A

Project Report

On

OBSTACLE DETECTION SYSTEM USING RASPBERRY PI

B. Tech. – Sem. VII

Bachelor of Technology

In

Information Technology

Submitted By:

Kruti J. Raval

Under the Guidance of

Prof. Vipul Dabhi

DEPARTMENT OF INFORMATION TECHNOLOGY

FACULTY OF TECHNOLOGY, DHARMSINH DESAI UNIVERSITY

COLLEGE ROAD, NADIAD- 387001

Table of Contents

CHAPTER 1

INTRODUCTON

1.1 Project Summary:

If you want to Detect any Obstacle in your way “Obstacle Detection System” is the best system for it!

This system is being built using Raspberry Pi Model B and Ultrasonic Sensor. This system can also be tracked using Android Application.

Obstacle Detection System is intended to help the users find a obstacle in their path which can be helpful to a blind person as well as in reverse driving of car. This document is meant to delineate the features of Obstacle Detection System, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other.

1.2 Purpose:

Obstacle Detection System is intended to help the users in following Functionalities:

- User can be alert of obstacle with the help of buzzer.
- Distance from the obstacle can be known at any point of time.
- User have different Buzzer sounds for diff. proximity of distances.
- User can know the Location of the System from anywhere.
- User can access the application from anywhere.

The main purpose of "Obstacle Detection System" is to provide good and easily accessible interface to the users specially blind people and their closed ones who can track them.

1.3 Scope:

a. We can directly notify user to reach to the location of the system if hit with an obstacle .

b. In future we will continue to add new features and updates.

1.4 Objective:

a. Every User gets an unique identity.

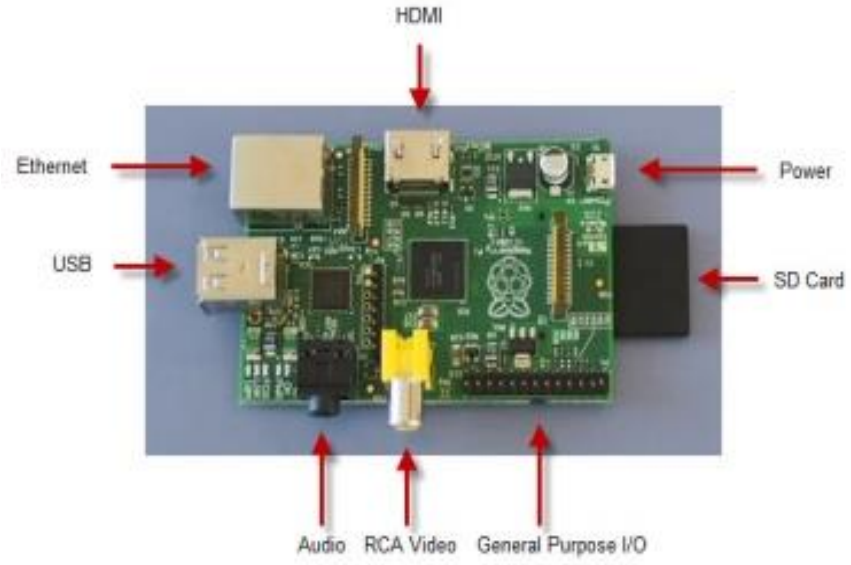
b. User is able to locate System and find the location of it.

c. User can also know the distance of the obstacle with the help of Ultrasonic Sensor.

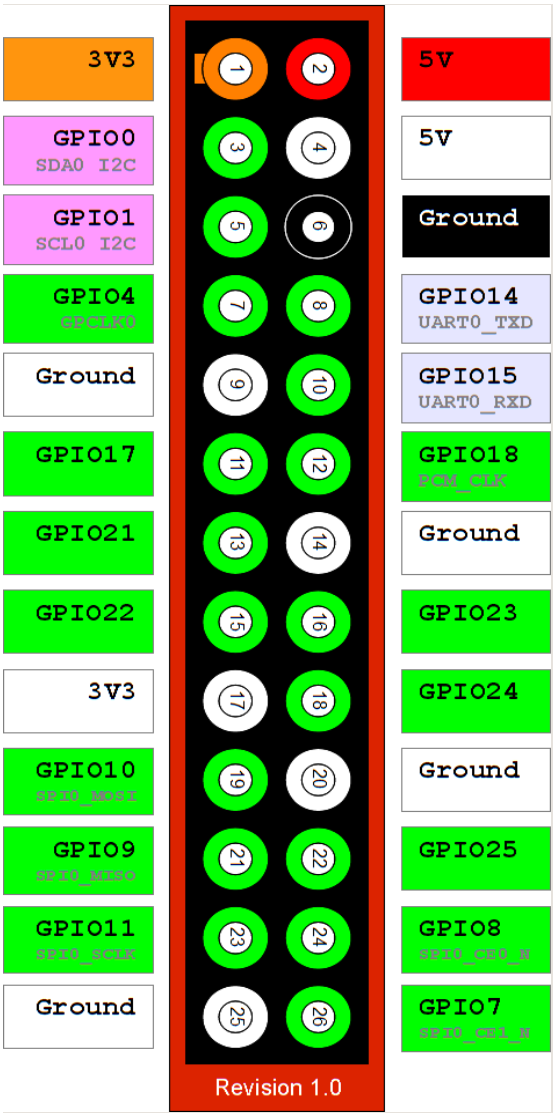
d. User can also get to know the range of closeness with the obstacle with the help of buzzer..

1.5 Technology And Literature Review:

RASPBERRY PI- The Raspberry Pi is a series of credit card-sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and developing countries. The Raspberry Pi is a credit card-sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high-definition video. We want to see it being used by kids all over the world to learn programming. Model B is the higher-spec variant of Raspberry Pi 1 (superseded by Raspberry Pi 2 Model B), with 512 MB of RAM, two USB ports and a 100mb Ethernet port.



GPIO PINS - One powerful feature of the Raspberry Pi is the row of GPIO (general purpose input/output) pins along the edge of the board, next to the yellow video out socket. These pins are a physical interface between the Pi and the outside world. At the simplest level, you can think of them as switches that you can turn on or off (input) or that the Pi can turn on or off (output). Seventeen of the 26 pins are GPIO pins; the others are power or ground pins.



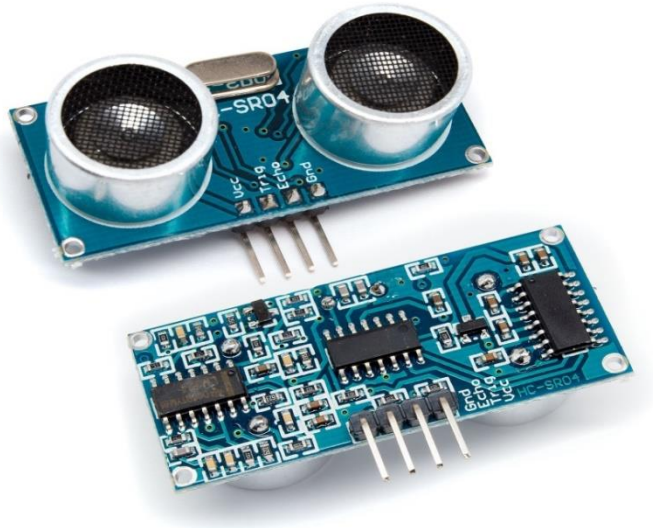
ULTRASONIC SENSOR (HC-SR04) - The Ultrasonic Sensor sends out a high-frequency sound pulse and then times how long it takes for the echo of the sound to reflect back. The sensor has 2 openings on its front. One opening transmits ultrasonic waves, (like a tiny speaker), the other receives them, (like a tiny microphone).

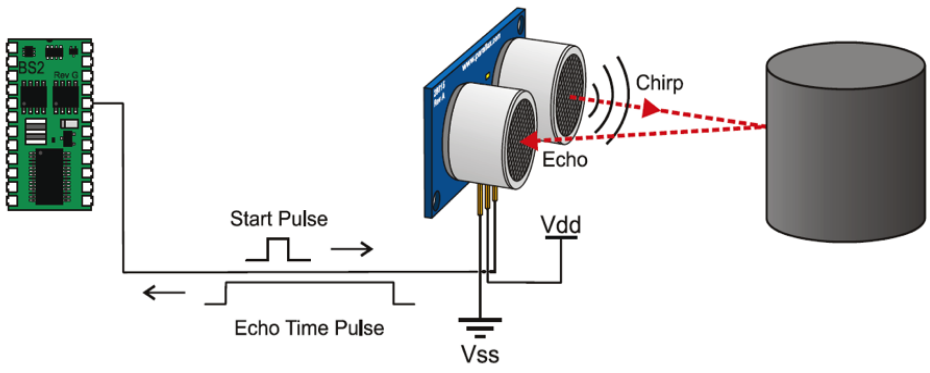
The speed of sound is approximately 341 meters (1100 feet) per second in air. The ultrasonic sensor uses this information along with the time difference between sending and receiving the sound pulse to determine the distance to an object. It uses the following mathematical equation:

Distance = Time x Speed of Sound divided by 2

Time = the time between when an ultrasonic wave is transmitted and when it is received

we divide this number by 2 because the sound wave has to travel to the object and back.





A chirp is emitted from the “speaker.” It bounces off of an object. The echo returns to the microphone. The time it takes to travel to the object and back is used to figure out the distance.

BUZZER- A buzzer or beeper is an audio signaling device, which may be mechanical,Electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm Devices, timers and confirmation of user input such as a mouse click or keystroke.

ANDROID-Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input.

CHAPTER 2

PROJECT MANAGEMENT

2.0 PROJECT MANAGEMENT

2.1 FEASIBILITY STUDY

The feasibility of software can be tested in for dimensions: **Technology**-is a project technically feasible? Our project is feasible and can be installed easily. **Finance**-Is it financially feasible? Does it have too much cost of development? Raspberry pi is a bit costly but it has its own advantages. **Time**-will it take too much time to complete? We have planned each phases and it seems to be controlled and within time, so no extra time cost will be added. **Resources**-Do we have sufficient resource to succeed? Raspberry Pi along with Ultrasonic Sensor and Buzzer is available.

There are four categories of feasibility tests: operational feasibility, technical feasibility, schedule feasibility and economic feasibility. Luckily we are able to meet all the tests successfully.

2.1.1 Technical Feasibility

The technical feasibility means that the project can be done with the current equipment, existing software technology and the current knowledge.

Our system is technically feasible. It is developed using Raspberry pi with Python language for coding and ultrasonic sensor which altogether makes our system technically feasible.

2.1.2 Operational Feasibility.

Operation feasibility deals with the acceptance of the users and their willingness to use the system. The system should be such that it is acceptable by all its users and no user hesitates to use it.

The system is such that it facilitates all the users of the system. The System is user friendly and can be used easily by any People.

2.1.3 Implementation Feasibility.

This system is built in Raspberry Pi with use of Ultrasonic Sensor and Buzzer. We cannot find any problem while implementing the project in this Technology. So system is feasible for implementing.

2.1.4 Time Schedule Feasibility.

The Project has simple working and the basic requirement can be satisfied within allotted time period so the time development feasibility is satisfied.

2.1.5 Cost Analysis.

Our team's estimation of the initial cost has been determined based on the individual components we have deemed necessary. The primary cost will be the hardware components. Each individual hardware component must be purchased separately.

Components	**Cost (in Rupees)**
Raspberry Pi	**3000**
Ultrasonic Sensor	**120**
Buzzer	**20**
Ethernet Cable	**100**
USB Cable	**50**
Breadboard and GPIO connectors	**100**
TOTAL	**3390**

2.2 PROJECT PLANNING

2.2.1 Project Development approach and justification.

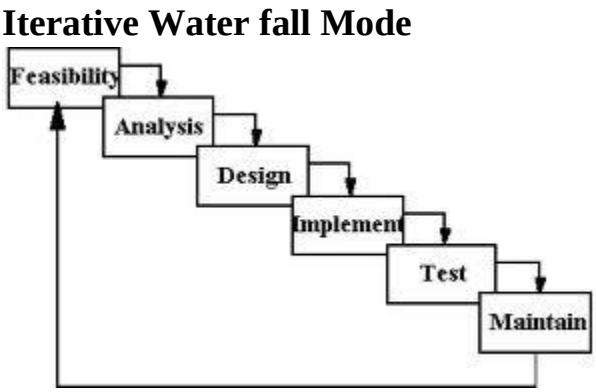


Figure 2.1 Waterfall Model

1. Requirements specification
2. Design
3. Construction (implementation or coding)
4. Integration
5. Testing and debugging
6. Installation
7. Maintenance

2.2.2 Project Plan

1. Gather the module definition.
2. Checking the time schedule feasibility.
3. Requirement gathering for module.
4. Analysis on gathered requirement.
5. Designing.
6. Coding.
7. Testing.
8. Management

2.2.3 Milestones and Deliverables

Management needs information. As software is intangible, this information can only be provided as documents that describe that state of the software being developed. Without this information, it is impossible to judge progress and cost estimates and schedules cannot be updated. When planning a project series of milestones are established.

Milestones:

I. Milestone is an end-point of the software process activity.
II. At each milestone there should be formal output, such as report, that can be represented to the management. The weekly report is submitted to project guide, which include day to day work report.
III. Milestone represents the end of the distinct, logical stage in the project.

Deliverables:

I. Deliverables is a project report that is delivered to the administrator of the project.
II. Deliverables are delivered to the administrators of our organization at the end of the some major project phase such as specification, design, etc.
III. Deliverables are usually milestone
IV. Milestones may be internal project results that are used by the project manager to check progress but which are not delivered to the administrator.

2.2.4 Roles & Responsibilities

Name	Role				
	Analysis	Designing	Coding	Testing	Documentation
Kruti Raval	✓	✓	✓	✓	✓
Tejas Shahu	✓	✓	✓	✓	✓

Table 2.1 Roles and Responsibilities

CHAPTER 3

SYSTEM REQUIREMENT STUDY

3.0 SYSTEM REQUIREMENT STUDY

3.1 Study of current system

The current system has a feature which allows user to know the distance from obstacle and also location of the system.

3.2 Problem and weaknesses of the current system

This system requires constant access of the internet if they want to track the location of the system.

3.3 User characteristics

This System is ideal for Blind users and users who want to know any obstacle in their way and the ones who want to know their location.

3.4 Hardware and software requirements

3.4.1 Software requirements

1. Android Studio
2. Raspbian OS

3.4.2 Hardware requirements

1. Raspberry Pi 1 Model B
2. Ultrasonic Sensor (HC-SR04)
3. Buzzer
4. Ethernet cable
5. USB cable
6. GPIO connectors
7. 1kΩ Resistor and 2kΩ Resistor

3.5 Constraints

3.5.1 Hardware Constraints

Ultrasonic Sensor HC-SR04 can measure the distance of obstacle in the range of 4cm to 400 cm and 30 degrees wide.

The user must possess an Android mobile having the package installer feature.

3.5.2 Higher Language Order Constraints

The application requires the front end to be the Android structure.

3.5.3 Reliability Requirements

The application must adhere to the reliability requirements and should run smoothly in the device.

3.5.4 Safety and Security Consideration

The application does not access any feature of the mobile which may cause it to compromise on the user's security or personal information.

CHAPTER 4

SYSTEM ANALYSIS

4.1 REQUIREMENTS OF NEW SYSTEM (SRS)

4.1.1 Functional Requirements

This section of the document illustrates different functions provided by the Application:

R1 : Obstacle Detection

Description: The Ultrasonic Sensor will sense if there is obstacle present or not.

Input: The input will be in the form of Sonar from Ultrasonic Sensor.

Output: According to distance between obstacle and system, it will give the distance in form digital number.

Process: The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. It operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

R2: Alert the User

Description: There will be one buzzer which will inform the user.

Input: It will take distance as input.

Output: It will buzz the buzzer according to distance between obstacle and system.

Process: Compare the distance like obstacle is nearby or not according to that buzzer will alert with user with different Intensity.

R3: Locate the System using Android Application

Description: There will be one android application which can track system on the basis of IP address.

Input: Android application will take latitude and longitude as an input.

Output: Google Map will show the location of the system.

Process: Android Application is tracking System on the basis of IP address. Code will send the IP address to http://freegeoip.net/json website. This website is returning the latitude and longitude as an output. Using Android Application it is fetching these two values from server database and giving location on Google Maps.

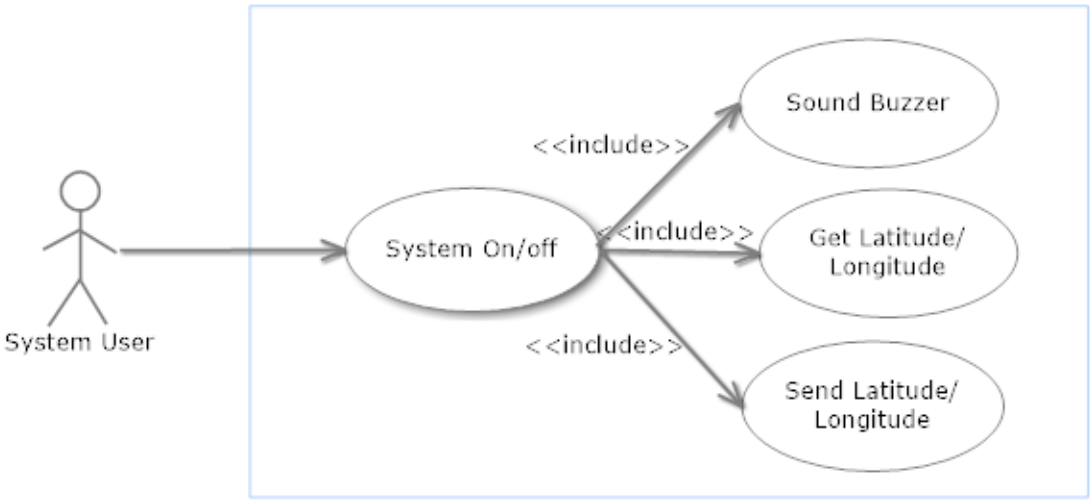
4.2 Features of new system

The features of new system are like following:

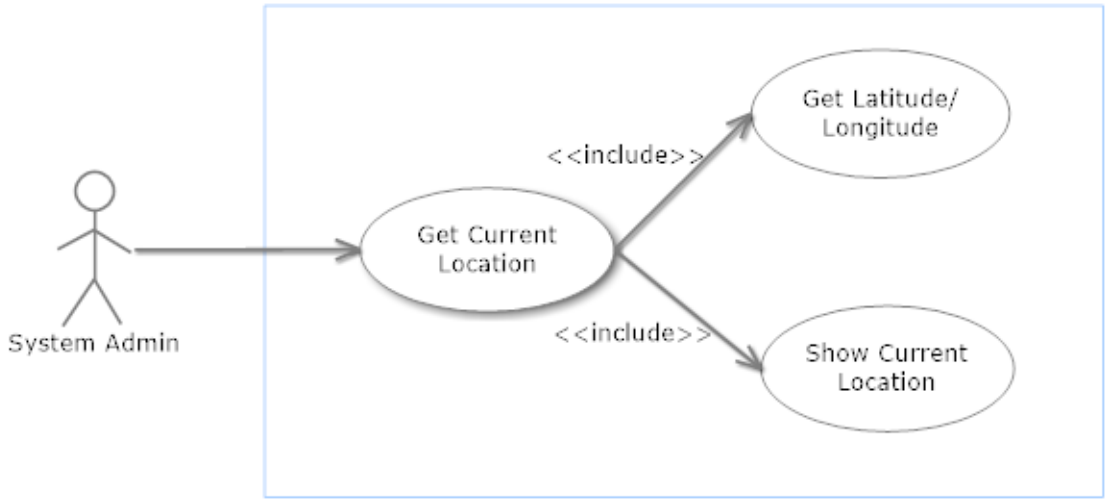
- User can know the Proximity of distance of obstacle.
- User will be able to locate the system.

4.3 Use case Diagram

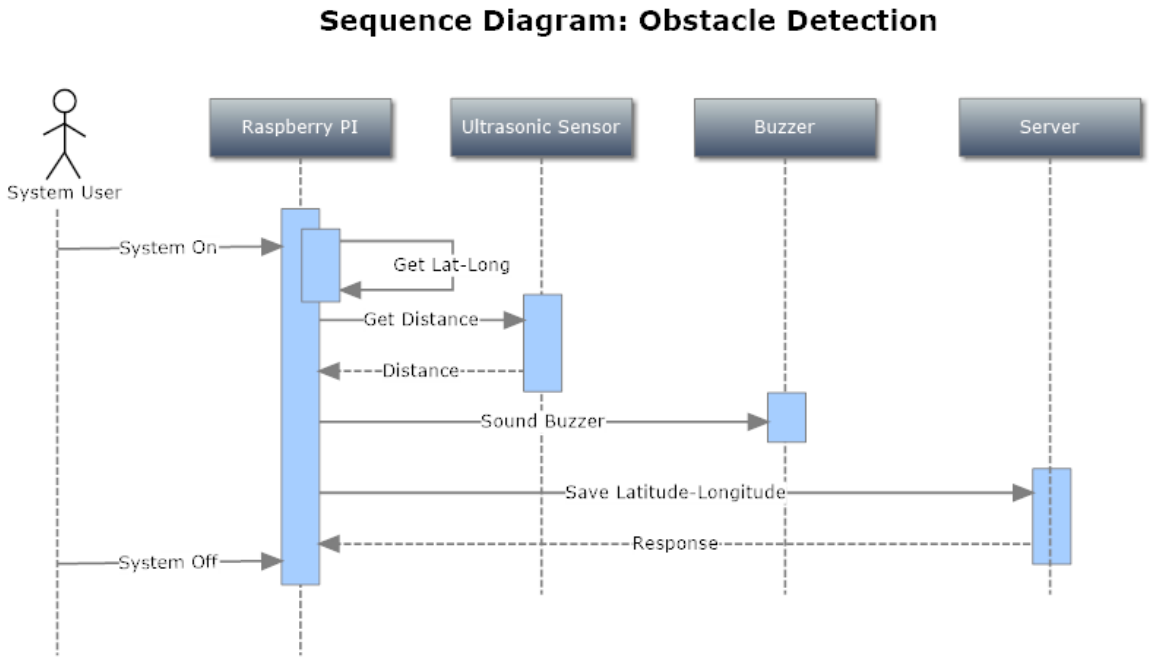
Use Case Diagram: Obstacle Detection

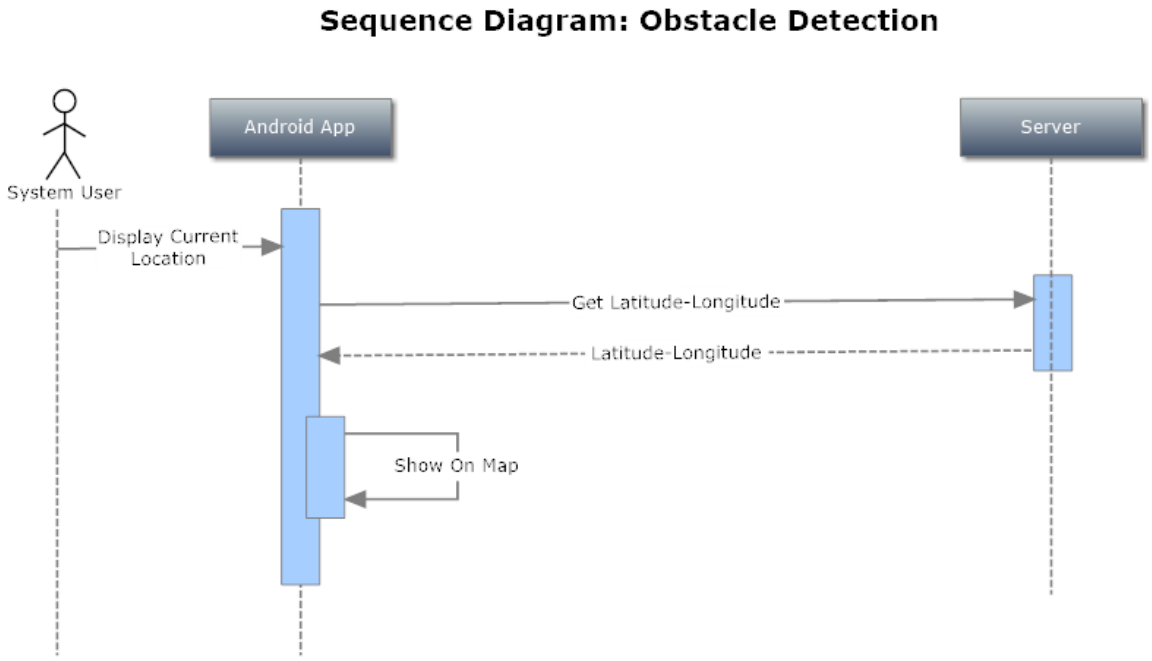


Use Case Diagram: Android App Obstacle Detection



4.4 Sequence Diagram





CHAPTER 5

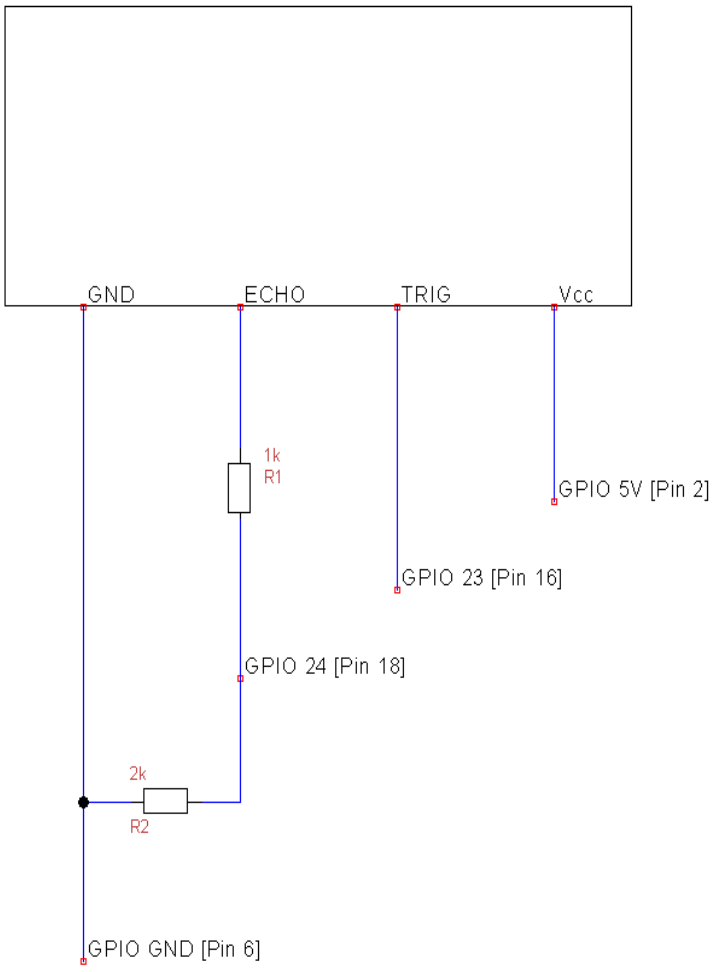
SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE DESIGN:

Assemble the Circuit:

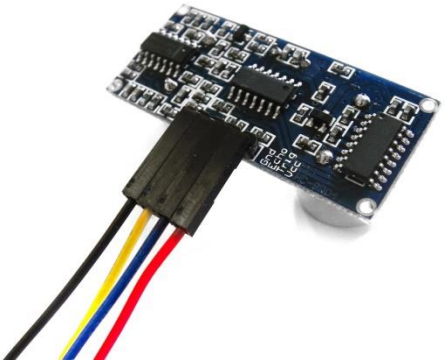
We'll be using four pins on the Raspberry Pi for this project: GPIO 5V [Pin 2]; Vcc (5V Power), GPIO GND [Pin 6]; GND (0V Ground), GPIO 23 [Pin 16]; TRIG (GPIO Output) and GPIO 24 [Pin 18]; ECHO (GPIO Input).

Connection of Ultrasonic Sensor with Raspberry Pi GPIO pins:



Connecting Raspberry Pi with Sensor HC-SR04:

1. Plug four of your male to female jumper wires into the pins on the HC-SR04 as follows: Red; Vcc, Blue; TRIG, Yellow; ECHO and Black; GND.



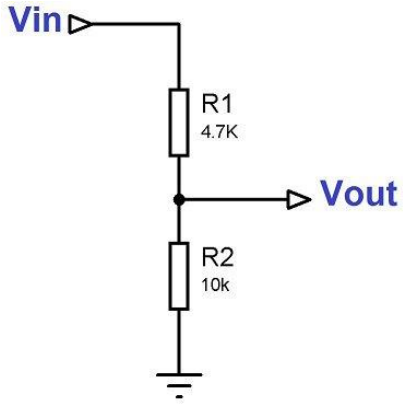
2. Plug Vcc into the positive rail of your breadboard, and plug GND into your negative rail.

3. Plug GPIO 5V [Pin 2] into the positive rail, and GPIO GND [Pin 6] into the negative rail.

4. Plug TRIG into a blank rail, and plug that rail into GPIO 23 [Pin 16]. (You can plug TRIG directly into GPIO 23 if you want).

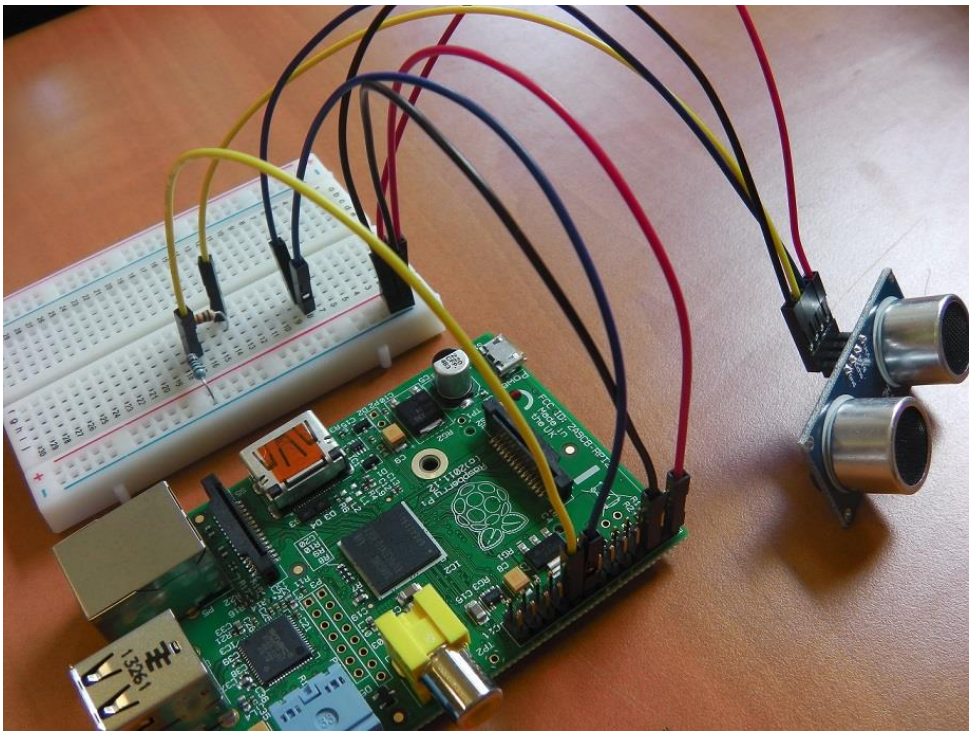
5. Plug ECHO into a blank rail, link another blank rail using R1 (1kΩ resistor)

6. Link your R1 rail with the GND rail using R2 (2kΩ resistor). Leave a space between the two resistors.



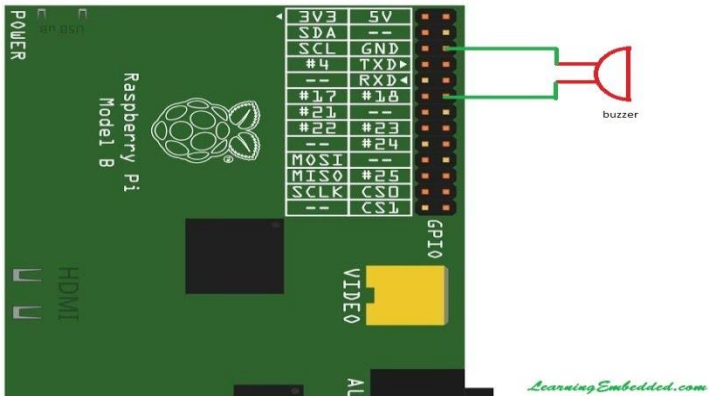
7. Add GPIO 24 [Pin 18] to the rail with your R1 (1kΩ resistor). This GPIO pin needs to sit between R1 and R2.

8. Sensor is now connected to Raspberry Pi.



Connecting Buzzer to the System:

1. Connect the piezoelectric buzzer as shown in figure.
2. The negative wire of Buzzer must be connected to the Ground [PIN 6] and the positive end to GPIO pin 12.



$$Speed = \frac{Distance}{Time}$$

$$34300 = \frac{Distance}{Time/2}$$

$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

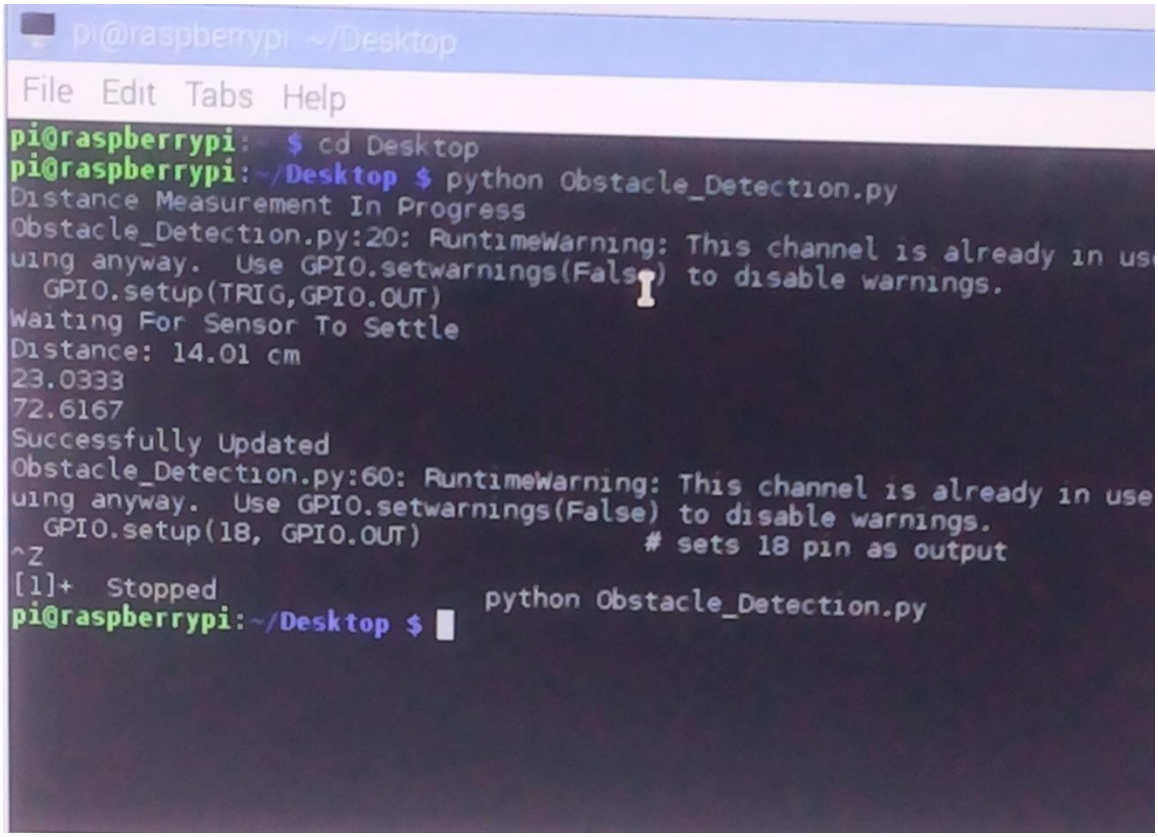
Thus from the above formula we can know the Distance of the Obstacle.

Sensorchec.py

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)

TRIG = 23
ECHO = 24

print "Distance Measurement In Progress"

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

GPIO.output(TRIG, False)
print "Waiting For Sensor To Settle"
time.sleep(2)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO)==0:
  pulse_start = time.time()

while GPIO.input(ECHO)==1:
  pulse_end = time.time()

pulse_duration = pulse_end - pulse_start

distance = pulse_duration * 17150

distance = round(distance, 2)

print "Distance:",distance,"cm"

GPIO.cleanup()
```

After executing the code **obstacleDetection.py** the output of Raspberry Pi will be as follows:

```
pi@raspberrypi ~ $ sudo python range_sensor.py
Distance Measurement In Progress
Waiting For Sensor To Settle
Distance: 12.52 cm
pi@raspberrypi ~ $
```



CHAPTER 6

TESTING

6.1 TESTING PLAN

What is 'Software Testing'?

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they don't happen when they should. It is oriented to 'detection'.

The need for Testing:

No matter how good a programmer is, no application will ever be one hundred percent correct. Testing was important to us in order to ensure that the application works as efficient as possible and conforms to the needs of the system. Testing was carried out throughout the development of the application, not just the application has been developed, as at this stage it took a great deal of effort to fix any bugs or design problems that were occurred.

6.2 TESTING STRATEGY

When our application was configured and customized in the system, the test was observed that this configuration or customization does not cause any improper processing or violation. The following care was taken when the application was developed at the local machine. The interface may have something not proper, which can be tested by this checklist:

- Number of input parameter equal to number of argument?
- Parameter and argument attributes match?
- Number of arguments transmitted to called forms equal to number of parameters?
- Attributes of arguments transmitted to called forms to attributes of parameters?
- Number attributes and order of arguments to built-in functions correct?

- The local data structures for a form are common source of errors. The following types of errors should be searched for,
- Improper or inconsistent typing
- Erroneous initialization or default values
- Incorrect (misspelled or truncated) variables names
- Inconsistent data types
- Underflow, overflow and addressing exception
- As far as unit testing is concerned we did it at the time of coding in an informal but extensive way, so as to reduce number of problems arising out of incorrect syntax, incorrect variable, function names etc.
- Close the database connection when not required.
- Care was taken to check for any infinite loop that exists in code before executing the code.

6.3 TESTING METHODS

While Box Testing

Also known as glass box, structural, clear box and open box testing. A software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do, it means that he must be completely aware that for particular input a particular output must be obtained. The main benefit of this type of testing is Tester can see if the program diverges from its intende
d goal. This test concentrates on the examination of the code rather than the specification. We have included three different forms of white box testing.

Statement Coverage Criterion:

This is the simplest coverage criterion. We are checking in it that each statement of the Program was executed “at least once”.

Branch Coverage Criterion:

An improvement over statement is **Branch Coverage**. In that we are running a series Of test to ensure that all branches are tested at least once.

Path Coverage Criterion:

There are many errors which were not detected by statement or branch testing. The reason is that some errors are related to some combination of branches and it may be not check in other test. We are checking in this test is all path of programs are executed or not.

Black Box Testing

Black-box and white-box are test design methods. Black-box test design treats the system as a "black-box", so it doesn't explicitly use knowledge of the internal structure. Black-box test design is usually described as focusing on testing functional requirements. Also known as behavioral, functional, opaque-box and closed-box.

Black Box Testing was helpful us to find error such as:

- Interface error
- Incorrect or missing functions.
- Errors in data structures or external database access.
- Performance Errors.

Unit Testing

Unit testing is a method of testing the correctness of a particular module of source code. The idea is to write test cases for every non-trivial function or method in the module so that each test case is separate from the others if possible. The developers mostly do this type of testing. In this method of testing we test all individual components

to ensure that they operate correctly. Each component are tested independently without other system components.

Integration Testing

It is the phase of software testing in which individual software modules we are combined and tested as a group. It follows unit testing and precedes system testing. The purpose of Integration testing is to verify functional, performance and reliability requirements placed on major design items.

It takes as its input modules that have been checked out by unit testing, groups them in larger aggregates, applies tests defined in an Integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

CHAPTER 7

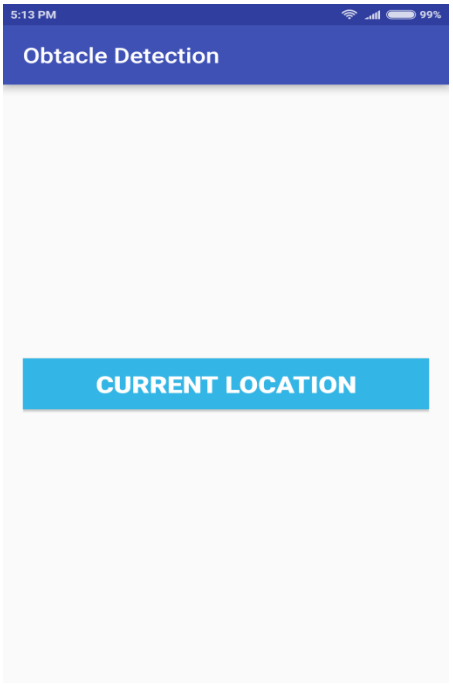
USER MANUAL

7. USER MANUAL

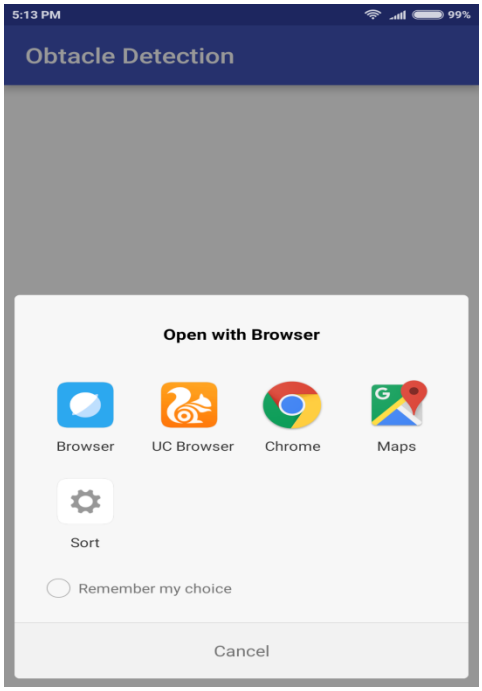
A user guide or user's guide, also commonly known as a manual, is a technical Communication document intended to give assistance to people using a particular system. It is usually written by a technical writer, although user guides are written by programmers, product or project managers, or other technical staff, particularly in smaller companies. User guides are most commonly associated with electronic goods, computer hardware and software. Our user guides contain both a written guide and the associated images. In the case of our application, it is usual to include screenshots of how the program should look. The language used is matched to the intended audience. We have prepared our user manual according to various user roles and module wise, so a novice user can understand system very quick.

Execution of Application:

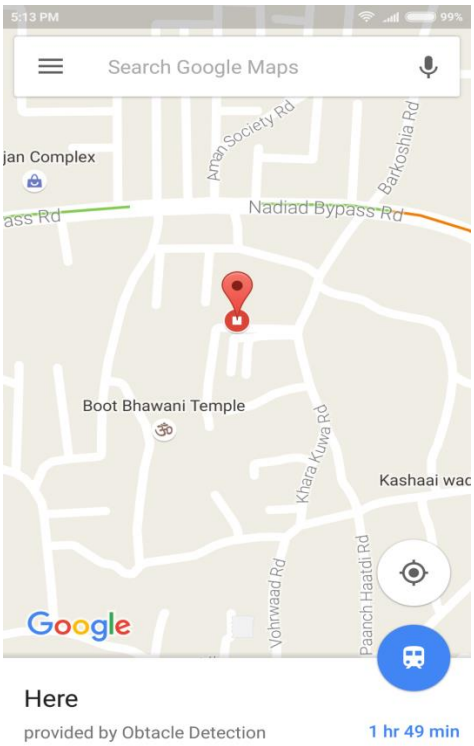
(1)Starting up the application



(2) Opening Location using app:



(3)Current Location in Google Map:



CHAPTER 8

LIMITATION & FUTURE ENHANCEMENT

8.1 LIMITATION

1. Ultrasonic Sensor can measure distance from 4cm to 400cm only.
2. Raspberry Pi needs constant power supply of 5volt.
3. Raspberry Pi must be plugged in with Internet for tracking this device into Android Application.
4. http://freegeoip.net/json website must be running because it is converting and giving required data to Android Application.

8.2 FUTURE ENHANCEMENT

1. High Range Ultrasonic Sensor will be used for better result.
2. GUI must be improved in future.
3. Mail will be received in mail account containing the data of last location of device.
4. Our application is efficient in RAM usage.
5. Our application's startup and loading must be as fast as possible.
6. Maps should be shown in the same application.

CHAPTER 9

CONCLUSION AND DISCUSSION

9.1 CONCLUSION AND FUTURE ENHANCEMENT

In the end, I concluded with these requirement cards.

1. A System like this is much needed today to help the blind people so that they can walk independently.

2. An application needs to be scalable as not only blind people but also in Reverse parking of car or in any automated machines.

3. Application needs to be error free, easy to use & reliable.

4. Maximum users use Android Kitkat or above. Hence priority would be to make an application compatible with Kitkat or above it. The feature of offers can decide upon what visitors buy in a mall since a majority of them do not have a predicated wish-list.

9.2 DISCUSSION

9.2.1 Self-Analysis of Project Viabilities

According to me, this project is completed with the primary functionalities as specified earlier but then again there is lot more than this which can be done. The project is well capable to handle the given job for some particular task but not all of them. So then it is a challenge to further develop it in to well flag product as it was challenge to develop up to this very stage.

9.2.2 Problem Encountered and Possible Solutions

There were many problems encountered during the design and the development phase of the project.

- The problem in fetching an API and getting the response.

9.2.3 Summary of Project work

We have completed my project work using software engineering and system Analysis and design approach. We have done work with preplanned scheduling related with time constraints and result oriented progress in project developmen

CHAPTER 10

REFERENCES

1. www.raspberrypi.org
2. www.stackoverflow.com
3. www.wikipedia.com
4. www.google.com
5. www.modmypi.com