speaker as part of a security process.

3.3.1 Factors on which Accuracy of SR depends

As mentioned earlier in this article, accuracy of speech recognition varies in the following:

✚ Error rates increase as the vocabulary size grows:

e.g. The 10 digits "zero" to "nine" can be recognized essentially perfectly, but vocabulary sizes of 200, 5000 or 100000 may have error rates of 3%, 7% or 45% respectively.

✚ Vocabulary is hard to recognize if it contains confusable words:

e.g. The 26 letters of the English alphabet are difficult to discriminate because they are confusable words (most notoriously, the E-set: "B, C, D, E, G, P, T, V, Z"); an 8% error rate is considered good for this vocabulary.

✚ Speaker dependence vs. independence:

A speaker-dependent system is intended for use by a single speaker.

A speaker-independent system is intended for use by any speaker, more difficult.

✚ Isolated, Discontinuous or continuous speech

With isolated speech single words are used, therefore it becomes easier to recognize the speech.

With discontinuous speech full sentences separated by silence are used, therefore it becomes easier to recognize the speech as well as with isolated speech.

With continuous speech naturally spoken sentences are used, therefore it becomes harder to recognize the speech, different from both isolated and discontinuous speech.

✚ Task and language constraints

e.g. Querying application may dismiss the hypothesis "The apple is red."

e.g. Constraints may be semantic; rejecting "The apple is angry."

e.g. Syntactic; rejecting "Red is apple the."

Constraints are often represented by a grammar.

✚ Read vs. Spontaneous Speech

When a person reads it's usually in a context that has been previously prepared, but when a person uses spontaneous speech, it is difficult to recognize the speech because of the disfluencies (like "uh" and "um", false starts, incomplete sentences, stuttering, coughing, and laughter) and limited vocabulary.

✚ Adverse conditions

Environmental noise (e.g. Noise in a car or a factory), Acoustical distortions (e.g. echoes, room acoustics) Speech recognition is a multi-levelled pattern recognition task.

- ✚ Acoustical signals are structured into a hierarchy of units;
e.g. Phonemes, Words, Phrases, and Sentences;
- ✚ Each level provides additional constraints;
e.g. Known word pronunciations or legal word sequences, which can compensate for errors or uncertainties at lower level;

3.4 Artificial intelligence

Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is an academic field of study which studies the goal of creating intelligence. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success John McCarthy, who coined the term in 1955, defines it as "the science and engineering of making intelligent machines".

AI research is highly technical and specialized, and is deeply divided into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues.

General intelligence is still among the field's long term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, mathematics, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology.

Some subfields focus on the solution of specific problems. Others focus on one of several possible approaches or on the use of a particular tool or towards the accomplishment of particular applications.

The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects. General intelligence is still among the field's long term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, mathematics, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology.

The field was founded on the claim that a central property of humans, intelligence—the sapience of Homo sapiens—"can be so precisely described that a machine can be made to simulate it. This raises philosophical issues about the nature of the mind and the ethics of creating artificial beings endowed with human-like intelligence, issues which have been addressed by myth, fiction and philosophy since antiquity. Artificial intelligence has been the subject of tremendous optimism but has also suffered stunning setbacks. Today it has become an essential part of the technology industry, providing the heavy lifting for many of the most challenging problems in computer science.

3.5 Extra Features Added in AIR v2.0

AIR v2 comes with many new features, some of them are listed below:

3.5.1 Language translation:

It can translate text from one language to another hence it removed the language barrier. So it's doesn't matter "who you are" & "where you are" air always be with you at every step you take.

3.5.2 Self-Power Backup:

Separate Power Always remains a big issue when you are on a move, But AIR v2.0 Comes with “Inbuilt Rechargeable Power Supply battery” That can powered upto 10 hours without any interruption in the flow of current

3.5.3 Run Up to 10 hours:

No need to worry about the supply. It can run up to 10 hour under critical condition and up to 15 hours under normal condition

3.5.4 Increased Stability:

Fixing various issues Faced by previous version leads to the development of new updated stable environment with advanced language translation Feature.

3.5.5 Fully Portable :

Portable makes Device ALIVE because when the device becomes portable it doesn't need to worry about its external component / Environment. It becomes undependable and can perform its processing without external influences. We made it compact as much possible which increase its portability Factors.

3.6.6 A mini Laptop

Air is powered by Raspberry pi B model, connected to a 3.5” resistive touchscreen, 2 speakers, 4 USB slots, a Wi-Fi adaptor and not to forget 10,000 mAh power bank!
So all you need is to connect a keyboard and mouse to Rpi and you have MINI LAPTOP running instantaneously.

References

References

- [1.] Simon Monk: *Programming the Raspberry Pi*. Getting Started with Python. Pages 1-157, Chapter 1-4.
- [2.] Myers, Courtney Boyd ed. (2009). *The AI Report*. Forbes June 2009.
- [3.] Rich, Elaine (1983). *Artificial Intelligence*. McGraw-Hill. Pages 7-33
- [4.] Python Coding. URL: <http://www.learnpython.org/>
- [5.] <http://www.codecademy.com/en/tracks/python>.
- [6.] Linux Commands: <http://linuxcommand.org/> & <http://linuxsurvival.com/>
- [7.] LCD Coding: <http://www.root9.net/2012/10/raspberry-pi-lcd-scroller-tutorial.html>
- [8.] Matt Richardson and Shawn Wallace :*Getting Started with Raspberry Pi*, Chapter 1-3 & chapter 7
- [9.] Rick Golden: *Raspberry Pi Networking Cookbook*, Pages 36-45
- [10.] Google Speech to text and text to speech api's: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>
- [11.] "*Tiny USB-Sized PC Offers 1080p HDMI Output*". February 2012.
- [12.] "*raspberrypi.org – ArmedSlack 13.37*". 16 September 2014.
- [13.] *Raspberry Pi For Dummies*; Sean McManus and Mike Cook; 32 pages; 2013
- [14.] *Getting Started with Raspberry Pi*; Matt Richardson and Shawn Wallace; 176 pages; 2013
- [15.] Fairhead, Harry (2 December 2011). "*Raspberry Pi or Programming – What shall we teach the children?*" I Programmer. 7 February 2012.

Appendix

Source Code

Module 1: c.py

```
import test
import search
import os
import sys
import remove
import offline_checker
import time
import pattern
import cmd_checker
import decimal
mquery=sys.argv[1]
print"You ask ... %s" %(mquery)
call cmd_checker with query
if (cmd_find==0):

    if (len(dquery)<2):
        fquery=mquery

    else:
        fquery=mquery

    squery=fquery.split()
    for i in range(len(squery)):

        f=test.check(squery[i])
        print f

    empty_list.append(f)
if not empty_list:
    print "Wait... check into database"
    offline_checker.query(mquery)
    #os.system("python qw.py '%s'" %(fquery))
    sys.exit()

for s in range(len(empty_list)):
    b=empty_list.count(empty_list[s])
```



```

        a.append(b)
# findind maximum code
try:
    found=max(a)
    print "found next"
    print found

except:
    print "error fgfg "
    sys.exit()
else:
    if (found<(len(squery)/2)+ad):
        print "Wait... check into database"
        calling offline checker database

        sys.exit()
    for position, item in enumerate(a):
        print "i am in"
        if item == found:
            collect all values

if(true==1):
    print empty_list[collect[0]]
    search.search(empty_list[collect[0]],mquery)
else:
    print "Wait... check into database"
    offline_checker.query(mquery)
    sys.exit()

```

Module 2: newb.sh

```

#!/bin/bash
echo "convert to flac"
rec if.flac
echo "Processing..."

```

```
wget -q -U "Mozilla/5.0" --post-file if.flac --heade -O - "http://www.google.com/s" | cut -d\" -f9
>stt1.txt
echo -n "You Said: "
cat stt1.txt
python c.py "$(cat stt1.txt)"
```

Module 3: duplicate_remover.py

```
#!/usr/bin/env python
import total_line_finder
import re

def check(q):
    s=total_line_finder.q()
    f=open("session_forall_q.txt", "r" ).readlines()
    if re.search(r"\b" + re.escape(q) + r"\b", d):
        print "Duplicate found"
        return 0

    else:
        print "NO Duplicate found"
        return 1
```

Module 4: auto_offline_db_maker.py

```
#!/usr/bin/env python

import sys
import unpredic_word
import re
import sys
def save(t):
    print t
    f = open("stt1.txt","r")
    fl = open("session_q.txt", "a")
```

```
v = open("stt1.txt", "r").readlines()
g=v[1]
g=g.replace("\n","")
d=unpredic_word.check(g)
print "value of d"
print d
```

check length

```
if(d==1):
    save to sessional database
else:
    pnot saved to database
```

Module 5: check.py

```
#!/usr/bin/python
i=0
h=open('stt.txt','r')
s= h.read()
for line in s:
    i=i+1
    if(line=="is"):
        print i

print s
d=open("new.txt","a")
d.write(s)
h.close()
```

Module 6: offline_checker.py

```
import sys
import test
import search
import os
import test2
import remove

def query(mquery):

    if (len(dquery)<2):
        fquery=mquery

    else:
        fquery=mquery

    fquery=remove.remove(fquery)
    squery=fquery.split()
    for i in range(len(squery)):
        f1=test2.check(squery[i])
        print f1
        empty_list1.append(f1)
    empty_list1=sum(empty_list1,[])
    print "empty : %s" %(empty_list1)

    if not empty_list1:
        call qw.py

    for s in range(len(empty_list1)):
        b=empty_list1.count(empty_list1[s])

        a1.append(b)

    l= len(a1)

    print a1
    find max a
```

```

true=1
for i in range(len(collect1)):
    if(true==1):
        if empty_list1[collect1[i-1]]==empty_list1[collect1[i]]:
            true=1
        else:
            pno=pattern.pattern(fquery)
            print pno[0]
            if (len(pno)==1):
                search.search(pno[0],mquery)
                sys.exit()

            true=0

```

```

if(true==1):
    print empty_list1[collect1[0]]
    search answer
else:
    call qw

```

Module 7: pattern.py

```

import sys
import re
def pattern(word):
    lines = open( "stt.txt", "r" ).readlines()

    for line in lines:
        i=i+1
        if re.search(r"\b" + re.escape(word) + r"\b", line):
            collect f
    return f

def cmd_pattern(word):
    i=0
    lines = open( "cmd_q.txt", "r" ).readlines()

    for line in lines:
        i=i+1

```

```
        if re.search(r"\b" + re.escape(word) + r"\b", line):
            collect f
    return f
s
```

Module 8: qw.py

```
#!/usr/bin/python

import wolframalpha
import auto_offline_db_maker
import session_forall

client = wolframalpha.Client(app_id)

query = ' '.join(sys.argv[1:])
res = client.query(query)

if len(res.pods) > 0:
    texts = ""
    pod = res.pods[1]
    pod2 = res.pods[0]
    if pod.text:
        texts = pod.text
    elif pod2.text:
        texts = pod2.text

    else:
        texts = "I have no answer for that"
    # to skip ascii character in case of error
    texts = texts.encode('ascii', 'ignore')

    print g
    save=duplicate_removal.check(g)
    if(save==1):
        session_forall.q(g)#to save every ask question
        session_forall.ans(lcd)
    else:
        print "Sorry, I am not sure."
```

Module 9: s.sh

```
#!/bin/bash
INPUT=$*
ary=($INPUT)
for key in "${!ary[@]}"
if [[ "$LENGTH" -lt "100" ]]; then

SHORT[$STRINGNUM]=${SHORTTMP[$STRINGNUM]}
else
STRINGNUM=$((STRINGNUM+1))
SHORTTMP[$STRINGNUM]="${ary[$key]}"
SHORT[$STRINGNUM]="${ary[$key]}"
fi
done
say $*
done
```

Module 10: search.py

```
import sys
import os
import cmd_int
import session_forall
import duplicate_remover

def search(i,q):
    lines = open("result.txt", "r" ).readlines()

    s= "%s" % (lines[i-1] )
    save=duplicate_remover.check(q)
    if(save==1):
        save into database
    print s

def offline_search(i,q):
    lines = open("session_ans.txt", "r" ).readlines()

    s= "%s" % (lines[i-1] )
```

```

s=s.replace("\n","")
save=duplicate_remover.check(q)
print s
if(save==1):
    save into database

def cmd_search(i,q):
    lines = open("cmd_ans.txt", "r" ).readlines()
    s= "%s" % (lines[i-1] )
    d=s.replace("\n", "")
    print "%s" %d
    if (d.isdigit()):
        print "xolo"
        cmd_int.cmd(d)

    else:
        os.system("./s.sh '%s'" %(lines[i-1] ))

def every_q(i):
    lines = open("session_forall_q.txt", "r" ).readlines()

    s= "%s" % (lines[i-1] )
    print s

def every_ans(i):
    lines = open("session_forall_ans.txt", "r" ).readlines()

    s= "%s" % (lines[i-1] )
    print s

```

Module 11: 501.py

```

#!/usr/bin/env python

import total_line_finder
import search
import sys

f=total_line_finder.q()

search.every_q(f)
sys.exit()

```


Module 12: 500.py

```
#!/usr/bin/env python

import total_line_finder
import search
import sys

f=total_line_finder.q()

search.every_q(f)
sys.exit()
```

Module 13: session_forall.py

```
def q(question):

    f = open("session_forall_q.txt","a")
    if len(question)==0:
        exit()
    f.writelines("%s \n"%(question))

def ans(answer):
    d = open("session_forall_ans.txt","a")
    if len(answer)==0:
        pass
    ans=answer.replace("\n", "")
    d.writelines("%s \n"%(ans))
```

Module 14: session_forall.py

```
#!/usr/bin/env python

def q():
    i=0
    question= open("session_forall_q.txt","r").readlines()
```

```

    for f in question:
        i=i+1
    return i

def a():
    i=0
    question= open("session_forall_ans.txt","r").readlines()

    for f in question:
        i=i+1
    return i

```

Module 15: unpredic_word.py

```

#!/usr/bin/env python
import re

def check(text):
    i=0
    lines
    =["today","yesterday","now","time","dollar","money","price","currency","temperature","stock"]
    print lines[0]
    out = regex.sub("", text)

    if re.search(r"\b" + re.escape(text) + r"\b", out):
        print "not predicable word present"
        return 1

    else:
        print "predicable word present"
        return 0

```

Module 16: cmd_int.py

```

#!/usr/bin/env python
import sys
import os

def cmd(n):
    print n

```

```
if (n=="500"): #repeat previous question
    print "hi"
    os.system("python 500.py")
elif (n=="501"): #repeat previous answer
    os.system("python 501.py")
elif (n==502): #database update
    os.system("")
elif (n==503): #da
    os.system("")

else:
    print "nothing found"
```