

# Chapter 1

## Introduction

---

1.1 Overview

1.2 Project Motivation

1.3 Main Idea

1.4 Objectives

1.5 Idea and approach

1.6 Block diagram

1.7 Requirements

1.8 Challenges

1.9 Related Work

1.10 Project Plan

1.11 Estimated Cost and Budget

1.12 Report Content

## **1.1 Overview**

In this chapter, we provided general information to understand our project. This includes an explanation of main idea, motivations, and the project objectives. After that a look at some related work was given.

## **1.2 Project motivation**

What inspired us for doing this project, was that we almost every day forget where we put our things or misplaced things, so spending a lot of time to find it, on average about 15 minutes each day searching for items like cell phones, keys and remote control, according to an online survey of 3,000 people published in 2012 by a British insurance company.<sup>(1)</sup>

Now what if this faces blind people you could imagine how much this will be hard for them especially if there is no one to help them.

To overcome this irritating problem we started searching on things can help us doing that, so we read about the RFID technology and this was the first time we hear about it.

## **1.3 Main Idea**

We intended to build a system that can help blind people to find their lost thing using RFID technology; this technology can locate any object attached with a tag, to achieve simplicity in use the blind will use voice order for searching.

## **1.4 Objectives**

We aimed in our project to achieve several objectives:

- 1) We would like to build a system makes the life of blind people easier.
- 2) Blind self-confidence development, through finding their things by themselves without assistance from others.
- 3) Reduce the time and the effort required to identify the position of the lost objects.

## 1.5 Idea and approach:

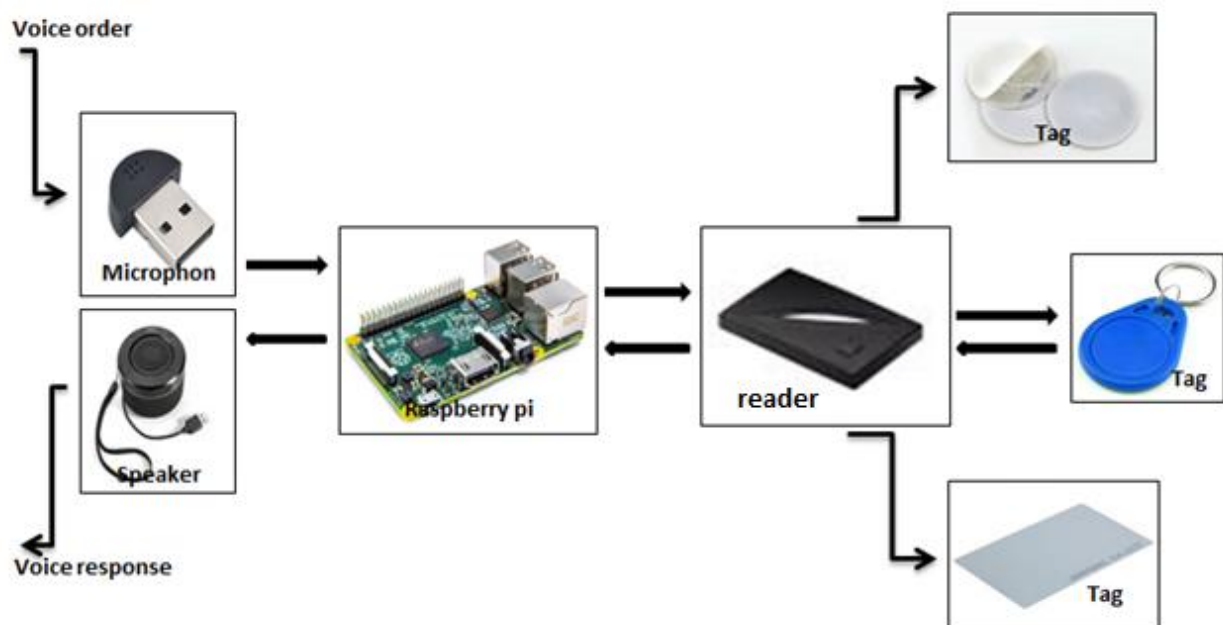
The object idea is standing on finding a location of an object using radio frequency identification (RFID) technology to help blind people, and this idea will be achieved as follow:

The system provided the blind an easy way to search through voice orders describe the wanted object, using voice recognition that performed at Raspberry Pi connected to RFID reader. When Raspberry Pi determine the RFID tag that is attached with required object it gives the reader ID for this tag.

The reader uses electromagnetic waves to automatically identify and track tags attached to objects. These waves will reach all tag exist in its range, after the tags response and give the reader the information that it carry, the Raspberry Pi will select the ID value for the wanted object, then the result will convert to voice to be audible for the blind.

## 1.6 Block diagram

Here is the main block diagram of the project as shown in figure1.1. It includes all components used in this project. We used microphone and speaker to in and out the voice, raspberry pi for processing and reader to scan the tags.



**Figure1.1:** Main block diagram

## **1.7 Requirements**

In order to design and implement the targeted system, the following hardware and software requirements were needed:

Hardware:

- RFID readers.
- RFID tags.
- RFID antennas.
- Speaker.
- Microphone.
- Raspberry Pi's.
- Wi-Fi adapter.

Software:

Programming software by using python language to program the raspberry pi which responsible for many task.

## **1.8 Challenges:**

We faced many challenges in this project and we summarized them below:

- 1) Availability of components, where we couldn't bring the Cottonwood RFID reader whose reading range is 6 m.
- 2) Dealing with new technology (RFID technology).
- 3) Make a RFID network.

## **1.9 literature review (Related Work):**

Radio Frequency Identification (RFID) becomes an extremely powerful enabling technology in many fields like electronic passport, animal tracking, supply chains, industrial automation, mining securities, hospital, asset management and pharmaceuticals. This widely using of it is due to its ease and efficiency in communication. Some related project and papers were studied and summarized to shown that.

### **1.9.1 Long Range UHF RFID Item Tracking System<sup>(2)</sup>**

This project aims to provide a method for tracking tools and other valuable items in workspaces, dens and garages or other places. By using Long Range Ultra High Frequency RFID technology to tag and monitor items.

Their system used Raspberry Pi 2, UHF RFID reader and Long Range UHF RFID antenna with passive RFID stickers jointly with Web Api service to make a tracking system by allow the consumer to register their RFID readers (Monitors) and Tags, also view the last known location of an item.

#### **Differences from our projects:**

- This project used a long range UHF RFID reader and we used a complete deferent reader whose range about 10 cm due to unavailability of UHF RFID reader, which mean dealing with it in a deferent way.

### **1.9.2 Design and implementation of library books search and management system using RFID Technology<sup>(3)</sup>**

Their system suggest an intelligent book search and management system based on RFID and Wi-Fi technology, using this system the location of the book precisely can be shown by a clear route guide picture promptly and accurately.

The system consists of books with HF passive RFID e-book tags, HF passive RFID readers, Wi-Fi networks, system server and customer devices. The HF passive RFID e-book tags are attached to every book recording relevant information like name, author and publishing date, and recorded data can be sent to RFID reader; HF passive RFID readers attached to each layer or case of the bookshelf can read the information sent through RFID e- tags, and pass the read information and bookshelf position to system server via Wi-Fi; the system server, RFID readers and terminal devices should be in the same Wi-Fi network range.

#### **Differences from our projects:**

- We used almost the same idea but in other field through employ this to help blind people and add voice order for searching, and changing the whole process.

- **1.9.3 An Indoor Localization System Based On Backscatter RFID Tag<sup>(4)</sup>**

The purpose of this paper is to introduce a prototype indoor localization system with high precision achieved at low cost (both in system price and computational price), by combining the angle of arrival (AOA) and phase of arrival (POA) methods to achieve the passive tag's localization by using trigonometry.

The system use one reader and consists of an antenna, RFID tags and a pan-tilt unit, and achieves mean accuracy of 23 cm but all their experiments are based on the assumption that the reader and the tag are on the same height. Therefore, the position estimation is in 2-dimensional. Rather than applying an antenna array or a virtual antenna array to estimate the AOA.

**Differences from our projects:**

- Because we have used a short range RFID reader we couldn't use neither of AOA or POA, so we depended on the data in the database to locate the tag place.

## **1.10 Project plan**

The project contains following stages:

### **Stage 1: Preparing the project**

The idea of the project is selected. Then required information has been collected. Discussion with supervisor and dividing task between the group members will happen.

### **Stage 2: Analysis and overview**

Here, a deep and complete study for all point of the project had made and a study of all possible design option to determine our own design.

### **Stage 3: Determine the project requirement**

After determine our design scheme we specify the entire needed requirement for the system, software and hardware.

### **Stage 4: Study of the principle**

In this stage, we studied how RFID system work, principle of Raspberry Pi and any other technologies or information needed.

### Stage 5: Documentation and writing

Writing and preparing the documentation of the project was start from the first stage, and continue until the end of the project.

### Stage 6: Raspberry pi programming

Run the raspberry pi and learn the python language for programming it.

### Stage 7: Software and hardware implementation

Install the required software on the raspberry pi and run them.

### Stage 8: System measurement

Here we built the project and made some testing.

### Stage 9: System testing

Here the system tested to conclude the system performance.

### Stage 10: Writing documentation

The documentation continued from the first phase till the end in parallel.

**Table 1.1** Timing plane for the first semester

Test/week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1															
S2															
S3															
S4															
S5															

**Table 1.2** Timing plane for the second semester

Test/week	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
<b>S6</b>															
<b>S7</b>															
<b>S8</b>															
<b>S9</b>															
<b>S10</b>															

### 1.11 Estimated cost and budget

The whole estimated cost will be approximately 180 JD, and the table below shows the cost of each hardware component.

**Table 1.3:** Estimated cost and budget

Category	Number of pieces	Price per piece	Total price
<b>RFID reader</b>	2	50	100
<b>RFID tag</b>	10	3	30
<b>USB microphone</b>	1	5	5
<b>USB speaker</b>	1	5	5
<b>Power bank</b>	2	20	40
<b>Total = 180 JD</b>			



## 1.12 Report contents

This project introduction is mainly divided into six chapters; each of them describes specific part of the project as following:

**Chapter One:** Includes the introduction, provides a general overview about the project, its objectives, importance, related works, challenges, time planning, estimated cost at the end of the report content.

**Chapter Two:** Discusses the theoretical background. It starts with general information about the project, the main component and types of RFID system, Wi-Fi adapter, speaker, and microphone and then discusses the important aspect of the system including Raspberry Pi microcontroller.

**Chapter Three:** Presents the general system design concept. It includes system objectives, general system block diagram, description of system design 'component and operation'.

**Chapter Four:** Shows the system design in detail. It includes all system hardware and software installation and their role in the project.

**Chapter Five:** It contains the result of searching and replying processes, testing to the whole system, and performance.

**Chapter Six:** This chapter will consider system achievement, real outcome, conclusion, and recommendation for developing the system in future.

# Chapter 2

## **Theoretical Background**

---

2.1 Overview

2.2 RFID

2.3 Raspberry Pi

2.4 Microphone

2.5 Speaker

2.6 Wi-Fi communication

2.7 Voice recognition

2.8 SQL database

## 2.1 Overview

In this chapter various technologies were discussed that used in our project, where each technology briefly explained and viewed in terms of advantages and disadvantages.

First RFID technology is discussed showing its importance, positives and negatives. Then Raspberry Pi will take its place in discussion, where its features will be mentioned, also its drawbacks.

## 2.2 RFID technology:

### 2.2.1 Definition of RFID

Radio frequency identification (RFID) is a form of wireless communication that uses electromagnetic waves to identify object carrying tags when they are in the reader range. An RFID system has readers and tags that communicate with each other by radio waves. Tags are small and need little power so they don't require a battery to store information and exchange data with readers, this makes applying tags to all kinds of things that people would like to identify or track easy and cheap.

### 2.2.2 RFID History:

Radio frequency identification has been around for decades. It developed from its roots in World War II radar systems to today's hottest supply chain technology.

**Table 2.1** <sup>(5)</sup>: The decade of RFID

The Decades of RFID	
Decade	Event
1940–1950	Radar refined and used, major World War II development effort. RFID invented in 1948.
1950–1960	Early explorations of RFID technology, laboratory experiments.
1960–1970	Development of the theory of RFID. Start of applications field trials.
1970–1980	Explosion of RFID development. Tests of RFID accelerate. Very early adopter implementations of RFID.
1980–1990	Commercial applications of RFID enter mainstream.
1990–2000	Emergence of standards. RFID widely deployed. RFID becomes a part of everyday life.
2000–	RFID explosion continues

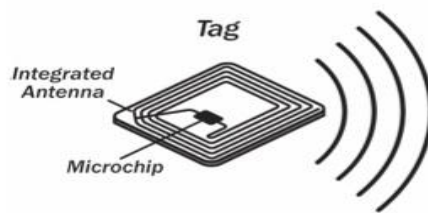
### 2.2.3 RFID System Components:

The basic components of an RFID system are:

- a. Tags
- b. Reader
- c. Antenna
- d. Reader Control and Application Software

#### 2.2.3. a RFID Tag

The tag is designed to be a transmitter and receiver. It is made up of two main parts: a microchip which is a small silicon chip with embedded circuit for storage and processing information, and an antenna to receive and transmit signals. Each tag is distinguished by specific serial number. Tags have many different forms depending on the application. RFID tags can be active, passive or semi-passive.



**Figure 2.1:** Tag parts

#### ♦ Tags Types

Active, semi-passive and passive are the three main tags types. Tags made up with few characteristics that may vary depending on type of tag, the selection of tags depends on the functional need of RFID application.

##### 1. Passive Tag <sup>(6)</sup>

A passive tag is an RFID tag that does not contain a battery; the power is supplied by the reader. When radio waves from the reader are encountered by a passive RFID tag, the coiled antenna within the tag forms a magnetic field. The tag draws power from it, energizing the circuits in the tag. The tag then sends the information encoded in the tag's memory.

♣ The major disadvantages of a passive RFID tag are:

- The tag can be read only at very short distances, typically a few feet at most.
- It may not be possible to include sensors that can use electricity for power.

♣ The advantages of a passive tag are:

- The tag functions without a battery; these tags have a useful life of twenty years or more.
- The tag is typically much less expensive to manufacture.
- The tag is much smaller (some tags are the size of a grain of rice).

## 2. Active tag: <sup>(6)</sup>

An RFID tag is an active tag when it is equipped with a battery that can be used as a partial or complete source of power for the tag's circuitry and antenna. Some active tags contain replaceable batteries for years of use; others are sealed units.

♣ The major advantages of an active RFID tag are:

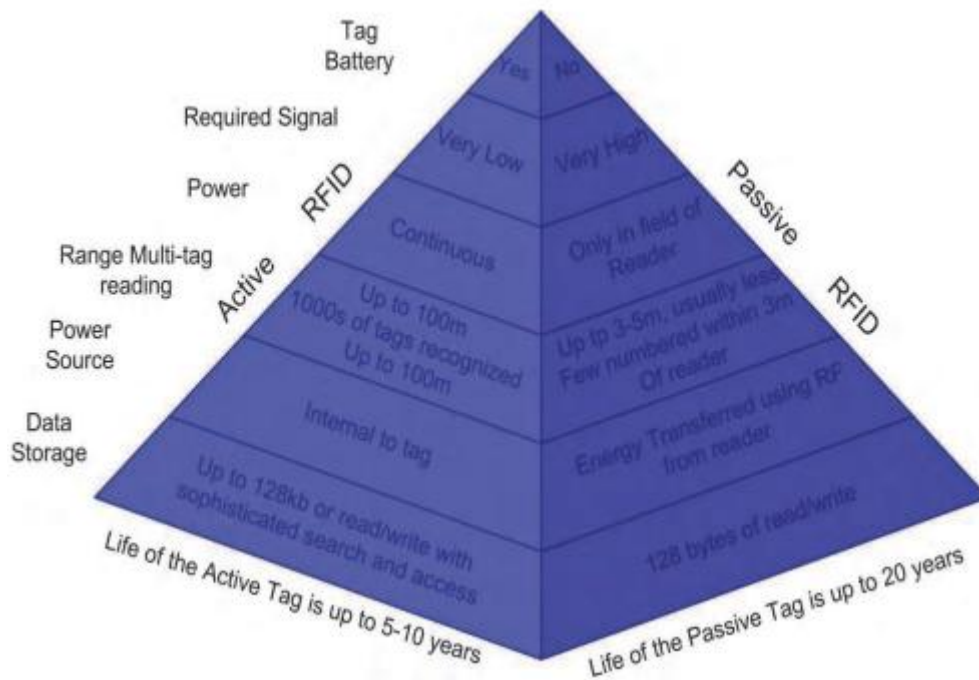
- It can be read at distances of one hundred feet or more, greatly improving the utility of the device
- It may have other sensors that can use electricity for power.

♣ The problems and disadvantages of an active RFID tag are:

- The tag cannot function without battery power, which limits the lifetime of the tag.
- The tag is typically more expensive.
- The tag is physically larger, which may limit applications.
- Battery outages in an active tag can result in expensive misreads.

♣ Active RFID tags may have all or some of the following features:

- Longest communication range of any tag.
- The capability to perform independent monitoring and control.
- The capability of initiating communications.
- The capability of performing diagnostics.
- The highest data bandwidth.



**Figure 2.2:** Comparison between passive and active RFID <sup>(7)</sup>

### 3. Semi-passive tag <sup>(8)</sup> :

Semi-passive (battery assisted backscatter) radio frequency (RF) tags are powered by a battery, offering a very good range of readability. While they do not contain an active transmitter, PNNL's semi-passive tags have been proven in real-world conditions to read and write from distances up to 100 meters. With power from small batteries similar to those found in watches, semi-passive tags can be used to monitor inputs from sensors, even when the tags aren't in the presence of a radio frequency field. As a result, the semi-passive RF tags also can control outputs. These systems can be used to monitor and activate or deactivate items remotely, making them ideal for applications such as alarms, seals, or thermostats.

#### ♣ The major Advantages of Semi-passive tag :

- Read and write from as far as 100 meters.
- Monitor external inputs such as temperature, pressure, chemicals and tamper detectors.
- Multiple tags can be read simultaneously.
- Control outputs such as valves and switches.
- Identifies an item's precise location.
- Operates with a battery lifetime of more than five years.

- Can be located to within 0.1 meters.
- By reading and writing through fences, walls, and boxes it is easier to track and identify items.

Classification of RFID tags is also possible with respect to their capabilities such as read-only, re-write and further data recoding. Further data recording examples are temperature, motion and pressure etc. Compiled tags classification into five classes is: <sup>(7)</sup>

- Class 0: Read only, contains an ID number that is written only once during manufacture.
- Class 1: Write once read many, which is manufactured with no data written in to the memory. Data can be written by manufacture or the user one time.
- Class 2: Read-Write, which is most flexible type of tag, that user can access to read and write data into the tag memory.
- Class 3: Read-write (with sensor), which is contain sensors for recording parameters like temperature, pressure and motion, which are either semi passive or active.
- Class 4: Read-Write (with integrated transmitters), which are like miniature radio devices without the present of the reader, these kind are active tags.

### **2.2.3. b RFID Reader**

RFID reader is external powered equipment used in RFID system for producing and accepting radio signals. A single reader can operate on multiple frequencies and this functionality depends on the vendor, it can have anti-collision algorithm/procedures for deducting multiple tags at one time. RFID reader works as middle-ware between tag and user application. Reader is the central part of the RFID system and communicates with tags and computer program, it supply tags information to a computer program after reading each tags unique ID. It can also perform writing on to tag, if the tag is supported. Although the reader can have multiple frequency capability but it works on a single frequency at a time. The reader can communicate with the computer program and need either wired or wireless connection with the computer. The reader can use a wire connection with any of the following: USB, RS-232, and RS485. Otherwise, the reader can connect with the computer through Wi-Fi. The reader provides various management techniques and functionality to computer programs through various built-in functions/components.

## ◆ Readers Types

There are two types of RFID readers <sup>(9)</sup>:

- RFID read-only readers: As the name suggests, these devices can only query or read information from a nearby RFID tag. These readers are found in fixed, stationery applications as well as portable, handheld varieties.
- RFID read-write readers: Also known as encoders, these devices read and also write (change) information in an RFID tag. Such RFID encoders can be used to program information into a "blank" RFID tag. A common application is to combine such a RFID reader with a barcode printer to print "smart labels". Smart labels contain a UPC bar code on the front with an RFID tag embedded on the back.

### 2.2.3. c Reader Antennas

RFID readers and reader antennas work together to read tags. Reader antennas convert electrical current into electromagnetic waves that are then radiated into space where they can be received by a tag antenna and converted back to electrical current. Just like tag antennas, there is a large variety of reader antennas and optimal antenna selection varies according to the solution's specific application and environment.

The two most common antenna types are linear- and circular-polarized antennas. Antennas that radiate linear electric fields have long ranges, and high levels of power that enables their signals to penetrate through different materials to read tags. Linear antennas are sensitive to tag orientation; depending on the tag angle or placement, linear antennas can have a difficult time reading tags. Conversely, antennas that radiate circular fields are less sensitive to orientation, but are not able to deliver as much power as linear antennas.

Choice of antenna is also determined by the distance between the RFID reader and the tags that it needs to read. This distance is called read range. Reader antennas operate in either a "near-field" (short range) or "far-field" (long range). In near-field applications, the read range is less than 30 cm and the antenna uses magnetic coupling so the reader and tag can transfer power. In near-field systems, the readability of the tags is not affected by the presence of dielectrics such as water and metal in the field.

In far-field applications, the range between the tag and reader is greater than 30 cm and can be up to several tens of meters. Far-field antennas utilize electromagnetic coupling and dielectrics can weaken communication between the reader and tags. <sup>(10)</sup>



### **2.2.3. d Reader Control and Application Software**

Reader control and application software, also known as middleware, helps connect RFID readers with the applications they support. The middleware sends control commands to the reader and receives tag data from the reader. <sup>(10)</sup>

### **2.2.4 RFID standards <sup>(11)</sup>**

Like many other technologies, RFID standards are used. RFID standards, as any other standards enable manufacturers to make the same products for a variety of markets and in this way gain the economies of scale.

RFID standards also enable products from different manufacturers to operate together. One example may be for tags which are used in very large quantities and manufacturers may want to source from two suppliers to provide reliability.

#### **♦ RFID standards bodies**

There are two main international RFID standards bodies or standardization bodies:

- ISO - International Standards Organization
- EPCglobal - Electronics Product Code Global Incorporated

Although these two organizations provide the main RFID standards organizations, there is also a plethora of other standards that apply to niche areas of RFID.

In terms of the standardization organizations ISO is the longest established. In 1996 it set up a joint committee with IEC to look at standardization for RFID technology.

The ISO RFID standards fall into a number of categories according to the aspect of RFID that they are addressing. These include: air interface and associated protocols; data content and the formatting; conformance testing; applications; and various other smaller areas.

In addition to the ISO RFID standards, there are also the standards from EPC Global. In 1999 a number of industrial companies with MIT set a consortium known as the Auto-ID consortium with the aim of researching and standardizing RFID technology.

In 2003 this organization was split with the majority of the standardization activities coming under a new entity called EPCglobal. The Auto-ID Center retained its activities associated with the research into RFID technologies.

#### ◆ **Auto-ID tag standards**

In order to be able to standardize the RFID tags, the Auto-ID Center devised a series of classes for RFID tags. Was generated and these still form the basis for a developed system of RFID tag classes seen today:

- **Class 0:** Basic read-only passive tag using backscatter where the tag was programmed at the time the tag chip was made.
- **Class 1:** Basic read-only passive tag using backscatter with one-time non-volatile program capability.
- **Class 2:** Passive backscatter tag with up to 65k of read-write memory.
- **Class 3:** Semi-passive tag with up to 65 k read-write memory and a battery incorporated to provide increased range.
- **Class 4:** Active tag using a battery to enable extra functionality within the tag and also to provide power for the transmitter.
- **Class 5:** An active tag that provides additional circuitry to communicate with other class 5 tags.

The responsibilities for Class 0 and Class 1 RFID tag definitions and standards were handed on to EPC Global in 2003.

Although other newer RFID tag standards and now available, reference is still made to these original tag classes.

#### ◆ **Gen2 RFID tag standard**

In 2004 EPCglobal began the creation of a second generation protocol, often referred to as EPCglobal Gen2. While the EPCglobal Gen2 RFID standard is not backwards compatible with the Class 0 and Class 1 tags, it aims to provide a worldwide RFID tag standard that is compatible with ISO standards.

#### ◆ ISO 18000 series RFID standards

The ISO 18000 series standards are a series of standards that define the air interface for the different RFID frequencies in use around the globe. There are a total of seven standards within the ISO 18000 series as outlined in the table below:

**Table 2.2:** ISO 18000 standard

ISO 18000 STANDARD	DETAILS OF THE PARTICULAR ISO 18000 SERIES STANDARD
ISO 18000-V1	Generic parameters for air interfaces for globally accepted frequencies
ISO 18000-V2	Air interface for 135 KHz
ISO 18000-V3	Air interface for 13.56 MHz
ISO 18000-V4	Air interface for 2.45 GHz
ISO 18000-V5	Air interface for 5.8 GHz
ISO 18000-V6	Air interface for 860 MHz to 930 MHz
ISO 18000-V7	Air interface at 433.92 MHz

RFID standards are now widespread in their use, and although EPCglobal and ISO are separate organizations, there are efforts to move towards a single RFID standards scenario, rather than having two sets of competing RFID standards.

#### 2.2.5 RFID Frequencies

RFID systems can be broken down by the frequency band within which they operate: low frequency, high frequency, and ultra-high frequency. In the sections below we will explore the frequencies of RFID systems.

Frequency refers to the size of the radio waves used to communicate between RFID systems components. RFID systems throughout the world operate in low frequency (LF), high frequency (HF) and ultra-high frequency (UHF) bands. Radio waves behave differently at each of these frequencies with advantages and disadvantages associated with using each frequency band.

If an RFID system operates at a lower frequency, it has a shorter read range and slower data read rate, but increased capabilities for reading near or on metal or liquid surfaces. If a system operates at a higher frequency, it generally has faster data transfer rates and longer read ranges than lower frequency systems, but more sensitivity to radio wave interference caused by liquids and metals in the environment.

### **1. LF RFID**

The LF (Low Frequency) band covers frequencies from 30 KHz to 300 KHz. Typically LF RFID systems operate at 125 KHz, although there are some that operate at 134 KHz. This frequency band provides a short read range of 10 cm, and has slower read speed than the higher frequencies, but is not very sensitive to radio wave interference.

### **2. HF RFID**

The HF (High Frequency) band ranges from 3 to 30 MHz, most HF RFID systems operate at 13.56 MHz with read ranges between 10 cm and 1 m. HF systems experience moderate sensitivity to interference.

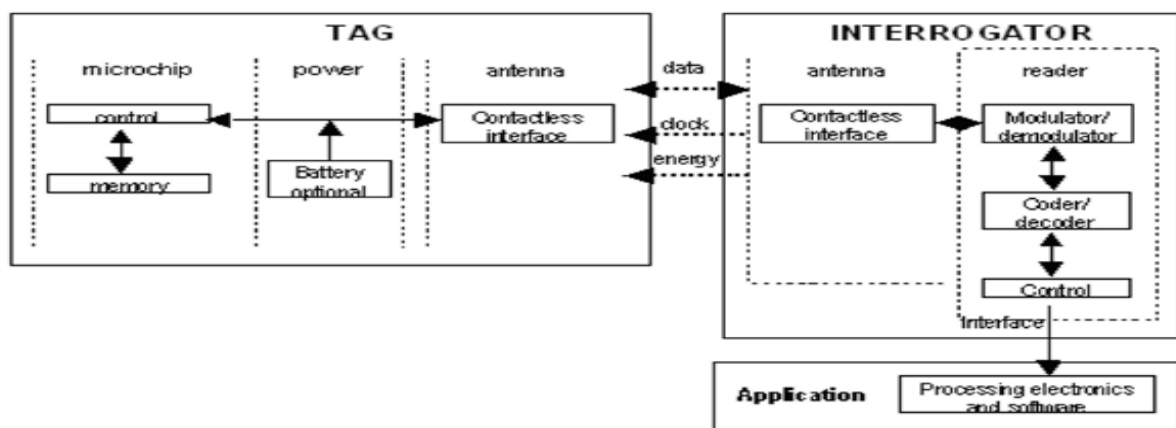
### **3. UHF RFID**

The UHF (Ultra High Frequency) frequency band covers the range from 300 MHz to 3 GHz. Systems complying with the UHF Gen2 standard for RFID use the 860 to 960 MHz band. While there is some variance in frequency from region to region. The read range of passive UHF systems can be as long as 12 m, and UHF RFID has a faster data transfer rate than LF or HF. UHF RFID is the most sensitive to interference, but many UHF product manufacturers have found ways of designing tags, antennas, and readers to keep performance high even in difficult environments. Passive UHF tags are easier and cheaper to manufacture than LF and HF tags.

### 2.2.6 How RFID system works? <sup>(7)</sup>

RFID system deducts tags within antennas' range and performs various operations onto each tag. The RFID system can only work effectively if all RFID components logically connect together and these components need to be compatible with each other. That's why understanding of these separate components is necessary. Implementation of complete RFID solution is only possible through integration of these components which needs understanding of compatibility for each component, realization of each component compatibility needs property study for these components. These components are gathered and defined as above.

- Tag has unique ID and use for unique identification; tags are attached with objects in RFID solutions.
- Antenna use for reading tags; antenna has its own magnetic field and antenna can only read tags within these magnetic fields.
- Reader works for handling antenna signals and manipulate tags' information.
- Communication infrastructure use for reader to communicate with IT infrastructure and work as middle layer between application software and reader.
- Application software is computer base software which enable user to see RFID information, this can be database, application routines or user interface.



**Figure 2.3:** System overview

### **2.2.7 Why we chose low frequency (LF) Passive radio frequency identification technology?**

In our view the LF Passive RFID is the most appropriate for our project. There are many reasons that is so, some of them are:

- High reliability: a RFID passive tag does not have a battery and their life of twenty years or more.
- Suitable reader size for our application to be portable.
- Availability of the reader in our country, where it was the only reader available in our country.
- The appropriate size and shape: a passive tag is small in size and exists in several shapes as label, card, button...etc. That is make attachment the tag at any object easy.

### **2.3 Raspberry Pi**

A Raspberry Pi is a credit-card sized computer originally designed for education, inspired by the 1981 BBC Micro. Creator Eben Upton's goal was to create a low-cost device that would improve programming skills and hardware understanding at the pre-university level. But thanks to its small size and accessible price, it was quickly adopted by tinkerers, makers, and electronics enthusiasts for projects that require more than a basic microcontroller (such as Arduino devices).

(12)

The Raspberry Pi is a computer, very like the computers with which you're already familiar. It uses a different kind of processor, so you can't install Microsoft Windows on it. But you can install several versions of the Linux operating system that look and feel very much like Windows. It also capable of offering basic office computing, low-level gaming, Internet and email access, media playback and many other features regularly expected from a computer in the 21st century, the Pi achieves all of this with a stripped-down component count, and ARM processor with a very low price. <sup>(13) (14)</sup>

RASPBIAN, PIDORA, OPENELEC, RASPBMC, RISC OS, and ARCH LINUX these are few software's (operating systems) which are used in RPi. All this software's can be downloaded easily and these are free from the official forum under the NOOBS (new out of the box software) category. It supports Python as the main programming language for functioning and coding. It also supports BASIC, C, C++, JAVA, and Perl and Ruby languages. <sup>(15)</sup>

The Raspberry Pi is an open hardware, with the exception of the primary chip on the Raspberry Pi, the Broadcom SoC (System on a Chip), which runs many of the main components of the board—CPU, graphics, memory, the USB controller, etc. Many of the projects made with a Raspberry Pi are open and well-documented as well and are things you can build and modify by yourself. <sup>(12)</sup>

### 2.3.2 What are the differences between models?

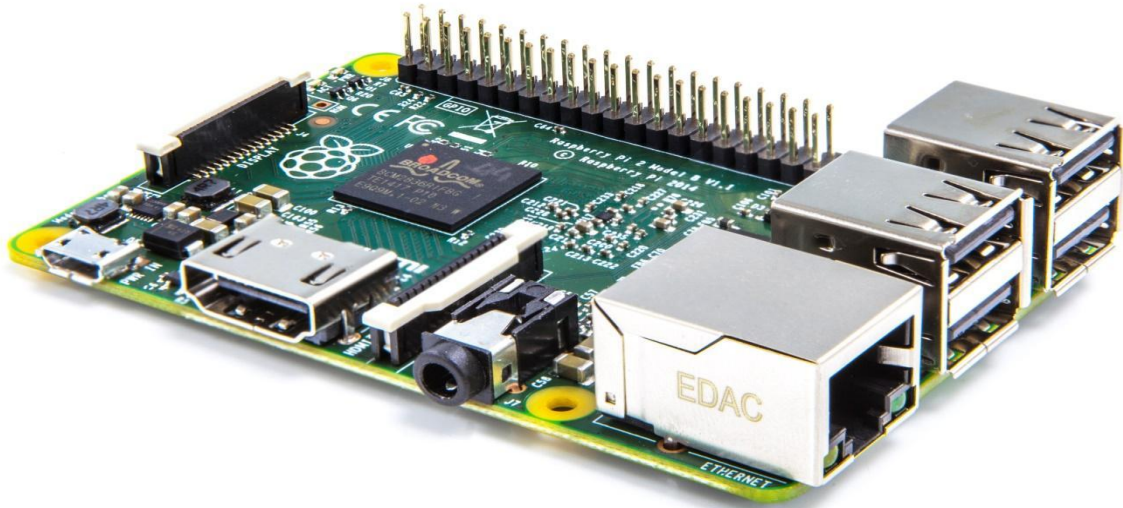
The current models of the Raspberry Pi available are: the Pi 3 Model B, the Pi 2 Model B, the Pi Zero, and the Pi 1 Model B+ and A+.

The following table 2.1 is giving you a breakdown of the current versions, their features and a summary of what they are good for. <sup>(16)</sup>

**Table 2.3:** Comparison between different models of Raspberry Pi

Name	Raspberry Pi 3 Model B	Raspberry Pi 2 Model B	Raspberry Pi Zero	Raspberry Pi 1 B+	Raspberry Pi 1 A+
Release date	2016 Feb 29	2015 Feb 1	2015 Nov 30	2012 Feb 15	2014 Nov 10
SoC type	Broadcom BCM2387	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835
Core type	Cortex-A53 64-bit	Cortex-A7	ARM1176JZF-S	ARM1176JZF-S	ARM1176JZF-S
No. of cores	4	4	1	1	1
CPU clock	1.2 GHz	900 MHz	1 GHz	700 MHz	700 MHz
RAM	1 GB	1 GB	512 MB	512 MB	256 MB
USB Ports	4	4	micro + micro OTG	2	1
Ethernet	Yes	Yes	No	Yes	No
HDMI	Yes	Yes	Yes – Mini HDMI	Yes	Yes

According to the above Raspberry Pi 3 looks and feel exactly like the Raspberry Pi 2 B, so for this project Raspberry Pi 2 B is used, because it is adequate and satisfies what is required, and it's available from previous project.



**Figure 2.4:** Raspberry Pi 2 model B <sup>[14]</sup>

### 2.3.3 Raspberry Pi Basic Hardware Setup

The Raspberry Pi board contains a processor and graphics chip, program memory (RAM) and various interfaces and connectors for external devices. Some of these devices are essential, others are optional. RPi operates in the same way as a standard PC, requiring a keyboard for command entry, a display unit and a power supply.

It also requires 'mass-storage', but a hard disk drive of the type found in a typical PC is not really in keeping with the miniature size of RPi. Instead we will use an SD Flash memory card normally used in digital cameras, configured in such a way to 'look like' a hard drive to RPi's processor. RPi will 'boot' (load the Operating System into RAM) from this card in the same way as a PC 'boots up' into Windows from its hard disk. <sup>(17)</sup>

The following are essential hardware components to get started:

- SD card containing Linux Operating system.
- USB keyboard.
- TV or monitor (with HDMI, DVI, Composite or SCART input).
- Power supply.



- Video cable to suit the TV or monitor used.

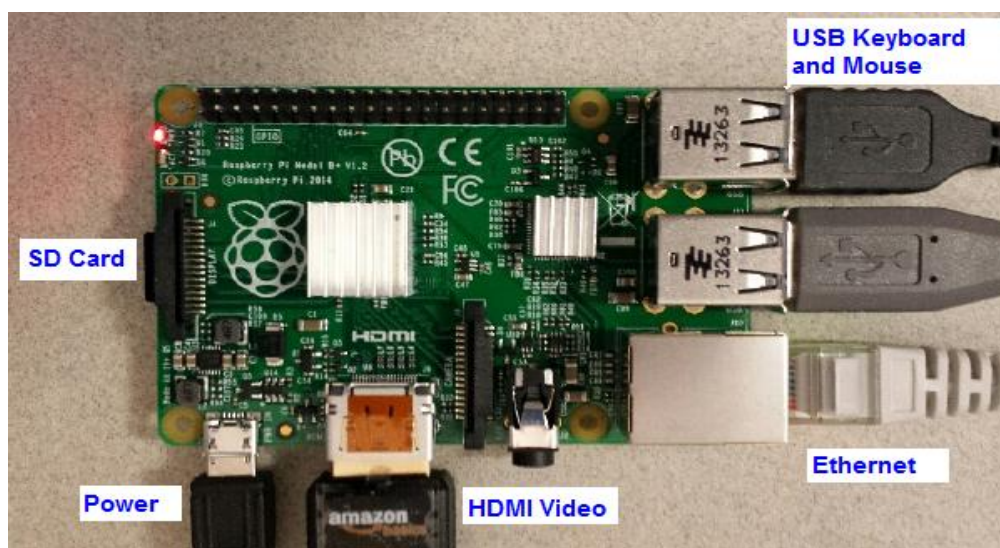
Recommended optional extras include:

- USB mouse
- Internet connection, Model A or B: USB Wi-Fi adaptor
- Internet connection, Model B only: LAN (Ethernet) cable
- Powered USB hub
- Case

### 2.3.4 Interfacing Raspberry Pi

The steps of connecting RPi are given below:

1. Begin by placing your SD card into the SD card slot on the RPi. It will only fit one way.
2. Next, plug your keyboard and mouse into the USB ports on the RPi.
3. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc).
4. Connect your HDMI cable from your RPi to your monitor or TV.
5. If you intend to connect your RPi to the internet, plug an Ethernet cable into the Ethernet port, or connect a Wi-Fi dongle to one of the USB ports (unless you have a Raspberry Pi 3).
6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your RPi.



**Figure 2.5:** Connecting Raspberry Pi <sup>(16)</sup>

## **2.3.5 Advantages and disadvantages<sup>(15)</sup>**

### **A. Advantages**

Some of the merits are:

- RPi is an inexpensive device with an easily affordable price.
- RPi has the size of a credit card. As we all know with technology, generally the smaller it is, the better.

### **B. Disadvantages**

Although it has merits but it has some demerits also some of them are:

- Raspberry Pi does not support X86 operating systems means hardware limitations do not allow for Raspberry Pi to run 32 bit operating systems such as Microsoft Windows, Max OS X or some varieties of Linux. This can be a huge loss for not-so computer friendly end users .For professional users; this is not much of a set back as Raspberry Pi supports other popular operating system.
- Some applications which necessitate high demands on CPU processing are off-limits. Such as “Model B took 107 ms to complete one calculation of the entirely synthetic prime number test and a mid-range desktop Core 2 Duo E8400 took only 0.85ms.”(By Collins, 2012)<sup>(15)</sup>
- Users must not use usual computer standards to judge Raspberry Pi. It can work as a private computer but still cannot replace it.

## **2.4 Microphone**

### **2.4.1 Microphone**

Microphone is a piece of equipment into which people speak or sing in order to record their voices or to make their sound louder.

### **2.4.2 How microphone works<sup>(18)</sup>**

A microphone is an example of a transducer, a device that changes information from one form to another. Sound information exists as patterns of air pressure; the microphone changes this information into patterns of electric current.

A variety of mechanical techniques can be used in building microphones. The two most commonly encountered in recording studios are the magneto-dynamic and the variable condenser designs.

## **2.5 Speaker**

Speaker is an output device that receives audio input from the computer's sound card and produce audio output in the form of sound waves.

Speakers are transducers that convert electromagnetic waves into sound waves. The audio input may be either in analog or digital form. Analog speakers simply amplify the analog electromagnetic waves into sound waves. Since sound waves are produced in analog form, digital speakers must first convert the digital input to an analog signal, and then generate the sound waves. <sup>(19)</sup>

## **2.6 Wi-Fi communication**

Wi-Fi is a very widely used type of medium-range wireless communication system. Medium-range in this case means that a typical Wi-Fi signal can carry about 100 meters.

Being a wireless protocol, Wi-Fi standard uses the ISM (Industrial, Scientific and Medical) band of frequency which is free to use and require no licensing. Launched in 2.4 GHz with transmission rates of 1-2 Mbps, Wi-Fi now works at 5 GHz frequency also with astounding data transmission rates reaching up to 54 Mbps at both frequencies.

Wi-Fi is a marketing term applied to 802.11b IEEE standard, but it now ubiquitously used for all the standards that fall under 802.11 category of Wireless LAN. So, Wi-Fi defines 802.11 x standards where x is the respective Wi-Fi version. Popular Wi-Fi version are a, b, g and n.

Data exchange in Wi-Fi can be summarized into three phases:

- Phase I: Where data is prepared for transmission; it is encoded; changed into frames (digital signals are sent in frames for better QoS). The frequency for data transmission is also chosen depending upon the technique used to send the signals wirelessly.
- Phase II: Where data is transmitted with air as the medium of wave transmission
- Phase III: Where data is received, decoded, acknowledged and then used.

All of these phases apply some of the popular digital communications spread spectrum techniques for signal multiplexing (FHSS, Infrared, OFDM etc.); make use of security methods (WEP, WPA). Let's find out the technical insides of the Wi-Fi legacy.<sup>(20)</sup>

We are going to choose Wi-Fi for communication because it satisfy our needed, it has low cost, good rang, secure and there is many resources explain who to deal with it.

## **2.7 Voice recognition**

Alternatively referred to as speech recognition, voice recognition is a computer software program or hardware device with the ability to decode the human voice. Voice recognition is commonly used to operate a device, perform commands, or write without having to use a keyboard, mouse, or press any buttons.<sup>(21)</sup>

## **2.8 SQL database**

SQL (Structured Query Language) is a standardized programming language used for managing relational databases and performing various operations on the data in them. Initially created in the 1970s, SQL is regularly used by database administrators, as well as by developers writing data integration scripts and data analysts looking to set up and run analytical queries.

The uses of SQL include modifying database table and index structures; adding, updating and deleting rows of data; and retrieving subsets of information from within a database for transaction processing and analytics applications. Queries and other SQL operations take the form of commands written as statements -- commonly used SQL statements include select, add, insert, update, delete, create, alter and truncate.<sup>(22)</sup>

# Chapter 3

## Design Concept

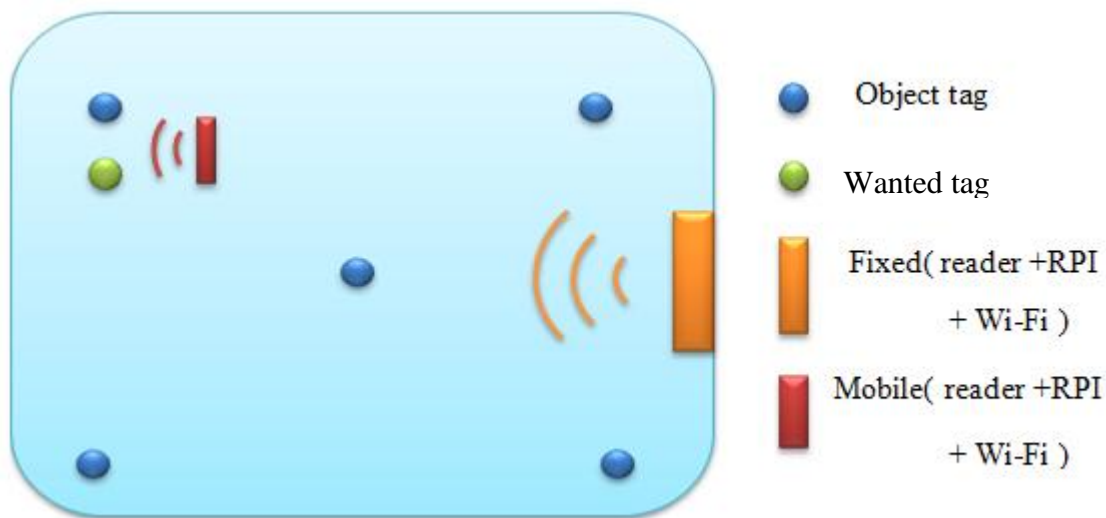
---

- 3.1 Overview.
- 3.2 Basic operation.
- 3.3 Main block diagram.
- 3.4 The main components of this project.
- 3.5 The generic process.
- 3.6 Wi-Fi communication

### 3.1 Overview

In this chapter, the main design of our system with more details and explanation were shown. Through an informative block diagrams, system main flow chart, project idea and discuss their relations with each other and with the overall work as shown in figure 3.1.

### 3.2 Basic operation



**Figure3.1:** structure of the system

The system consists of two RFID readers one is fixed in the room and the other is mobile and number of the tags attached to the objects.

At the time the blind person wants to find his misplaced thing, he holds the mobile reader and says the name of this thing, by voice recognition the name will translate into word, then the Raspberry Pi will take the corresponding tag ID of the word.

Using Wi-Fi communication between the two Raspberry Pi's the ID send to the fixed one to begin its role, if the missing object is not in the range of the mobile reader.

The two readers cooperate with each other to find where the missing thing is and reply to the blind if they find its object or not.

### 3.3 Main block diagram

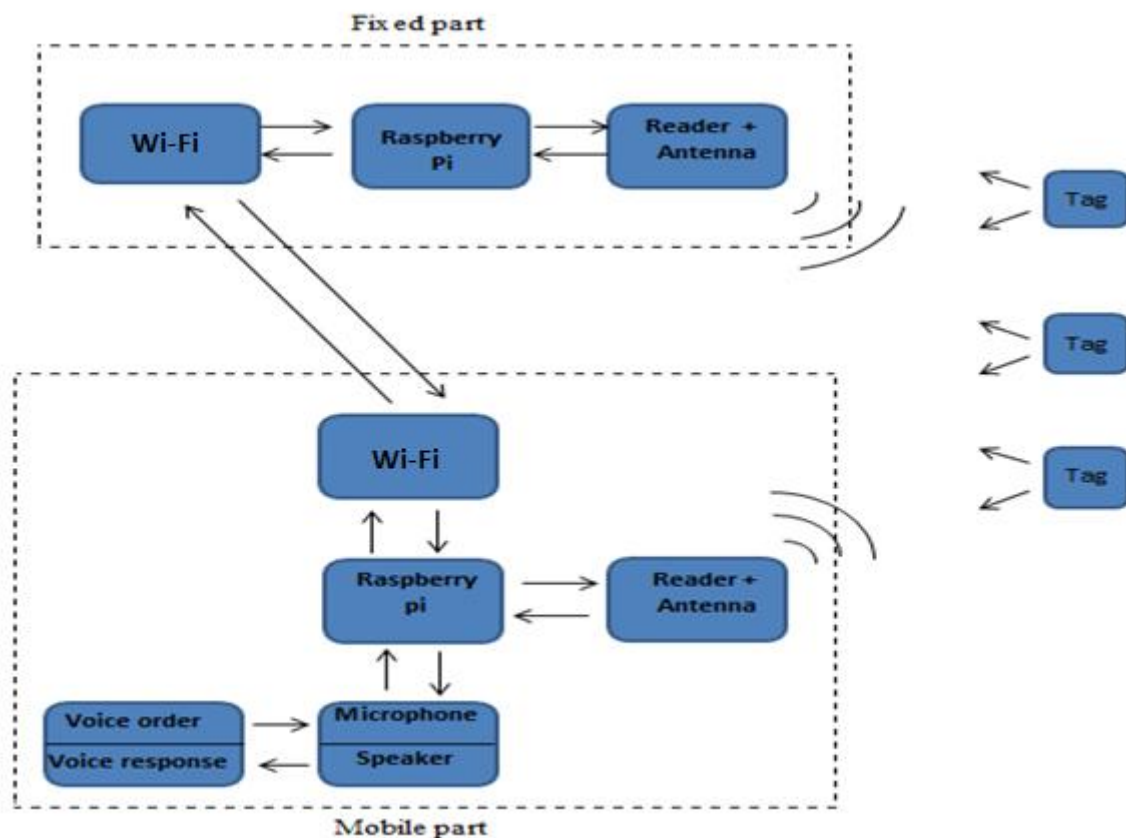


Figure3.2: Main block diagram.

### 3.4 The main components of project

In this section we described the function of each component that used in our project.

#### 3.4.1 RFID reader

Two readers were used in the project one was fixed and the other was mobile to be portable with the blind, the two readers were from the same type that was RFID Access Control ID Card Reader 125 KHz Wiegand 26 Security.



Figure3.3: RFID Access Control ID Card Reader 125 KHz Wiegand 26 Security.

The characteristics of this reader are: <sup>(23)</sup>

- 125 KHz proximity card reader.
- It's a standard wiegand 26 bit reader.
- Read range: up to 10cm.
- External LED control.
- External buzzer control.
- Indoor / Outdoor operation.
- Solid epoxy potted.
- Weatherproof IP65.
- Reverse polarity protection.
- Support RFID cards (EM4100 standard ID cards).
- Durable keytop with blue backlight.
- Reading Time (Card)  $\leq 300$  ms.

The basic functions of the reader in the project is to send data and command to the tags with build in antenna and read the information on it, then transmit reading result to an RFID program installed on Raspberry Pi.

### 3.4.2 RFID tag

The passive tags in the project were attached to the wanted object, we used different shape of the tag, card and keyfob as shown in figure 3.4 & 3.5.



**Figure3.4:** key fob tag



**Figure3.5:** card tag



The characteristics of the tags are: <sup>(24)</sup>

- 64bit memory array laser programmable
- Several options of data rate and coding available
- On chip resonance capacitor
- On chip supply buffer capacitor
- On chip voltage limiter
- Full wave rectifier on chip
- Large modulation depth due to a low impedance modulation device
- Operating frequency 100 - 150 kHz
- Very small chip size convenient for implantation
- Very low power consumption

Theses tags were placed in two locations. One was in the domain of the first reader and the other place in the domain of the second reader. The tags identify the object by the unique ID that it holds.

### 3.4.3 Raspberry Pi

For this system two Raspberry Pi 2 model B were used, one attached to the fixed RFID reader and the other attached to the mobile reader.



**Figure3.6:** Raspberry Pi 2 model B

The characteristics of this Raspberry Pi are: <sup>(16)</sup>

- ◆ A 900MHz quad-core ARM Cortex-A7 CPU.

- ◆ 1GB RAM.
- ◆ 4 USB ports.
- ◆ 40 GPIO pins.
- ◆ Full HDMI port.
- ◆ Ethernet port.
- ◆ Combined 3.5mm audio jack and composite video.
- ◆ Camera interface (CSI).
- ◆ Display interface (DSI).
- ◆ Micro SD card slot.
- ◆ Video Core IV 3D graphics core.

The Raspberry Pi was the heart of the system, because the entire component i connected to it. Moreover it did all the processing and management of the data.

#### **3.4.4 Microphone**

An instrument for converting sound waves into electrical energy variations which may then be amplified, transmitted, or recorded. For our system USB Microphone was used.

The microphone used to take the voice signal from the blind person and sent it to the Raspberry Pi to do the processing on it.

#### **3.4.5 Speaker**

Is a hardware device connected to a computer's sound card that outputs sound generated by the computer. In our system Mini Portable Speaker for the Raspberry Pi was used.

The speaker's job was telling the blind person if the wanted object found or not, by using sentences already recorded.

#### **3.4.6 Wireless adapter**

A wireless adapter is a hardware device that is generally attached to a computer or other workstation device to allow it to connect to a wireless system.

For our system two Wi-Fi adapters were used, the first one connected to the fixed Raspberry Pi and the other one connected to the mobile Raspberry Pi, so that the two Raspberry Pi's can be in touch.



**Figure3.7:** Wireless Adapter

The characteristics of wireless adapter is: <sup>(25)</sup>

- Provide Wireless-N capability.
- Flexibility of selectable dual-band Wireless-N--connect to either a 2.4 GHz or 5 GHz wireless network.
- Set up Adapter in a few easy steps with the included software's simple setup wizard.
- Keep Wi-Fi freeloaders and Internet threats at bay with customizable security settings, including WPA/WPA2 Personal and WPA/WPA2 Enterprise.

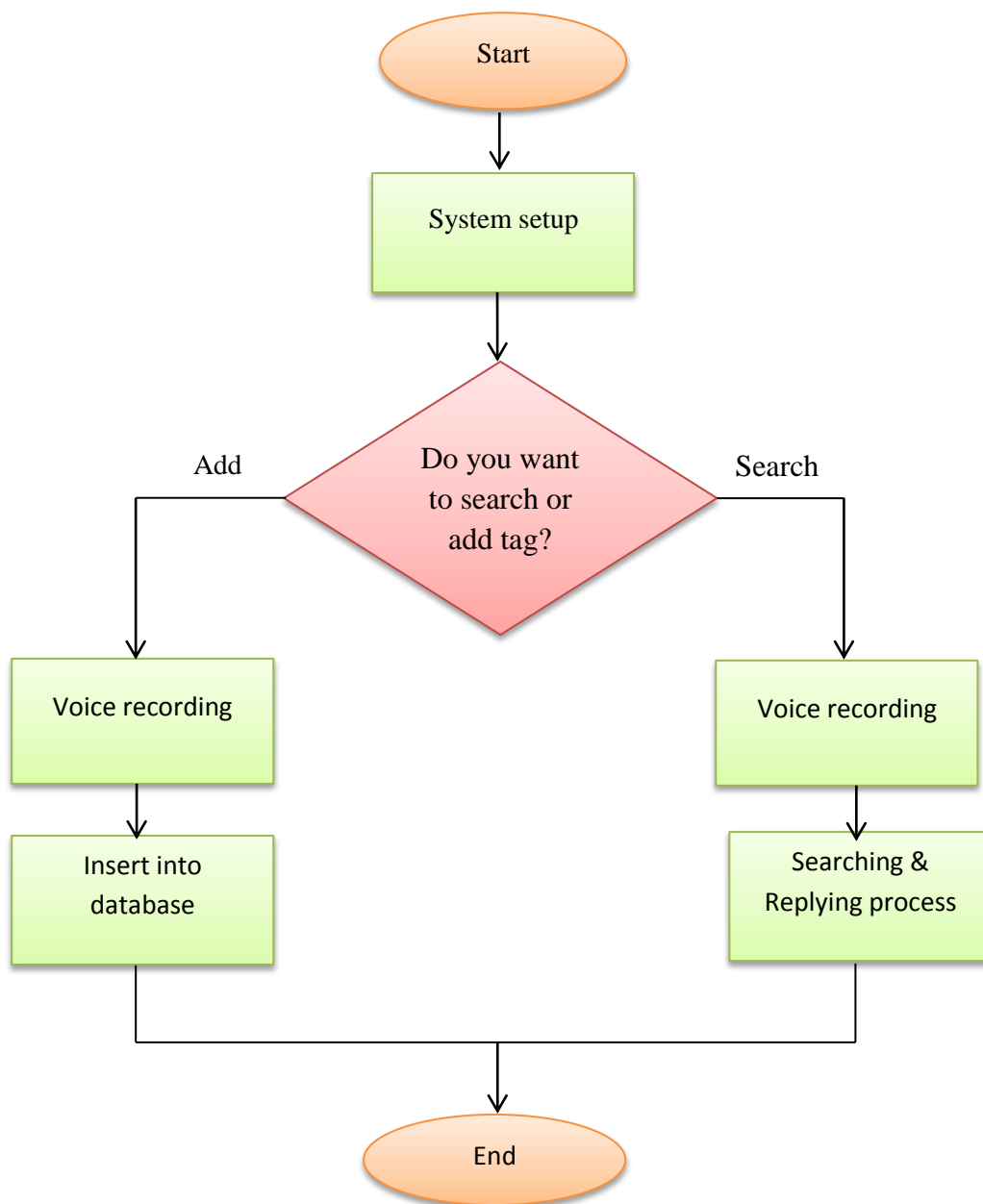
The wireless adapters were connected to the two RPi's to allow the communication between them.

### **3.4.7 Voice recognition**

Voice recognition or speech recognition is a computer software program device with the ability to decode the human voice.

It needed to convert voice to text, and this text recorded in database in order to get the tag ID of the corresponding word.

### 3.5 The generic process



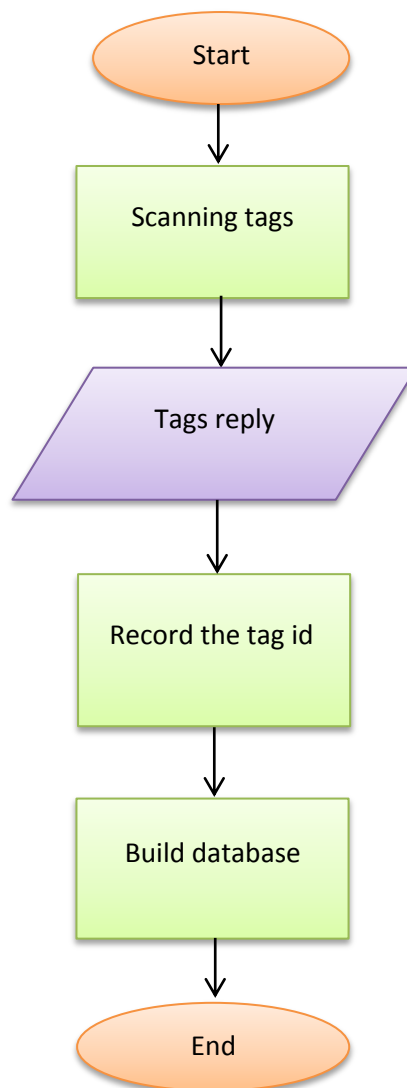
**Figure 3.8:** the generic process

The whole system can be divided into four processes:

- System setup process.
- Voice recording process.
- Searching and replying process.
- Adding process.

All this processes explained in detail as follow:

### 3.5.1 System setup process



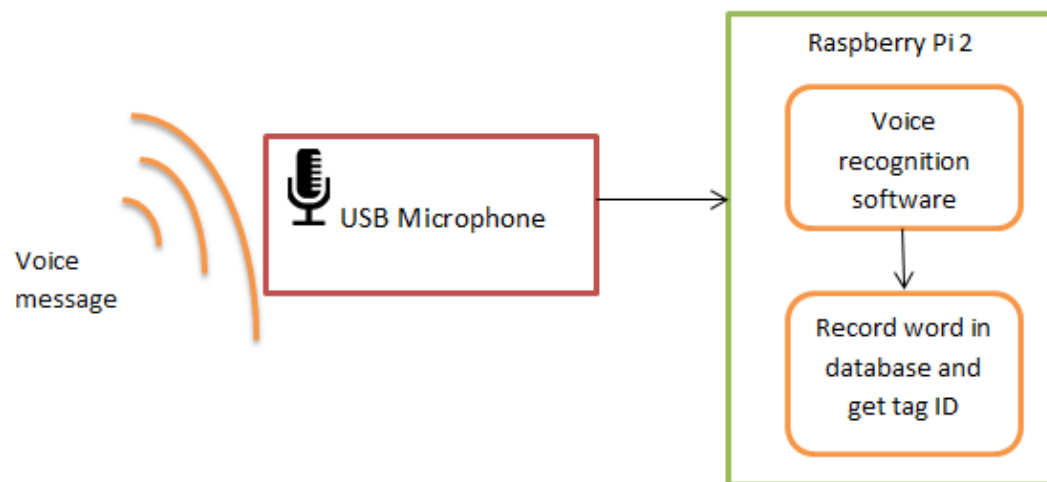
**Figure 3.9:** System setup process

#### 3.5.1.1 Explanation of system setup process

This process occurred at the beginning of the system. And it executed as follow:

- Each reader scans the tags in its range, by sending electromagnetic fields to them.
- All tags reply to the reader and the reader recode their id number this number present the object, that means each object has its own unique id number.
- Build database and insert the id number and the object name into it.

### 3.5.2 Voice recording process



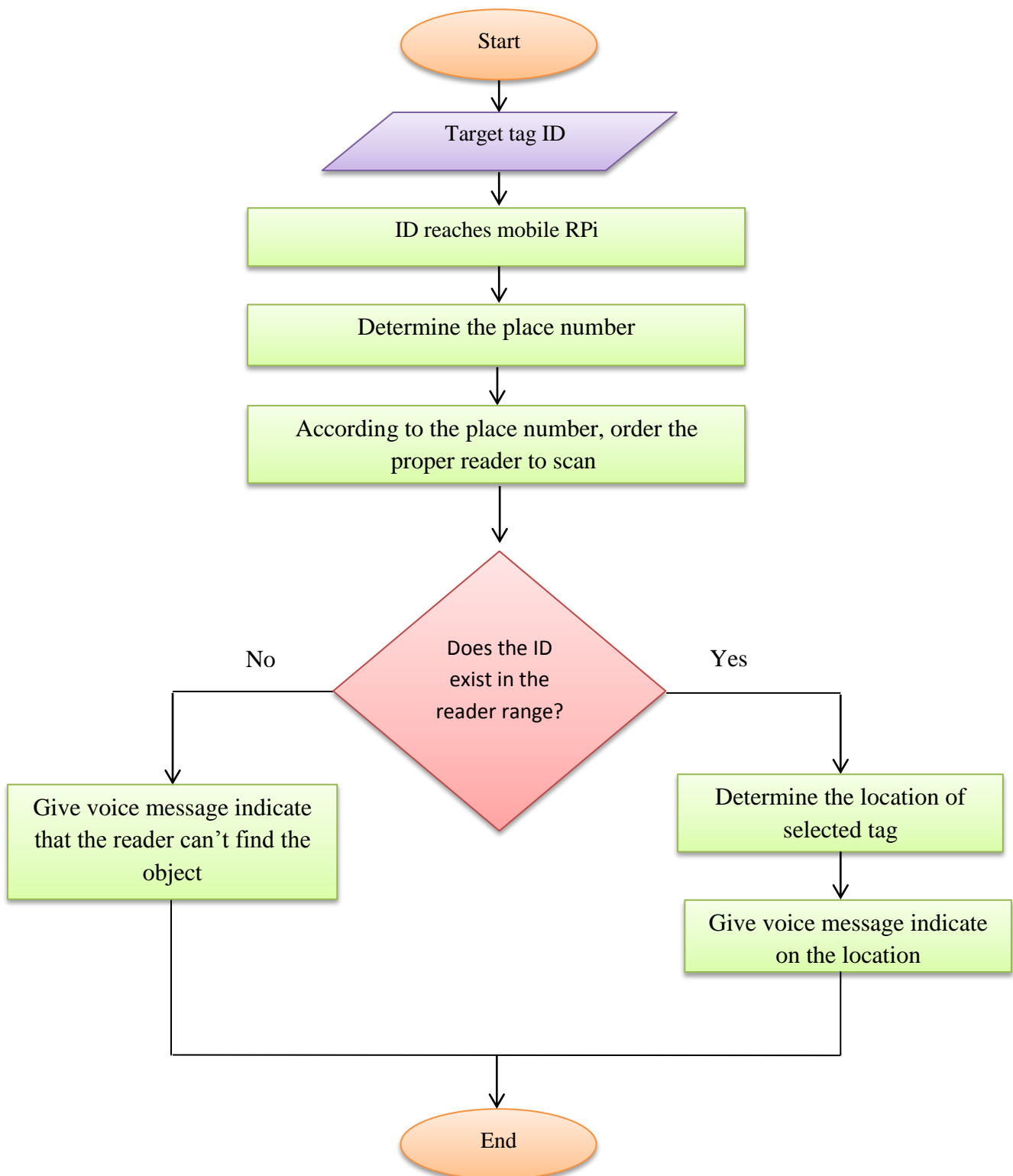
**Figure 3.10:** Voice recording process

As shown in the figure the voice recording process will obtain by:

- USB microphone takes a voice message from the blind.
- Then the message translated into the corresponding text by voice recognition.
- From this text the search in the database begin to determine the ID of the wanted tag.

### 3.5.3 Searching & replying process

The aim of search and reply process was to determine the location of the target tag and it happens as follow:



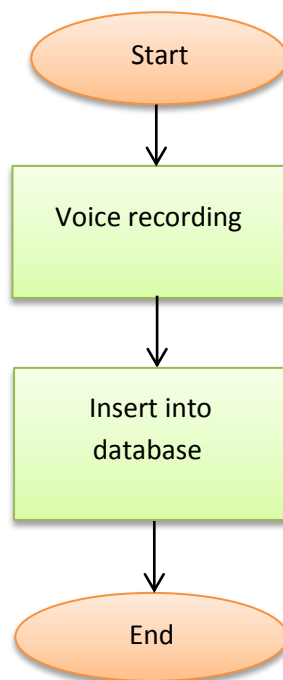
**Figure 3.11:** Searching & replying process

### 3.5.3.1 Explanation of searching & replying process

The target tag ID is available at mobile RPi from the previous process; searching process start in the database to determine the place number of this ID.

If the place number was for the mobile RPi, it orders the reader to start scanning else the mobile RPi send an order to the fixed RPi to start scanning via Wi-Fi communication. After the reader receives the order it read all tag in its range and resend the result to the RPi. Finally the mobile RPi convert a saved text indicate on the location to voice to be audible for the blind. In case that the reader can't find the object, mobile RPi tell the blind that the wanted object is out of the readers range.

### 3.5.4 Adding process



**Figure 3.12:** Adding process

#### 3.5.4.1 Explanation of add process

When the user select the add option, the system ask him to say the name of the object then record its id to insert it into database.



### 3.6 Wi-Fi communication

Two Raspberry Pi's with Wi-Fi adapter talking to each other by socket programming.



**Figure 3.13:** RPi with Wi-Fi adapter

Each Wi-Fi adapter connected to each RPi through the USB port on it, so the RPi's communicate with each other using socket programming.

# Chapter 4

## Hardware and software Implementation

---

- 4.1 Overview
- 4.2 Software Design
- 4.3 Hardware Design
- 4.4 Initial version of the project

## 4.1 Overview

After viewing the general system design in the previous chapter, it is now the time for presenting the construction and testing processes.

When the collecting of all the necessary information related to the project and analysing them become ready, the group started to build the system step by step, as will be shown.

## 4.2 Hardware Design

### 4.2.1 Setting up Raspberry Pi

We first connected a monitor, keyboard and mouse to the Raspberry Pi to program it. Then we installed the most recent version of the Raspberry Pi operation system (raspbian jessie with pixel) on the SD card. Then we upgraded and updated the system.



**Figure4.1:** Setting up Raspberry pi

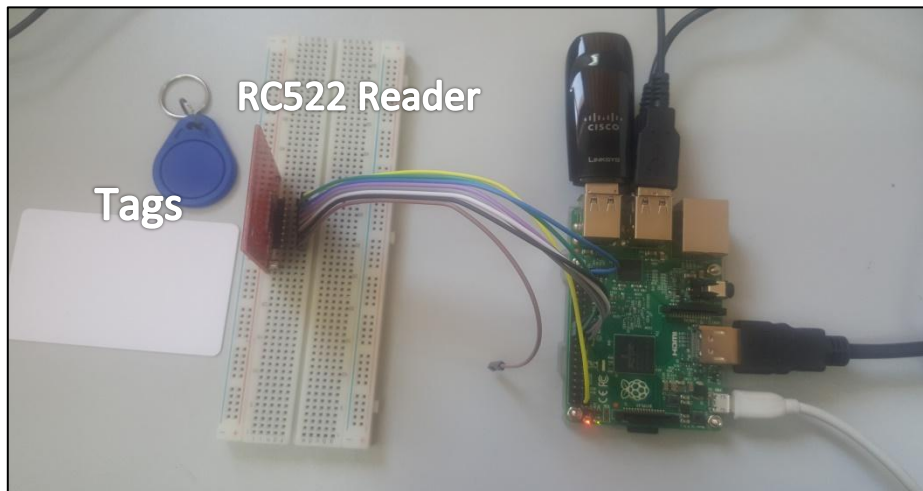
### 4.2.2 Circuits description

In this section the hardware circuit described in details, by showing the connection and interfacing between the Raspberry Pi and the RFID reader.

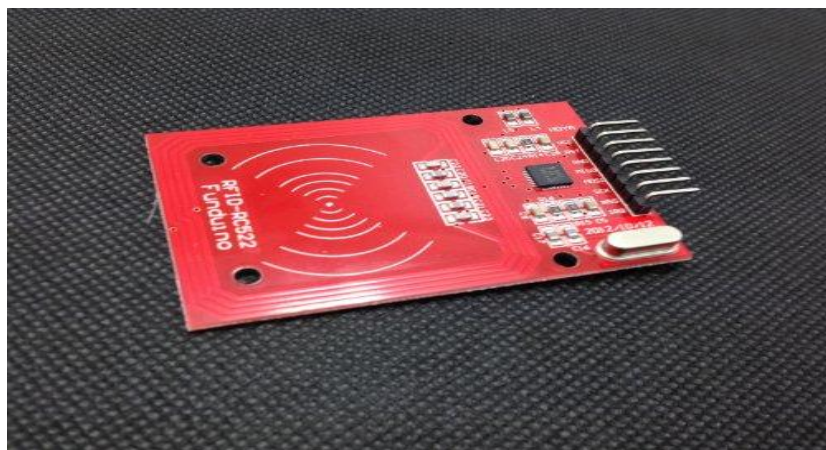
#### 4.2.2.a Raspberry Pi with RC522 RFID reader

RC522 RFID reader is a RFID near field card reader which work on 13.56 MHz frequency. It used to do an experiment on it, so making us familiar with dealing and connecting the reader

with Raspberry Pi. We connected the RC522 RFID reader directly to the Raspberry Pi on the GPIO pins as shown:



**Figure4.2:** RC522 reader connection



**Figure4.3:** RC522 RFID reader

Wiring:

**Table4.1:** RC522 reader wiring

Name	Pin Number	Pi0n name
VCC	1	3V3
RST	22	GPOI 25
GND	Any	Any Ground
MISO	21	GPIO 9
MOSI	19	GPIO 10
SCK	23	GPIO 11
NSS	24	GPIO 8
IRQ	None	None

#### 4.2.2.b Raspberry Pi with microphone and speaker

Here the microphone and the speaker were connected to the Raspberry Pi using USB microphone adapter to deal with voice recognition as will be shown later.



**Figure4.4:** Microphone and speaker connection

#### 4.2.2.c Raspberry Pi with Wi-Fi

To enable Wi-Fi communication between the two Raspberry Pi's cisco Wi-Fi adapter drivers were attached to each Raspberry Pi.



**Figure4.5:** Wi-Fi adapter connection

## 4.3 Software Design

### 4.3.1 Voice recognition

Voice recognition was used to convert the voice which was taken from the microphone to text. For performing two libraries were used: SpeechRecognition and PyAudio.

SpeechRecognition is a Library for performing speech recognition, with support for several engines and APIs, online and offline. In this design Google speech API was employed. This API converts spoken text (microphone) into written text (Python strings), briefly Speech to Text. You can simply speak in a microphone and Google API will translate this into written text. The API has excellent results for English language.

PyAudio module is required to use microphone input.

#### Installation

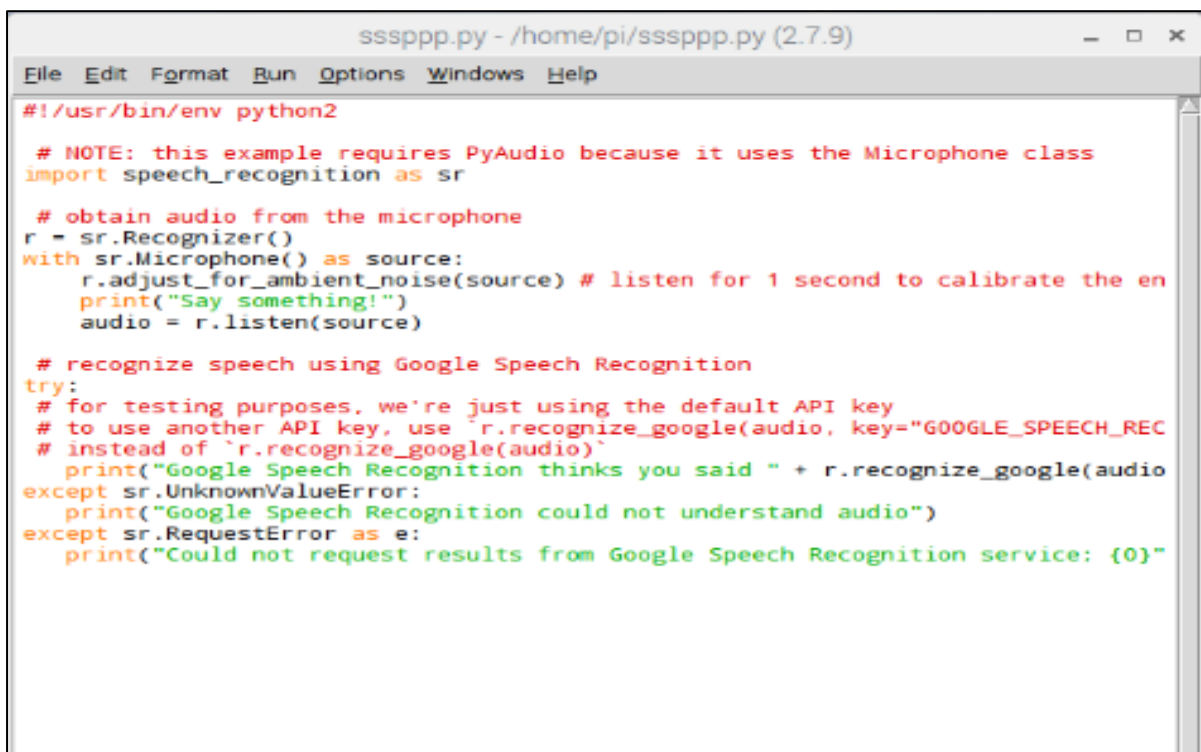
This is the installation guide for PyAudio and SpeechRecognition.

```
git clone http://people.csail.mit.edu/hubert/git/pyaudio.git
cd pyaudio
sudo python setup.py install
sudo apt-get install libportaudio-dev
sudo apt-get install python-dev
sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev
sudo pip3 install SpeechRecognition
```

#### Program

The following figure 4.6 show the program which record audio from microphone, send it to the speech API and return a Python string.

The audio was recorded using the speech recognition module; the module was included on top of the program. Then the recorded speech was send to the Google speech recognition API which then returns the output. **r.recognize\_google(audio)** returns a string.

A screenshot of a text editor window titled 'sssp.py - /home/pi/sssp.py (2.7.9)'. The window contains a Python script for voice recognition. The script starts with a shebang line, followed by a comment and an import statement for 'speech\_recognition as sr'. It then initializes a recognizer and a microphone, adjusts for ambient noise, and listens for 1 second. After that, it attempts to recognize the speech using Google Speech Recognition. The script includes try-except blocks to handle 'UnknownValueError' and 'RequestError' exceptions, providing feedback messages for each case.

```
#!/usr/bin/env python2

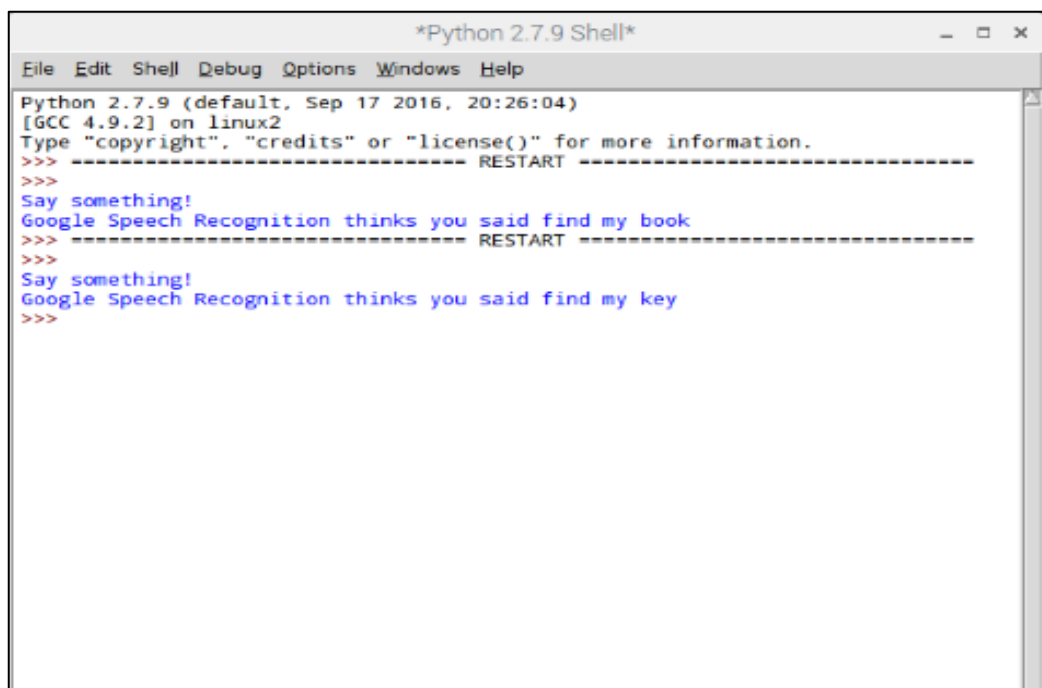
# NOTE: this example requires PyAudio because it uses the Microphone class
import speech_recognition as sr

# obtain audio from the microphone
r = sr.Recognizer()
with sr.Microphone() as source:
    r.adjust_for_ambient_noise(source) # listen for 1 second to calibrate the en
    print("Say something!")
    audio = r.listen(source)

# recognize speech using Google Speech Recognition
try:
    # for testing purposes, we're just using the default API key
    # to use another API key, use `r.recognize_google(audio, key="GOOGLE_SPEECH_REC
    # instead of `r.recognize_google(audio)`
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio)
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service: {0}")
```

**Figure4.6:** Voice recognition program

After running the program find my book and find my key were told to it, the result was as following:

A screenshot of a Python 2.7.9 Shell window. The window shows the output of the voice recognition program. It starts with the shell's header and version information. Then, it shows the program's output: 'Say something!', 'Google Speech Recognition thinks you said find my book', and 'RESTART'. This sequence is repeated once more with the input 'find my key'.

```
*Python 2.7.9 Shell*
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ----- RESTART -----
>>> Say something!
Google Speech Recognition thinks you said find my book
>>> ----- RESTART -----
>>> Say something!
Google Speech Recognition thinks you said find my key
>>>
```

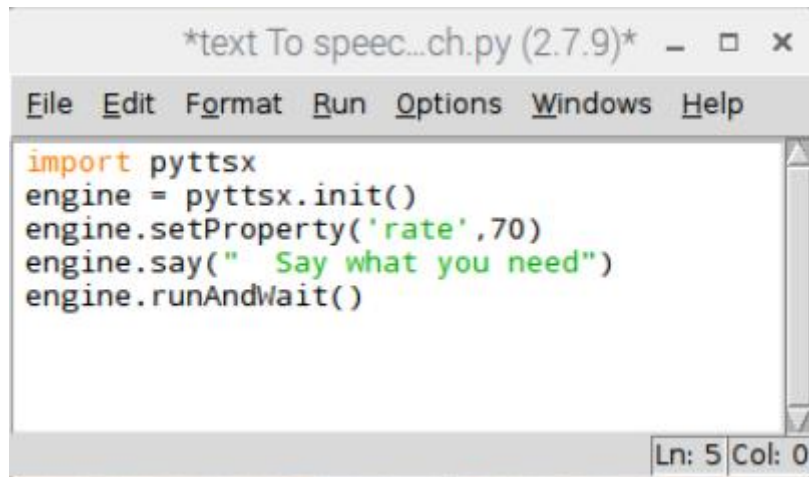
**Figure4.7:** The result of voice recognition

### 4.3.2 Speech Synthesis

Speech synthesis was used for saying message to the user using the speaker. For this purpose pyttsx library was installed to convert the text to voice by this command.

```
$ sudo pip install pyttsx
```

The code was:



```
*text To speech...ch.py (2.7.9)*  
File Edit Format Run Options Windows Help  
import pyttsx  
engine = pyttsx.init()  
engine.setProperty('rate',70)  
engine.say(" Say what you need")  
engine.runAndWait()  
Ln: 5 Col: 0
```

**Figure4.8:** text to speech code

Where rate mean integer speech rate in words per minute and we set it here to value 70.

### 4.3.3 Socket programming

For network communication socket programming was used between the two RPI's to communicate via Wi-Fi.

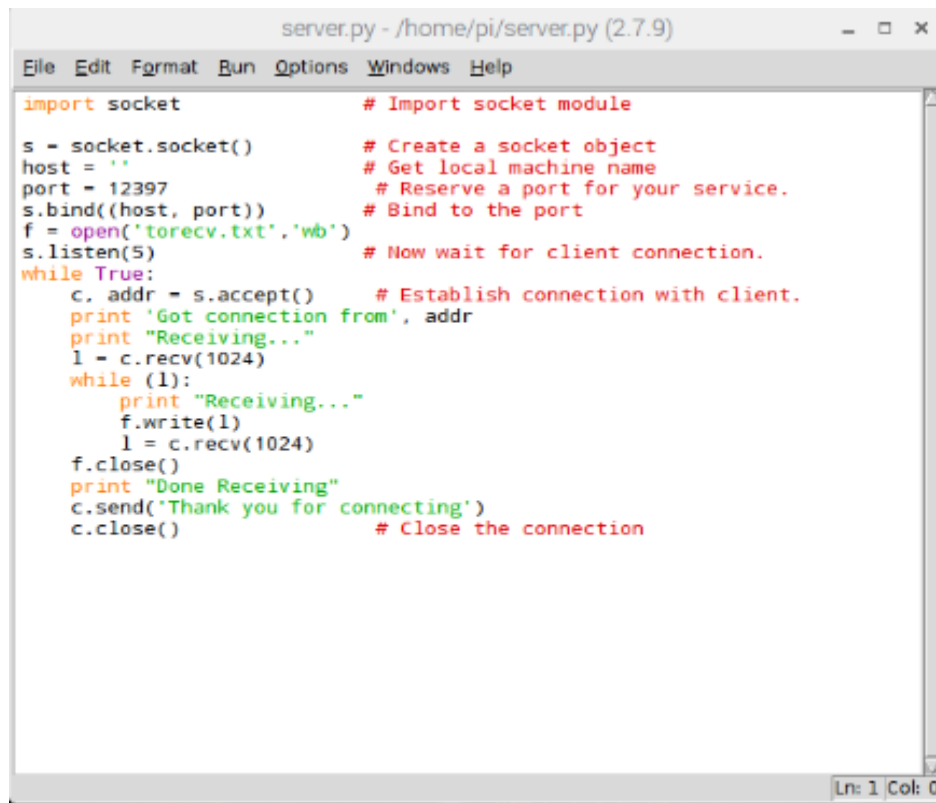
Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

We wrote two codes using socket module to make one of raspberry pi acts as server and the other as client to transfer file between them.

The server code and the result of running it shown in figures 4.9 & 4.10:

The server accepts the connection from the client and then writes the received data into file.



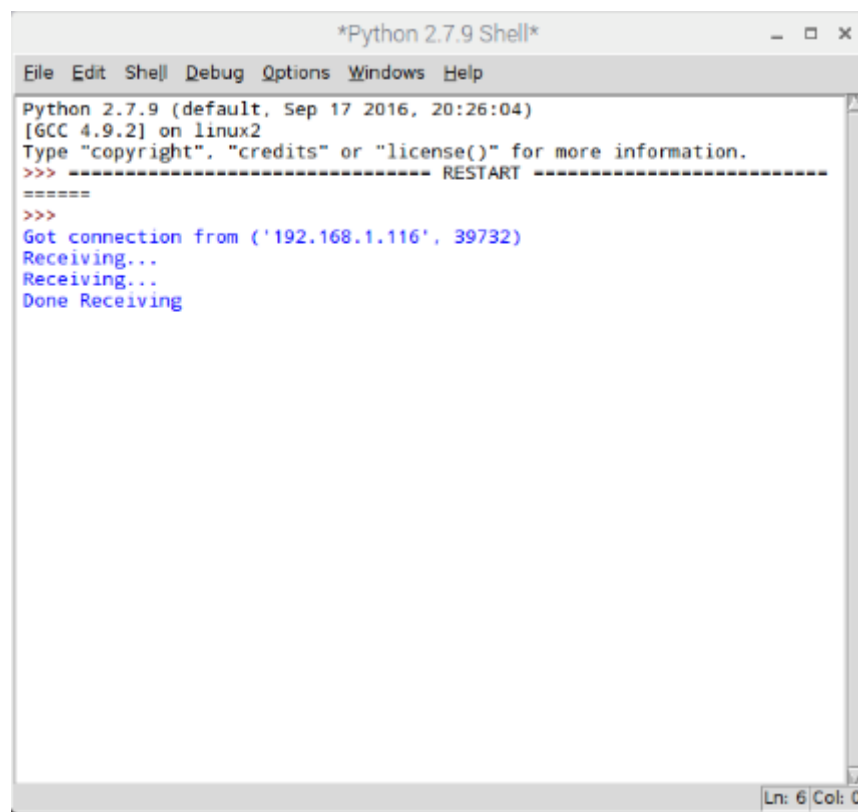
A screenshot of a text editor window titled "server.py - /home/pi/server.py (2.7.9)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code is written in Python and includes comments. It imports the socket module, creates a socket object, binds it to the local machine name and port 12397, and then enters a loop to accept connections. When a connection is accepted, it prints the address, receives data in a loop, and sends a thank you message before closing the connection.

```
import socket                # Import socket module

s = socket.socket()          # Create a socket object
host = ''                   # Get local machine name
port = 12397                # Reserve a port for your service.
s.bind((host, port))        # Bind to the port
f = open('torecv.txt', 'wb') # Now wait for client connection.
s.listen(5)

while True:
    c, addr = s.accept()     # Establish connection with client.
    print 'Got connection from', addr
    print "Receiving..."
    l = c.recv(1024)
    while (l):
        print "Receiving..."
        f.write(l)
        l = c.recv(1024)
    f.close()
    print "Done Receiving"
    c.send('Thank you for connecting')
    c.close()                # Close the connection
```

**Figure 4.9:** Server code

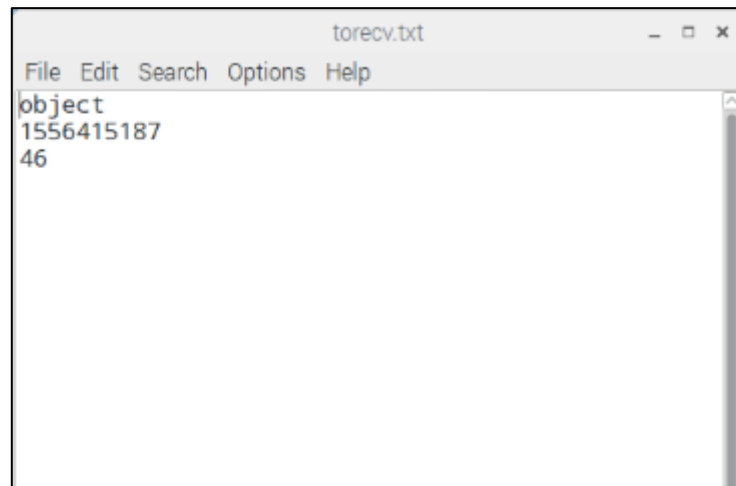
A screenshot of a Python 2.7.9 Shell window. The window title is "\*Python 2.7.9 Shell\*". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The shell output shows the Python version, GCC version, and the date. It then shows the execution of the server code, which prints "Got connection from ('192.168.1.116', 39732)", "Receiving...", "Receiving...", and "Done Receiving".

```
*Python 2.7.9 Shell*

Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ----- RESTART -----
>>>
>>> Got connection from ('192.168.1.116', 39732)
>>> Receiving...
>>> Receiving...
>>> Done Receiving
```

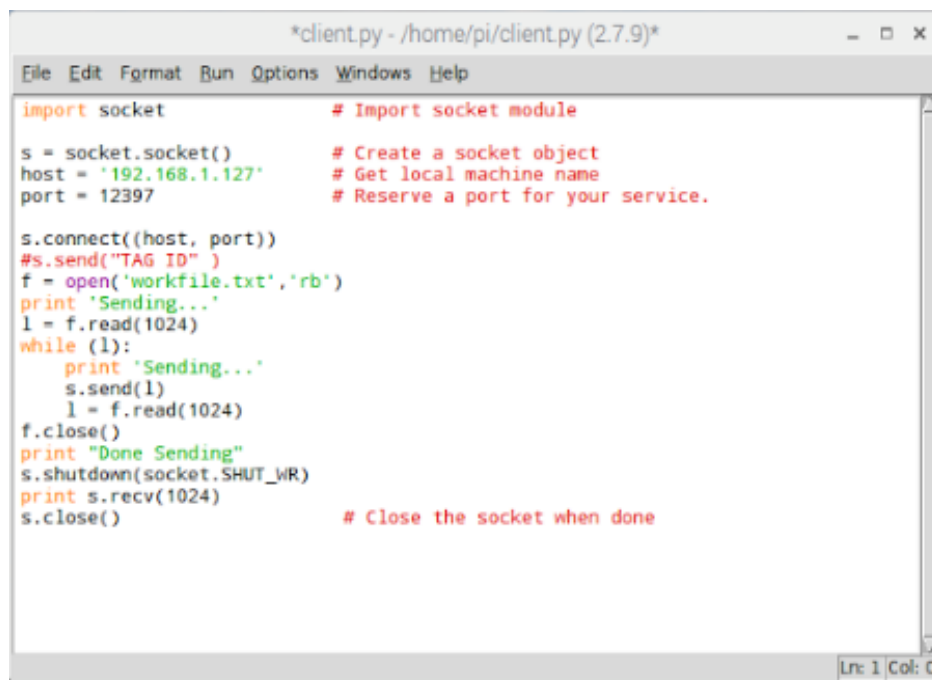
**Figure 4.10:** Running the server code

The file that received from the client was:



**Figure4.11:** Received file

The client connects to the server and then sends the file to the server as shown in figure 4.12.



**Figure4.12:** Client code

#### 4.3.4 SQL database

The SQL database was employed to store the entire tag id and its corresponding name (object name).

First we install the database on the Raspberry Pi.

```
sudo apt-get install python-MySQLdb
```

Second we create a database with the required field.

```
import MySQLdb

connection = MySQLdb.connect (host = "localhost",
                               user = "taguser",
                               passwd = "passwordhiba",
                               db = "rfid_tags")

sql_command = """
CREATE TABLE tags (
item INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
tag_type CHAR(20),
tag_id CHAR(20),
place INT);"""

cursor.execute(sql_command)
```

Third we write a code to insert the tag id which was read by the RFID reader and the object name to the database.

```
def insert():
    #inserting
    file_content1 = file.readline().rstrip('\n')
    file_content2 = object.readline().rstrip('\n')
    query = "INSERT INTO tags (tag_type,tag_id) VALUES (%s,%s)"
    cursor.execute(query, (file_content2,file_content1))
```

### 4.3.5 RFID-RC522

After connecting the RC522 RFID reader to the Raspberry Pi, we made few configuration changes on the Raspberry Pi to allow communication with RFID module.

First enable The Serial Peripheral Interface (SPI), which is a communication protocol used to transfer data between micro-computers like the Raspberry Pi and peripheral devices.

Second we downloaded the MFRC522-python library as shown:

```
cd~
```

```
git clone https://github.com/mxgxw/MFRC522-python.git
```

## 4.4 Initial version of the project

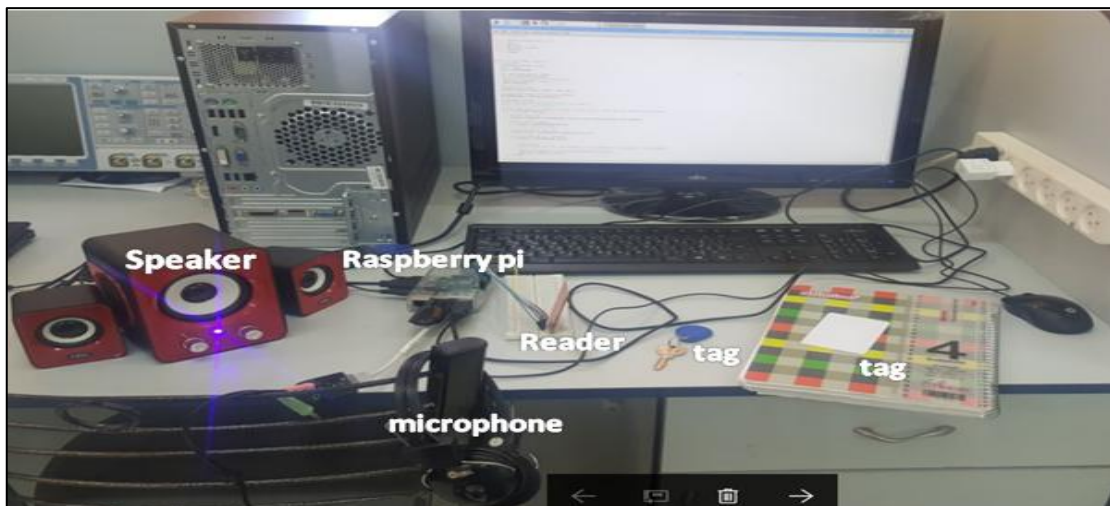
We did an experiment and built a simplified system that simulates our graduation project which locates the location of the missing object using RFID technology. We have done the work in three stages the first stage was voice recognition (convert speech to text) then reading the tag by RFID reader and finally voice synthesis (convert text to speech).

The hardware and the software that required for this experiment were:

Hardware Required:

- Raspberry Pi 2
- MFRC522 RFID Reader
- RFID Tags
- USB Microphone
- Speaker

Hardware setup is pretty simple and it was as shown in figure 4.13:

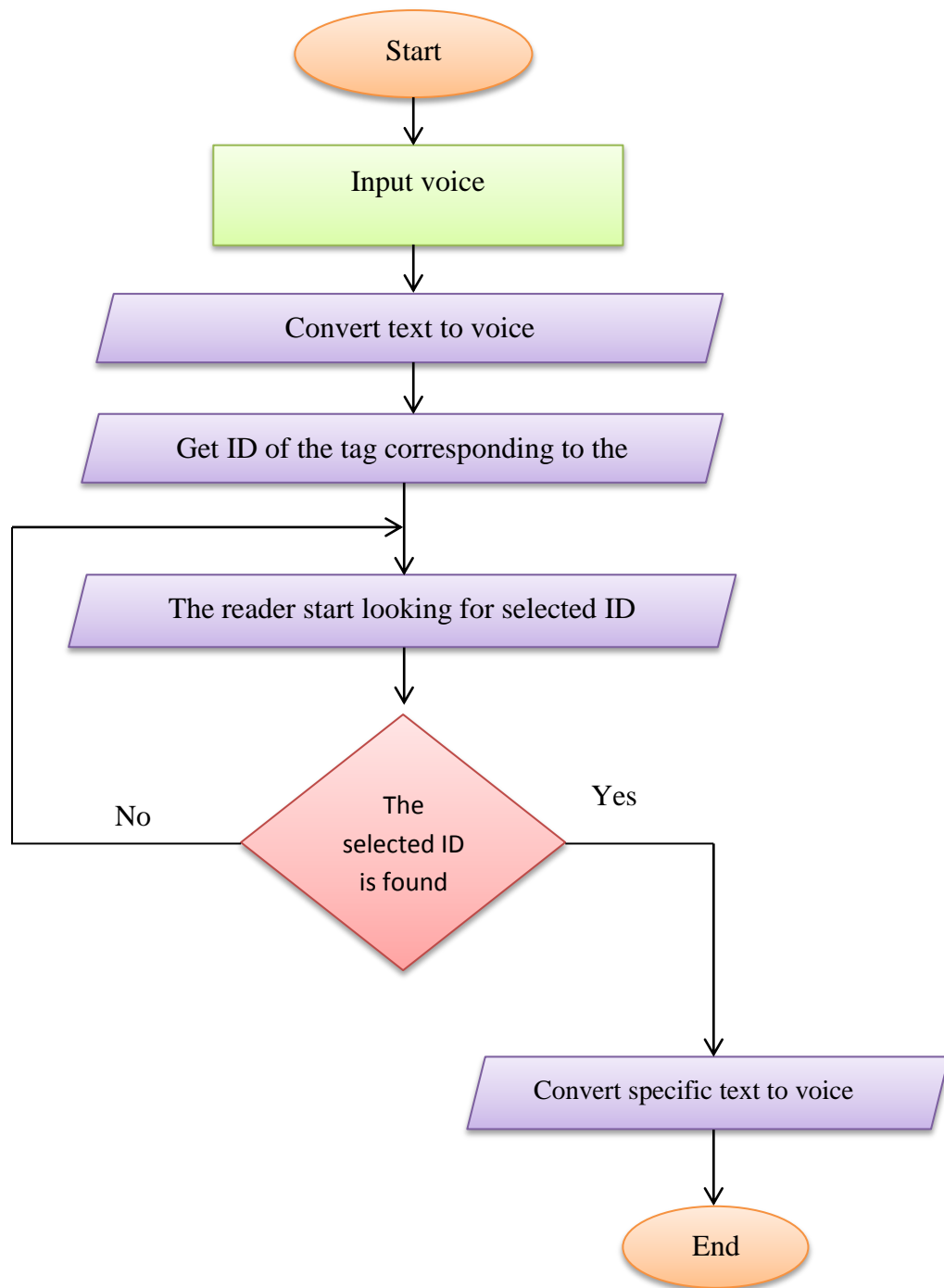


**Figure 4.13:** Hardware setup

Software required:

- Python pyttax library
- Python speech recognition library
- Python RC522 library

The system operate as illustrated in the flowchart



**Figure 4.14:** The process of the initial version of the project

To locate the place of a missing thing we complete the hardware setup phase then wrote a python code into the raspberry Pi.

First we define two objects by connecting tags to them: book which tag id is 1556415187 and key which tag id is 16616334126 and recorded these tags id into database.

Second voice recognition was used to take the voice from the person using microphone and convert this voice to text, and then take this text.

If this text was “book” the reader read the tags in its rang and see if the reading tag has the same book tag id, if so use Pyttsx, which is a software convert text to voice, to make the speaker say “I find your book”, and if the tag id was not the same then say “I can’t find your book move the reader.”. As shown:

```
if (word == 'book' or word == 'book book' or word == 'find my book' or
word == 'find my puppy'):
    if num == '1556415187' :
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say("I find your book")
        engine.runAndWait()

    elif num != '1556415187':
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say("I can not find your book move the reader")
        engine.runAndWait()
```

If this text was “key” the reader start reading and see if the reading tag has the same key tag id if so use Pyttsx to make the speaker says “I find you’re key”, and if the tag id was not the same then say “I can’t find your key move the reader”. As shown:

```
if (word == 'monkey' or word == 'mikey' or word == 'find my key' or word ==
'find monkey' or word == 'find party' or word == 'find my teeth') :
    print(word)
    if num == '16616334126' :
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say("I find your key")
        engine.runAndWait()
        y = False
    elif num != '16616334126':
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say("I can not find your key move the reader")
        engine.runAndWait()
        y = True
```

# Chapter 5

## Testing and Results

---

- 5.1 Overview
- 5.2 Testing and results
  - 5.2.1 RFID reader testing
  - 5.2.2 Voice recognition testing
- 5.3 Performance evaluation and analysis



## 5.1 Overview

The final stage to achieve the project is to test the system to get result and measure its performance. This chapter show all testing needed to evaluate the performance of our system.

## 5.2 Testing and results

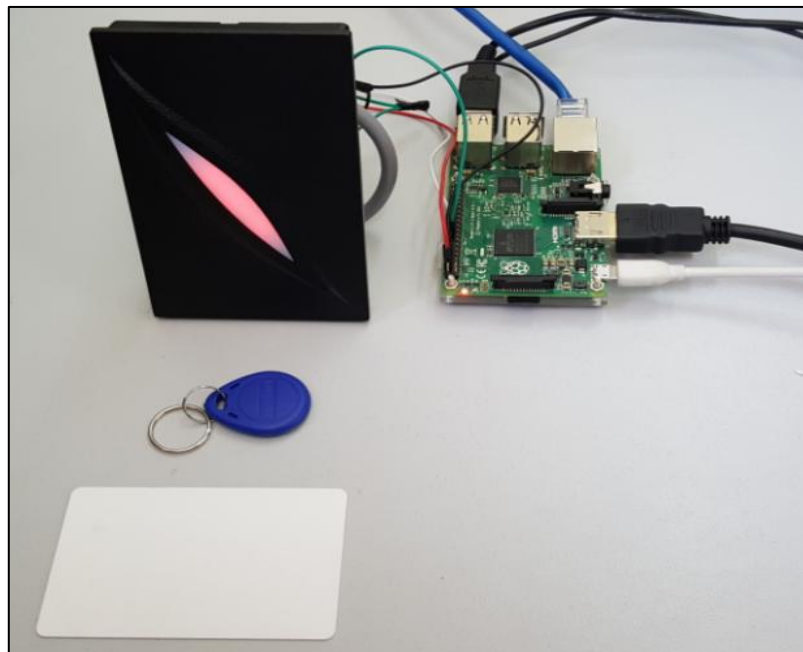
Checking and testing of the raspberry pi and RFID reader were done in this section including the tasting results.

### 5.2.1 RFID reader testing

We did several experiments to test the RFID reader work. These experiments were conducted to test three parameters they are:

1. The range of the RFID reader
2. Multi-tags reading.
3. The error rate of detecting a specific tag

To do the first test we connected the reader with the raspberry pi as shown in figure5.1 and the wiring as in table 5.1 then running the code to start scanning.



**Figure 5.1:** Reading range

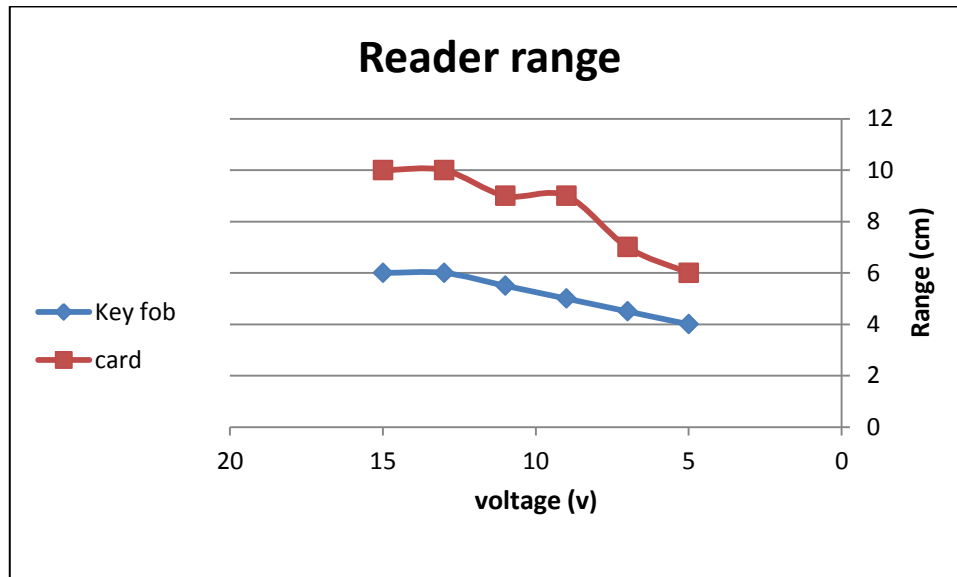
**Table5.1:** Wiring w26 reader with raspberry pi

Name	Pin Number	Pin Name
VCC	4	5.0 v
GND	6	0 v
Data0	16	GPIO 23
Data1	18	GPIO 24

We tested the range of the reader at different supplied voltages. The result of the first experiment was as follow:

**Table 1:** the range of w26 reader

	Key fob tag	Card tag
5 volt	4 cm	6 cm
7 volt	4.5 cm	7 cm
9 volt	5 cm	9 cm
11 volt	5.5 cm	9 cm
13 volt	6 cm	10 cm
15 volt	6 cm	10 cm



The result show that the range of the RFID reader depends on the supplied voltage and the type of the tag. Where increasing the voltage increases the readings range. This is because the reader must generate energy. This energy must be coupled from the reader to the tag, and the tag must use it efficiently. Therefore, maximum reader power output, the coupling of the energy [58]

from reader to tag become better and the range improved. The tag's antenna aperture also effect on the range. For this reason the card tag has reading range larger than key fob tag.

### Theory and Equations:

Here we are going to go into some theory. The equation for the Power Transmission Coefficient,  $\tau$ , which describes the impedance match. Here, as  $\tau$  tends to unity, the better the impedance match between the chip and antenna, as follows:

$$\tau = \frac{4R_c R_a}{|Z_c + Z_a|^2}$$

Here,  $R_c$  and  $R_a$  are chip and antenna resistance, respectively.  $Z_c$  and  $Z_a$  are chip and antenna impedance, respectively. Additionally, by making use of the Friis free-space equation, we can also obtain an equation for read range,  $r$ :

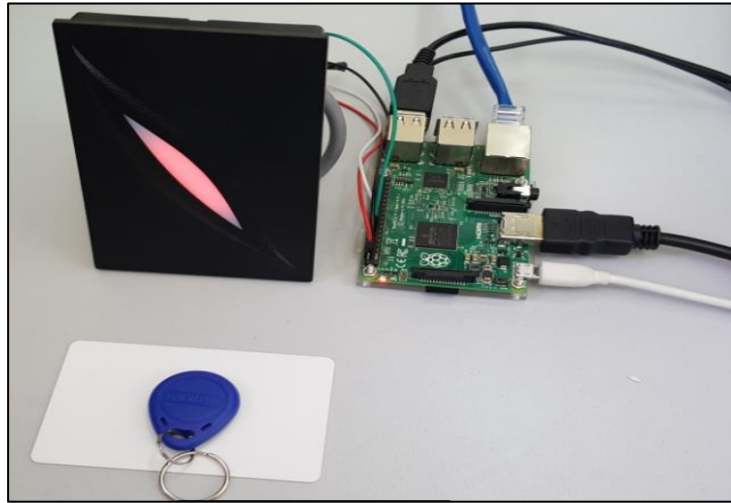
$$r = \frac{\lambda}{4\pi} \sqrt{\frac{P_r G_r G_a \tau}{P_{th}}}$$

Here,  $\lambda$  is the wavelength,  $P_r$  is the power transmitted by the reader,  $G_r$  is the reader antenna gain,  $G_a$  is the gain of the receiving tag antenna, and  $P_{th}$  is the minimum threshold power. The peak read range,  $r$ , across a frequency range can be referred to as the tag's resonance and will coincide with the maximum power transmission coefficient,  $\tau$ .<sup>(26)</sup>

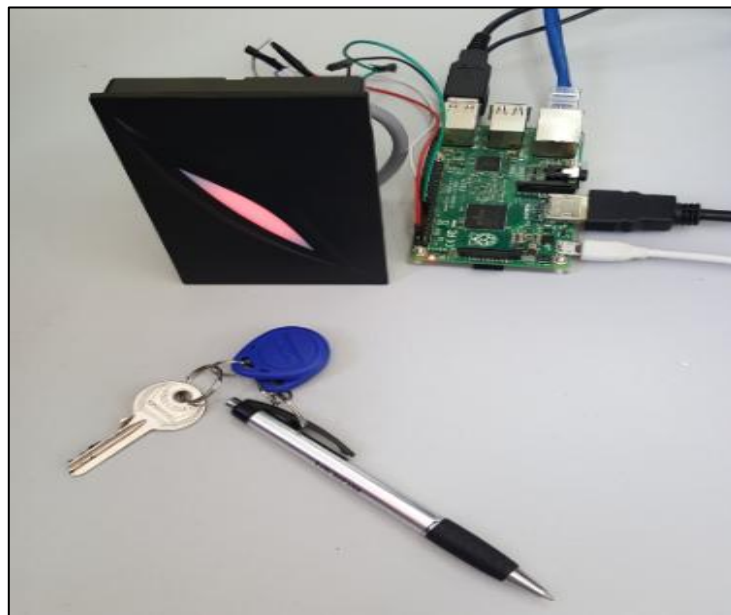
The variables  $\lambda$ ,  $G_r$ ,  $G_a$ ,  $\tau$  and  $P_{th}$  are constant where  $P_r$  is variable according to the voltage feed to the reader. So changing the voltage lead to change the read range.

What we conclude from this experience is that increasing the effort given to the reader leads to increasing the distance of the reading without exceeding the maximum amount of voltages borne by the reader

In the second test we wanted to check the ability of the reader to detect multi-tag at the same time, for that two tags were put above each other as in the figures 5.2 and 5.3



**Figure 5.2 :** Card and key fob tags



**Figure 5.3:** Two key fob tags

In the first case the card tag was always detected due to the difference of antenna aperture where the antenna aperture of the card is bigger than key fob, so it absorbs most of the transmitted energy from the reader as shown in figure 5.4. In the second case the upper tag was always detected which was attached with key as shown in figure 5.5 and that because the upper tag absorb all the transmitted signal. This proves that the reader can't read multi-tag at the same time.

```

pi@raspberrypi:~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo python /home/pi/main.py
('Initializing reader 1...'. ' ')
WARNING:root:Kernel interface for GPIO 23 already exists.
WARNING:root:Kernel interface for GPIO 24 already exists.
Done !
('Initializing reader 2...'. ' ')
WARNING:root:Kernel interface for GPIO 8 already exists.
WARNING:root:Kernel interface for GPIO 7 already exists.
Done !
Ready to go !
10111100101100010000011
011110010110001000001
[reader] Frame of length (23): 10111100101100010000011 (994369) OK K0I
the card is detected
[reader] Frame of length (3):110 DROPPED
1111100101100010000011
11110010110001000001
[reader] Frame of length (22): 1111100101100010000011 (994369) OK K0I
the card is detected
[reader] Frame of length (3):110 DROPPED

```

**Figure5.4:** Card detected

```

pi@raspberrypi:~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo python /home/pi/main.py
('Initializing reader 1...'. ' ')
Done !
('Initializing reader 2...'. ' ')
Done !
Ready to go !
10111100101011110001111
011110010101111000111
[reader] Frame of length (23): 10111100101011110001111 (994247) OK K0I
the key is detected
[reader] Frame of length (2):10 DROPPED
10111100101011110001111
011110010101111000111
[reader] Frame of length (23): 10111100101011110001111 (994247) OK K0I
the key is detected
[reader] Frame of length (3):110 DROPPED

```

**Figure5.5:** key detected

The third test aimed to measure the error rate of detecting a specific tag, firstly we examined card tag by scanning the card 20 times, the correct reading was 16 time. Secondly we examined key fob tag and the correct reading was 19 time. The error rate was as follow:

Error rate (card) = 20%

Error rate (key fob) = 5%

## 5.2.2 Voice recognition testing

In this section, we tested Google speech recognition to determine the error rate. For the test we said about fifty word and sentence. Here is a sample of words and sentences that have been tried(written in red) in figure 5.6

```

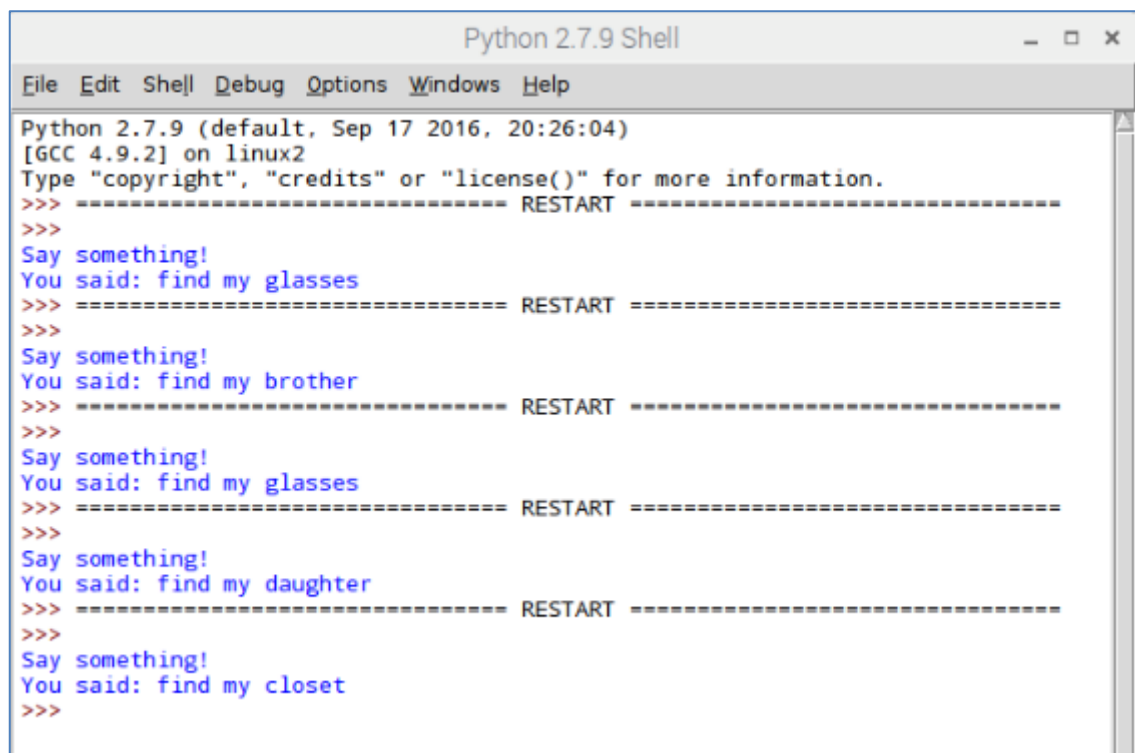
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>> Say something! hello
You said: hello
>>> ===== RESTART =====
>>> Say something! Find my book
You said: find my phone
>>> ===== RESTART =====
>>> Say something! book
Google Speech Recognition could not understand audio
>>> ===== RESTART =====
>>> Say something! Find my phone
You said: find my phone
>>> ===== RESTART =====
>>> Say something! laptop
You said: laptop
>>> ===== RESTART =====
>>> Say something! Find my key
You said: find Mikey
>>> ===== RESTART =====
>>> Say something! Find my glasses
You said: find my blood type
>>> ===== RESTART =====
>>> Say something! Find my book
You said: find my book
>>> ===== RESTART =====
>>> Say something! remote
You said: remote
>>> ===== RESTART =====
>>> Say something! bag
Google Speech Recognition could not understand audio
>>> ===== RESTART =====
>>> Say something! Find my bag
You said: find my bed
>>> ===== RESTART =====
>>> Say something! Find my bag
You said: find my bag
>>>

```

**Figure5.6:** sample of testing voice recognition

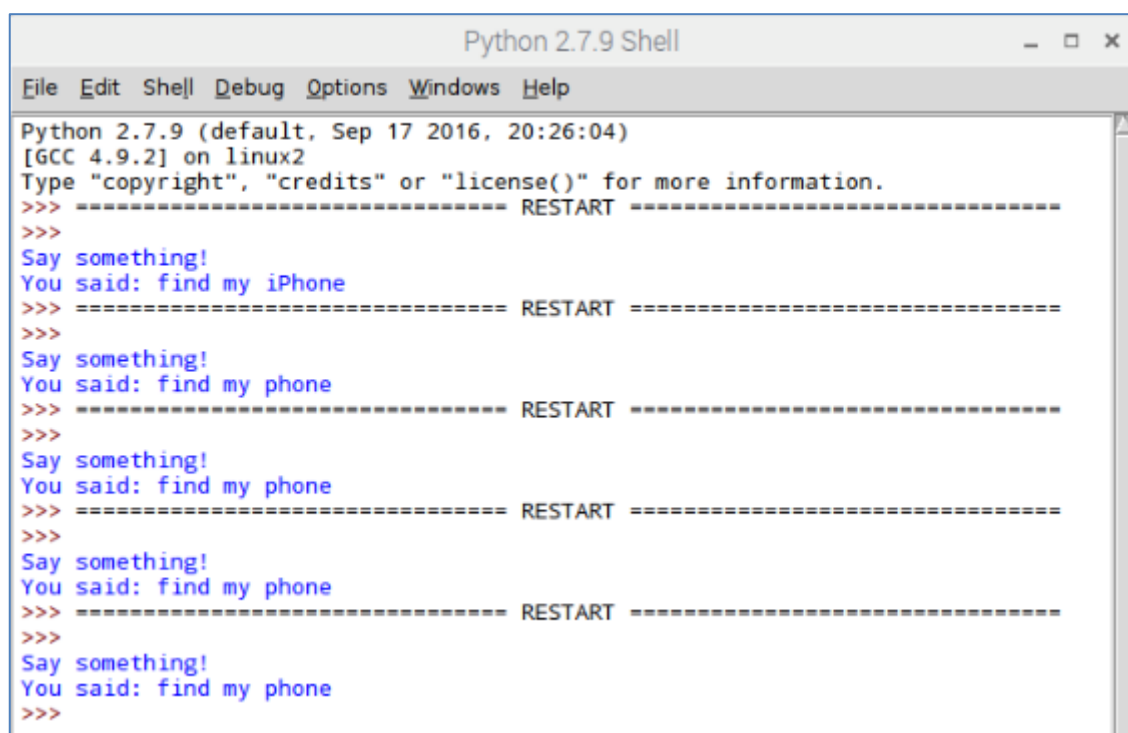
The error rate after all trials was around 50%, which is high. To override this problem we checked all the possible choices for the same word. These possibilities were taken in consideration in building the database.

As example, we said ' find my glasses ' and ' find my phone ', then the most repetitive possibilities added to the database. These possibilities emerged in figure5.7 and 5.8



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Say something!
You said: find my glasses
>>> ===== RESTART =====
>>>
Say something!
You said: find my brother
>>> ===== RESTART =====
>>>
Say something!
You said: find my glasses
>>> ===== RESTART =====
>>>
Say something!
You said: find my daughter
>>> ===== RESTART =====
>>>
Say something!
You said: find my closet
>>>
```

**Figure5.7:** Recognition errors ‘find my glasses’



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Say something!
You said: find my iPhone
>>> ===== RESTART =====
>>>
Say something!
You said: find my phone
>>> ===== RESTART =====
>>>
Say something!
You said: find my phone
>>> ===== RESTART =====
>>>
Say something!
You said: find my phone
>>> ===== RESTART =====
>>>
Say something!
You said: find my phone
>>>
```

**Figure5.8:** Recognition error for ' find my phone’

As a result of taking the most repetitive possibilities, the error rate decreased to 25%.

### 5.3 Performance evaluation and analysis

To evaluate the work of the system we tested all stages one after another where we select the wanted process, searching process and adding process.

#### 5.3.1 Select the wanted process

The first process that the user of the system do it, is selecting the desired operation either searching or adding. Two buttons connected to Raspberry Pi, one for searching and the other for adding.

Figure 5.9 indicates that the first button is pressed to start searching and Figure 5.10 indicates that the second button is pressed.

```
press button one to search, two to add new tag
Button 1 Pressed
say what you need
Google Speech Recognition thinks you said find my book
find my book
item=1,tag_type=find my book,tag_id=994369, place=1
994369
('Initializing reader 0...', '')
/home/pi/cardReader.py:44: RuntimeWarning: This channel is already in use,
nuing anyway. Use RPIO.setwarnings(False) to disable warnings.
  RPIO.setup(self.GPIO_0, RPIO.IN)
/home/pi/cardReader.py:45: RuntimeWarning: This channel is already in use,
nuing anyway. Use RPIO.setwarnings(False) to disable warnings.
  RPIO.setup(self.GPIO_1, RPIO.IN)
Done !
Ready to go !
```

Figure 5.9: Button 1 pressed

```
press button one to search, two to add new tag
Button 2 Pressed
say what to add
Google Speech Recognition thinks you said laptop
laptop
repeat again
Google can not understand audio
```

Figure 5.10: Button 2 pressed

#### 5.3.2 Searching process

After pressing the first button and selecting the search process, we said the name of the lost object. This object may be in place one or place two or out of readers range.



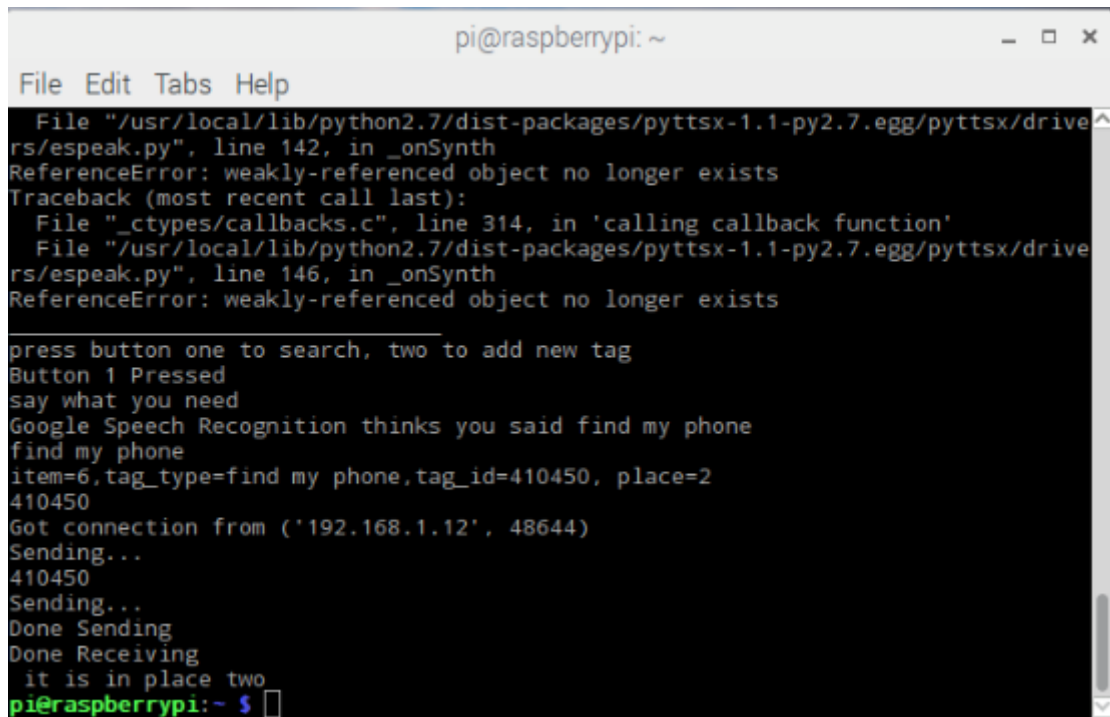
We said find my book which is in place 1, the result was as in figure5.11

```
press button one to search, two to add new tag
Button 1 Pressed
Google Speech Recognition thinks you said find my book
find my book
item=1,tag_type=find my book,tag_id=994369, place=1
994369
('Initializing reader 0...', '')
WARNING:root:Kernel interface for GPIO 23 already exists.
WARNING:root:Kernel interface for GPIO 24 already exists.
Done !
Ready to go !
01111001011000100000111
111100101100010000011
[reader] Frame of length (23): 01111001011000100000111 (1988739) - PARITY CHECK
FAILED
[reader] Frame of length (2):10 DROPPED
1011110010110001000011
011110010110001000001
[reader] Frame of length (22): 1011110010110001000011 (497185) OK KOI
1011110010110001000011
011110010110001000001
('994369', '497185', 1)
[reader] Frame of length (2):10 DROPPED
10111100101100010000011
011110010110001000001
[reader] Frame of length (23): 10111100101100010000011 (994369) OK KOI
10111100101100010000011
011110010110001000001
('994369', '994369', 2)
I find the wanted object
pi@raspberrypi:~ $
```

**Figure5.11:** searching in place 1

There is an error in the reading as shown in the previous figure, the correct reading or detecting the object is done after three reading two of them were wrong. To overcome this problem, we have repeated the process of scanning the tags until the desired thing is found.

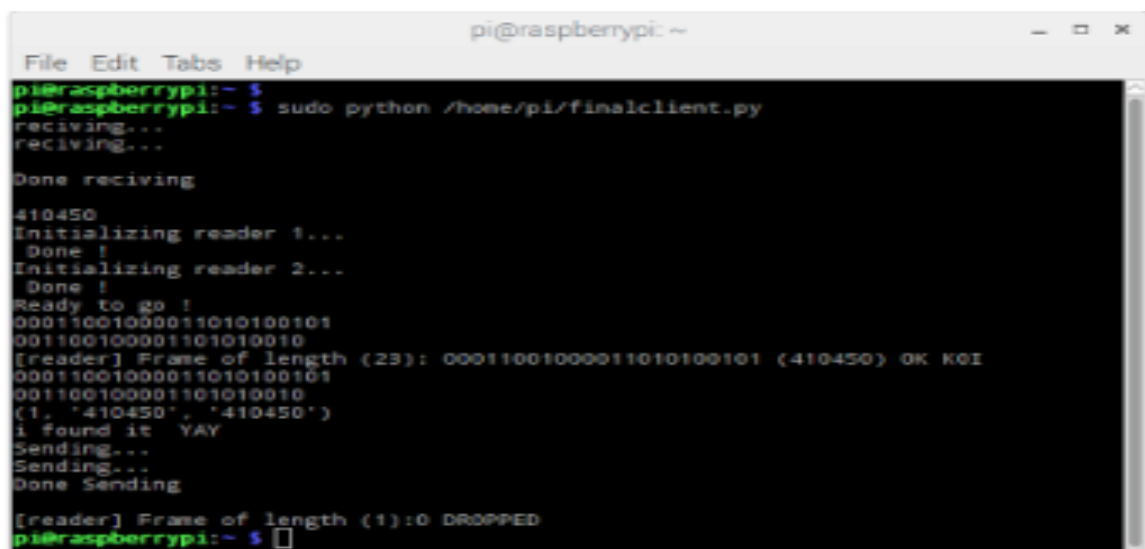
Then we said find my phone which is in place 2, the result was as follow in figure 5.12



```
pi@raspberrypi: ~  
File Edit Tabs Help  
File "/usr/local/lib/python2.7/dist-packages/pytttsx-1.1-py2.7.egg/pytttsx/drive  
rs/espeak.py", line 142, in _onSynth  
ReferenceError: weakly-referenced object no longer exists  
Traceback (most recent call last):  
  File "_ctypes/callbacks.c", line 314, in 'calling callback function'  
  File "/usr/local/lib/python2.7/dist-packages/pytttsx-1.1-py2.7.egg/pytttsx/drive  
rs/espeak.py", line 146, in _onSynth  
ReferenceError: weakly-referenced object no longer exists  
press button one to search, two to add new tag  
Button 1 Pressed  
say what you need  
Google Speech Recognition thinks you said find my phone  
find my phone  
item=6,tag_type=find my phone,tag_id=410450, place=2  
410450  
Got connection from ('192.168.1.12', 48644)  
Sending...  
410450  
Sending...  
Done Sending  
Done Receiving  
it is in place two  
pi@raspberrypi:~$
```

**Figure5.12:** Searching in place2 (server side)

My phone was in place two so the reading order was sent to client side (place 2). The reader in place two completed the scanning and returned the result to the server side (place 1). The server side presents the mobile reader while the client side presents the fixed reader.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~$  
pi@raspberrypi:~$ sudo python /home/pi/finalclient.py  
receiving...  
receiving...  
Done receiving  
410450  
Initializing reader 1...  
Done !  
Initializing reader 2...  
Done !  
Ready to go !  
00011001000011010100101  
001100100001101010010  
[reader] Frame of length (23): 00011001000011010100101 (410450) OK K0I  
00011001000011010100101  
001100100001101010010  
(1, '410450', '410450')  
i found it YAY  
Sending...  
Sending...  
Done Sending  
[reader] Frame of length (1):0 DROPPED  
pi@raspberrypi:~$
```

**Figure5.13 :** Searching in place2 (client side)

Finally, the case of not existing the object in the readers range has been treated , the figures below illustrate that .

```
press button one to search, two to add new tag
Button 1 Pressed
say what you need
Google Speech Recognition thinks you said find my book
item=1,tag_type=find my book,tag_id=994369, place=1
994369
('Initializing reader 0...', '')
Done !
Ready to go !
10111100101011110001111
011110010101111000111
[reader] Frame of length (23): 10111100101011110001111 (994247) OK KOI
10111100101011110001111
011110010101111000111
('994369', '994247', 1)
[reader] Frame of length (3):110 DROPPED
10111100101011110001111
011110010101111000111
[reader] Frame of length (23): 10111100101011110001111 (994247) OK KOI
10111100101011110001111
011110010101111000111
('994369', '994247', 2)
[reader] Frame of length (3):110 DROPPED
10111100101011110001111
011110010101111000111
[reader] Frame of length (23): 10111100101011110001111 (994247) OK KOI
10111100101011110001111
011110010101111000111
('994369', '994247', 3)
I can not find it, out of range
pi@raspberrypi:~ $
```

**Figure5.14** : Out of range mobile reader

Figure5.15 and figure5.16 shows the result if the object was out of range fixed reader.

```
press button one to search, two to add new tag
Button 1 Pressed
say what you need
Google Speech Recognition thinks you said find my phone
item=7,tag_type=find my phone,tag_id=410450, place=2
410450
Got connection from ('192.168.1.9', 47654)
Sending...
410450
Sending...
Done Sending
Done Receiving
pi@raspberrypi:~ $
```

**Figure5.15** : Out of range fixed reader (Server side)

```

pi@raspberrypi:~$ sudo python /home/pi/finalClient.py
receiving...
receiving...
Done receiving
410450
Initializing reader 1...
Done !
Initializing reader 2...
Done !
Ready to go !
10111100101100010000011
011110010110001000001
[reader] Frame of length (23): 10111100101100010000011 (994369) OK KOI
10111100101100010000011
011110010110001000001
(1, '994369', '410450')
[reader] Frame of length (3):110 DROPPED
10111100101100010000011
011110010110001000001
[reader] Frame of length (22): 10111100101100010000011 (994369) OK KOI
10111100101100010000011
011110010110001000001
(2, '994369', '410450')
[reader] Frame of length (3):110 DROPPED
10111100101100010000011
011110010110001000001
[reader] Frame of length (23): 10111100101100010000011 (994369) OK KOI
10111100101100010000011
011110010110001000001
(3, '994369', '410450')
not exist
----Out of the range-----
Sending...
Sending...
Done Sending
pi@raspberrypi:~$

```

Figure5.16 : Out of range fixed reader (Client side)

### 5.3.3 Adding process

The adding process used to add new object to the data base with its id and place. For adding the second button was pressed.

The figure bellow shows the original data base before adding a new tag .

```

pi@raspberrypi: ~
File Edit Tabs Help
mysql> select * from tags:
+-----+-----+-----+-----+
| item | tag_type | tag_id | place |
+-----+-----+-----+-----+
| 1 | find my book | 994369 | 1 |
| 2 | book | 994369 | 1 |
| 3 | book book | 994369 | 1 |
| 4 | find Michael | 994369 | 1 |
| 5 | find my puppy | 994369 | 1 |
| 6 | find my phone | 410450 | 2 |
| 7 | find my iphone | 410450 | 2 |
| 8 | phone | 410450 | 2 |
| 9 | find my remote | 988216 | 2 |
| 10 | 120 mos | 988216 | 2 |
| 11 | find my vitamix | 988216 | 2 |
| 12 | remote | 988216 | 2 |
| 13 | find my key | 994247 | 1 |
| 14 | find my keys | 994247 | 1 |
| 15 | find monkey | 994247 | 1 |
| 16 | find molly | 994247 | 1 |
+-----+-----+-----+-----+
16 rows in set (0.00 sec)
mysql>

```

Figure5.17: Database before adding

To add new tag we firstly said the name of the new object more than one time for error handling in voice recognition, and then the new tag was scanned to get the id of this tag. Finally we selected the place number. These steps are obvious in figure 5.18

```

press button one to search, two to add new tag
Button 2 Pressed
say what to add
Google Speech Recognition thinks you said laptop
laptop
repeat again
Google Speech Recognition thinks you said laptop
laptop
('Initializing reader 0...', '')
Done !
Ready to go !
10011001010101010011001
001100101010101001100
[reader] Frame of length (23): 10011001010101010011001 (415052) OK KOI
10011001010101010011001
001100101010101001100
('1000', '415052', 1)
I add the tag
say the place number
Google Speech Recognition thinks you said list two
list two
415052
laptop
415052
laptop
pi@raspberrypi:~ $ sudo python2 /home/pi/finalV3.py

```

Figure 5.18: Adding steps

The data base after adding the new object become as in figure 5.19

```

pi@raspberrypi: ~
File Edit Tabs Help
Database changed
mysql> select * from tags;
+-----+-----+-----+-----+
| item | tag_type | tag_id | place |
+-----+-----+-----+-----+
| 1 | find my book | 994369 | 1 |
| 2 | book | 994369 | 1 |
| 3 | book book | 994369 | 1 |
| 4 | find Michael | 994369 | 1 |
| 5 | find my puppy | 994369 | 1 |
| 6 | find my phone | 410450 | 2 |
| 7 | find my iphone | 410450 | 2 |
| 8 | phone | 410450 | 2 |
| 9 | find my remote | 988216 | 2 |
| 10 | 120 mos | 988216 | 2 |
| 11 | find my vitamix | 988216 | 2 |
| 12 | remote | 988216 | 2 |
| 13 | find my key | 994247 | 1 |
| 14 | find my keys | 994247 | 1 |
| 15 | find monkey | 994247 | 1 |
| 16 | find molly | 994247 | 1 |
| 17 | laptop | 415052 | 2 |
| 18 | laptop | 415052 | 2 |
+-----+-----+-----+-----+
18 rows in set (0.00 sec)

mysql>

```

Figure5.19: Database after adding

If a search is requested and the required object is not present in the database, the system will ask the blind to add this object to the database as follow in the figures 5.20, 5.21 and 5.22

```

press button one to search, two to add new tag
Button 2 Pressed
say what to add
Google Speech Recognition thinks you said computer
computer
repeat again
Google Speech Recognition thinks you said find my computer
find my computer
('Initializing reader 0...', '')
Done !
Ready to go !
10011001000110101010110
001100100011010101011
[reader] Frame of length (23): 10011001000110101010110 (411307) OK K0I
10011001000110101010110
001100100011010101011
('1000', '411307', 1)
I add the tag
say the place number
Google Speech Recognition thinks you said lyrics to
lyrics to
411307
computer
411307
find my computer
0
pi@raspberrypi:~ $

```

Figure5.20 : The steps for adding

item	tag_type	tag_id	place
1	find my book	994369	1
2	book	994369	1
3	book book	994369	1
4	my book	994369	1
5	find Michael	994369	1
6	find my puppy	994369	1
7	find my phone	410450	2
8	find my iphone	410450	2
9	phone	410450	2
10	find my remote	988216	2
11	120 mos	988216	2
12	find my vitamix	988216	2
13	remote	988216	2
14	find my key	994247	1
15	find my keys	994247	1
16	find monkey	994247	1
17	find molly	994247	1

17 rows in set (0.01 sec)

mysql>

Figure5.21 : Before adding

item	tag_type	tag_id	place
1	find my book	994369	1
2	book	994369	1
3	book book	994369	1
4	my book	994369	1
5	find Michael	994369	1
6	find my puppy	994369	1
7	find my phone	410450	2
8	find my iphone	410450	2
9	phone	410450	2
10	find my remote	988216	2
11	120 mos	988216	2
12	find my vitamix	988216	2
13	remote	988216	2
14	find my key	994247	1
15	find my keys	994247	1
16	find monkey	994247	1
17	find molly	994247	1
18	computer	411307	2
19	find my computer	411307	2

19 rows in set (0.00 sec)

mysql>

Figure5.22 : After adding

# Chapter 6

## Recommendations and Conclusion

---

- 6.1 Overview
- 6.2 System achievements
- 6.3 Real learning outcomes
- 6.4 Recommendations
- 6.5 Conclusion



## **6.1 Overview**

The project has been done step by step, for employ the RFID technology to locate things. The project was making the life of the blind much easier through practical way to find the missing things. Meanwhile we have some recommendations and suggestions for future work that can be taken to make the system more efficient.

## **6.2 System achievement**

Almost all the goals of our system have been achieved. In this point the main achievements of the system are discussed and the ways of achieving it.

We deal with the RFID technology and programme the RFID reader to read the tags.

We use the voice recognition and the text to speech to make the system easy to use.

We built a database and insert tags into it and search in it.

We structure an Object Locator using RFID technology, using all these achievements.

We create network between two RFID reader for searching in many places.

## **6.3 Real learning outcomes**

After the implementation of the project we have become an expert in the following points:

- Learn how to programme the Raspberry Pi and use it
- Learn how to deal and programme the RFID technology
- Learn how to make socket programming
- Learn how to use the voice recognition and speech to text software
- Learn how to build a database using SQL
- Learn how to solve many problems that we face in voice recognition and RFID reader



## **6.4 Recommendations**

After we worked on this project, and faced many problems during the implementation, we saw the following points may be good improvement for this project in order to make it more reliable:

- An improvement to the system could be applied, by using Cottonwood: UHF Long distance RFID reader module to increase the reading range up to 5 meters.
- Try to use the Arabic language in searching and replying process.

## **6.5 Conclusion**

Finally at the end of the long work in this project, we built the object locator using RFID technology for blind people, and it has efficiently worked in the way we imagine. We also solve the problems and challenges that we faced.

The general challenges include how to deal with the RFID reader and programme it to work in the way we want, another challenge was how to collect all the hardware's and software's and built the system and the biggest challenge was the first moment we turn on the system, and start finding the way to solve the critical case, which how to understand what the blind say.

And the specific challenges were:

1. Voice recognition translate differently the same word each time we say it
2. Tell the reader to stop reading the tags when the wanted tag found
3. Increase the reader range

But at the end of the day it was nice thing to see our dream and effort in building this system becomes true, also it was an interesting work in this group.

## **Finally in few words ...**

We really hope that using this system the life of the blind people become easier and make them more dependent on themselves.

## References:

1. **Reddy, Sumathi.** Why we keep losing our keys. *wsj.com*. [Online] Apr 14, 2014. [Cited: Dec 7, 2016.] <http://www.wsj.com/articles/SB10001424052702304117904579501410168111866>.
2. **Payette, Carey.** Long Range UHF RFID Item Tracking System. *hackster.io*. [Online] Sep 27, 2015. [Cited: Dec 2016, 1.] <https://www.hackster.io/careypayette/long-range-uhf-rfid-item-tracking-system-fc5372>.
3. *Design and implementation of library books search and management system using RFID Technology.* **Haiming Cheng, Ling Huang, He Xe, Yifan Hu, Xu An Wang.** Nanjing, China : IEEE, 2016. 978-1-5090-4124-4.
4. *An Indoor Localization System Based On Backscatter RFID Tag.* **Jun Wang, Yiyin Wang, Xinping Guan.** Shanghai, China : IEEE, 2016. 1558-2612.
5. **LANDT, JEREMY.** The history of RFID. *IEEE*. OCTOBER/NOVEMBER 2005.
6. What is RFID? *TechNovelgy.com*. [Online] [Cited: 11 13, 2016.] <http://www.technovelgy.com/ct/Technology-Article.asp?ArtNum=1>.
7. [book auth.] Kamran Ahsan. *RFID Components, Applications and System Integration with Healthcare Perspective*. s.l. : Deploying RFID - Challenges, Solutions, and Open Issues, 2011.
8. Radio Frequency Identification Tags - Semi Passive. / *Pacific Northwest National Laboratory*. [Online] [Cited: 11 13, 2016.] <http://availabletechnologies.pnnl.gov/technology.asp?id=90>.
9. **Firefly University.** RFID System Components. *firefly RFID solution*. [Online] [Cited: 11 13, 2016.] <http://www.fireflyrfidsolutions.com/firefly-university/what-is-rfid-copy/>.
10. How Do RFID Systems Work. *IMPINJ*. [Online] [Cited: 11 13, 2016.] <http://www.impinj.com/resources/about-rfid/how-do-rfid-systems-work/>.
11. RFID Standards. *Radio-Electronics.com*. [Online] Adrio Communications Ltd. [Cited: 11 13, 2016.] <http://www.radio-electronics.com/info/wireless/radio-frequency-identification-rfid/iso-epcglobal-iec-standards.php>.
12. **Baker, Jason.** opensource.com. [Online] February 2, 2015. [Cited: November 2, 2016.] <https://opensource.com/resources/what-raspberry-pi>.
13. **Graham Hastings, Michael Kölling.** *The Raspberry Pi Education Manual Version 1.0*. s.l. : BCS, the chartered institute of IT, 2012.
14. **Cawley, Christian.** *MakeUseOf*. [Online] February 2013. [Cited: November 2, 2016.] <http://www.makeuseof.com/tag/great-things-small-package-your-unofficial-raspberry-pi-manual/#chapter-1>.
15. *Raspberry Pi Technology: A Review.* **Chaudhari, Harshada.** 3, India : International Journal of Innovative and Emerging Research in Engineering, 2015, Vol. 2.
16. *Raspberry Pi Foundation*. [Online] [Cited: November 2, 2016.] <https://www.raspberrypi.org/>.
17. *Raspberry Pi Getting Started Guide Vsn 1.0*. s.l. : RS Components, 3/2012.

18. **Elsea, Peter.** Microphones. *artsites.ucsc.edu*. [Online] 1996. [Cited: November 11, 2016.] [http://artsites.ucsc.edu/ems/music/tech\\_background/te-20/teces\\_20.html](http://artsites.ucsc.edu/ems/music/tech_background/te-20/teces_20.html).
19. **Christensson, Per.** Speakers Definition. *techterms.com*. [Online] February 27, 2010. [Cited: November 11, 2016.] <http://techterms.com/definition/speakers>.
20. Wi-Fi Protocol: Networking, Frame Formats, Security, Attributes. *www.engineersgarage.com*. [Online] [Cited: April 24, 2017.] <https://www.engineersgarage.com/articles/what-is-wifi-technology>.
21. Voice recognition. *www.computerhope.com*. [Online] April 26, 2017. [Cited: May 5, 2017.] <https://www.computerhope.com/jargon/v/voicereco.htm>.
22. SQL (Structured Query Language). *www.techtarget.com*. [Online] September 2016. [Cited: April 30, 2017.] <http://searchsqlserver.techtarget.com/definition/SQL>.
23. RFID Access Control ID Card Keypad Reader 125KHz Wiegand 26 Security. *www.ebay.com*. [Online] [Cited: April 26, 2017.] <http://www.ebay.com/itm/RFID-Access-Control-ID-Card-Keypad-Reader-125KHz-Wiegand-26-Security-/262017939247>.
24. Read Only Contactless Identification Device . *www.sunrom.com*. [Online] [Cited: April 29, 2017.] [www.sunrom.com/get/522300](http://www.sunrom.com/get/522300).
25. Cisco-Linksys AE1000 High-Performance Wireless-N Adapter. *www.amazon.com*. [Online] [Cited: April 29, 2017.] <https://www.amazon.com/Cisco-Linksys-AE1000-High-Performance-Wireless-N-Adapter/dp/B003B20F5E>.
26. *Impedance Matching of Tag Antenna to Maximize RFID Read Ranges*. **M S Yeoman, M A O'Neill**. United Kingdom : Excerpt from the Proceedings of the 2014 COMSOL Conference in Cambridge, 2014.
27. *RFID Privacy & Security Issues*. **Muir, Brent**. Melbourne Area, Australia, Australia : s.n., Feb 25, 2014.
28. **Ha, Vipul Chawla and Dong Sam.** AN OVERVIEW OF PASSIVE RFID. *IEEE Applications & Practice*. 2007.
29. *Localization Systems using Passive UHF RFID*. **Jae Sung Choi, Hyun Lee, Ramez Elmasri, Daniel W. Engels**. Arlington, TX USA : IEEE computer society, 2009.

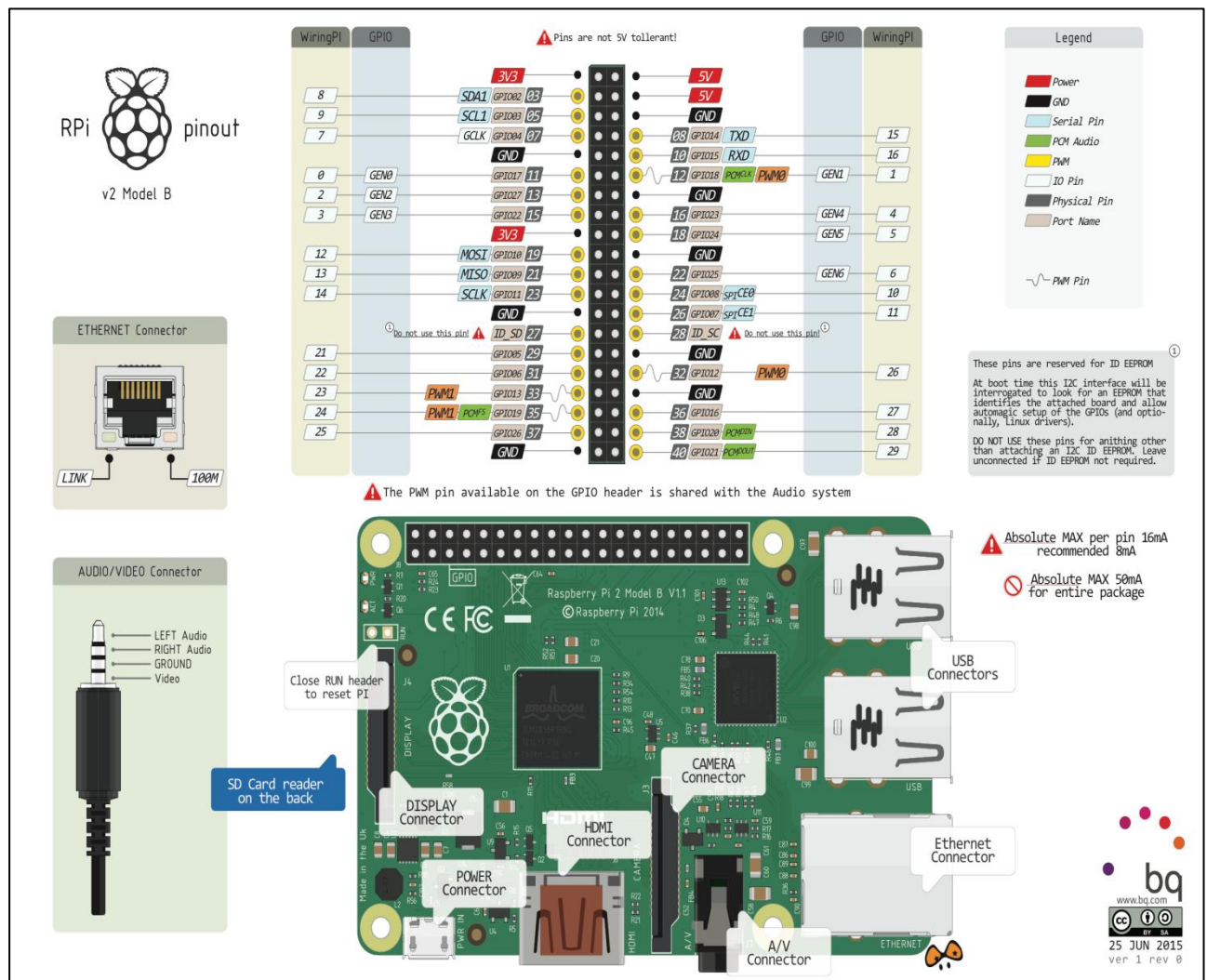
# Appendix A

---

Raspberry Pi 2

RFID Access Control ID Card Reader 125 KHz Wiegand 26

# Raspberry Pi v2 Mod B Pin specification



## Weatherproof Security Door Black Wiegand 26 RFID ID Card Reader 125KHz EM4100

### Features:

- Frequency: 125Khz
- Standard wiegand26 bit output
- Easy to install on Metal Door Frame or Mullion
- Reverse Polarity Protection
- Indoor / Outdoor Operation
- External LED Control
- External Buzzer Control
- Solid Epoxy Potted
- Waterproof IP65



### Specifications:

Model	KR100E	KR100M
Read Range	Up to 10CM	Up to 5CM
Reading Time (Card)	≤300ms	
Power / Current	DC 6-14V / Max.70mA	
Input Port	2ea (External LED Control, External Buzzer Control)	
Output Format	26bit Wiegand (default)	
LED Indicator	2 Color LED Indicators(Red and Green)	
Beeper	Yes	
Operating Temperature	-20° to +65°C	
Operating Humidity	10% to 90% relative humidity non-condensing	
Color	Black	
Material	ABS+PC with texture	
Dimension(W x H x T)mm	116*75*17.3	
Weight	120g	
Index of Protection	IP65	

# Appendix B

---

## Codes

initial version of the project

server code

CardReader that was called by server code

Client code

CardReader that was called by client code

The code of the initial version of the project:

```
#!/usr/bin/python

import speech_recognition as sr
import sys
import pyttsx
import RPi.GPIO as GPIO
import MFRC522
import signal

#this is the reader function
def reading():
    continue_reading = True
    global num
    num = 00000000000

    def end_read(signal,frame):
        global continue_reading
        print "Ctrl+C captured, ending read."
        continue_reading = False
        GPIO.cleanup()

    # Hook the SIGINT
    signal.signal(signal.SIGINT, end_read)

    # Create an object of the class MFRC522
    MIFAREReader = MFRC522.MFRC522()

    # Welcome message
    print "Welcome to the MFRC522 data read example"
    print "Press Ctrl-C to stop."

    # This loop keeps checking for chips. If one is near it will get the
    UID and authenticate
    while continue_reading:

        # Scan for cards
        (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
```



```

# If a card is found
if status == MIFAREReader.MI_OK:
    print "Card detected"

# Get the UID of the card
(status,uid) = MIFAREReader.MFRC522_Anticoll()

# If we have the UID, continue
if status == MIFAREReader.MI_OK:

    # Print UID
    print "Card read UID:
"+str(uid[0])+",""+str(uid[1])+",""+str(uid[2])+",""+str(uid[3])

    num = str(uid[0])+str(uid[1])+str(uid[2])+str(uid[3])
    continue_reading = False
return num

engine = pyttsx.init()
engine.setProperty('rate',70)

# obtain audio from the microphone
def take_voice():
    global r
    global audio
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source) # listen for 1 second to
calibrate the energy threshold for ambient noise levels
        engine.say(" Say what you need")
        engine.runAndWait()
        print("Say what you need :")
        audio = r.listen(source)
    return r,audio

```

```

take_voice()

# recognize speech using Google Speech Recognition
try:
    print("Google Speech Recognition thinks you said " +
r.recognize_google(audio))
    word = r.recognize_google(audio)
    print(word)
    # call reading function
    reading()
except sr.UnknownValueError:
    print('Google can not understand audio')
    engine = pyttsx.init()
    engine.setProperty('rate',70)
    #engine.say("I can not understand audio")
    engine.say("I can not understand audio   Repeat again")
    engine.runAndWait()
    take_voice()
    print("Google Speech Recognition thinks you said " +
r.recognize_google(audio))
    word = r.recognize_google(audio)
    # call reading function
    reading()

print num

def find_book():
    global x
    x = True
    if (word == 'book' or word == 'book book' or word == 'find my book' or
word == 'find my puppy'):
        if num == '1556415187' :
            engine = pyttsx.init()
            engine.setProperty('rate',70)
            engine.say("I find your book")
            engine.runAndWait()

```

```

        x = False
    elif num != '1556415187':
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say("I can not find your book move the reader")
        engine.runAndWait()
        x = True
    return x

find_book()
if x== True :
    reading()
    find_book()

def find_key():
    global y
    y = True
    if (word == 'monkey' or word == 'mikey' or word == 'find my key' or word
    == 'find monkey' or word == 'find party' or word == 'find my teeth') :
        print(word)
        if num == '16616334126' :
            engine = pyttsx.init()
            engine.setProperty('rate',70)
            engine.say("I find your key")
            engine.runAndWait()
            y = False
        elif num != '16616334126':
            engine = pyttsx.init()
            engine.setProperty('rate',70)
            engine.say("I can not find your key move the reader")
            engine.runAndWait()
            y = True
    return y
find_key()
if y == True:
    reading()
    find_key()

```

Server Code (Mobile reader)

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

```

import MySQLdb
import re , csv, sys
import time
import os
import speech_recognition as sr
import sys
import pytttx
import RPi.GPIO as GPIO
import signal
import RPIO
import socket
import atexit
from cardReader import CardReader

my_id = 0
my_place = 0
add_place = 0

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

FT = open('search.txt','w')          # open text file to write reading id
FR = open('result_reader2.txt','w')  # open text file to write reading result from reader2
FA = open('add_tag.txt','w')         # open text file to write all possible choice of the word
FP = open('place_no.txt','w')        # open text file to write tag place for the added tag

#####
# function to initiate speech synthesis
def start():
    engine = pytttx.init()
    engine.setProperty('rate',80)
    engine.say("press one to search two to add ")
    engine.runAndWait()
#####

#####
# function to create the database
def build_db():
    try:
        connection = MySQLdb.connect (host = "localhost",
                                       user = "taguser",
                                       passwd = "passwordhiba",
                                       db ="rfid_tags" )
    except mc.Error as e:
        print("Error %d: %s" % (e.args[0], e.args[1]))
        sys.exit(1)

```

```

cursor = connection.cursor()

cursor.execute ("DROP TABLE IF EXISTS tags")

sql_command = """
CREATE TABLE tags (
item INT NOT NULL PRIMARY KEY AUTO_INCREMENT ,
tag_type CHAR(20),
tag_id CHAR(20),
place INT);"""

cursor.execute(sql_command)

sql_command = """
INSERT INTO tags( tag_type, tag_id, place) VALUES
('find my book','994369' , 1),
('book', '994369', 1),
('book book', '994369', 1),
('my book', '994369', 1),
('find Michael','994369', 1),
('find my puppy', '994369', 1),
('find my phone','410450' , 2),
('find my iphone', '410450', 2),
('phone', '410450', 2),
('find my remote', '988216', 2),
('120 mos', '988216', 2),
('find my vitamix', '988216', 2),
('remote', '988216', 2),
('find my key', '994247', 1),
('find my keys','994247' , 1),
('find monkey', '994247', 1),
('find my teeth', '994247', 1),
('find molly', '994247', 1);"""

cursor.execute(sql_command)
print(sql_command)
connection.commit()
cursor.close()
connection.close()
#####

#####
# function to search in database
def searchINdb(word):
    global my_id
    global my_place
    try:
        connection = MySQLdb.connect (host = "localhost",

```

```

        user = "taguser",
        passwd = "passwordhiba",
        db = "rfid_tags" )
except mc.Error as e:
    print("Error %d: %s" % (e.args[0], e.args[1]))
    sys.exit(1)

cursor = connection.cursor()
sql = "SELECT * FROM tags WHERE tag_type = '%s' " % (word)
try:
    cursor.execute(sql)
    # Fetch all the rows in a list of lists.
    results = cursor.fetchall()
    for row in results:
        item= row[0]
        tag_type= row[1]
        tag_id = row[2]
        my_id = row[2]
        place = row[3]
        my_place = row[3]
        print "item=%s,tag_type=%s,tag_id=%s, place=%s" % (item, tag_type,
tag_id,place)
        FT.write(my_id)
        FT.flush()
        return my_id,my_place
    for row in not results :
        print('unable to fetch data')
except:
    print "Error: unable to fetch data"
    speech = "press button two to add new object"
    engine = pyttsx.init()
    engine.setProperty('rate',80)
    engine.say(speech)
    engine.runAndWait()
    speech = "it is not in database"
    engine = pyttsx.init()
    engine.setProperty('rate',80)
    engine.say(speech)
    engine.runAndWait()
    switches()

    connection.commit()
#####

#####
# function to obtain audio from the microphone
def take_voice():
    global r
    global speech
    global audio
    global word

```

```

word = 'null'
r = sr.Recognizer()
with sr.Microphone() as source:
    r.adjust_for_ambient_noise(source) # listen for 1 second to calibrate the energy threshold for
ambient noise levels
    engine = pytttsx.init()
    engine.setProperty('rate',80)
    engine.say(speech)
    print(speech)
    engine.runAndWait()
    audio = r.listen(source)
return r,audio
#####

#####
# function to begin reading process
def scanning_tag():
    engine = pytttsx.init()
    engine.setProperty('rate',70)
    engine.say("start scanning")
    engine.runAndWait()

global end
end = ""
global Tag_ID
BIT_TRANSMISSION_TIME = 0.002          #From wiegand specification
FRAMESIZE = 26                        #Supposed size of received frame
FRAMETIME = FRAMESIZE * BIT_TRANSMISSION_TIME #Theoric time necessary to
transfer a frame
ALLOWANCE = 10                        #Auhtorized allowance for the transmission time in
percent
TIMEOUT = FRAMETIME*(1+ALLOWANCE/100)    #Real time allowed for the
transmission
readersList = [
    CardReader("reader", 23, 24, TIMEOUT)]    #Creating readers

def closeProgram(signal, frame):
    """ Close fonction """
    print("\nResseting GPIO...", end)
    RPIO.cleanup()          #Reset every channel that has been set up by this
program, and unexport interrupt gpio interfaces
    print(" ok")
    print("exiting")
    sys.exit(0)

#Starting readers
readersCount = 0
for reader in readersList:
    print("Initializing reader " + str(readersCount) + "...", end )
    reader.registerReader()
    print(" Done !")
    readersCount += 1

```

```

#Ready message
print("Ready to go !")
RPIO.wait_for_interrupts()
#####

#####
# function to send reading order to the second reader
def send_order():
    x = True
    FT = open('search.txt','r')      # open text file to write reading id
    FR = open('result_reader2.txt','w')
    s = socket.socket()              # Create a socket object
    host = "                          # Get local machine name
    port = 9985                       # Reserve a port for your service.
    s.bind((host, port))              # Bind to the port
    s.listen(5)                       # Now wait for client connection.
    while x:
        c, addr = s.accept()          # Establish connection with client.
        print 'Got connection from', addr
        print "Sending..."
        l = FT.read(1024)
        print l
        while (l):
            print "Sending..."
            c.send(l)
            l = FT.read(1024)
        FT.flush()
        print "Done Sending"
        c.shutdown(socket.SHUT_WR)
        line = c.recv(1024)
        while(line):
            FR.write(line)
            line = c.recv(1024)
            FR.flush()
        print "Done Receiving"
        x = False
    c.close()                        # Close the connection
#####

#####
# function for searching and adding process
def switches():
    while True:
        global add_place
        global speech
        input_state = GPIO.input(17)    #pin 11 GND 9
        if input_state == False:
            print('Button 1 Pressed')

```



```

time.sleep(0.2)
speech = "say what you need"
take_voice()
#####
# recognize speech using Google Speech Recognition
try:
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
    word = r.recognize_google(audio)
    searchINdb(word)
    print(my_id)

except sr.UnknownValueError:
    print('Google can not understand audio')
    engine = pyttsx.init()
    engine.setProperty('rate',70)
    engine.say("I can not understand audio Repeat again")
    engine.runAndWait()
    take_voice()
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
    word = r.recognize_google(audio)
    searchINdb(word)
#####
if (my_place==1) :
    scanning_tag()          # call scanning_tag function
elif (my_place == 2):
    send_order()
    FR = open('result_reader2.txt','r')
    result_reader2 = FR.read()
    if (result_reader2 == 'exist') :
        speech = " it is in place two"
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say(speech)
        print(speech)
        engine.runAndWait()
    else:
        speech = "I can not find it in place two"
        engine = pyttsx.init()
        engine.setProperty('rate',70)
        engine.say(speech)
        engine.runAndWait()
break

input_state2 = GPIO.input(18)    #pin 12 GND 14
if input_state2 == False:
    print('Button 2 Pressed')
    time.sleep(0.2)
    speech = "say what to add"
    take_voice()
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
    word = r.recognize_google(audio)
    print(word)

```

```

FA.write(word)
FA.write('\n')
#####
# recognize speech using Google Speech Recognition
try:
    for no_saying in range (1):
        speech = "repeat again"
        take_voice()
        print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
        word = r.recognize_google(audio)
        print(word)
        FA.write(word)
        FA.write('\n')
        FA.flush()
    FT.write('%d' % 1000)
    FT.flush()
    scanning_tag()
    speech = "say the place number"
    take_voice()
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
    word = r.recognize_google(audio)
    print(word)
    if( word == 'place one' or word == 'this one' or word == 'list one' or word == 'fun' or
word == 'placement' or word == 'one'):
        add_place = 1
    else :
        add_place = 2

# To add a new tag into database
except sr.UnknownValueError:
    print('Google can not understand audio')
    engine = pyttsx.init()
    engine.setProperty('rate',70)
    engine.say("I can not understand audio Repeat again")
    engine.runAndWait()
    take_voice()
    print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
    word = r.recognize_google(audio)
try:
    connection = MySQLdb.connect (host = "localhost",
                                user = "taguser",
                                passwd = "passwordhiba",
                                db = "rfid_tags" )
except mc.Error as e:
    print("Error %d: %s" % (e.args[0], e.args[1]))
    sys.exit(1)

cursor = connection.cursor()
ID = open('/home/pi/getTag_ID.txt', 'r') # read the id of the added tag
object = open('add_tag.txt','r')      # read all possible choice of the word

for i in range (2):

```

```

ID.seek(0)
file_content1 = ID.readline().rstrip('\n')
print file_content1
file_content2 = object.readline().rstrip('\n')
print file_content2
query = "INSERT INTO tags (tag_type,tag_id, place) VALUES (%s,%s,%s)"
cursor.execute(query, (file_content2,file_content1,add_place))
connection.commit()

break
#####

start()          # call start function
switches()       # call switches function

```

CardReader code that was called by server code

```

import RPIO
import threading
import sys
import signal
import pyttbx

f = open("getTag_ID.txt",'w')
FT = open('search.txt','r')

global end
global my_id
end = " "
tag_id = 0
count_read = 0

class CardReader(object):

    """Class representing a reader. One object should be instantiated for each physical
    reader"""

    def __init__(self, name, GPIO_0, GPIO_1, TIMEOUT):
        #Pins used to receive 0s and 1s
        self.name = name
        self.GPIO_0 = GPIO_0
        self.GPIO_1 = GPIO_1

        self.tag = "" #The buffer used to store the RFID Tag
        self.TIMEOUT = TIMEOUT #Real time allowed for the transmission
        return super(CardReader,self).__init__()

    def addBitToTag(self, gpio_id, val):
        #Beginning of a new frame, we start the timer
        if self.tag == "":
            self.t = threading.Timer(self.TIMEOUT, self.processTag)
            self.t.start()

        #We check wether we received a 0 or a 1
        if gpio_id == self.GPIO_0:
            self.tag += "0"
        elif gpio_id == self.GPIO_1:
            self.tag += "1"

    def registerReader(self, edge = 'falling', pull_up_down=RPIO.PUD_UP):
        RPIO.setup(self.GPIO_0, RPIO.IN)
        RPIO.setup(self.GPIO_1, RPIO.IN)
        RPIO.add_interrupt_callback(self.GPIO_0, self.addBitToTag, edge = edge,
pull_up_down = pull_up_down)
        RPIO.add_interrupt_callback(self.GPIO_1, self.addBitToTag, edge = edge,
pull_up_down = pull_up_down)

        #Initializing timer

```

```

        self.t = threading.Timer(0.1, self.processTag)
        self.t.start()

    def removeReader(self):
        RPIO.del_interrupt_callback(self.GPIO_0)
        RPIO.del_interrupt_callback(self.GPIO_1)

    #Method triggered after Timer tick that prints out the tag
    def processTag(self):
        global count_read
        if self.tag == "":
            return
        elif len(self.tag) < 10:
            print "[" + self.name + "] Frame of length (" + str(len(self.tag)) +
"): " + self.tag + " DROPPED")
        elif self.verifyParity(self.tag):
            print "[" + self.name + "] Frame of length (" + str(len(self.tag)) +
"): " + self.tag + " (" + str(CardReader.binaryToInt(self.tag)) + ") OK KOI" )
            count_read += 1
            tag_id = str(CardReader.binaryToInt(self.tag))
            f.write(tag_id)
            f.flush()
            my_id = FT.readline().rstrip()
            FT.seek(0) # to read from the first line
            print (my_id,tag_id,count_read)
            if(tag_id == my_id):
                print('I find the wanted object')
                engine = pyttsx.init()
                engine.setProperty('rate',80)
                engine.say('I find it ')
                engine.runAndWait()
                RPIO.stop_waiting_for_interrupts()
            elif(my_id == '1000'):
                print('I add the tag')
                engine = pyttsx.init()
                engine.setProperty('rate',80)
                engine.say('I add it ')
                engine.runAndWait()
                RPIO.stop_waiting_for_interrupts()
            elif(tag_id != my_id and count_read == 3):
                print('I can not find it, out of range')
                engine = pyttsx.init()
                engine.setProperty('rate',80)
                engine.say('I can not find it ')
                engine.runAndWait()
                RPIO.stop_waiting_for_interrupts()

        self.tag = ""

```

```

def verifyParity(self, binary_string):
    first_part = binary_string[0:13]
    second_part = binary_string[13:]
    parts = [first_part, second_part]
    bitsTo1 = [0, 0]
    index = 0

    for part in parts:
        bitsTo1[index] = part.count('1')
        index += 1

    if bitsTo1[0] % 2 != 0 or bitsTo1[1] % 2 != 1:
        print "[" + self.name + "] Frame of length (" + str(len(self.tag)) + "):
" + self.tag + " (" + str(CardReader.binaryToInt(self.tag)) + ") - PARITY CHECK FAILED")
        return False
    return True

#Method to convert the RFID binary value into a readable integer
@staticmethod
def binaryToInt(binary_string):
    print(binary_string)
    binary_string = binary_string[1:-1] #Removing the first and last bit (Non-data
bits)

    print(binary_string)
    result = int(binary_string, 2)
    return result

```

```

import signal
import socket
import RPIO
import sys
from cardReader import CardReader

f = open('result.txt', 'w') # to write result on it


s = socket.socket()      # Create a socket object
host = '192.168.1.6'     # Get local machine name
port = 9985              # Reserve a port for your service.

s.connect((host, port))
fi = open('reciving_id.txt','wb')
print 'reciving...'
li = s.recv(1024)
while (li):
    print 'reciving...'
    fi.write(li)
    li = s.recv(1024)
    print li
fi.close()
print "Done reciving"
print s.recv(1024)
order_file = open('reciving_id.txt','r') # to get the order
order = order_file.readline().rstrip("\n")
print order


BIT_TRANSMISSION_TIME = 0.002 #From wiegand specification
FRAMESIZE = 26 #Supposed size of received frame
FRAMETIME = FRAMESIZE * BIT_TRANSMISSION_TIME #Theoric time necessary to
transfer a frame
ALLOWANCE = 10 #Auhtorized allowance for the transmission time in percent
TIMEOUT = FRAMETIME*(1+ALLOWANCE/100) #Real time allowed for the transmission


#Creating readers
readersList = [
CardReader("reader", 23, 24, TIMEOUT),
CardReader("arduino", 8, 7, TIMEOUT)
]


def closeProgram(signal, frame):
    """ Close fonction"""
    print("\nRessetting GPIO...")
    RPIO.cleanup() #Reset every channel that has been set up by this program, and
unexport interrupt gpio interfaces
    print(" ok")
    print("exiting")
    sys.exit(0)

```

```

signal.signal(signal.SIGINT, closeProgram)

#Starting readers
readersCount = 1
for reader in readersList:
    print("Initializing reader " + str(readersCount) + "...")
    reader.registerReader()
    print(" Done !")
    readersCount += 1

#Ready message
print("Ready to go !")

RPIO.wait_for_interrupts()

f = open('result.txt','rb')
print 'Sending...'
l = f.read(1024)
while (l):
    print 'Sending...'
    s.send(l)
    l = f.read(1024)
f.close()
print "Done Sending"
s.shutdown(socket.SHUT_WR)
print s.recv(1024)
s.close()          # Close the socket when done

```

CardReader code that was called by client code



```

import RPIO
import threading

f = open('reciving_id.txt','r')
fr = open('result.txt', 'w')
tag_id = 0
count_read = 0

class CardReader(object):
    """Class representing a reader. One object should be instantiated for each physical
    reader"""

    def __init__(self, name, GPIO_0, GPIO_1, TIMEOUT):
        #Pins used to receive 0s and 1s
        self.name = name
        self.GPIO_0 = GPIO_0
        self.GPIO_1 = GPIO_1

        self.tag = "" #The buffer used to store the RFID Tag
        self.TIMEOUT = TIMEOUT #Real time allowed for the transmission
        return super(CardReader , self).__init__()

    def addBitToTag(self, gpio_id, val):
        #Beginning of a new frame, we start the timer
        if self.tag == "":
            self.t = threading.Timer(self.TIMEOUT, self.processTag)
            self.t.start()

        #We check wether we received a 0 or a 1
        if gpio_id == self.GPIO_0:
            self.tag += "0"
        elif gpio_id == self.GPIO_1:
            self.tag += "1"

    def registerReader(self, edge = 'falling', pull_up_down=RPIO.PUD_UP):
        RPIO.setup(self.GPIO_0, RPIO.IN)
        RPIO.setup(self.GPIO_1, RPIO.IN)
        RPIO.add_interrupt_callback(self.GPIO_0, self.addBitToTag, edge = edge,
pull_up_down = pull_up_down)
        RPIO.add_interrupt_callback(self.GPIO_1, self.addBitToTag, edge = edge,
pull_up_down = pull_up_down)

        #Initializing timer
        self.t = threading.Timer(0.1, self.processTag)
        self.t.start()

    def removeReader(self):
        RPIO.del_interrupt_callback(self.GPIO_0)
        RPIO.del_interrupt_callback(self.GPIO_1)

    #Method triggered after Timer tick that prints out the tag
    def processTag(self):

```

```

        global count_read
        if self.tag == "":
            return
        elif len(self.tag) < 10:
            print "[" + self.name + "] Frame of length (" + str(len(self.tag)) +
"): " + self.tag + " DROPPED")
        elif self.verifyParity(self.tag):
            print "[" + self.name + "] Frame of length (" + str(len(self.tag)) +
" + self.tag + " (" + str(CardReader.binaryToInt(self.tag)) + ") OK KOI" )
            tag_id = str(CardReader.binaryToInt(self.tag))
            my_id = f.read()
            f.seek(0)
            count_read += 1
            print(count_read,tag_id,my_id)
            if(tag_id == my_id ):
                print("i found it YAY")
                fr.write('exist')
                fr.flush()
                RPIO.stop_waiting_for_interrupts()
        elif(tag_id != my_id and count_read == 3):
            print('not exist')
            print('-----Out of the range-----')
            fr.write('not exist')
            fr.flush()
            RPIO.stop_waiting_for_interrupts()

        self.tag = ""

def verifyParity(self, binary_string):
    first_part = binary_string[0:13]
    second_part = binary_string[13:]
    parts = [first_part, second_part]
    bitsTo1 = [0, 0]
    index = 0

    for part in parts:
        bitsTo1[index] = part.count('1')
        index += 1

    if bitsTo1[0] % 2 != 0 or bitsTo1[1] % 2 != 1:
        print "[" + self.name + "] Frame of length (" + str(len(self.tag)) +
" + self.tag + " (" + str(CardReader.binaryToInt(self.tag)) + ") - PARITY CHECK FAILED")
        return False
    return True

#Method to convert the RFID binary value into a readable integer
@staticmethod
def binaryToInt(binary_string):
    print(binary_string)
    binary_string = binary_string[1:-1] #Removing the first and last bit (Non-data
bits)
    print(binary_string)

```

```
result = int(binary_string, 2)  
return result
```