THE SMART HOME

A Mini Project Report

Submitted to the Faculty of Engineering of
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITYKAKINADA, KAKINADA

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

By

Ch. Jeevana Jyothi
(17481A0529)

G. Gayathri
(17481A0549)

Ch. Bhargavi
(18485A0501)

B. Jaswanth
(17485A0524)

Under the Enviable and Esteemed Guidance of
Ms. K. Bhargavi, M.Tech
Assistant Professor, Departmentof CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute Permanently affiliated to JNTUK)
SESHADRIRAO KNOWLEDGE VILLAGE
GUDLAVALLERU – 521356
ANDHRA PRADESH
2019-20

GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
SESHADRI RAO KNOWLEDGE VILLAGE, GUDLAVALLERU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**THE SMART HOME**" is abonafide record of work carried out by Ch. Jeevana Jyothi (17481A0529), G. Gayathri (17481A0549), Ch. Bhargavi (18485A0501), B. Jaswanth (17481A0524) under the guidance and supervision of Ms. K. Bhargavi in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Jawaharlal Nehru Technological University Kakinada, Kakinada during the academic year 2019-20.

Project Guide	**Head of the Department**
(Ms. K. Bhargavi)	**(Dr. S. NARAYANA)**

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragements crown all the efforts with success.

We would like to express our deep sense of gratitude and sincere thanks to **Ms. K. Bhargavi, Assistant Professor**, Department of Computer Science and Engineering for **her** constant guidance, supervision and motivation in completing the project work.

We feel elated to express our floral gratitude and sincere thanks to **Dr. S. Narayana**, Head of the Department, Computer Science and Engineering for his encouragements all the way during analysis of the project. His annotations, insinuations and criticisms are the key behind the successful completion of the project work.

We would like to take this opportunity to thank our beloved principal **Dr. P. RavindraBabu** for providing a great support for us in completing our project and giving us the opportunity for doing project.

Our Special thanks to the faculty of our department and programmers of our computer lab. Finally, we thank our family members, non-teaching staff and our friends, who had directly or indirectly helped and supported us in completing our project in time.

Team members

Ch. Jeevana Jyothi (17481A0529)
G. Gayathri (17481A0549)
Ch. Bhargavi (18485A0501)
B. Jaswanth (17481A0524)

ABSTRACT

In smart home, the mirror which we provide looks like normal mirror but, when someone stands in front of it the scene changes. The mirror provides a functional, user friendly and interactive UI to it's user to display messages. We can also access our daily home appliances like fans and lights using Alexa. Here we use Raspberry pi and Alexa as back end which connects to front end i.e, javascript and python. Thus, we can create a smart home for a human in their daily routine.

Index Terms: Smart home, Python, Raspberry pi, Alexa.

INDEX

LIST OF FIGURES **PAGENO**

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION:

In smart home, the mirror which we provide looks like normal mirror but, when someone stands in front of it the scene changes. The mirror provides a functional, user friendly and interactive UI to it's user to display messages. We can also access our daily home appliances like fans and lights using Alexa. Here we use Raspberry pi and Alexa as back end which connects to front end i.e, javascript and python. Thus, we can create a smart home for a human in their daily routine.

1.2 PROBLEM STATEMENT:

Now a days, human are busy as per their schedule. As they are leading a busy life, people are not able to know current issues and do small things in their lives.

1.3EXISTING SYSTEM:

There is no such system that human can do the simple things in their routine without wasting their time. Human should spend some time to know the current things in the world.

1.4DISADVANTAGES:

- It needs man power.
- It wastes the lots of time.
- Even though human spend a lots of time there is no guarantee for complete knowledge.

1.5 PROPOSED SYSTEM:

We propose to build this smart piece by keeping in mind all the requirements and drawbacks of existing system. Along with time, date, and weather forecast related information, we also aim at implementing module which can feed news headlines from various sources.

Using Alexa we are implementing access to the home appliances like fans, lights, Tele visions, etc.

1.6 ADVANTAGES:

- The system is very easy to use.
- No need to allot time separately.
- In smart home, mirror is the able to display useful information without needing to open apps or do anything. You simply look at your mirror and the information is available on it.

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 FUNTIONAL REQUIREMENTS:

The main function of our system is to turn on/off the light and fan through voice commands using Alexa and providing a smart mirror with date time and temperature on it.

Other Functional Requirements:

The system should be available when we connected to it.

The system should be connected to the internet to access.

The system is not easy to maintain.

The system should be reusability to add the requirements.

System performance is good when the mobile is connected to internet.

2.2 Non-Functional Requirements:

2.2.1 Usability

This system has a usable to anyone connected to internet having this software and now a day most people prefer to work less to save energy these systems reduce the human power by simply getting ready infront of mirror, knowing some details in it and can simply on/off the electronicgadgets.

2.2.2 Performance

This system will give response to the user very quickly when people gives commands through voice, that system must connected to internet.

2.2.3 Reliability

Due to wired connectivity, reliability can be guaranteed.

2.2.4 Documentation

Everything that is done for designing our system is documented in an understandable manner.

2.2.5 System Modifications

Our system is flexible so that only the authorized persons can do any further modifications required for the system.

2.2.6 Error Handling

Our system is error free it responds only for the specified actions given by the user.

2.3 Software Requirement Specifications:

This document, Software Requirements Specification (SRS), details the requirements of our project which controls the project. In our system implementation Python and JavaScript code is the software requirement.

2.3.1 Purpose:

This Software Requirements Specification (SRS) provides a description of all the functions, specifications, external behaviour, design constraints, requirements (function and non-functional) and other factors necessary to provide a complete and comprehensive description of the proposed system.

2.3.2 Scope:

The scope of the System is to provide a smart mirror with some details displayed on it. This system can be used to further development in the System, initiate interest in houses helps to use the system. System hav an interactive environment that incorporates technology to assist in Activities of Daily Living (ADLs) and is subject to the regulations and constraints placed on the system.

2.3.3 Software Requirements:

- Python
- Javascript

2.2 HARDWARE SPECIFICATIONS:

- Raspberry pi
- Alexa
- channel relay

CHAPTER 3
DESIGN

3.1 SYSTEM ARCHITECTURE

From a project management point of view, software design is conducted in two steps:-

- Preliminary design is concerned with transformation of requirements into data and software architecture.
- Detailed design focuses on refinements to the architectural representation that leads to detailed data structure and algorithmic representations of software.

3.1.1 Fundamental Design Concepts

- **Data Abstraction** -It is an act of representing essential features without including the background details or explanations.
- **Refinement** - It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a step-wise fashion until programming language statements are reached.. Abstraction and Refinement are complementary concepts.
- **Modularity** - Software architecture is divided into components called modules.
- **Software Architecture** - It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system.
- **Control Hierarchy** - A program structure that represents the organization of a program component and implies a hierarchy of control.
- **Data Structure** - It is a representation of the logical relationship among individual elements of data.

- **Software Procedure** - It focuses on the processing of each module individually.
- **Information Hiding** - Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

3.2 UNIFIED MODELLING LANGUAGE (UML)

- UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The goal is for UML to become a common language for creating models of object oriented computer software
- The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
- UML can be applied to diverse application domains (e.g., banking, finance, internet, aerospace, healthcare, etc.) It can be used with all major object and component software development methods and for various implementation platforms (e.g., J2EE, .NET).
- UML is a standard modelling **l**anguage, not a software development process. explained that process:
- provides guidance as to the order of a team's activities,
- specifies what artefacts should be developed,
- directs the tasks of individual developers and the team as a whole,
- Offers criteria for monitoring and measuring a project's products and activities.
- UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case

driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).

The purpose of the UML diagrams is to present multiple views of a system and this set of multiple views is called a model. The most important diagram of UML is class diagram.

3.2.1 USECASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. It’s a great starting point for any project discussion because you can easily identify the main factors involved and the main processes of the system. You can create use case diagrams using our tool and/or get started instantly using our use case templates.

3.2.1 Actors:

Actor is something external to the system and interacts with the system. Actor may be a human being, device or some other software system.

For Library system, actors might be: 1. User

2. Controller

3.2.1 Use – Case

A use-case represents sequence of actions performed by the system that yields an observable result of value for a particular actor. Use-case represents a functional requirement of a system.

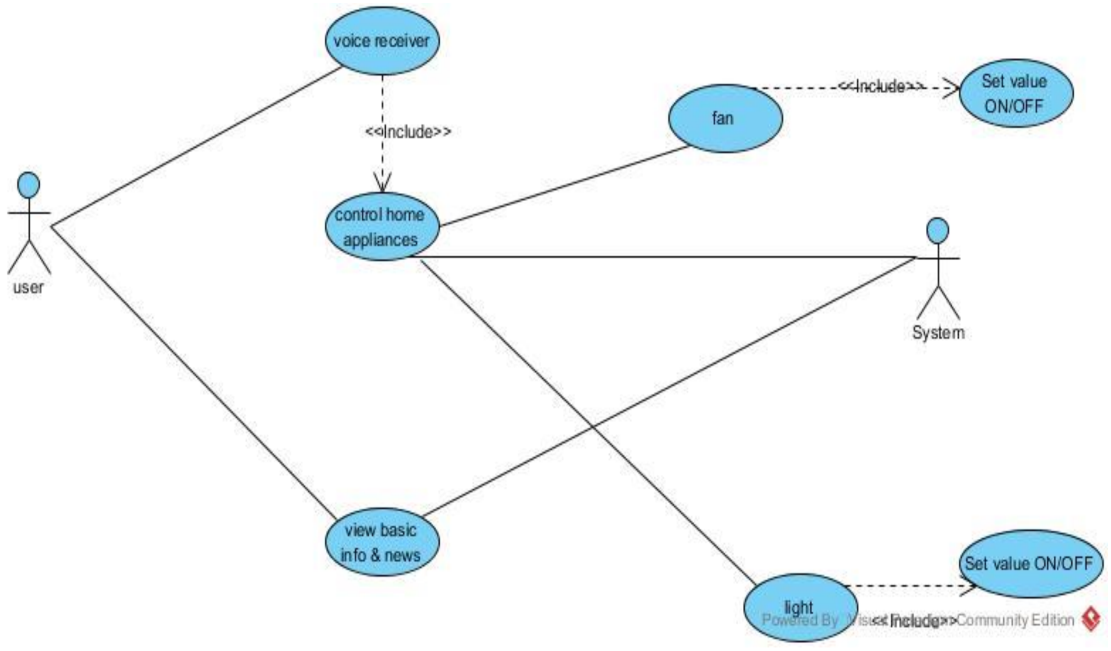


Fig. 3.1 Usecase Diagram

3.2.2 CLASS DIAGRAM

Class diagrams are the main building block of any object oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows. gives an overview of a software system by displaying classes, attributes, operations, and their relationships. This Diagram includes the class name, attributes, and operation in separate designated compartments.

Class Diagram defines the types of objects in the system and the different types of relationships that exist among them. It gives a high-level view of an application. This modeling method can run with almost all Object-Oriented

Methods. A class can refer to another class. A class can have its objects or may inherit from other classes.

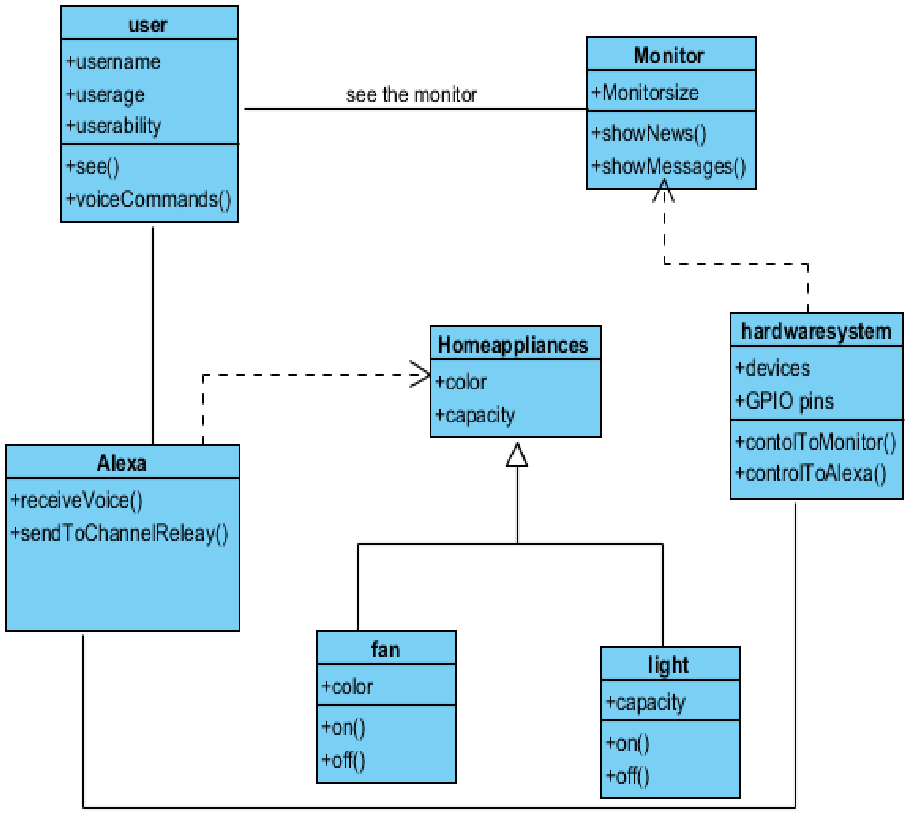


Fig. 3.2 Class Diagram

3.2.3 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case.. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behavior diagrams. An activity diagram is a behavioral diagram i.e. it depicts the behavior of a system. It is used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case.

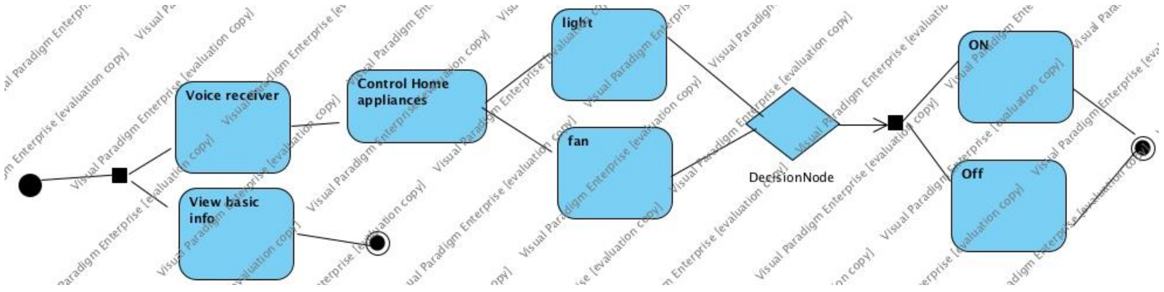


Fig. 3.3 Activity Diagram

3.2.4 SEQUENCE DIAGRAM

A sequence diagram is the most commonly used interaction diagram. A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

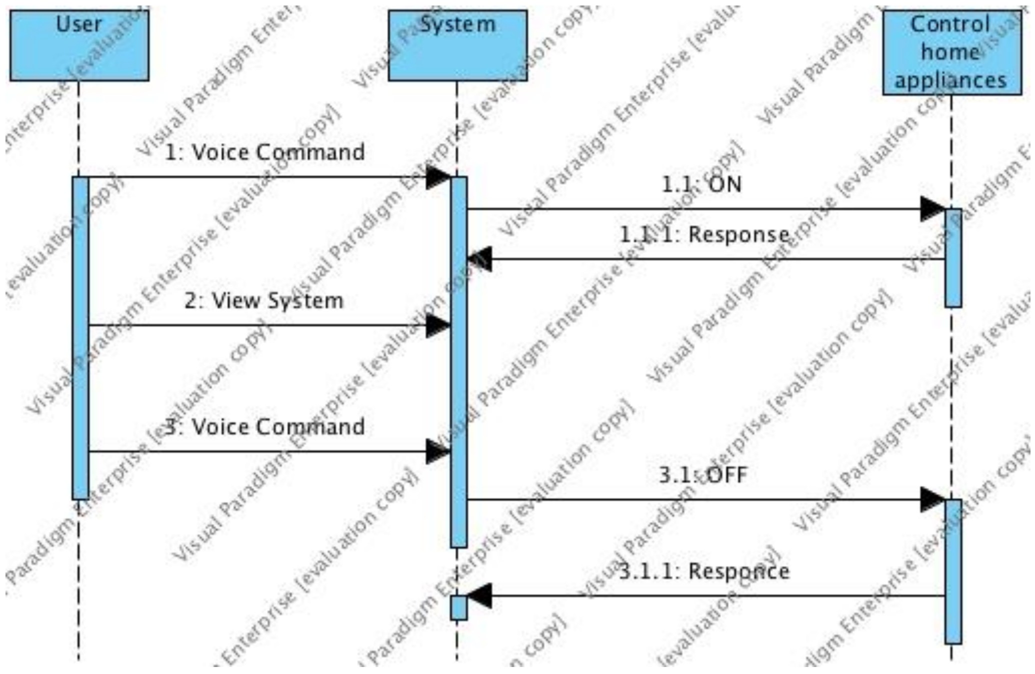


Fig. 3.4 Sequence Diagram

Uses of sequence diagrams –

- Used to model and visualise the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualise how messages and tasks move between objects or components in a system.

CHAPTER 4

IMPLEMENTATION

4.1 TECHNOLOGY DESCRIPTION

4.1.1 Raspberry pi

The **Raspberry Pi** is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

To use a Raspberry Pi we will need

- A **Raspberry Pi** computer with an SD card or micro SD card.
- A monitor with a cable (and, if needed, an HDMI adaptor)
- An USB keyboard and mouse.
- power supply.
- Headphones or speakers (optional)
- An ethernet cable (optional)

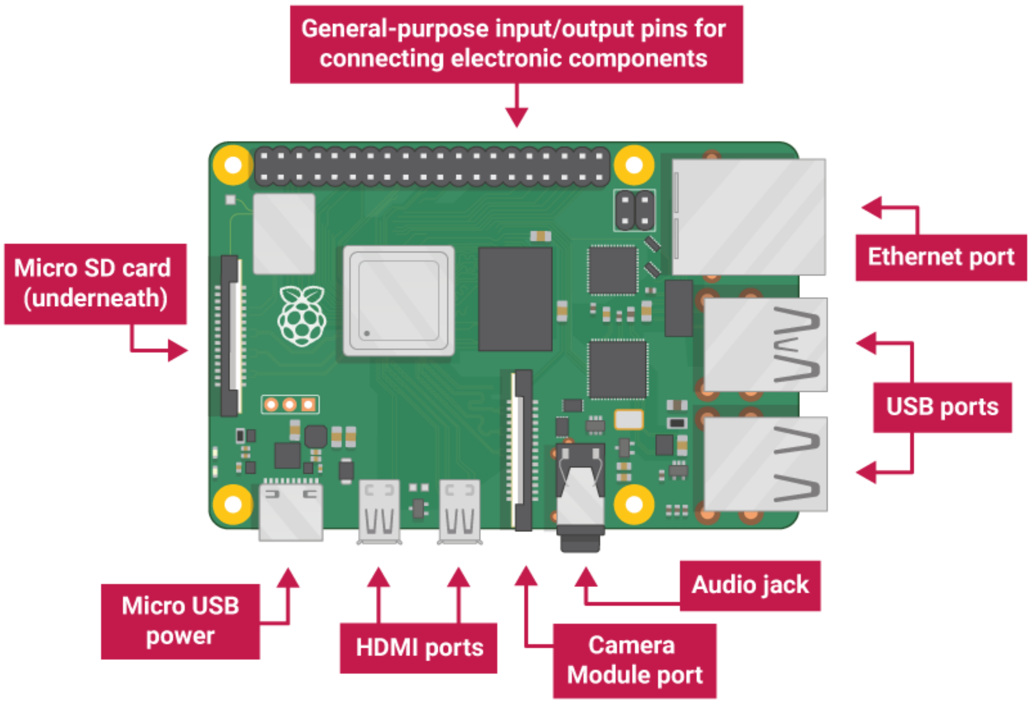

General-purpose input/output pins for connecting electronic components
Micro SD card (underneath)
Ethernet port
USB ports
Audio jack
Micro USB power
HDMI ports
Camera Module port

Fig. 4.1 Motherboard Raspberry Pi

4.1.2 Channel Relay

The Single **Channel Relay Module** is a convenient board which can be used to control high voltage, high current load such as motor, solenoid valves, lamps and AC load. It is designed to interface with microcontroller such as Arduino, PIC and etc. ... It also comes with a LED to indicate the status of **relay**.

Fig. 4.2 Channel Relay

4.1.3 Alexa

Use Raspberry Pi 3 as home automation device with Alexa. The system allow us to control multiple devices connected to Raspberry Pi 3 with voice command. You are able to control GPIO pins thus control GPIO connected devices.

Fig. 4.3 Alexa

4.1.4 Monitor

Generally, monitor is used to display the messages and content. Here the monitor is directly connected to Raspberry Pi via High Definition Multimedia Interface(HDMI) thus we can provide our required display to user.

Fig. 4.4 Monitor

4.2 INSTALLATION STEPS:

4.2.1 Installation Steps

- Download and install the latest *Node.js* version:

```
curl -sL https://deb.nodesource.com/setup_10.x | sudo -
Ebash–sudo apt install -y nodejs
```

- Redirect to the current working directory

```
cdMagicMirror/
```

- Install the application: **npm install**

- Make a copy of the config sample file:

```
cpconfig/config.js.sampleconfig/config.js
```

- Start the application:

```
npm run start
```

4.3 PROCEDURE FOR EXECUTION:

4.3.1 Connecting to Raspberry Pi

- Check the slot on the underside of your Raspberry Pi to see whether an SD card is inside. If no SD card is there, then insert an SD card with Raspbian installed (via NOOBS).

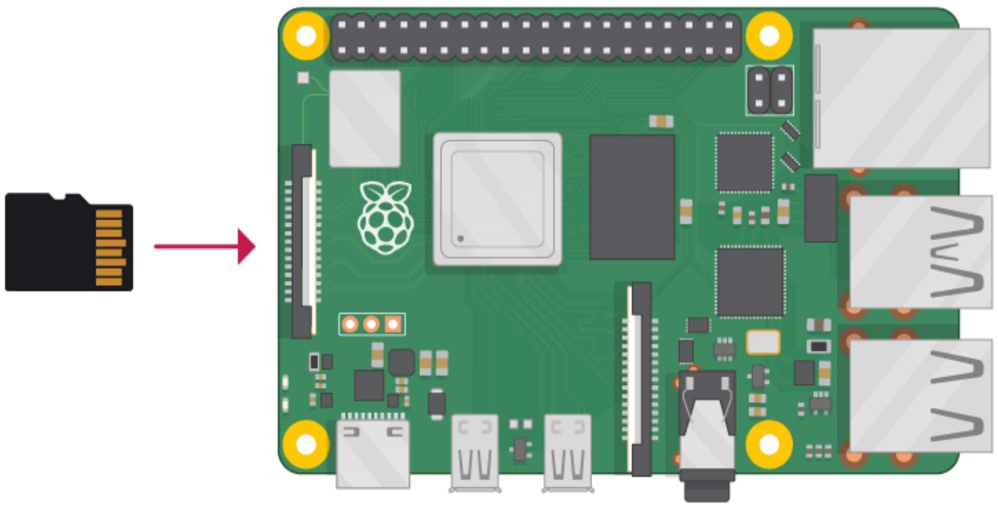


Fig. 4.5 Connecting To Raspberry Pi

Many microSD cards come inside a larger adapter, you can slide the smaller card out using the lip at the bottom.

Fig. 4.6 SD Card

Installing Raspbian using the Raspberry Pi

- Find the USB connector end of your mouse's cable, and connect the mouse to a USB port on your Raspberry Pi (it doesn't matter which port you use).

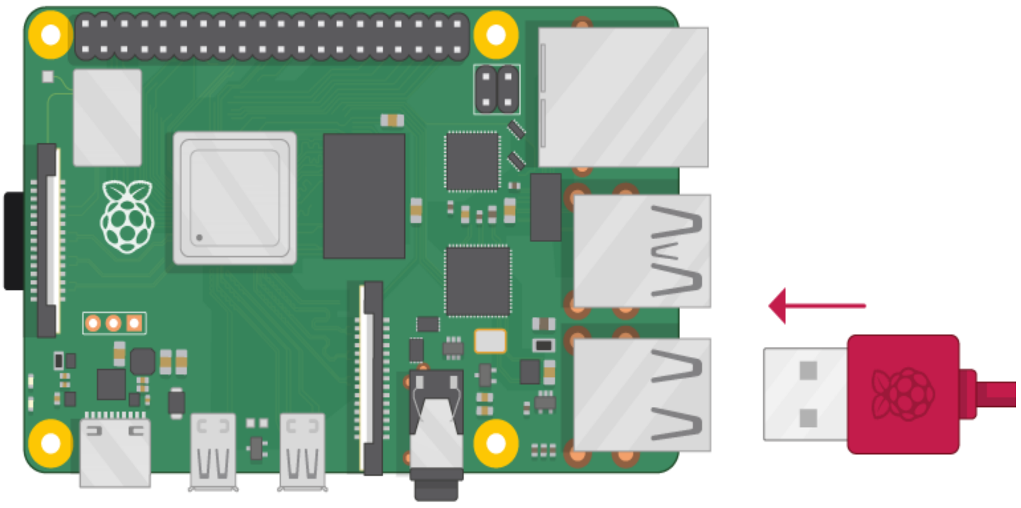


Fig. 4.7 USB Port Connection

- Connect the keyboard in the same way.

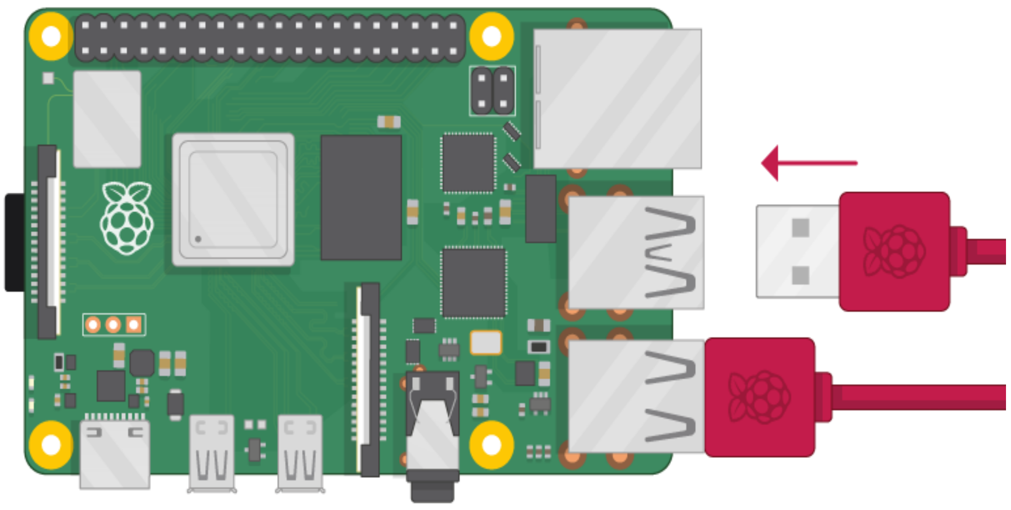


Fig. 4.8 Connecting To Keyboard

- Make sure your screen is plugged into a wall socket and switched on.
- Look at the HDMI port(s) on your Raspberry Pi — notice that they have a flat side on top.
- Use a cable to connect the screen to the Raspberry Pi's HDMI port — use an adapter if necessary.
- Connect your screen to the first of Raspberry Pi 4's HDMI ports, labelled **HDMI0**.

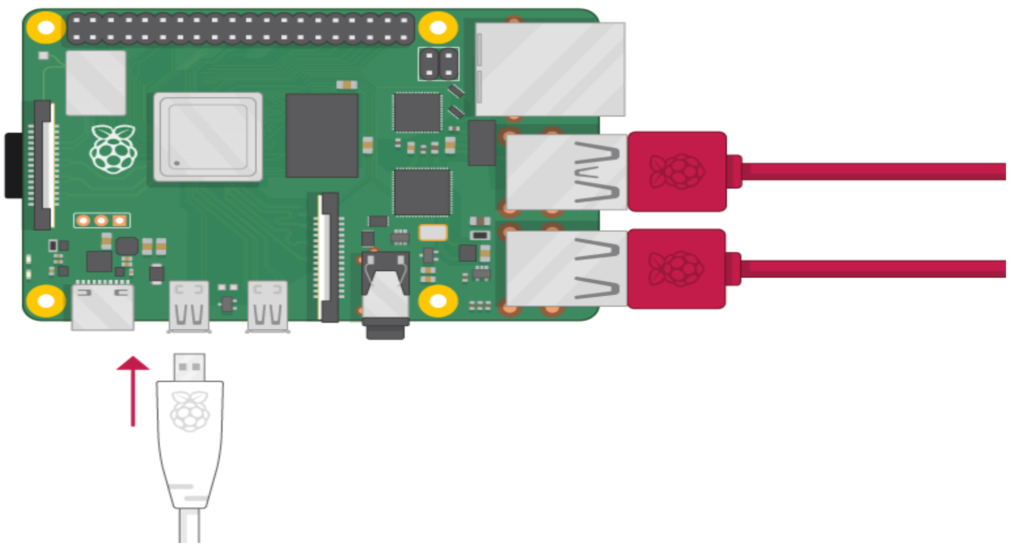


Fig. 4.9 Connecting To HDMI Ports

- You could connect an optional second screen in the same way.

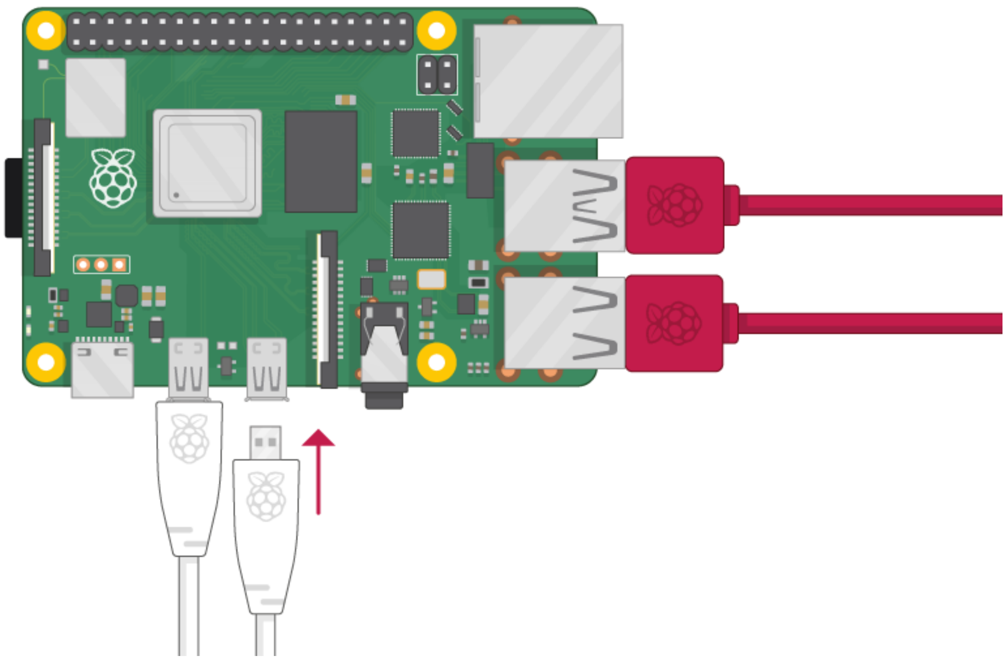


Fig. 4.10 Connecting To Optional Screen

- Connect your screen to the single HDMI port.

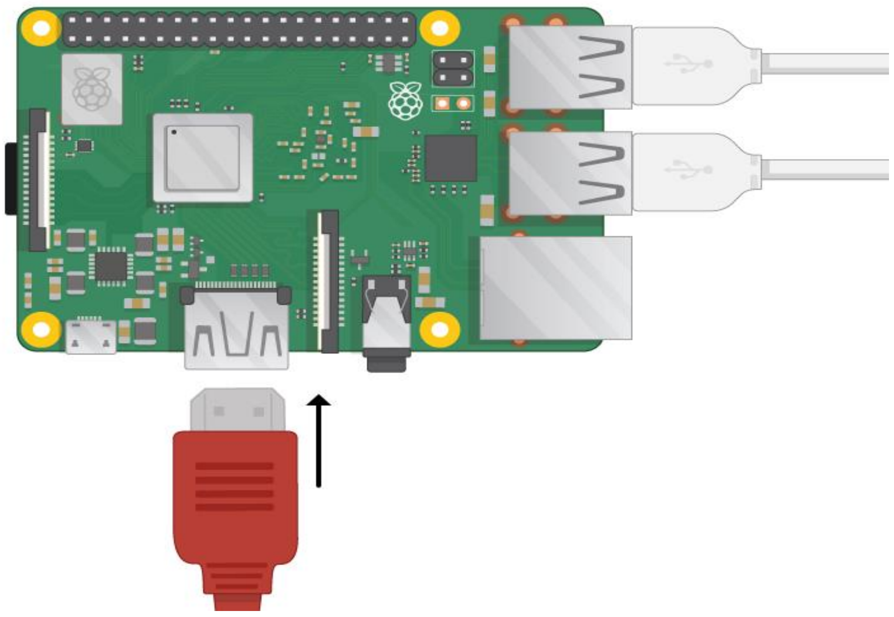


Fig. 4.11 Connecting The Pi To The Internet Via Ethernet

- If you want to connect the Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on the Raspberry Pi to an Ethernet socket on the wall or on your internet router. You don’t need to do this if you want to use wireless connectivity, or if you don’t want to connect to the internet.

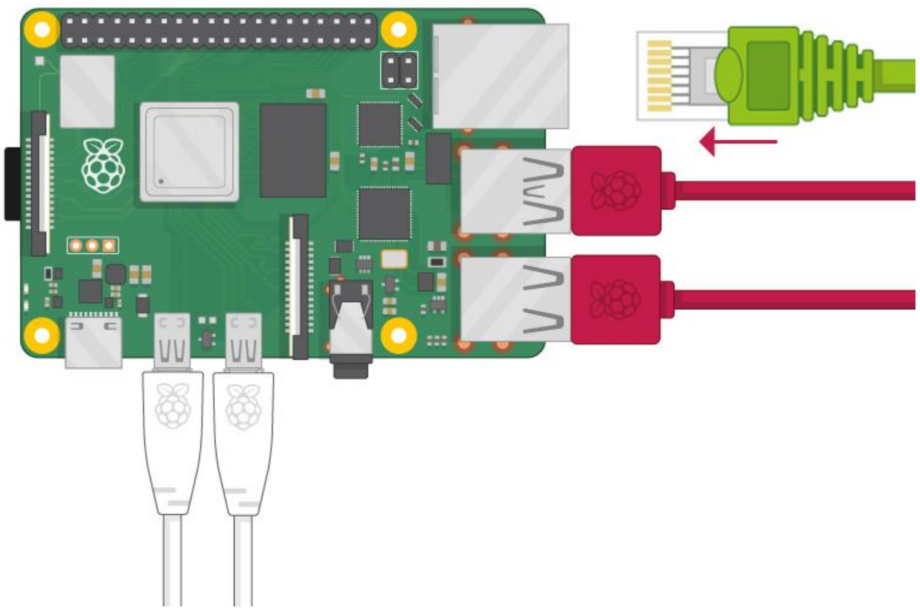


Fig. 4.12 Connecting To Speakers

- If your screen has speakers, your Raspberry Pi can play sound through these. Or you could connect headphones or speakers to the audio port.

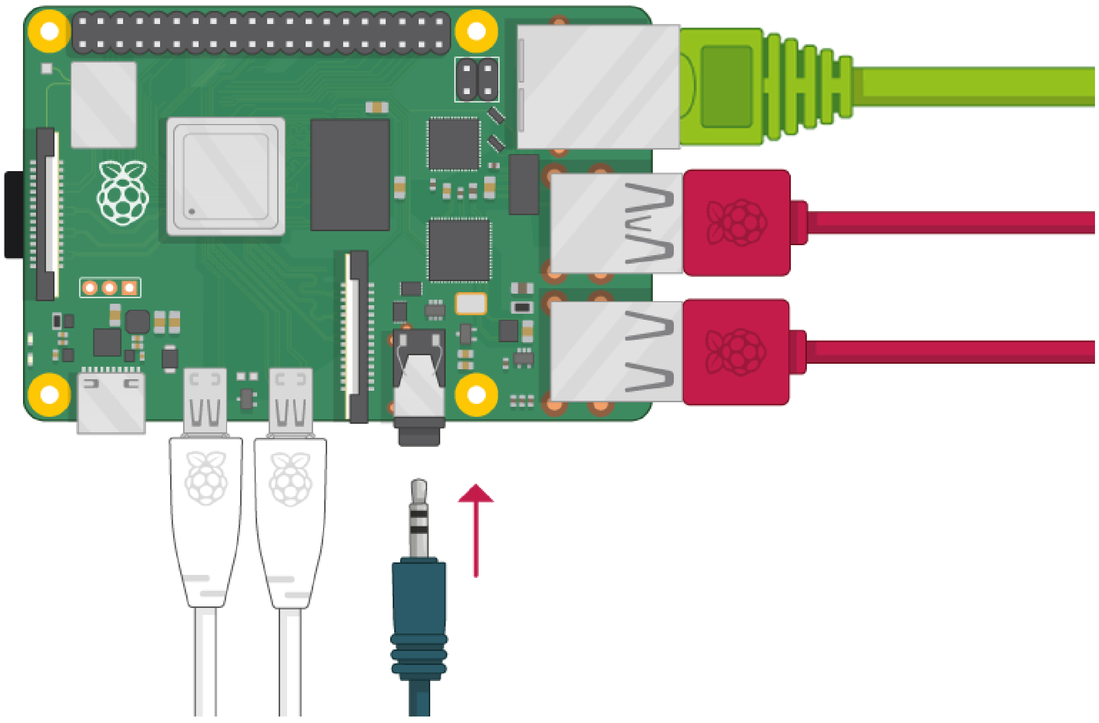


Fig. 4.13 Power Supply

- Plug the power supply into a socket and then connect it to your Raspberry Pi's USB power port.

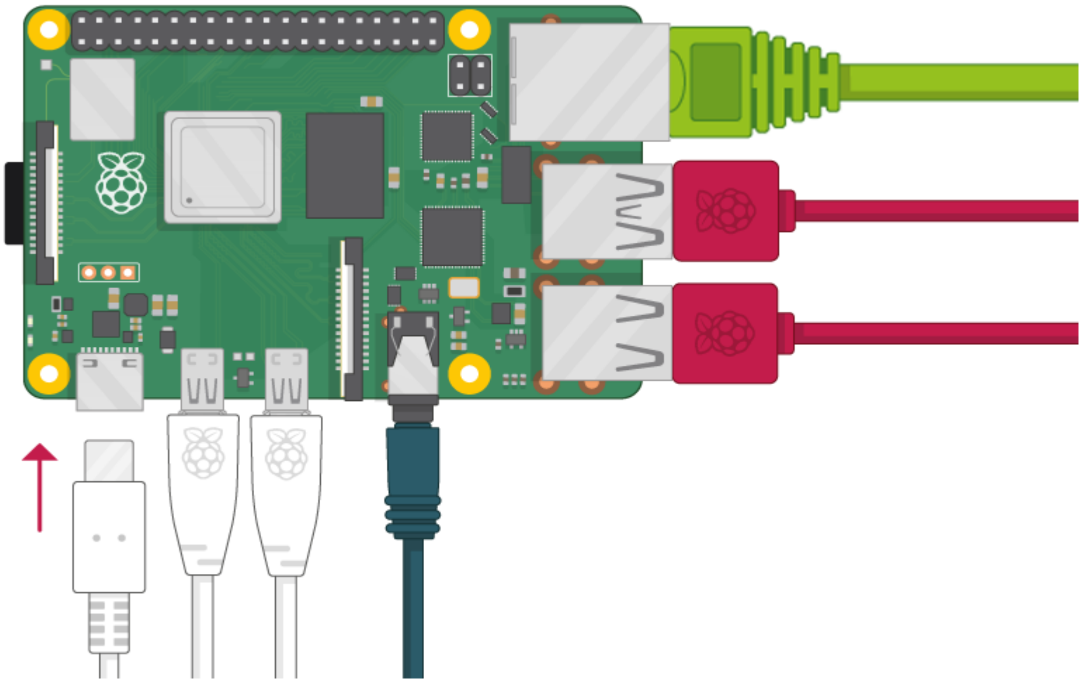


Fig. 4.14 Pi Motherboard

- You should see a red light on your Raspberry Pi and raspberries on the monitor.
- Your Raspberry Pi then boots up into a graphical desktop.

Fig. 4.15 Display

GPIO PINS

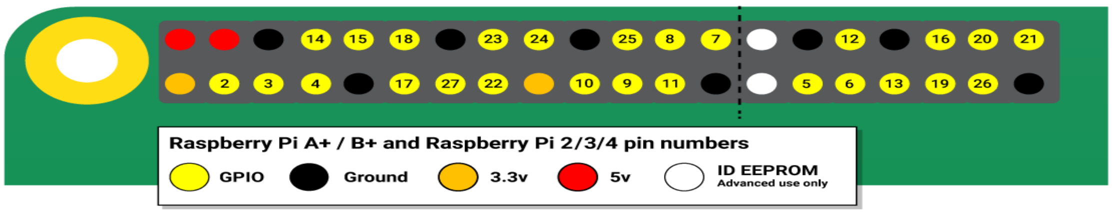


Fig. 4.15 GPIO Pins

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.

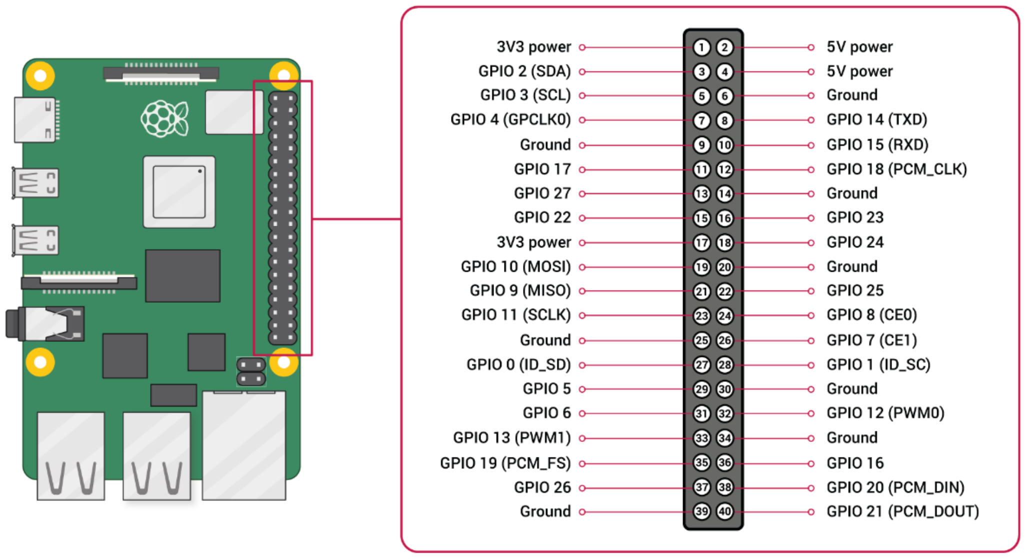


Fig. 4.16 GPIO Pins - Purpose

4.3.1 Code For Execution:

name_port_gpio.py

Usage: chip@chip:~/echo$ sudo python name_port_gpio.py devicename 51000 XIO-P2
Usage: chip@chip:~/echo$ sudo python name_port_gpio.py coffeebrew 51001 XIO-P3

```
import fauxmo
import logging
import time
import sys
import CHIP_IO.GPIO as GPIO ## Import GPIO library

from debounce_handler import debounce_handler

logging.basicConfig(level=logging.DEBUG)

class device_handler(debounce_handler):
    """Publishes the on/off state requested,
       and the IP address of the Echo making the request.
    """
    #TRIGGERS = {str(sys.argv[1]): int(sys.argv[2])}
    TRIGGERS = {"office": 52000}

    def act(self, client_address, state, name):
        print("State", state, "from client @", client_address)
        GPIO.setmode(GPIO.BOARD) ## Use board pin
numbering
        GPIO.setup(str(sys.argv[3]), GPIO.OUT)   ## Setup GPIO
Pin to OUTPUT
        GPIO.output(str(sys.argv[3]), not state) ## State is
true/false
        GPIO.cleanup(str(sys.argv[3]))
        return True

if __name__ == "__main__":
    # Startup the fauxmo server
    fauxmo.DEBUG = True
    p = fauxmo.poller()
    u = fauxmo.upnp_broadcast_responder()
    u.init_socket()
    p.add(u)

    # Register the device callback as a fauxmo handler
    d = device_handler()
    for trig, port in d.TRIGGERS.items():
        fauxmo.fauxmo(trig, u, p, None, port, d)

    # Loop and poll for incoming Echo requests
    logging.debug("Entering fauxmo polling loop")
    while True:
        try:
            # Allow time for a ctrl-c to stop the process
            p.poll(100)
            time.sleep(0.1)
        except Exception as e:
            logging.critical("Critical exception: "+ e.args )
            break
```

debounce_handler.py

```
import time
class debounce_handler(object):
"""Use this handler to keep multiple Amazon Echo devices from
reacting to the same voice command."""
    DEBOUNCE_SECONDS = 0.3
    def __init__(self):
        self.lastEcho = time.time()
    def on(self, client_address, name):
        if self.debounce():
            return True
        return self.act(client_address, True, name)
    def off(self, client_address, name):
        if self.debounce():
            return True
        return self.act(client_address, False, name)
    def act(self, client_address, state):
        pass
    def debounce(self):
"""If multiple Echos are present, the one most likely to respond
first is the one that can best hear the speaker... which is the
closest one. Adding a refractory period to handlers keeps us
from worrying about one Echo overhearing a command meant
for another one."""
        if (time.time() - self.lastEcho) <
self.DEBOUNCE_SECONDS:
            return True
        self.lastEcho = time.time()
        return False
```

name_port_gpio.py

```
import fauxmo
import logging
import time
import sys
import RPi.GPIO as GPIO ## Import GPIO library
from debounce_handler import debounce_handler
logging.basicConfig(level=logging.DEBUG)
class device_handler(debounce_handler):
    """Publishes the on/off state requested,
       and the IP address of the Echo making the request."""
    #TRIGGERS = {str(sys.argv[1]): int(sys.argv[2])}
    #TRIGGERS = {"office": 52000}
    TRIGGERS = {"kitchen": 52000, "living room": 51000,
"office": 53000, "room": 52002, "tv": 52003, "pc": 52004,
                "xbox": 52005, "light": 52006}

    def act(self, client_address, state, name):
        print("State", state, "from client @", client_address)
        # GPIO.setmode(GPIO.BOARD) ## Use board pin
numbering
        # GPIO.setup(int(7), GPIO.OUT)   ## Setup GPIO Pin to
OUTPUT
```

```
        # GPIO.output(int(7), state) ## State is true/false
        if name=="kitchen":
            GPIO.setmode(GPIO.BOARD) ## Use board pin
numbering
            GPIO.setup(int(7), GPIO.OUT)   ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(7), state) ## State is true/false
        elif name =="living room":
            GPIO.setmode(GPIO.BOARD) ## Use board pin
numbering
            GPIO.setup(int(11), GPIO.OUT)   ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(11), state) ## State is true/false
        elif name =="office":
            GPIO.setmode(GPIO.BOARD) ## Use board pin
numbering
            GPIO.setup(int(13), GPIO.OUT)   ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(13), state) ## State is true/false
        elif name == "room":
            GPIO.setmode(GPIO.BOARD)  ## Use board pin
numbering
            GPIO.setup(int(5), GPIO.OUT)  ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(5), state)  ## State is true/false
        elif name == "tv":
            GPIO.setmode(GPIO.BOARD)  ## Use board pin
numbering
            GPIO.setup(int(15), GPIO.OUT)  ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(15), state)  ## State is true/false
        elif name == "pc":
            GPIO.setmode(GPIO.BOARD)  ## Use board pin
numbering
            GPIO.setup(int(8), GPIO.OUT)  ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(8), state)  ## State is true/false
        elif name == "xbox":
            GPIO.setmode(GPIO.BOARD)  ## Use board pin
numbering
            GPIO.setup(int(12), GPIO.OUT)  ## Setup GPIO Pin to
OUTPUT
```

```
            GPIO.output(int(12), state)  ## State is true/false
        elif name == "light":
            GPIO.setmode(GPIO.BOARD)  ## Use board pin
numbering
            GPIO.setup(int(10), GPIO.OUT)  ## Setup GPIO Pin to
OUTPUT
            GPIO.output(int(10), state)  ## State is true/false
        else:
            print("Device not found!")
        return True
if __name__ == "__main__":
    # Startup the fauxmo server
    fauxmo.DEBUG = True
    p = fauxmo.poller()
    u = fauxmo.upnp_broadcast_responder()
    u.init_socket()
    p.add(u)
    # Register the device callback as a fauxmo handler
    d = device_handler()
    for trig, port in d.TRIGGERS.items():
        fauxmo.fauxmo(trig, u, p, None, port, d)
        logging.debug("Entering fauxmo polling loop")
    while True:
        try:
            # Allow time for a ctrl-c to stop the process
            p.poll(100)
            time.sleep(0.1)
        except Exception as e:
            logging.critical("Critical exception: "+ e.args  )
            break
```

CHAPTER 5
TESTING

5.1 BLACK BOX TESTING

The black box is a powerful technique to check the application under test from the user's perspective. Black box testing is used to test the system against external factors responsible for software failures. This testing approach focuses on the input that goes into the software, and the output that is produced. The testing team does not cover the inside details such as code, server logic, and development method.

Black Box Testing Techniques:

Boundary Value Analysis

It is the widely used black-box testing, which is also the basis for equivalence testing. Boundary value analysis tests the software with test cases with extreme values of test data. BVA is used to identify the flaws or errors that arise due to the limits of input data.

For example: Taking inputs for a test case data for an age section should accept a valid data of anything between 1-100. According to BVP analysis, the software will be tested against four test data as -1, 1, 100, and 101 to check the system's response using the boundary values.

Equivalence partitioning

This test case designing techniques checks the input and output by dividing the input into equivalent classes. The data must be tested at least once to ensure maximum test coverage of data. It is the exhaustive form of testing, which also reduces the redundancy of inputs.

For example: Taking inputs for a test case data for the example mentioned above will have three classes from which one data will be tested.

Valid class: 1 to 100 (any number), Invalid class: -1 (checking the lowest of lowest), Invalid class: 101(highest of highest)

State Transition Testing

This testing technique uses the inputs, outputs, and the state of the system during the testing phase. It checks the software against the sequence of transitions or events among the test data.

Based on the type of software that is tested, it checks for the behavioral changes of a system in a particular state or another state while maintaining the same inputs.

For example, A login page will let you input username and password until three attempts. Each incorrect password will be sent the user to the login page. After the third attempt, the user will be sent to an error page. This state transition method considers the various states of the system and the inputs to pass only the right sequence of the testing.

Decision Table Testing

This approach creates test cases based on various possibilities. It considers multiple test cases in a decision table format where each condition is checked and fulfilled, to pass the test and provide accurate output. It is preferred in case of various input combinations and multiple possibilities.

For example, A food delivery application will check various payment modes as input to place the order — decision making based on the table.

Case1: If the end-user has a card, then the system will not check for cash or coupon and will take action to place the order.

Case2: If the end-user has a coupon will not be checked for a card or cash and action will be taken.

Case3: if the end-user has cash, the action will be taken.

Case4: If the end-user doesn’t have anything, then action will not be taken.

Graph-Based Testing:

It is similar to a decision-based test case design approach where the relationship between links and input cases are considered.

Error Guessing Technique:

This method of designing test cases is about guessing the output and input to fix any errors that might be present in the system. It depends on the skills and judgment of the tester.

Comparison testing

This method uses the two different versions of the same software to compare and validate the results.

5.1.1 TEST CASES

In Software Development, we need to check each functionality or events in the developing product to make sure it fulfils the actual requirements. Test Cases are the set of conditions or variables for checking functionality. For each scenario there will be test cases, and these set of conditions are planned by the tester.

Sample Template of the Test cases is….

TEST CASE FIELD	**DERSCRIPTION**
Test case ID	• Each test case should be represented by a unique ID. To indicate test type follow some convection like “TC_UI_1” indicating “User Interface Test Case#1.”
Test Priority:	• It is useful while executing the test. ➢ Low ➢ Medium ➢ High
Name of the Module	• Determine the name of the main module or submodule being tested
Test designed by	• Testers Name
Date of test designed	• Date when test was designed
Test Executed by	• Who executed the test-tester

Date of the Test Execution	• Date when test needs to be executed
Name or Test Title	• Title of the test case
Description of test	• Determine the summary or test purpose in brief
Pre-condition	• Any requirement that needs to be done before execution of this test case. To execute this test case list all pre-conditions
Dependencies	• Determine any dependencies on test requirements or other test cases.
Test steps	• Mention all the test steps in details and write in the order in which it require to be executed. While writing test steps ensure that you provide detail.

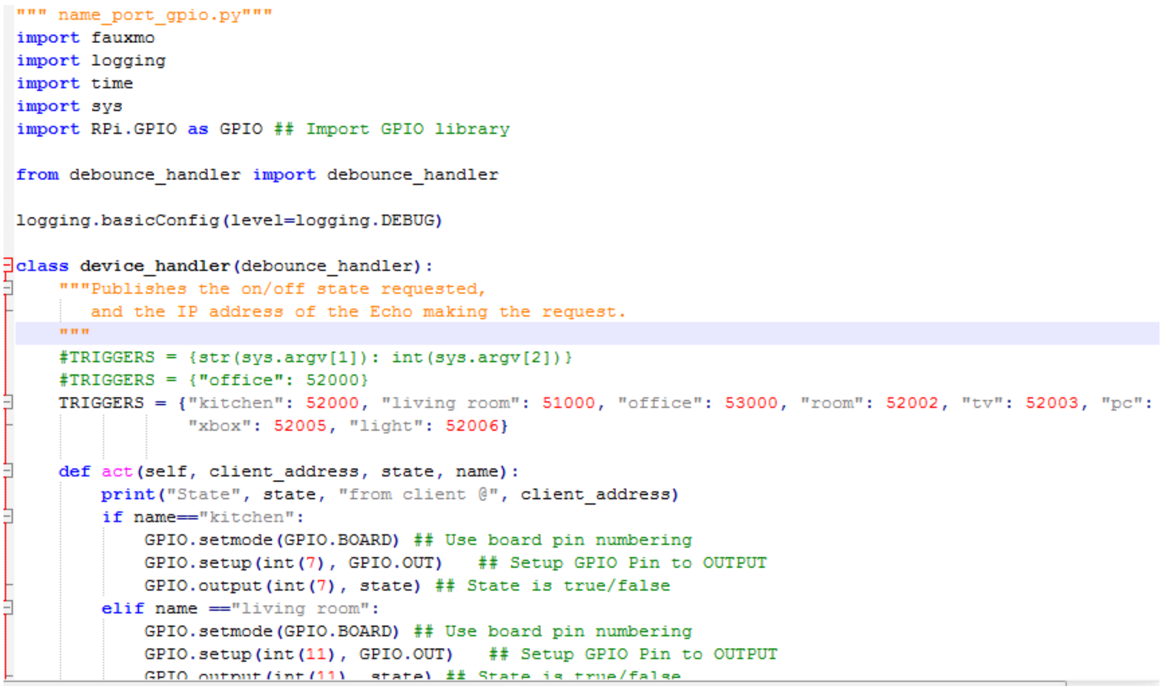
Table. 5.1 Sample Template For Testcase

Req id	**Risks**	**Req Type**	**Req Description**	**Trace from User req/Trace to system req**	**Traceto design specification**	**UT**	**IT**	**ST**	**AT**	**Trace to test script**
1	Technical aspect	Functional	Should perform	No	Yes	Y	Y	Y	N	Y
2	GUI	Interface	Easy use	Yes	Yes	Y	Y	N	N	Y
3	Design	Design	Overall design	Yes	Yes	Y	Y	Y	Y	Y
4	Updating information	Maintainability	Add or Delete	No	Yes	N	N	N	N	N

Table. 5.2 Test Case Example

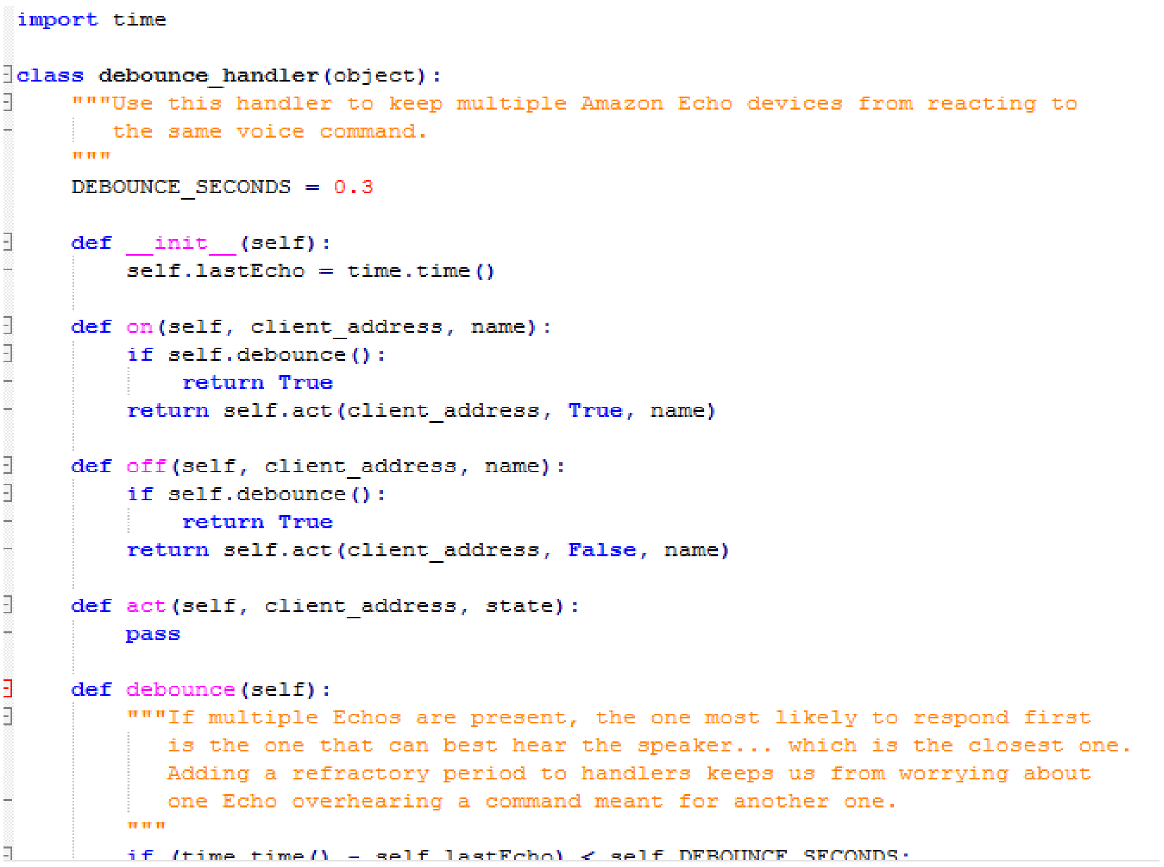
CHAPTER 6

SCREENSHOTS



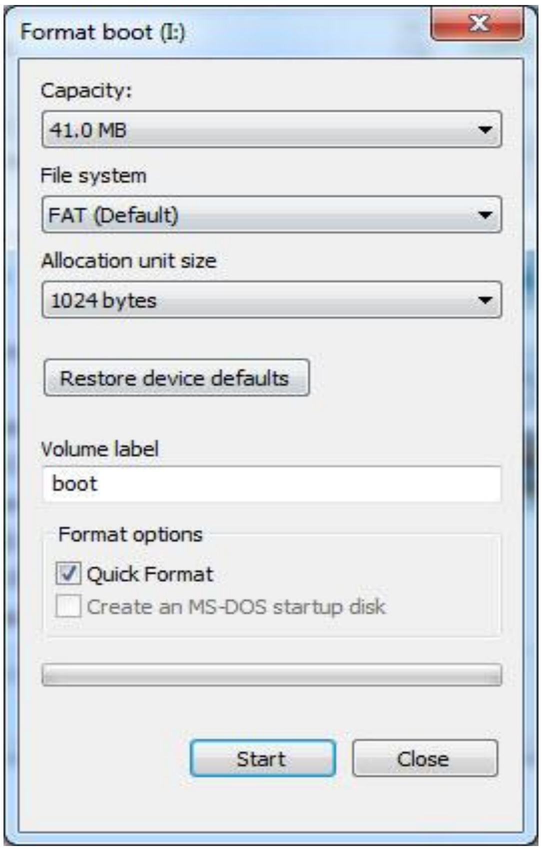
```
""" name_port_gpio.py"""
import fauxmo
import logging
import time
import sys
import RPi.GPIO as GPIO ## Import GPIO library

from debounce_handler import debounce_handler

logging.basicConfig(level=logging.DEBUG)

class device_handler(debounce_handler):
    """Publishes the on/off state requested,
       and the IP address of the Echo making the request.
    """
    #TRIGGERS = {str(sys.argv[1]): int(sys.argv[2])}
    #TRIGGERS = {"office": 52000}
    TRIGGERS = {"kitchen": 52000, "living room": 51000, "office": 53000, "room": 52002, "tv": 52003, "pc":
                "xbox": 52005, "light": 52006}

    def act(self, client_address, state, name):
        print("State", state, "from client @", client_address)
        if name=="kitchen":
            GPIO.setmode(GPIO.BOARD) ## Use board pin numbering
            GPIO.setup(int(7), GPIO.OUT)   ## Setup GPIO Pin to OUTPUT
            GPIO.output(int(7), state) ## State is true/false
        elif name =="living room":
            GPIO.setmode(GPIO.BOARD) ## Use board pin numbering
            GPIO.setup(int(11), GPIO.OUT)   ## Setup GPIO Pin to OUTPUT
            GPIO.output(int(11), state) ## State is true/false
```

Fig. 6.1 Code for GPIO Pins



```
import time

class debounce_handler(object):
    """Use this handler to keep multiple Amazon Echo devices from reacting to
       the same voice command.
    """
    DEBOUNCE_SECONDS = 0.3

    def __init__(self):
        self.lastEcho = time.time()

    def on(self, client_address, name):
        if self.debounce():
            return True
        return self.act(client_address, True, name)

    def off(self, client_address, name):
        if self.debounce():
            return True
        return self.act(client_address, False, name)

    def act(self, client_address, state):
        pass

    def debounce(self):
        """If multiple Echos are present, the one most likely to respond first
           is the one that can best hear the speaker... which is the closest one.
           Adding a refractory period to handlers keeps us from worrying about
           one Echo overhearing a command meant for another one.
        """
        if (time.time() - self.lastEcho) < self.DEBOUNCE_SECONDS:
```

Fig. 6.2 Code For Debounce Handler



Capacity: 41.0 MB
File system: FAT (Default)
Allocation unit size: 1024 bytes
Volume label: boot
Format options:
[✓] Quick Format
[ ] Create an MS-DOS startup disk

Fig. 6.3 Format boot

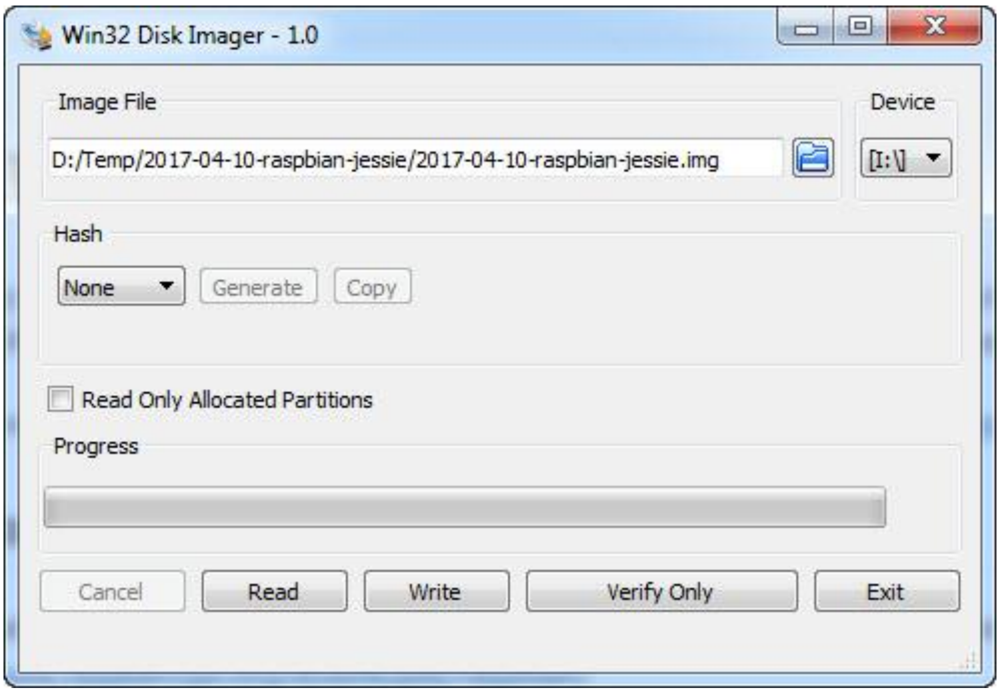
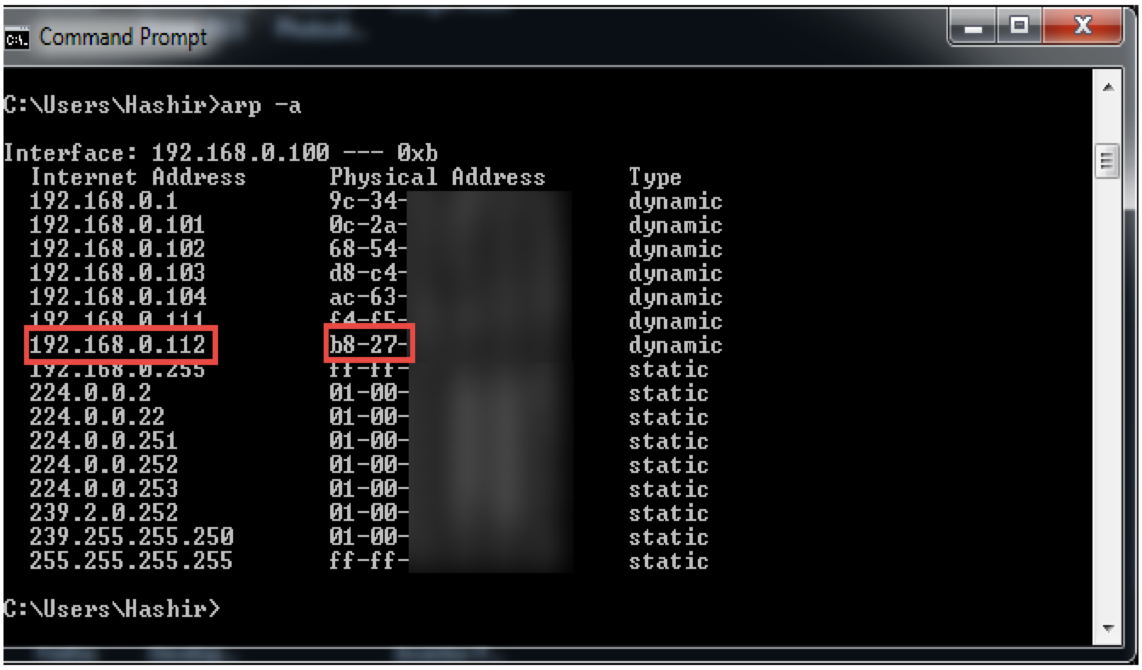


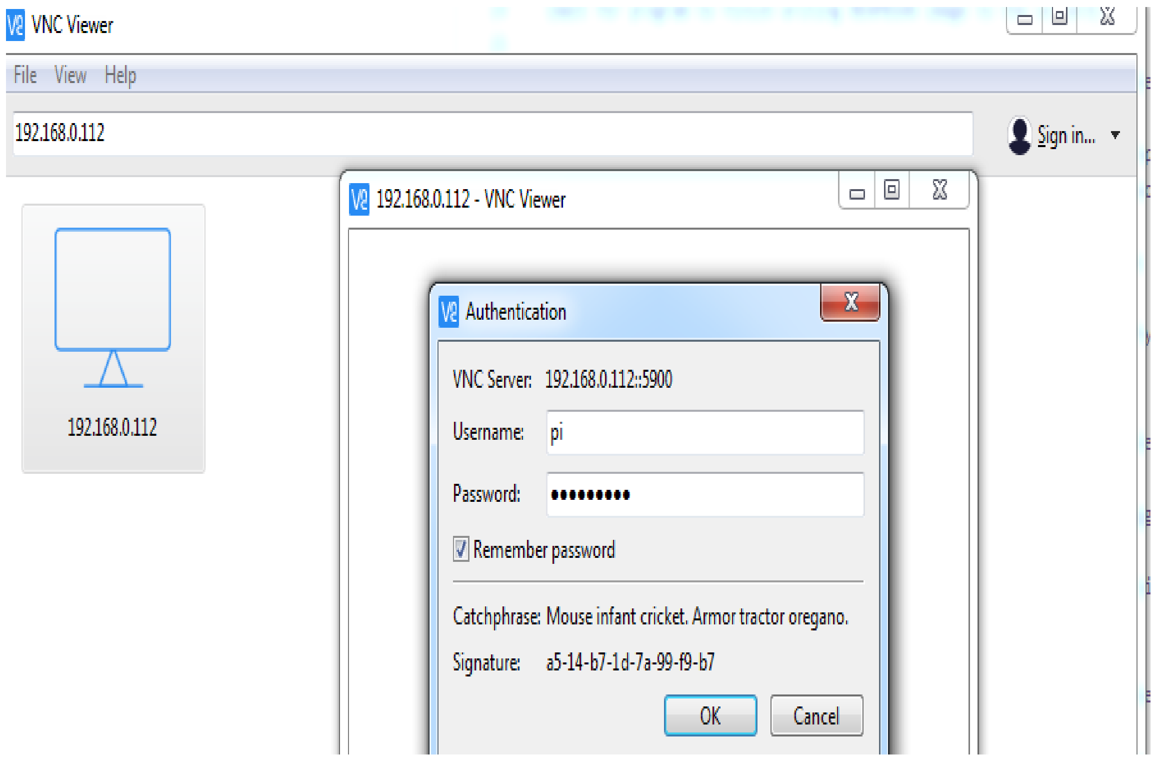
Image File: D:/Temp/2017-04-10-raspbian-jessie/2017-04-10-raspbian-jessie.img
Device: [I:\]
Hash: None
[ ] Read Only Allocated Partitions

Fig. 6.4 Disk Imager



```
C:\Users\Hashir>arp -a

Interface: 192.168.0.100 --- 0xb
  Internet Address      Physical Address      Type
  192.168.0.1           9c-34- [redacted]     dynamic
  192.168.0.101         0c-2a-                dynamic
  192.168.0.102         68-54-                dynamic
  192.168.0.103         d8-c4-                dynamic
  192.168.0.104         ac-63-                dynamic
  192.168.0.111         f4-f5-                dynamic
  192.168.0.112         b8-27-                dynamic
  192.168.0.255         ff-ff-                static
  224.0.0.2             01-00-                static
  224.0.0.22            01-00-                static
  224.0.0.251           01-00-                static
  224.0.0.252           01-00-                static
  224.0.0.253           01-00-                static
  239.2.0.252           01-00-                static
  239.255.255.250       01-00-                static
  255.255.255.255       ff-ff-                static

C:\Users\Hashir>
```

Fig. 6.5 For IP Address Tracing



VNC Viewer

File View Help

192.168.0.112

Sign in...

192.168.0.112

192.168.0.112 - VNC Viewer

Authentication

VNC Server: 192.168.0.112::5900

Username: pi

Password: •••••••••

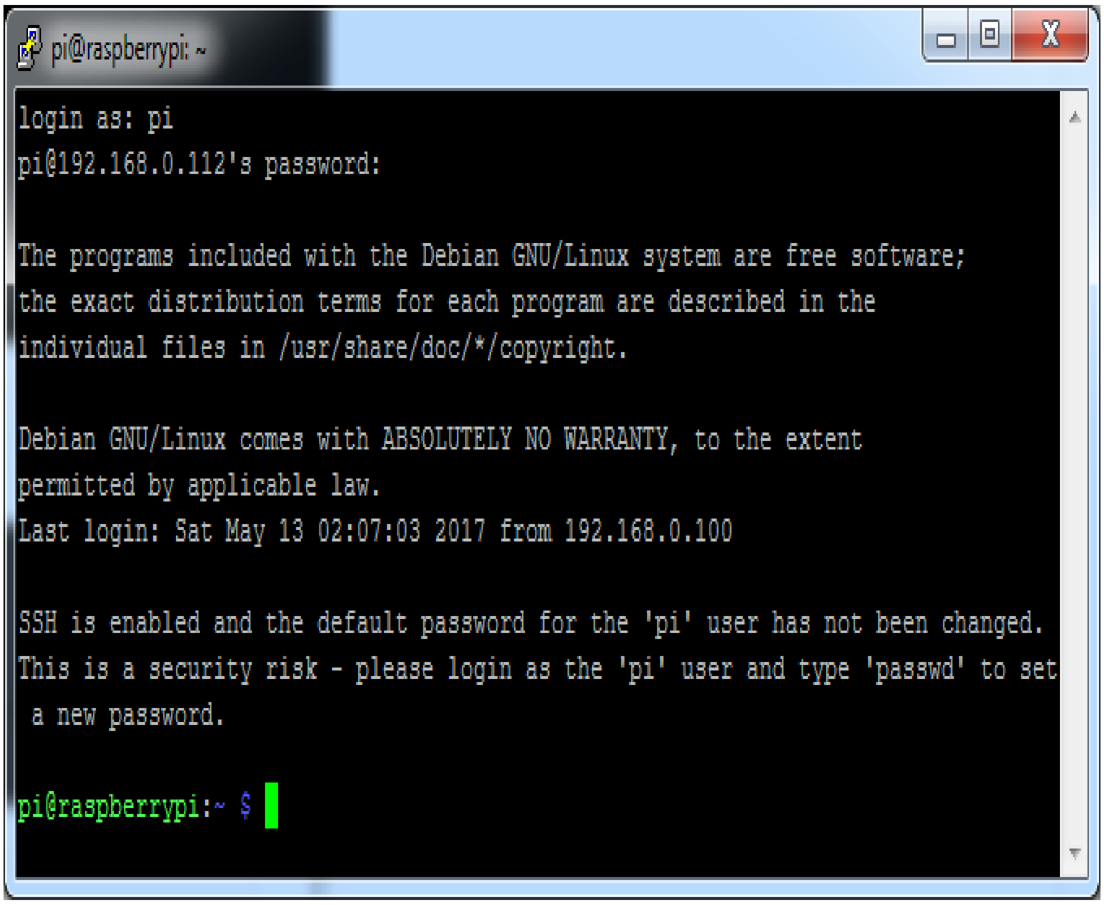
☑ Remember password

Catchphrase: Mouse infant cricket. Armor tractor oregano.

Signature: a5-14-b7-1d-7a-99-f9-b7

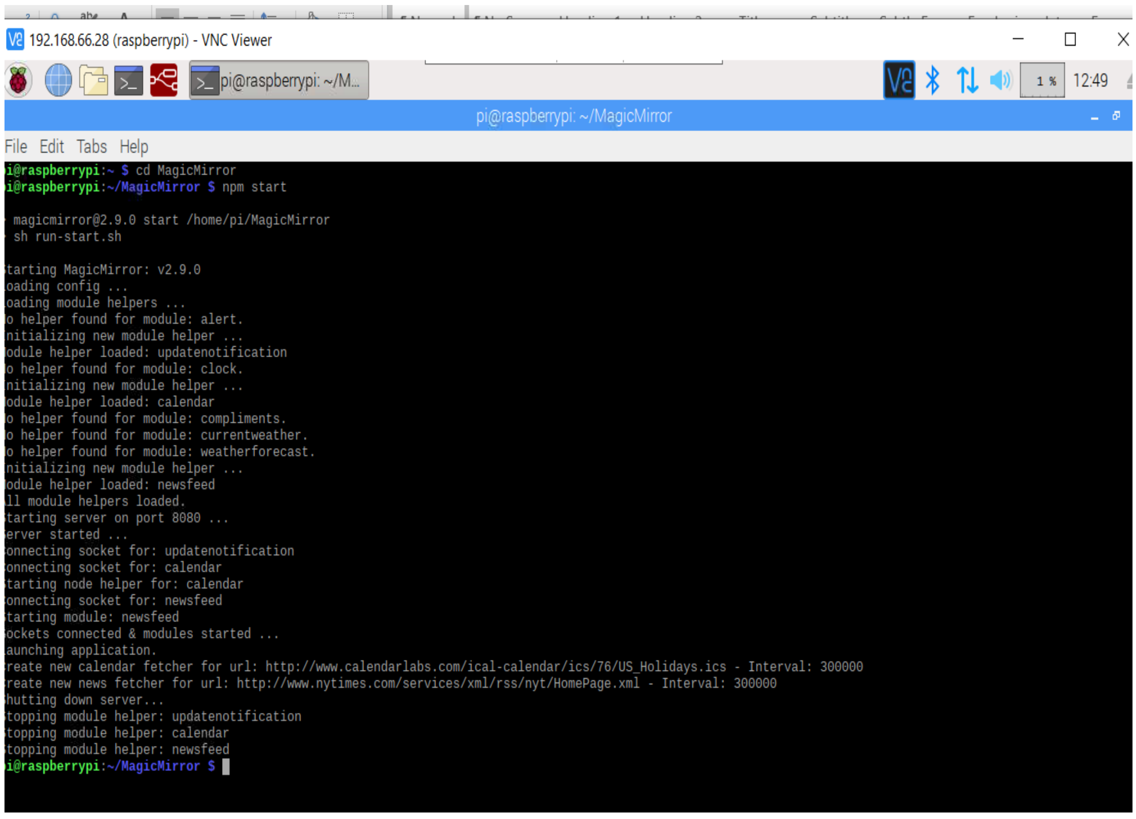
OK Cancel

Fig. 6.6 Connecting To Raspberry Pi



```
login as: pi
pi@192.168.0.112's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 13 02:07:03 2017 from 192.168.0.100

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $
```

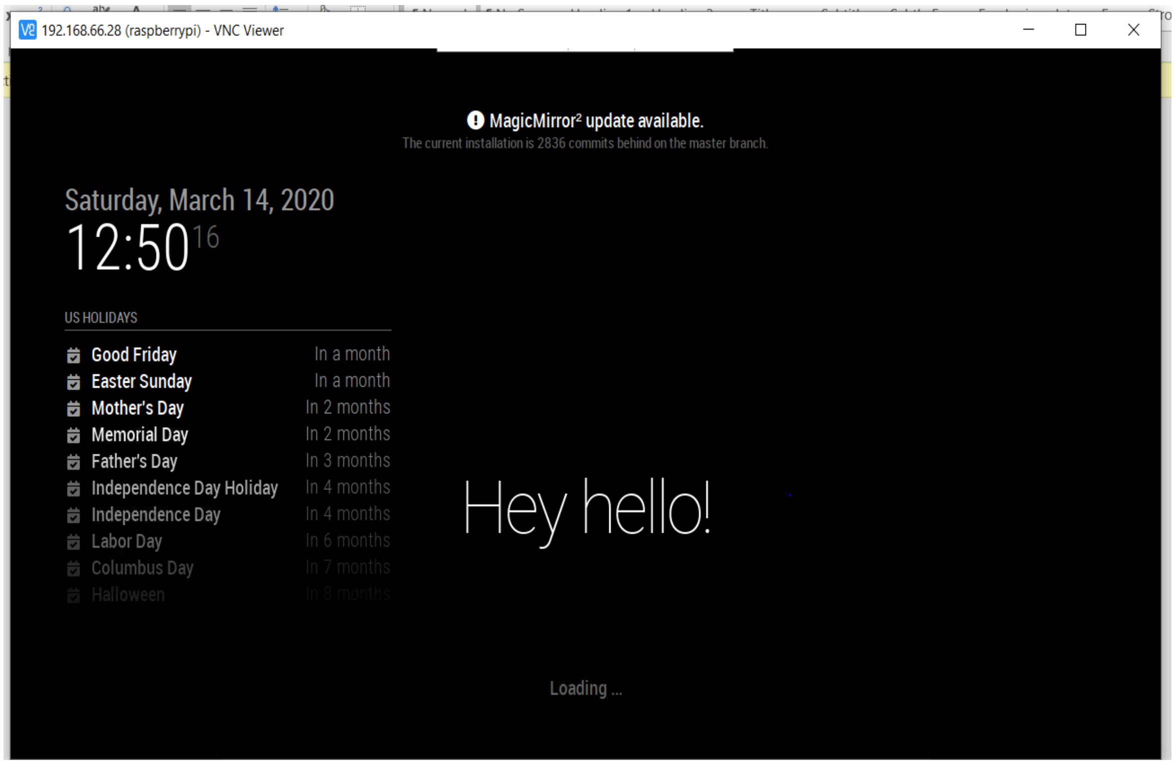
Fig. 6.7 Raspberry Pi Command Prompt



192.168.66.28 (raspberrypi) - VNC Viewer
pi@raspberrypi: ~/M...
12:49
pi@raspberrypi: ~/MagicMirror
File Edit Tabs Help

```
i@raspberrypi:~ $ cd MagicMirror
i@raspberrypi:~/MagicMirror $ npm start

 magicmirror@2.9.0 start /home/pi/MagicMirror
 sh run-start.sh

tarting MagicMirror: v2.9.0
oading config ...
oading module helpers ...
o helper found for module: alert.
nitializing new module helper ...
odule helper loaded: updatenotification
o helper found for module: clock.
nitializing new module helper ...
odule helper loaded: calendar
o helper found for module: compliments.
o helper found for module: currentweather.
o helper found for module: weatherforecast.
nitializing new module helper ...
odule helper loaded: newsfeed
ll module helpers loaded.
tarting server on port 8080 ...
erver started ...
onnecting socket for: updatenotification
onnecting socket for: calendar
tarting node helper for: calendar
onnecting socket for: newsfeed
tarting module: newsfeed
ockets connected & modules started ...
aunching application.
reate new calendar fetcher for url: http://www.calendarlabs.com/ical-calendar/ics/76/US_Holidays.ics - Interval: 300000
reate new news fetcher for url: http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml - Interval: 300000
hutting down server...
topping module helper: updatenotification
topping module helper: calendar
topping module helper: newsfeed
i@raspberrypi:~/MagicMirror $
```

Fig. 6.8 Redirecting To Folder



192.168.66.28 (raspberrypi) - VNC Viewer

MagicMirror² update available.
The current installation is 2836 commits behind on the master branch.

Saturday, March 14, 2020
12:50 16

US HOLIDAYS

Good Friday	In a month
Easter Sunday	In a month
Mother's Day	In 2 months
Memorial Day	In 2 months
Father's Day	In 3 months
Independence Day Holiday	In 4 months
Independence Day	In 4 months
Labor Day	In 6 months
Columbus Day	In 7 months
Halloween	In 8 months

Hey hello!

Loading ...

Fig. 6.9 Monitor Display

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION:

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. This system is working properly and meeting all user requirements.

7.2 FUTURE SCOPE:

This system can be further extended by adding new occupations and different techniques and made it available to all those who are very interested to learn.

REFERENCES

- http://www.geeksforgeeks.org
- https://opensource.com/resources/raspberry-pi
- https://maker.pro/raspberry-pi/projects/how-to-create-a-smart-mirror-using-raspberry-pi-and-magic-mirror
- https://magpi.raspberrypi.org/articles/build-a-magic-mirror
- https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams

GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)
Seshadri Rao Knowledge Village, Gudlavalleru

Department of Computer Science and Engineering

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions, component, or software to meet the desired needs.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs):

1. Apply the knowledge of circuit design, analog, digital electronics, coding, Arduino IDE.
2. Analyze, design and develop control systems, industrial drives and power systems using modern tools.

PROJECT PROFORMA

Classification of Project	Application	Product	Research	Review
		√		

Note: Tick Appropriate category

Project Outcomes	
Course Outcome (CO1)	Identify and analyze the problem statement using prior technical knowledge in the domain of interest.
Course Outcome (CO2)	Design and develop engineering solutions to complex problems by employing systematic approach.
Course Outcome (CO3)	Examine ethical, environmental, legal and security issues during project implementation.
Course Outcome (CO4)	Prepare and present technical reports by utilizing different visualization tools and evaluation metrics.

Mapping Table

CS2515 : MINI PROJECT															
Course Outcomes	Program Outcomes and Program Specific Outcomes														
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12		PSO 1	PSO 2
CO1	3	3	1		1		2	2	2	2				1	1
CO2	3	3	3	3	3	1		2	2	2		1		3	3
CO3	2	2	3	2	2	3	3	3	2	2	2			3	
CO4	2		1		3		1		3	3	2	2		2	2

Note: Map each project outcomes with POs and PSOs with either 1 or 2 or 3 based on level of mapping as follows:

1-Slightly (Low) mapped 2-Moderately (Medium) mapped 3-Substantially (High) mapped