2/6/2014

VOICE ASSISTANT FOR VISUALLY IMPAIRED

STAFFORDSHIRE UNIVERSITY

kyritsis@teilar.gr | Kyritsis Panagiotis
k_kokkinos@teilar.gr | Kokkinos Konstantinos

Table of Contents

1 Introduction 5
1.1 Introduction 5
1.2 Scope of the thesis 6
1.3 Thesis structure 6
2 Background 7
2.1 Sensor Networks 8
2.1.1 RFID Sensing 9
2.1.2 RFID Classification 10
2.1.3 RFID Components 13
2.2 WiFi 13
2.3 Bluetooth 15
3 Problem Statement and Design Considerations 18
3.1 Problem Statement 18
3.2 Design Considerations 18
4 Related Work 19
4.1 Introduction 19
4.2 RoboCard 20
4.3 ShopTalk 22
4.4 GroZi 24
4.5 iCare 25
4.6 Trinetra 26
4.7 IBM Patent 27
4.8 Comparison of Solutions 28
5 Hardware and Architecture 29
5.1 Hardware Components 29
5.2 Architecture 29
5.3 Components Communication 30
6 Methodology 32
6.1 Requirements 32
6.2 Implementation Categories 33
6.3 Interpretation 37
7 Results 39
7.1 Experiment Methodology 39
7.2 Experimental Results 39
7.3 Software Evaluation 43

8 Future Work 44
9 Conclusions 45
9.1 Hardware 45
9.2 Software 45
10 References 46
11 Appendices 49

List of Figures

Figure 2.1: Ubiquitous Sensor Networks (Figure adopted from [12])....8
Figure 2.2: RFID connects digital and physical worlds (Figure adopted from [16]) 9
Figure 2.3: Antenna regions (Figure adopted from [21]) 11
Figure 2.4: Passive RFID tags 11
Figure 2.5: Active RFID tags....12
Figure 2.6: Piconet....15
Figure 2.7: Scatternet....16
Figure 4.1: RoboCard (adopted from [30])....20
Figure 4.2: Belkin keypad on handle adopted from [31]....21
Figure 4.3: ShopTalk (adopted from [32])....23
Figure 4.4: MSI barcode on shelf scanned (adopted from [32])....23
Figure 4.5: MoZi Box (adopted from [33]) 24
Figure 4.6: GroZi hand glove (adopted from [33])....24
Figure 4.7: iCare Framework (adopted from [35]) 25
Figure 4.8: Trinetra's hardware components (adopted from [36]) 26
Figure 4.9: Trinetra's software components (adopted from [36]) 27
Figure 5.1: Components Communication Model 31
Figure 6.1: RFID tags distribution in the environment....32
Figure 6.2:Methoology Summery....37
Figure 6.3: Methodology flow chart....38

List of Tables

Table 2.1: Advantages and disadvantages of passive RFID tags....12
Table 2.2: Advantages and disadvantages of active RFID tags....12
Table 2.3: Data Rates Supported by IEEE 802.11 Standards....14
Table 2.4: Bluetooth technical specifications....17
Table 4.1: Comparison of solutions....28
Table 6.1: Methodology methods and actions summary....36
Table 7.1: Navigate from point A to point B in seconds....40
Table 7.2: Find and scan the barcode on shelf in seconds....41
Table 7.3: Find and scan the barcode on product....42

Abstract

According to the World Blind Union (WBU), there are estimated about 285 million people worldwide who are totally blind or visually impaired. Many of them have received expert training in order to improve their skills that allow them to orientate and navigate in indoor or outdoor environments. However, even the very well trained, blind people face huge problems when they have to learn new routes, especially in environments which they are not familiar with, or in environments that have been changed semantically. The researchers have tried in the recent past, to implement navigation and localization systems which aim to improve the everyday lives of visually impaired people, but only a few of them are available in the marketplace, and most of them use expensive and heavy equipment. Partially, this is the reason that these systems have adopted by a very low rate.

This thesis approaches the blind people navigation problem in indoor environments from a different view and proposes a system which exploits the technical characteristics of cheap and common devices. With this solution, a visually impaired person can have direct and interactive communication with the environment surrounding him, by using natural language orders. Also, he can access information that in other cases he could only access with the assistance of a sighted person. Furthermore, the system does not require the installation of complex sensors in the environment and it is based on cheap equipment for both visually impaired and indoor environment. Although the solution can be implemented in any regularly structured indoor environment, the deployment of this project performed in a supermarket environment.

In the following pages, it is presented a low cost and easily deployable system, which promotes and in parallel accesses navigation for the visually impaired people through a RFID transmitter/receiver technology whereas, nodes transmit relevant information about products and they communicate uni-directionally with a receiver device attached on a cane of the impaired person. Furthermore, the system combines the above methodology with a recognition mechanism based on a barcode-exploration technique from the shelved products in order to overcome the storing restrictions of the first method. The identification of barcodes is achieved through a Windows 8 and above smart phone with an embedded camera. Experimental results on the field verified our technique in practice. The end-product can also be implemented in other smart-device operating systems that support RFID readers.

Summarizing, this thesis adopts four hypotheses:

i. The implementation does not require complex sensors to be embedded in the environment
ii. The indoor environment is regularly structured
iii. The system can identify data from the environment and supply information in natural language
iv. The system can accept orders in natural language.

1 Introduction

1.1 Introduction

The navigation and mobility of a blind person without the assistance of a sighted one is a matter of personal independence. This feel of independence has a huge impact in blind's mental health. For a blind person the dependence of his mobility on a sighted one can result in reduced mobility and that has been noted to cause depression [1]. A research that took place in Great Britain [2] published that only 51% of visually impaired under the age of 60 had left their home on foot without the company of a sighted person. Furthermore, many of older people, who have vision loss, acts like prisoners in their homes, because they do not feel confidence and independent [3].
On the other hand, there are some environments which are difficult for visually impaired people to be navigated, because of their large size or lack of structure. There are also environments which are regularly structured, but the tasks that must be performed in them are very complex. For example, in a typical supermarket there are about 45.000 products and a median store size is 4.529 square meters [4]. In such an environment, a visually impaired man needs the assistance of a family member or a store employee to complete his shopping procedure.
The rapid evolution of new technologies can aid, among others, the independent shopping for blinds. A currently created concept, called Internet of Things (IoT), supports that everything around us have the ability to communicate with other objects and connect to a network. Convenient technologies, such as Radio Frequency Identification (RFID) can ensure that communication and information systems will be embedded in the environment around us in an invisible way.
In general, the term "Internet-of-Things" refers to:

- the resulting global network which interconnects smart objects using extended Internet technologies
- the set of supporting technologies which facilitate the realization of such a vision (e.g., RFIDs, sensor/actuators, machine-to-machine communication devices, etc.) and
- the number of applications and services that use these technologies to create new business and market opportunities [5, 6].

Taking under consideration the problems that visually impaired people face in their daily lives and the deployment of the IoT concept, this thesis presents a Windows phone application, which exploits the wide usage of RFID tags and QR codes and helps the shopping procedure for an impaired man.

1.2 Scope of the thesis

The objective of this project is to offer a Windows phone application, which will be able to supply independent shopping to visually impaired people, without the need of currying any other equipment except of a mobile phone and a cane. The solution is based on cheap equipment for both visually impaired and stores. The implementation of our project is performed in a supermarket environment, since it is regularly structured. But it can be performed in any regularly structured indoor environment.

1.3 Thesis structure

The following pages are structured as follows. Section 2 notes the background knowledge that was used to complete the project. Our problem statement and some design considerations are presented in Section 3. In Section 4 we discuss the State of the art in shopping systems for visually impaired people. Section 5 displays the hardware component and the architecture of the system and Section 6 analyzes the methodology of the solution. In Section 7 the results of the testing procedure are presented, followed by Section 8 where future work is highlighted. Finally, in Section 9 there are some conclusions noted.

2 Background

The increasing number of mobile devices in conjunction with the growing demand for location identification solutions that recognize signals from the environment have raised the attention of many researchers and mobile device enterprises in this area [7]. Nevertheless, most of the invented prototypes are focused on the recognition of environment situation and set aside the usableness of the solution. This section discusses the theoretical basis of the technologies that have been used and are applicable to this project.

2.1 Sensor Networks

The following generation of sensing and processing procedures will set aside the traditional desktop. The Internet-of-Things (IoT) [8] prospect leads to an era where every item can be connected in the network and exchange information. RFID and sensor network technologies encourage this project, by supporting invisibility of communication that can be embedded in the environment. Under this consideration everyday products (such as drinks, food, electric devices, health monitoring equipment, etc.) will be able to communicate with one another [9, 10]. This triggers a new internet that allows object-to-object communication rather than the old machine-to-machine kind [11].
With the aid of the increasing use of Internet wireless technologies such as WiFi, WiMax and LTE, the transportation towards **U**biquitous **S**ensor **N**etworks (USN) (Figure 2.1) will be really smooth. Some of the very first steps towards USN have already been accomplished they are obvious today [12]. Nevertheless, the IoT prospect requires:

- The implementation of a computing standard that will be able to connect objects and embed intelligence into the environment [13]
- The invisibility of sensors to the users while on the other hand communicated with every item in their range.

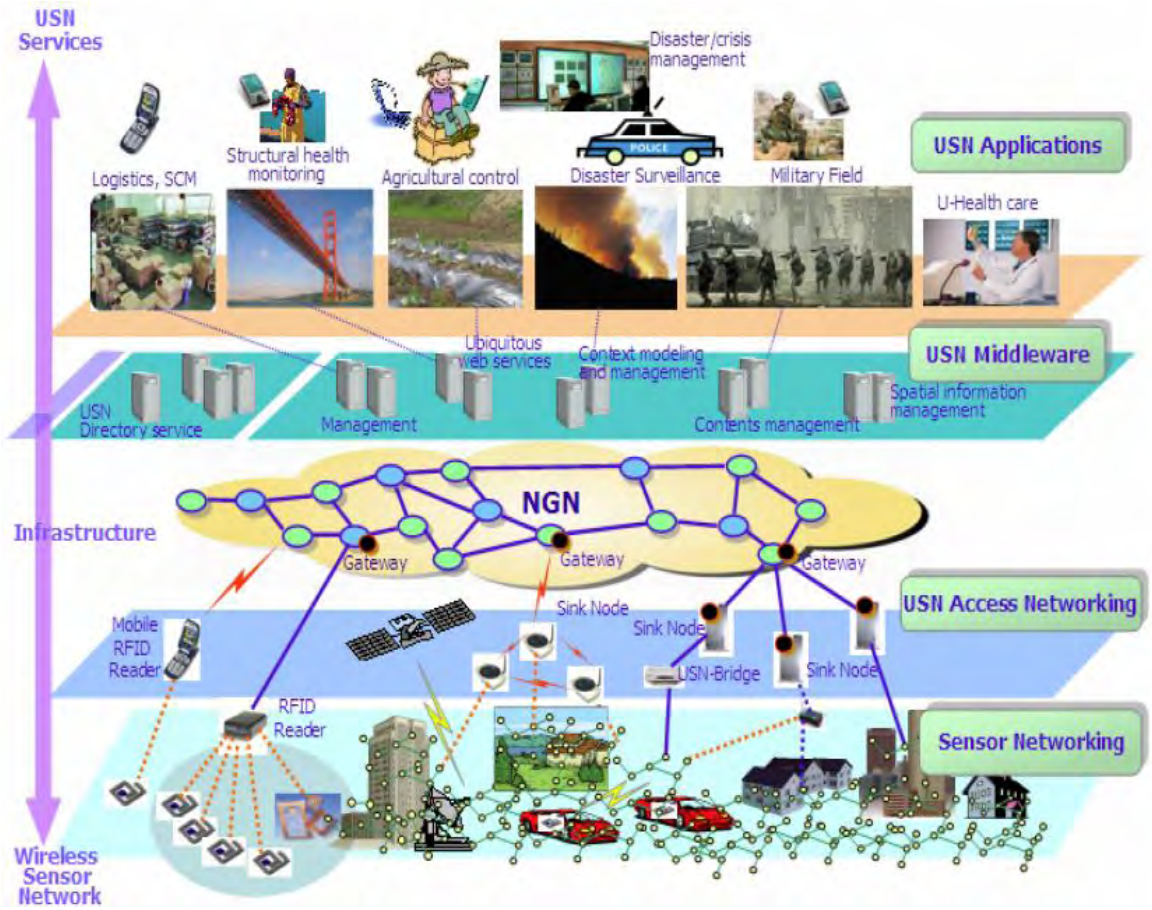

Figure 2.1: Ubiquitous Sensor Networks (Figure adopted from [12])

2.1.1 RFID Sensing

The advantages of RFID systems make the researchers believe that in near future, the devices which include sensors in embedded form will overwhelm the market. These devices will be able to gather and send information about people, weather, available stock, etc. that refers to identification, sensing and information processing. In 2011, IDTechEx reported that the value of the RFID market was estimated at about 5.84 billion American dollars, up from 5.63 billion dollars of 2010. This market includes RFID tags, readers, labels, fobs and software services related with RFID cards. This growth of the RFID market is expected to drift the USN market which in the next decade is estimated to increase its value by more than 500% [14].

The main reasons that the RFID technology attracts the interest of the markets is that it can remove boundaries limited by old solutions and that it can supply the traditional Line of Sight (LoS) barcode technology [15]. The deployment of RFID in the sectors of manufacturing and retail automation, asset tracking and supply-chain management managed to connect to some extend the digitally world and the physical world (Figure 2.2) and in the near future, RFID tags is expected to overwhelm our environment [16].

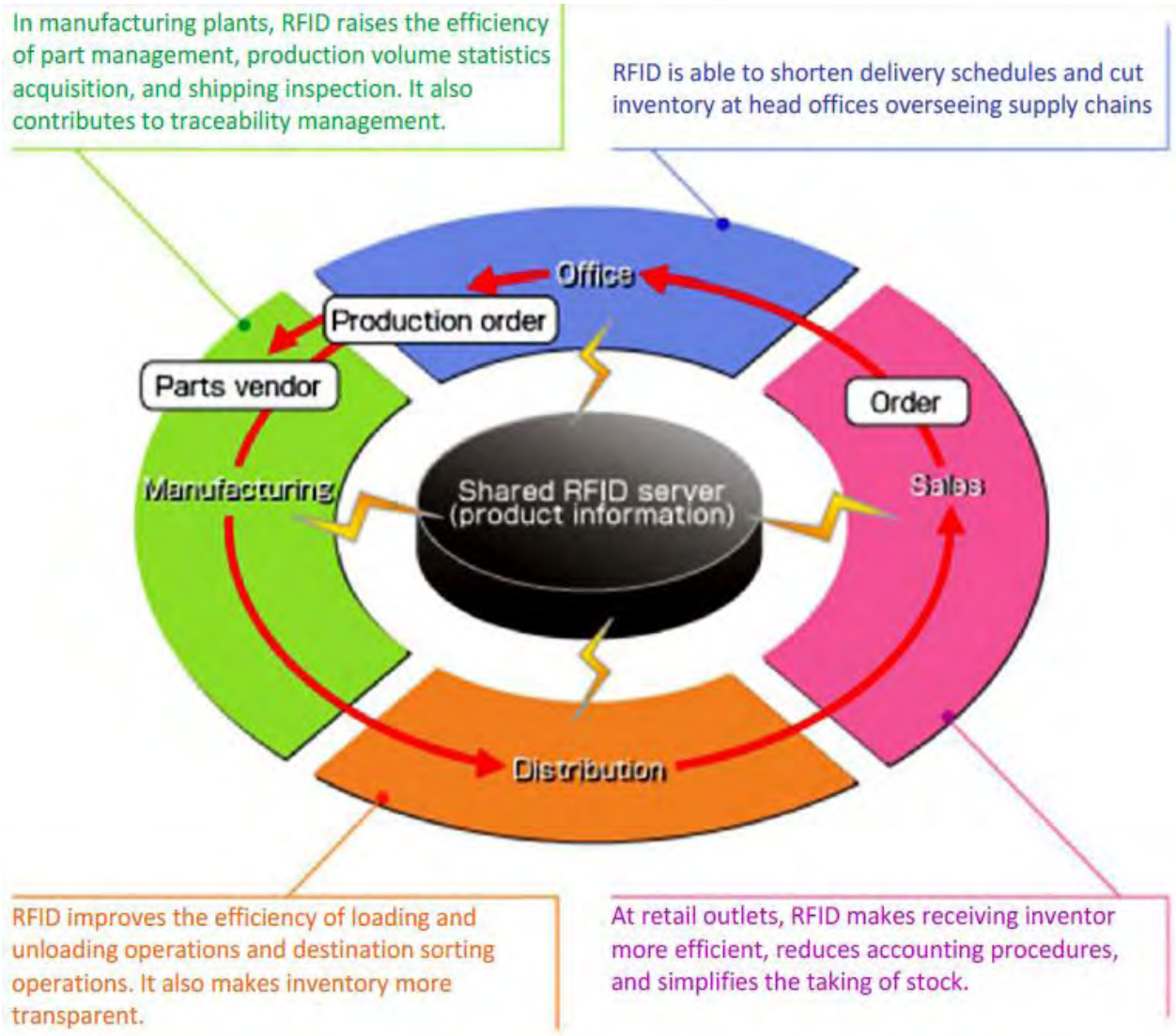

Figure 2.2: RFID connects digital and physical worlds (Figure adopted from [16])

On the other hand, a RFID reader can access information even if the RFID tag is hidden (sometimes security reasons may require this). This raises some concerns about privacy that did not exist with barcode technology, since the products embedded with RFID tags can be monitored beyond the intended use of manufacturers. As a scenario it can be mentioned that advertising agencies could influence the technology for directed selling or an ever more terrifying one that security agencies could monitor their customers. This security hole can be partially covered by RFID ASIC which supports detection only from authorized readers.
The RFID systems consist of two components:

- wireless tags and
- readers.

Wireless Tags: Each tag possesses a unique identification number and a storage capacity which is used for information such as product name, product ingredients, manufacturer etc., or environmental factors such as temperature, humidity etc. Usually, the tags are either attached or embedded into the objects which need to be detected [17].
Readers: A reader is used to read or write data to tags through wireless transmissions technologies. It can identify an object by reading the unique identification number of the tag and then access a database which supplies the matching between products and tad IDs [18].

2.1.2 RFID Classification

In recent years, the researchers concentrated their efforts on creating a classification of RFID systems. However, this project is very difficult since RFID systems are still evolving and new technologies are embedded into these systems. Generally, the factors that characterize RFID systems are:

- The operating frequency,
- The utilization,
- The reading distance,
- The protocol they adopt,
- The power supplied to the tag, and
- The process for sending data from the tag to the reader [19].

The most widely accepted classifications are presented below.

2.1.2.1 Transmitting Power Classification Model

In terms of the method of transmitting power from the reader to the tag, the RFID systems are classified into:

- Near Field
- Far Field [20]

The area surrounding the reader antenna is divided into the Near-Field sector and the Far-Field sector (Figure 2.3). In the far-field, electric and magnetic fields, which are perpendicular to each other, transmit as an electromagnetic wave and the angular distribution is not affected by the distance from the antenna. In near field the components have a range of angular dependence. Furthermore, the Near Field sector can be sub-divided into:

- Radiating (the angular field distribution depends on the distance), and
- Reactive (the energy is not radiated).

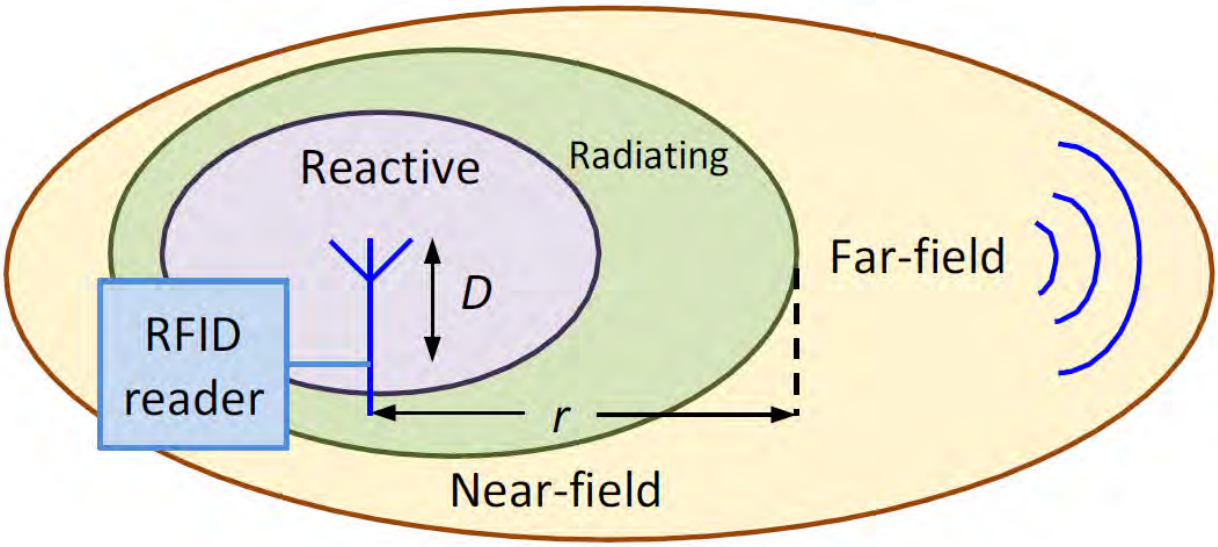

Figure 2.3: Antenna regions (Figure adopted from [21])

2.1.2.2 Powering Up Process Classification Model

In terms of the process of powering up the tags, the RFID systems are classified into:

- Passive,
- Active and
- Semi Active [20].

Passive Systems: In Passive RFID systems, the reader communicates first, followed by the tag. The reader is necessary for a passive tag to transmit its data, since does not have any power supply but they use the power from the reader to gain energy [22]. A passive tag usually consists of a microchip and an antenna and its range can reach 10 meters. Some passive RFID tags are shown in Figure 2.4. The main strong points and drawbacks of passive tags are presented in Table 2.1.

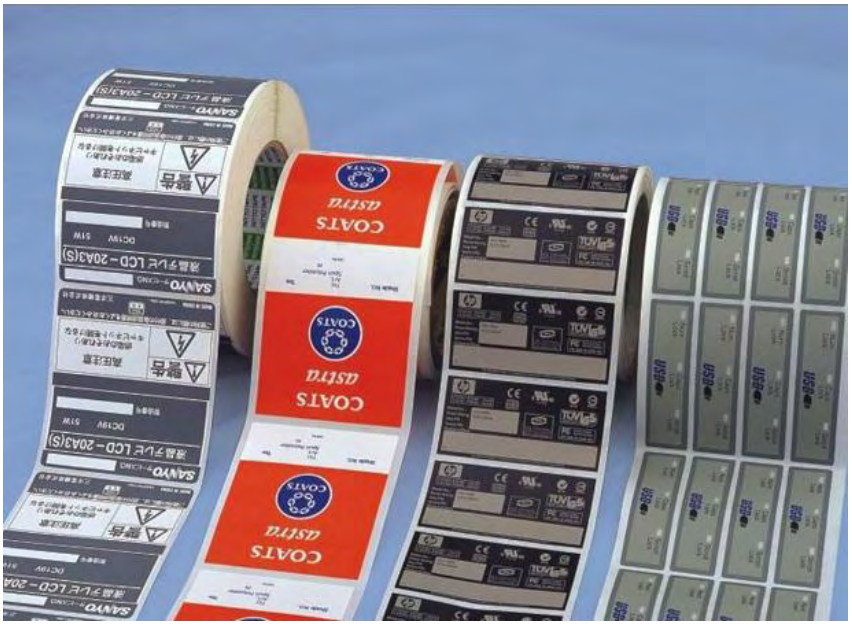

Figure 2.4: Passive RFID tags

Advantages	Disadvantages
Low price	Short reading distance
Resistant to environment conditions	Requires high power readers
Unlimited operational lifetime	
Light weight	

Table 2.1: Advantages and disadvantages of passive RFID tags

Active Systems: In active RFID systems, the tag communicates first, followed by the reader. The reader is useless in data transmission as the active tags use batteries for their power needs. These tags broadcast their data until their power is off. Another kind of active tags can enter a sleep mode in which does not broadcast any information. The reader can wake the tag and require data. This technology can increase the battery life of the tag. The range of these tags can be greater than 30 meters [23]. Some active RFID tags are shown in Figure 2.5, while their advantages and disadvantages are presented in Table 2.2.

Figure 2.5: Active RFID tags

Advantages	Disadvantages
Wide range distance	Heavy weight
Independency from the reader	Short lifetime
	High price
	Affected by the environmental conditions

Table 2.2: Advantages and disadvantages of active RFID tags

2.1.3 RFID Components

From the RFID tags hardware perspective, the tag consists of:

- A microchip
- An antenna
- A substrate

Microchip: it is the processor of the tag. Usually it is extremely small in size, and the functions it can execute are simple, since the microchip is responsible only for transmitting the tag's ID, which is stored in the microchip's memory. The antenna's power activates the microchip and its logic circuit retrieves the ID from the memory. Finally the microchip broadcasts the ID information by using backscattering modulation. The latest RFID tags' microchips curry an extra memory, which can be written by the user [24].
Antenna: this is the most important component of the RFID tag. It is responsible for receiving and transmitting RF waves which enables the communication between the tag and the reader. The size of the antenna designates the size and the cost of the RFID tag [25].
Substrate: it holds the components of the system. The substrate could be characterized as the RFID tag's motherboard.

2.2 WiFi

IEEE 802.11 suite of standards, widely known as Wi-Fi, defined the initial standard for Wireless Local Area Networks (WLANs). Currently, there are two signaling frequencies used by Wi-Fi networks:

- 2.4 GHz – This frequency includes 14 channels which are transmitting in a bandwidth of 20 to 22 MHz.
- 5 GHz - This frequency includes 13 channels which are transmitting in a bandwidth of 20 MHz.

Signals of higher frequencies face higher attenuation when passing through objects (e.g. walls) than signals of lower frequencies. This means that higher frequencies loose bigger amounts of their strength. Moreover, the signal strength that reaches the receiver varies from timeslot to timeslot, since the transmission attributes are dynamic and unpredictable. A very important feature of the received signal is the Signal-to-Noise-Ratio (SNR). SNR compares the level of a desired signal to the level of noise and it is usually expressed in decibels.
Another basic characteristic that comes under every network is throughput. Throughput is the rate of successful message delivery over a communication channel. It is measured in bits per second (bps) and sometimes in data packets per time slot. A feature that can be used interchangeably is data rate. Data rate is the average number of bits per unit time and it is also measured in bits or bytes per second (bps/Bps). Next table displays the supported data rates of IEEE 802.11 standards.

802.11 Extension	Supported Data Rates
802.11	1, 2 Mbps
802.11a	6, 9, 12, 18, 24, 36, 48, 54 Mbps 6, 12, and 24 Mbps are mandatory
802.11b	1, 2, 5.5, 11 Mbps
802.11g	1, 2, 5.5, 11, 6, 9, 12, 18, 22, 24, 33, 36, 48, 54 Mbps 1, 2, 5.5, 11, 6, 12 and 24 Mbps are mandatory 22 and 33 Mbps are typically not supported
802.11n	1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54, 121.5, 130, 144.44, 270, 300 Mbps

Table 2.3: Data Rates Supported by IEEE 802.11 Standards

Each Wi-Fi network is defined by a name and can communicate only on predefined channel frequency. The WLANs are based on wireless stations which should be configured in the following ways:

- Operating Mode: the operating mode can be infrastructure or ad-hoc. The infrastructure modes can be created only by access points or routers. On the other hand, in an ad-hoc mode any Wi-Fi capable device can join or even create it.
- Operating Channel: country regulatory agencies and the 802.11 extension determine the available channels. In 802.11b/g, the most commonly selected channels to avoid collisions are channels1, 6 and 11.
- Network Name: the network's name refers to the Service Set IDentifier (SSID). The network broadcasts its SSID, in order to be visible for the Wi-Fi capable devices in range.

The above characteristics enable the creation of a Wi-Fi network, but what is the use of a network if no devices are joined it? The network must be visible for the devices and the devices have to be capable to scan for visible networks. Scanning is a searching procedure for available networks within range of the device. In more details, on scanning, the wireless devices are "listening" for beacon frames, which are the MAC frames that an access point broadcasts [26].

2.3 Bluetooth

The IEEE 802.15.1, known as Bluetooth, is a wireless technology standard for data communication between mobile devices. This technology actually builds personal area networks (PANs) for these devices but can only cover a short range (at most 10 meters). It operates in license-free ISM frequency band between 2.402 GHz and 2.480 GHz(central frequency 2.45GHz) and divides the frequency band into 79channels (each 1MHz wide), while it does not require Light Of Sight (LOS). The Bluetooth radio module uses Gaussian Frequency Shift Keying (GFSK), which represents the binary one as a positive frequency deflection and the binary zero as a negative frequency deflection [27].
Bluetooth is a packet-based protocol which adopts a master-slave structure the Bluetooth enabled devices can support both point to point and point to multipoint connections. Each device connect to others in piconets (Figure 2.6), which are created by a master device simultaneously connecting up to seven active slave devices in an Ad-hoc mode. The slave devices are identified by a 3 bit address. The topology supplies up to 256 parked slaves devices which are synchronized to the master clock. The parked state slave devices are identified by an 8 bit address and the piconet is defined by a unique hopping sequence.

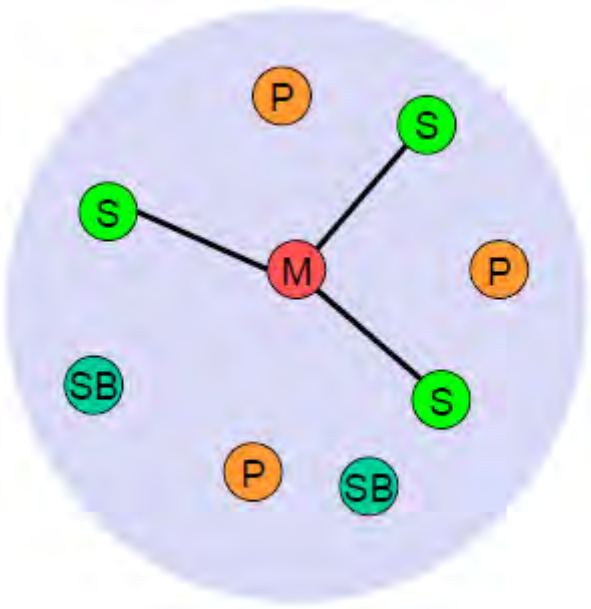

Figure 2.6: Piconet

In order to establish communication between the devices in a piconet, each device must recognize the hopping pattern of the device it wants to connect to and the phase within that pattern. The master device broadcasts its unique identification number, the devices, which are interested in connecting with it, capture this number, and then become slaves and provide all the devices with the correct hopping pattern. The master device transmits its clock information to the slave devices in the piconet, including the clock information into the hopping pattern [28].

In case that there are a number of piconets with overlapping coverage areas, the topology is called scaternet (Figure 2.7). So, scatternet is the link of multiple co-located piconets through the sharing of common master or slave devices. Devices can be slave in one piconet and master of another piconet [Achieved by TDM]. A single device can participate in multiple piconets and can have more than one master device. Up to 10 active piconets can coexist in a scatternet. The communication between piconets is achieved by allowing the devices "jumping" back and forth between the piconets.

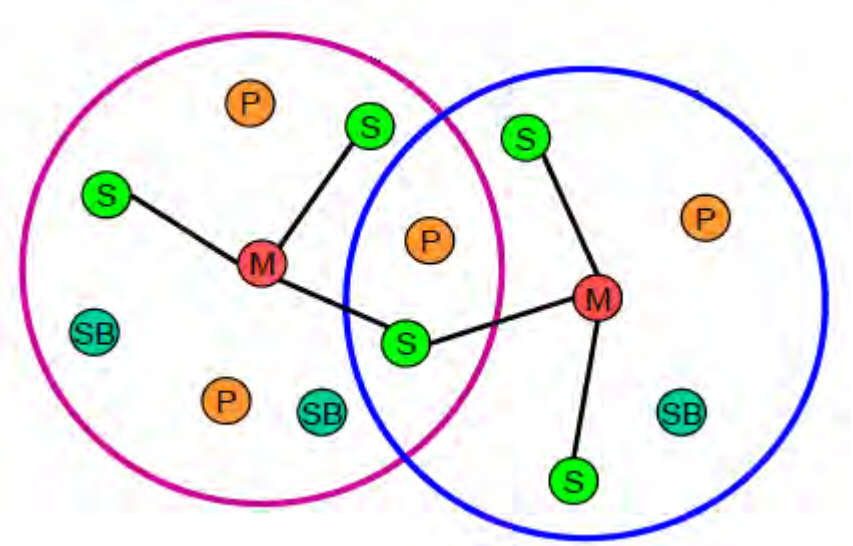

Figure 2.7: Scatternet

The following Table displays a summary of the technical specifications of Bluetooth.

SPECIFICATIONS	
Frequency band	2.4 GHz ISM band
Modulation	Gaussian shaped BFSK
Range	10 -100 m
Physical layer	FHSS
Coverage	Omni-directional. Non line of sight transmission
Data rate	1 Mbps/723 Kbps
Hopping rate	1600 hops/sec at 1 hop/packet
Channels	79/23 channels
Channel length	625 microseconds long

Data packet	Up to 2,745 bits in length
Reliable and secure	Good. Link layer authentication and encryption
Cost	$ 20 aims at $5 endpoint
Power	0.1 W (Active)
Acceptance	SIG have about 2500 member companies
Data / Voice support	One asynchronous data channel (732.2 kbps and reverse 57.6 kbps) OR Three simultaneous synchronous voice channels (64 kbps) OR Simultaneous asynchronous and synchronous channels.
Piconet	1 master and 7 slaves
Scatternet	Up to 10 piconets in a scatternet
Links	SCO and ACL links

Table 2.4: Bluetooth technical specifications

3 Problem Statement and Design Considerations

3.1 Problem Statement

The objective of this project is to offer independent shopping to visually impaired people, without the need of currying any other equipment except of a mobile phone and a cane. The solution is based on a Windows phone application and on cheap equipment for both visually impaired and stores. The implementation of the project performed in a regularly structured environment, in order to meet the requirements of a supermarket environment.

3.2 Design Considerations

The main considerations was to supply many levels of identification within the indoor environment (supermarket), in order to enable the visually impaired person to find the passageway he is interested in and the shelf that contains the product he wants to buy (e.g. healthcare product). Then the shopper must be able to detect the right product between the different products of the same kind (toothpastes, toothbrushes). Up to now, a passageway-level and a shelf-level tagging of products are necessary. When the shelf is identified, the visually impaired can use the QR codes (barcodes), which are displayed on the products, to distinguish between various products of the same kind.

Taking advantage of the existing infrastructure: one of the most important goals was to implement a low-cost solution for both visually impaired and supermarket. In order to reach this goal, the existing infrastructure elements that are already available in supermarkets had to be exploited. The products in supermarkets are tagged with QR codes and the use of RFID tags for market identification has been discussed for serious advantages (store significant amount of data information, re-programmability of tags, location accuracy etc.)[29].
Usability: another goal was to make the solution easy to be used by a visually impaired. To ensure that, the project had to:

- adopt a format for the user interface that is conventional to the user
- ensure the portability
- offer a solution that does not attract attention to the user (avoid large and heavy equipment or unusual audio signals).

The use of a mobile phone with an embedded application that supports test-to-speech conversion and barcode scanning ensures the first two goals. The third goal can be achieved with the use of a headphone or a Bluetooth head-set.

4 Related Work

4.1 Introduction

Independent shopping is a very challenging function for blind people. There have been implemented a number of shopping systems that tried to address the problem of blind shopping. The most important of the existing approaches are identified in this chapter with their design requirements.

4.2 RoboCard

The first accessible blind shopping project started in 2004 by Kulyukin [30]. This project, called *RoboCard*, uses a device aimed at assisting visually impaired people to shop independently, providing the shopper with effective interfaces to the locomotor and haptic spaces of a supermarket. The project is based on the idea that robots can act as effective interfaces to haptic and locomotor spaces in modern supermarkets and permits three product selection modalities - browsing, typing and speech, which allow the blind shopper to select the desired product from a list of products (Figure 3.1).

Figure 4.1: RoboCard (adopted from [30])

RoboCard, consists of:

1) a Pioneer 2DX platform with an attached polyvinyl chloride pipe (PVC)
2) a laptop
3) a SICK laser range finder,
4) a RFID reader,
5) a 200mm x 200mm RFID antenna,
6) a shopping basket.

The localization was succeeded through passive RFID tags that were deployed at various locations in a store. In more detail, the tags were placed at the beginning and end of every passageway and at three different locations within each passageway. These tags allowed a robot to estimates its position in each passageway and to correct global Markov localization errors. There were developed several laser range finding techniques for obstacle avoidance. Moreover, single subject experiments were executed with two visually impaired participants. Two years later, the authors included a small 10-key Belkin keypad and a handheld wireless IT4600 SR barcode scanner in their solution. The keypad was attached to the handle for product selection by list browsing and for robot control (Figure 3.2).

Figure 4.2: Belkin keypad on handle adopted from [31]

After these changes, the researchers tested their results by asking from a visually impaired person to:

a) find RoboCart in the cart area,
b) use the keypad to select three grocery items,
c) navigate to shelf sections,
d) retrieve the selected items from the shelves by scanning shelf barcodes and reaching over them to retrieve the items,
e) place the items into RoboCart's basket,
f) navigate to a cash register,
g) place the items on the conveyer belt,
h) pretend to pay with a credit card,
i) navigate to the exit,
j) remove the shopping bags from RoboCart, and
k) leave the store.

The visually impaired man successfully completed all the challenges [31]. Finally, RoboCard project offered us a generic idea on the way we can handle the information gathered from RFID tags and implement it in our project.

4.3 ShopTalk

ShopTalk [32] is a wearable system designed to help blind shoppers find products on shelves in grocery stores. It uses synthetic verbal route directions and descriptions of the store layout and leverages the orientation and mobility skills of independent blind people to direct them to passageways with target products. Inside the passageways areas, an off-the-shelf barcode scanner is used in combination with a software data structure, called a barcode connectivity matrix, to locate target product on shelves. The system consists of:

1) an OQO computer,
2) a Belkin numeric keypad,
3) a Hand Held Products IT4600 SR wireless barcode scanner with two plastic stabilizers,
4) a USB hub to connect all components.

The equipment was placed in a backpack. The keypad was attached to either left or right shoulder strap, depending on whether the shopper was left- or right-handed. Furthermore, a headphone had to be worn by the visually impaired to hear the voice instructions (Figure 3.3). An important component of the ShopTalk's architecture is the Barcode Connectivity Matrix (BCM) (Figure 3.4). The BCM associates the barcodes with passageways, passageways sides, groups of shelves, specific shelves in a shelf group, and positions on shelves. The BCM is used to generate the store navigation, product search and retrieval instructions. The experimental results of ShopTalk were very promising. The visually impaired man can execute verbal template-based route and product search instructions in supermarkets with 100% accuracy.

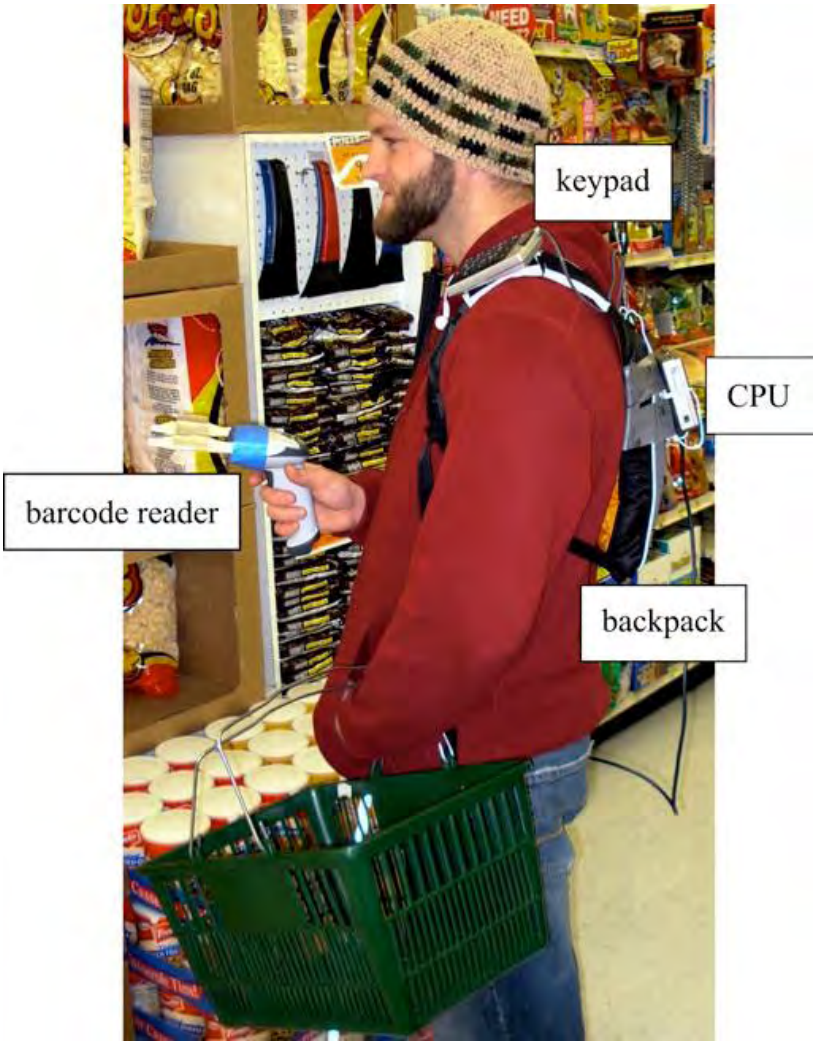

Figure 4.3: ShopTalk (adopted from [32])

Figure 4.4: MSI barcode on shelf scanned (adopted from [32])

4.4 GroZi

The GroZi project [33] exploits a portable handheld device that is used as human eyes, allowing visually impaired person to navigate in indoor environments and locate objects and locations of interest. GroZi is focused on the deployment of a navigational feedback device that combines a mobile visual object recognition system with haptic feedback. The system gives a shopper the ability to navigate himself in a supermarket, find a specific passageway, read passageways' labels, scan the passageway for products on the shopping list and navigate the shopper to find the product. GroZi consists of:

1) an accessible web site for users to creat grocery shopping lists in their homes
2) computer vision software for recognizing products and signs in stores
3) a MoZi Box (Figure 3.5) and the GroZi hand glove (Figure 3.6) with a small portable camera and vibrating motors to allow the execution of computer vision algorithms and give the user haptic and verbal feedback.

With the GroZi system, a blind user enters a web site to compile a shopping list of products and uploads it on the portable device. The user has to curry the device with him in the supermarket to receive directions for each product in the shopping list. For example, if the user wears the glove and points at a passageway, it will identify the components of that passageway and navigate the user to the right products. In GroZi, access to the shopping list is taken for granted, and the user cannot modify the shopping list in the supermarket. Another drawback is that the user needs to curry a powerful laptop on his back, which increases the already high system's cost.

Figure 4.5: MoZi Box (adopted from [33])

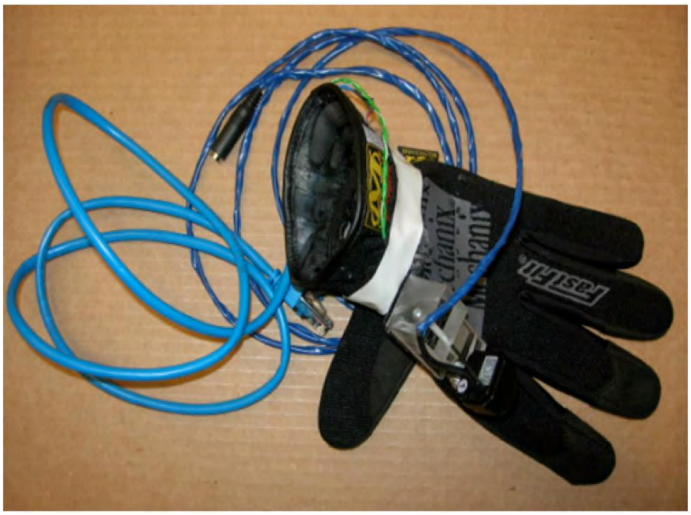

Figure 4.6: GroZi hand glove (adopted from [33])

4.5 iCare

iCARE [34, 35] is an accessible shopping project which aims to design an ambient interactive shopping environment for visually impaired people. The ambient environment can provide indoor navigation, understanding of locations and contents of different sections and a user interface for querying product databases (Figure 3.7). iCare consists of:

1) a PDA with Bluetooth,
2) Wi-Fi,
3) a screen reader,
4) a hand glove with an embedded RFID reader.

While the user walks around the supermarket, the RFID reader in the hand glove reads IDs from the products, searches for them in the database of the supermarket through a Wi-Fi connection, and gives instructions to the user. The main objective of the device is supermarket browsing. When the user moves his hand towards a shelf, the PDA transports messages such as "passing healthcare section", etc. When the user points to a product, he will hear the price of the product, its weight, its ingredients and nutritional data presumably from the RFID tag read from the product. This project is focused on product browsing, and it is not clear if the supermarket's navigation and product search have been evaluated in real-world's requirements. Also, iCare has high implementation cost for both visually impaired and supermarkets.

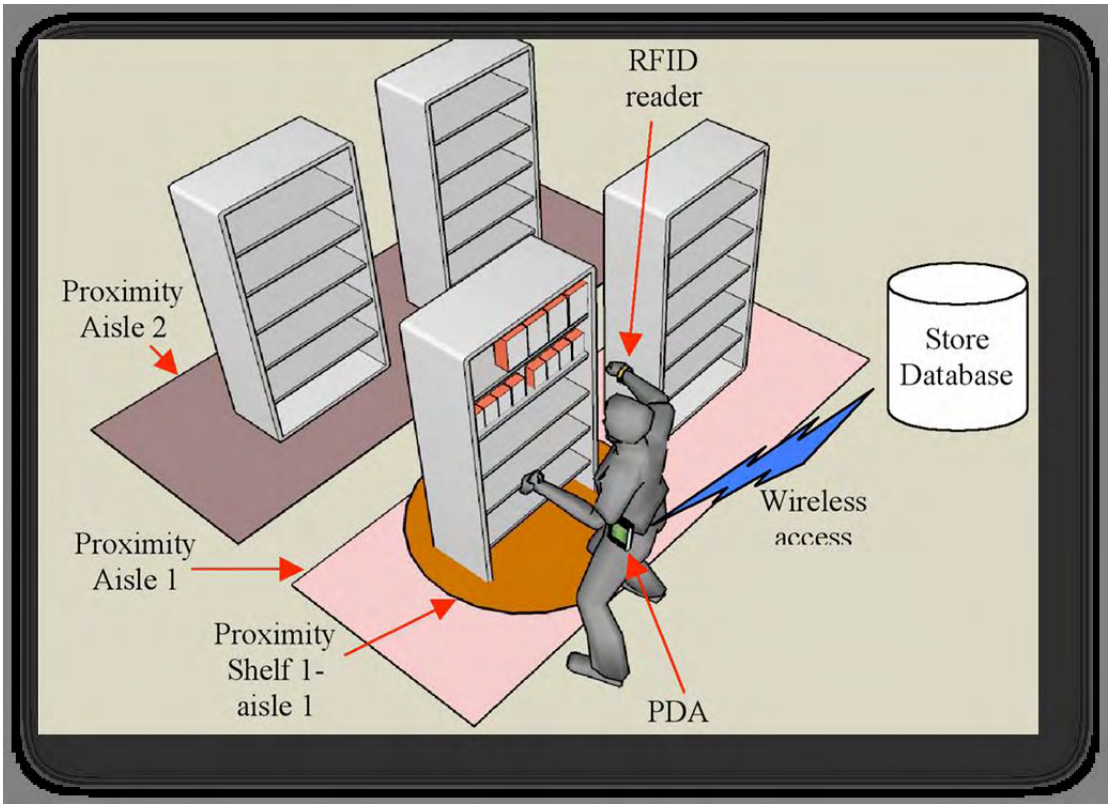

Figure 4.7: iCare Framework (adopted from [35])

4.6 Trinetra

Trinetra project [36, 37] is a cost-effective technology which tried to offer independent shopping for visually impaired in a greater degree than other solutions. The objective of this system is to utilize the collective capability of diverse networked embedded devices to support navigation, grocery shopping, transportation, etc. It is a barcode-based solution comprising a combination of off-the-shelf components, such as an Internet- and Bluetooth-enabled cell phone, text-to-speech software and a portable barcode reader. Trinetra consists of:

1) a Nokia mobile phone,
2) a Bluetooth headset,
3) a Baracoda IDBlue Pen,
4) a Baracoda Pencil,
5) a Windows-based server

The Baracoda Pencil is used to scan barcodes, and the Baracoda IDBlue Pen is used to scan RFID tags. The user scans a UPC barcode with the Baracoda Pencil and retrieves a description of the product. The data are passed wirelessly to and from the mobile phone with the Bluetooth technology. The phone checks the local cache to see if the barcode has previously been scanned. If the barcode is not in the mobile application's cache, a request is sent to a server, which also has a cache of UPC barcodes. If the barcode is in the server's cache, the information is returned to the phone. In case that the barcode does not belong to any cache, a public UPC database is contacted by the server to retrieve the necessary information. The project's hardware and software components are displayed in Figures 3.8 and 3.9 respectively.

Figure 4.8: Trinetra's hardware components (adopted from [36])

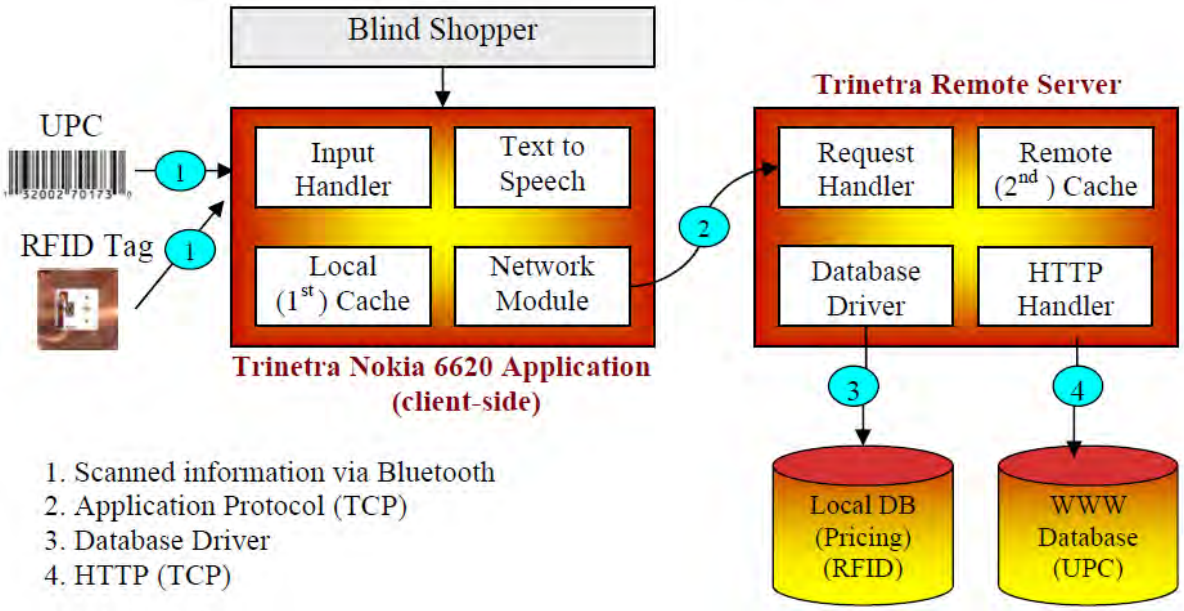

Figure 4.9: Trinetra's software components (adopted from [36])

Trinetra can only identify the products but it cannot navigate a user in the supermarket [38]. Furthermore, product searching is impossible and the implementation cost is high for this project too. Nevertheless, the way the system access the server and search in it was very useful for the implementation of our solution.

4.7 IBM Patent

IBM also, made an effort to provide visually impaired users with location and product identification at sites that have barcode labels. Their patent uses a portable unit to help visually impaired users at supermarkets, by providing them with information on their locations from various barcode labels, through speech synthesis [39]. The location of each barcode label is stored in a database on the portable unit. The user selects a product from the shopping list on the portable unit, and this unit navigates the user from his current location to the location of the product through voice instructions. This approach needs barcode directories at both ends of each passageway similar to building directories. Each barcode directory will list all barcode labels in a specific passageway.

Although this patent can navigate the user in the supermarket, the user can select products only from the target shopping list that already resides on the device. The user is not able to search through products and identify them, except from those which are included in the list. Furthermore, IBM has never built a prototype of this device, so it is not tested to see if it meets the design requirements in a real supermarket.

4.8 Comparison of Solutions

The following table offers a comparison of the existing solutions. The plus sign (+) characterizes a system which meets a design requirement. The minus sign (-) is assigned to systems that do not meet a design requirement. The asterisk sign (*) corresponds to systems that partially meets a requirement and the question mark (?) to systems that it is unknown if they meets a specific requirement. The final column presents if a system was tested in real world.

	Product selection	Store navigation	Product search	Product identification	Utilization of existing devices	Minimal environmental adjustment	Tested
RoboCart	*	+	*	*	-	-	+
ShopTalk	-	+	+	*	-	*	+
GroZi	-	*	*	*	-	?.	-
iCARE	-	*	*	+	*	-	-
Trinetra Barcode	*	-	-	*	+	+	-
Trinetra RFID	*	-	-	+	+	-	-
IBM	-	+	*	?.	?.	-	-

Table 4.1: Comparison of solutions

5 Hardware and Architecture

5.1 Hardware Components

Before presenting the architecture of the solution, it would be necessary to distinguish the hardware components that this project requires. The solution includes:

- RFID tags,
- a Windows mobile phone with an embedded camera,
- an RFID reader attached on the cane,
- a Server (or computer acting as a remote server),
- a modem/router supplying Wi-Fi connection.

For our convenience, during the testing procedure of the solution, we used an NFC enabled mobile phone attached on the cane, in order to substitute the RFID reader.

5.2 Architecture

This project can be divided into three major components:

- The database,
- The software in windows phone, and
- The RFID technology.

Database: the system uses a SQL Server 2008 database to store the information associated with tags. However, the operation of the database is not bound on a one and only software. In more details, the database contains information about products' ingredients, expiration date, name, weight, price and a description. Each product is recognized by a unique identifier. For example, the product with ID equals "31cf4230-e858-4e1a-ab08-9f3c1e24b3a6" corresponds to the product named "Tomatoes Batala" with specific values in weight, price, description and expiration date characteristics.
Software: The software component, written in Visual Studio 2013, is a windows phone 8 application, which was designed to handle communication between RFID stuff and user and also between database and user. The software first connected to database through the WiFi connection. Then, the user is able to give voice commands about the navigation or scanning procedures. The application "listens" for RFID signals and as soon as a signal is detected, it communicates with the database to retrieve data. The results of the database search are returned to the application, which announce them through speech synthesis. Moreover, into this component falls the software created to send messages from the RFID reader to the windows phone application. In our tests the RFID reader was presented by a NFC enabled windows phone. The NFC enabled phone connects via Bluetooth with the windows phone application and sends messages about the tags that are detected.
RFID technology: RFID technology was used to enable the recognition of the user's location. This category is composed of the RFID reader and the RFID tags. The reader is attached on the cane and the tags are placed on the floor to represent points of interest.

5.3 Components Communication

The components, as mentioned before, refer to the database, the software and the RFID technology. What about the communication model between them? How do the components transmit and receive data? This section demonstrates the technologies that facilitates the communication between the components and ensures the integrity of transmitted data. The choice of the most appropriate technologies is based on the requirements of a low cost and easy deployment system, which is built to meet the real-time demands.

The communication part of the system components can be is classified into two domains:

- The RFID reader – RFID tags communication
- The RFID – Windows phone application communication
- The Windows phone application – Database communication

The communication between the reader and the tags is achieved through radio signals. The RFID tags transmit signals on a continuous base. The reader is responsible for detecting and transforming them to useful information. More details about the communication between RFID components have been presented in Section 2.

The communication between the RFID components and the windows phone application is achieved via Bluetooth technology. Bluetooth is responsible for data transmission between the RFID reader and the windows phone application with minimum losses in data integrity and time domains. As soon as the application initializes its main function, a background thread works to trigger the Bluetooth service. The service exchanges signals (such as 3-way handshake) at regular intervals in order to ensure the secure communication between the RFID reader and the application. If a handshake signal is not received, the communication attempt has failed and the Bluetooth connection is broken. This fires the execution of a scanning procedure, which searches for new devices. When a new device is identified, a new connection is available and the user can demand the Bluetooth coupling.

On the other hand, the communication between the windows phone application and the database server cannot be achieved through Bluetooth technology, since the distance that separates these components can be out of the Bluetooth communication range. For this kind of communication, the Wi-Fi technology appeared to be the most suitable module. Wi-Fi is responsible for data transmission between the windows phone application and the database server. As soon as the application starts, a background thread triggers the Wi-Fi searching procedure. If a connection is identified, the user can establish the connection or not. Through Wi-Fi, the application can transmit demands for searching in the database data and retrieve the results from the database server. If the Wi-Fi connection has failed during the execution of the application, the system will inform the user about the loss and a new background thread will fire the method to identify the available Wi-Fi connections.

Figure 5.1 presents a summary of the system's architecture with the technology used for the communication between components.

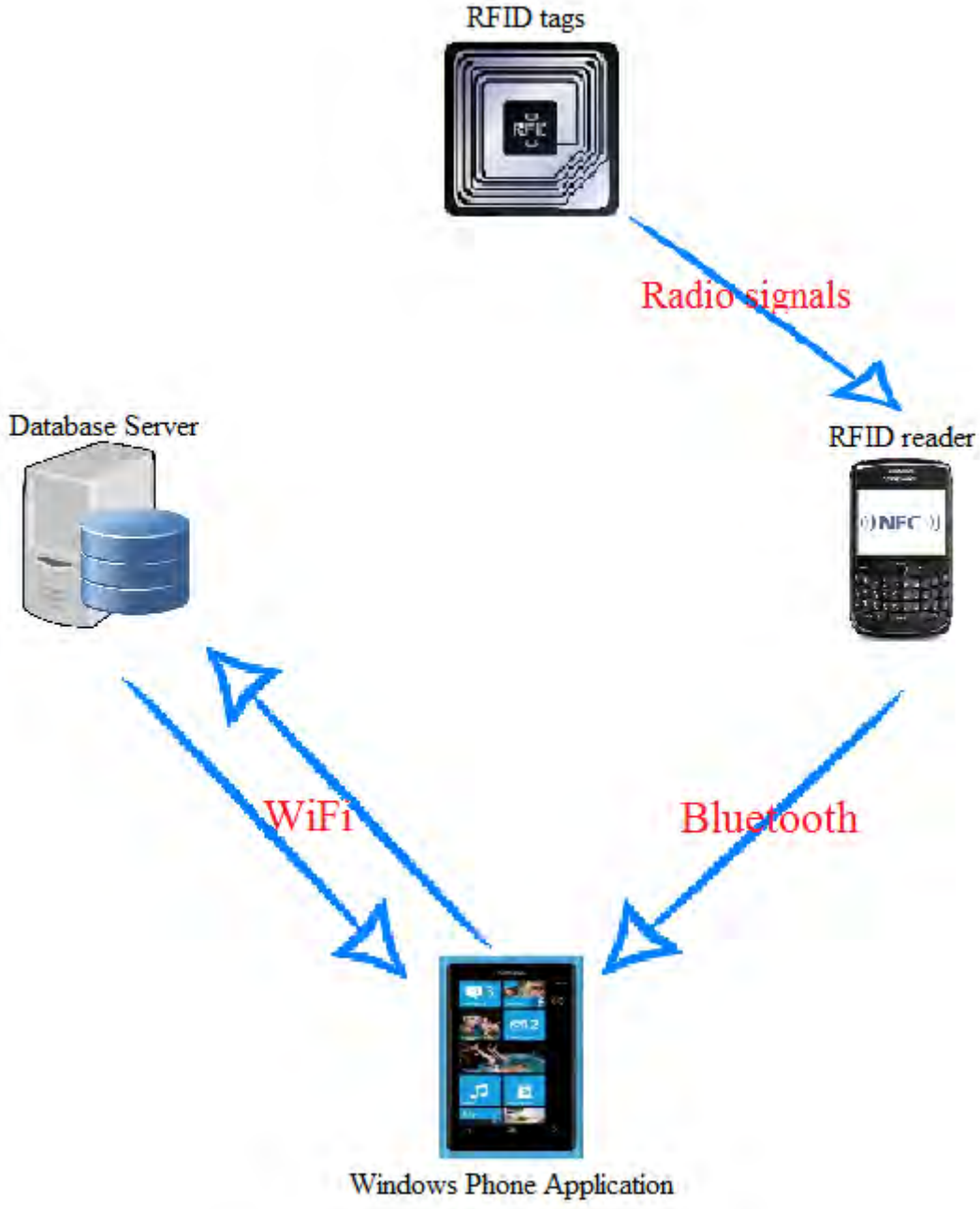

Figure 5.1: Components Communication Model

6 Methodology

6.1 Requirements

The supermarket must divide its area into product's category sections. Each passageway must refer to a specific section, in order to make the searching procedure of a product easier. At the beginning and at the end of each passageway, RFID tags must be attached on the floor (Figure 6.1). These RFID tags must contain the information about the category that each section falls into (e.g. healthcare section). In addition to these, the supermarket has to display its products in a way that all the products on a specific shelf are of the same kind (e.g. toothpastes). For convenience, the QR codes (or barcodes) of the products must be displayed on shelves. Furthermore, the supermarket has to supply Wi-Fi coverage at any point it extends and must maintain a Server. The Server will contain all the information about the supermarket's structure and the products. That means that the Server has to know where each RFID tag is placed and at which section a specific tag refers to. Also, the Server must provide a short description of each product in the supermarket, its name and some details about it.
From the visually impaired man perspective, he must curry a Windows mobile phone. The visually impaired will execute the application, which will enable the camera, Bluetooth and Wi-Fi technologies. Moreover, a RFID reader must be attached on his cane, which will detect the RFID tags.

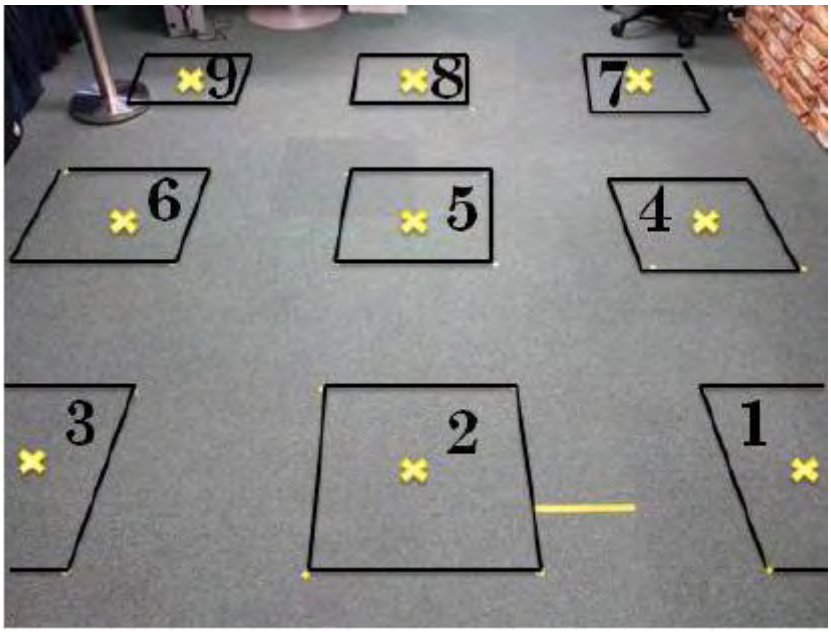

Figure 6.1: RFID tags distribution in the environment

6.2 Implementation Categories

At this point, we are able to divide the solution into four domains:

- the category searching domain,
- the category identification domain,
- the product searching domain, and
- the product identification domain.

The *category searching* domain refers to all the technologies and methods that had to be deployed to ensure that a RFID tag will be correctly recognized by the reader and the Server.
The *category identification* domain includes the functions to identify that the sensed tag was the tag which curries specific information about a supermarket section.
The *product searching* domain refers to the technologies and methods that are used to recognize a QR code in both mobile phone and Server.
The *product identification* domain contains the appropriate functions to identify that the QR code (or barcode) captured by the camera corresponds to a product with specific features.
Each domain has a different goal to achieve. The functionality of each area is specified by a different method in the Windows phone 8 application's source code. In category identification area, it has deployed a method which is responsible for reading the passageways' signals and sending the unique identification number of the RFID tag. A small piece of the code that achieves this is given below. The PeerFinder method is used to advertise the presence of the device that hosts the application in order to enable peers to find the device. In line 6, the application is searching for available devices and creates a list of them, to allow the user chose the one that will try to connect via Bluetooth (method FindAllPeersAsync). The PeerFinder method is necessary to be called before the FindAllPeersAsync. In line 15, the application tries a Bluetooth connection and in the next four lines the application receives the RFID tag identification number, as soon as it is detected.

```
1   private async void AppToApp(string message)
2   {
3       PeerFinder.Start();
4       try
5       {
6           var pairedDevices = await PeerFinder.FindAllPeersAsync();
7           if (pairedDevices.Count == 0)
8           {
9               Debug.WriteLine("No paired devices were found.");
10          }
11          else
12          {
13              MessageBox.Show("Connect with device " +
                pairedDevices[0].DisplayName);
14              PeerInformation selectedDevice = pairedDevices[0];
15              StreamSocket socket = new StreamSocket();
16              await socket.ConnectAsync(selectedDevice.HostName, "1");
17              DataWriter dataWriter = new DataWriter(socket.OutputStream);
18              dataWriter.WriteString(message);
19              await dataWriter.StoreAsync();
20          }
21          }
22      catch (Exception ex)
```

```
23                {
24                        if ((uint)ex.HResult == 0x8007048F)
25                        {
26                                MessageBox.Show("Bluetooth is turned off");
27                        }
28                }
29        }
```

To enable the product searching domain functionality, the application uses a method which continuously “listens” for voice command from the user. This method exploits the advantages of Visual Studio 2013 and its latest windows phone 8 SDK. At the beginning, a Voice Command Definition (VCD) file was created. This is an XML document that defines all the spoken commands that users can say to fire actions when executing an application. Then, source code was added to initialize the VCD file with the phone's speech feature. Finally, a piece of code which handles the navigation and execution of voice commands. In the example of the system’s code given below, we can see in lines 6-13 the way the application recognizes the voice command “Product” (lines 8, 9) and navigates to the QR code scanning page. Similarly, in lines 15-21 and 23-28, the application navigates to main and product information page, “listening” for “Navigate” or “Info” respectively.

```
1 <VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.0">
2   <CommandSet xml:lang="en-US">
3    <CommandPrefix>Go</CommandPrefix>
4    <Example> product </Example>
5
6    <Command Name="Product">
7      <Example> product </Example>
8      <ListenFor> [and] go [to] product </ListenFor>
9      <ListenFor> [and] product </ListenFor>
10       <!-- Listens for VoiceAssistant Product -->
11       <Feedback> Going to product scaning... </Feedback>
12       <Navigate Target="Scan.xaml" />
13     </Command>
14
15     <Command Name="Navigate">
16       <Example> navigate </Example>
17       <ListenFor> navigate </ListenFor>
18       <!-- Listens for VoiceAssistant Navigate -->
19       <Feedback> Going to navigate... </Feedback>
20       <Navigate Target="MainPage.xaml" />
21     </Command>
22
23     <Command Name="Info">
24       <Example> Info </Example>
25       <ListenFor> [and] info </ListenFor>
26       <Feedback> Going to product info... </Feedback>
27       <Navigate Target="ProductInfoPivotPage.xaml" />
28     </Command>
29
30   </CommandSet>
31 </VoiceCommands>
```

The last two domains are based on similar methods. Both of them exploit the camera of the smart phone device to execute barcode scanning procedure. The main functions of these methods refer to the activation of the phone' embedded camera, the scanning of the barcode and the result identification. By default, the barcode scanner will scan every supported barcode type. If a new barcode is found, device activates vibration. The source code of each function is presented below.

Camera Initializer

```
1   void cam_Initialized(object sender,Microsoft.Devices.CameraOperationCompletedEventArgs e)
2   {
3     if (e.Succeeded)
4     {
5       this.Dispatcher.BeginInvoke(delegate()
6       {
7         _phoneCamera.FlashMode = FlashMode.Off;
8         _previewBuffer = new WriteableBitmap((int)_phoneCamera.PreviewResolution.Width
                                                (int)_phoneCamera.PreviewResolution.Height);
9
10          _barcodeReader = new BarcodeReader();
11          _barcodeReader.TryHarder = true;
12
13          _barcodeReader.ResultFound += _bcReader_ResultFound;
14          _scanTimer.Start();
15        });
16    }
17    else
18    {
19      Dispatcher.BeginInvoke(() =>
20      {
21        MessageBox.Show("Unable to initialize the camera");
22      });
23    }
24  }
```

Barcode Scanner

```
1     private void ScanForBarcode()
2     {
3             //take a snapshot of the camera
4             _phoneCamera.GetPreviewBufferArgb32(_previewBuffer.Pixels);
5             _previewBuffer.Invalidate();
6
7             //scan the snapshot for barcodes
8             //fire ResultFound event
9             _barcodeReader.Decode(_previewBuffer);
10    }
```

Result Found

```
1     void _bcReader_ResultFound(Result obj)
2     {
3       if (!obj.Text.Equals(tbBarcodeData.Text))
4       {
5             VibrateController.Default.Start(TimeSpan.FromMilliseconds(100));
6             tbBarcodeType.Text = obj.BarcodeFormat.ToString();
7             tbBarcodeData.Text = obj.Text;
8             Speak(obj.Text);
9       }
10      else
11      {
```

```
12                  Speak(obj.Text);
13              }
14          }
```

The Speak method that appears in lines 8 and 12 above is the method which converts a text message to speech. The code for this is given below.

```
1           private async void Speak(string message)
2           {
3                   synth = new SpeechSynthesizer();
4                   await synth.SpeakTextAsync(message);
5           }
```

The sequence of actions that describes the methodology, in conjunction with the used methods and the domains each one fall into, are summarized in the next table.

No.	Stage	Method	Command Conveyance	Domain
0	User walks into the supermarket			category searching
1	Locate passageway	Read passageway signs	Speech synthesis	category identification
2	Enable Scanning	Listen voice commands	Speech recognition	product searching domain
3	Locate Product Shelf	Read shelf signs	Speech synthesis	product searching domain
4	Product confirmation	Scan QR codes	Speech synthesis	product identification

Table 6.1: Methodology methods and actions summary

6.3 Interpretation

The understanding of our methodology will be much easier with the following use case scenario. Suppose that a blind shopper wants to visit a supermarket and buy toothpaste. He has a Windows mobile phone and his cane is equipped with an RFID reader. The RFID reader can communicate with the mobile phone through Bluetooth. When the impaired man enters the supermarket, he is connected with the Windows-based Server that the supermarket has installed, via Wi-Fi. As blinds use their cane to navigate, the RFID reader on the cane will detect the RFID tags that will be placed on the floor. While walking in the supermarket, the reader detects a tag that is placed on a section. When the reader detects a tag, it communicates with the mobile phone through Bluetooth. So, the mobile phone sends this information to the Server via the Wi-Fi connection. The Server identifies the location of the tag, according to the information that has passed in it at the installation procedure. Then, the Server sends a message to the mobile phone that the passageway in front of the user contains healthcare products. A speech synthesis function informs the user about this information, so, he decides to walk this passageway. On the shelves the QR code of each healthcare product is displayed. The user activates the camera of his mobile phone and takes a picture of a QR code. Then, the application automatically sends the picture's information to the Server, which replies with the description of the product, its price and maybe its ingredients. The speech synthesis function announces the information to the user, who decides either to buy this toothpaste or not. The flow of the actions executed in the use case could be easily recognized by the flow chart presented in Figure 6.3. Moreover, Figure 6.2 represents a summary of the methodology used in this project.

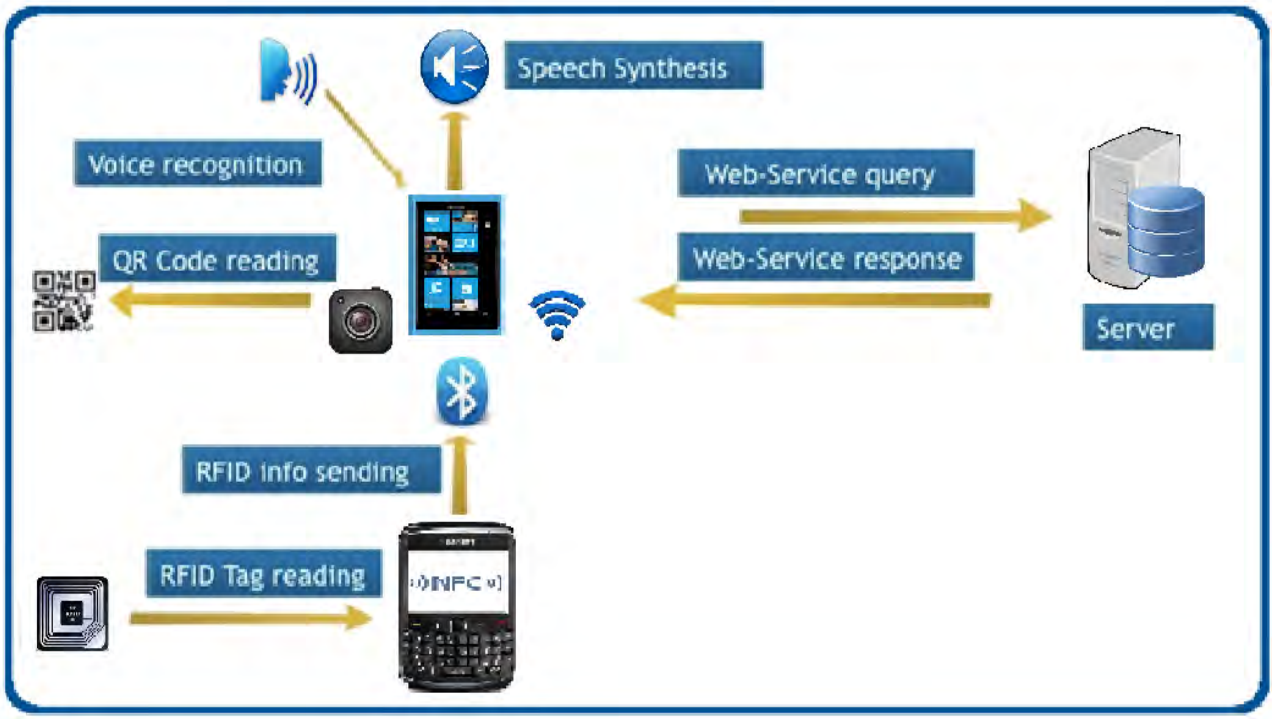

Figure 6.2:Methoology Summery

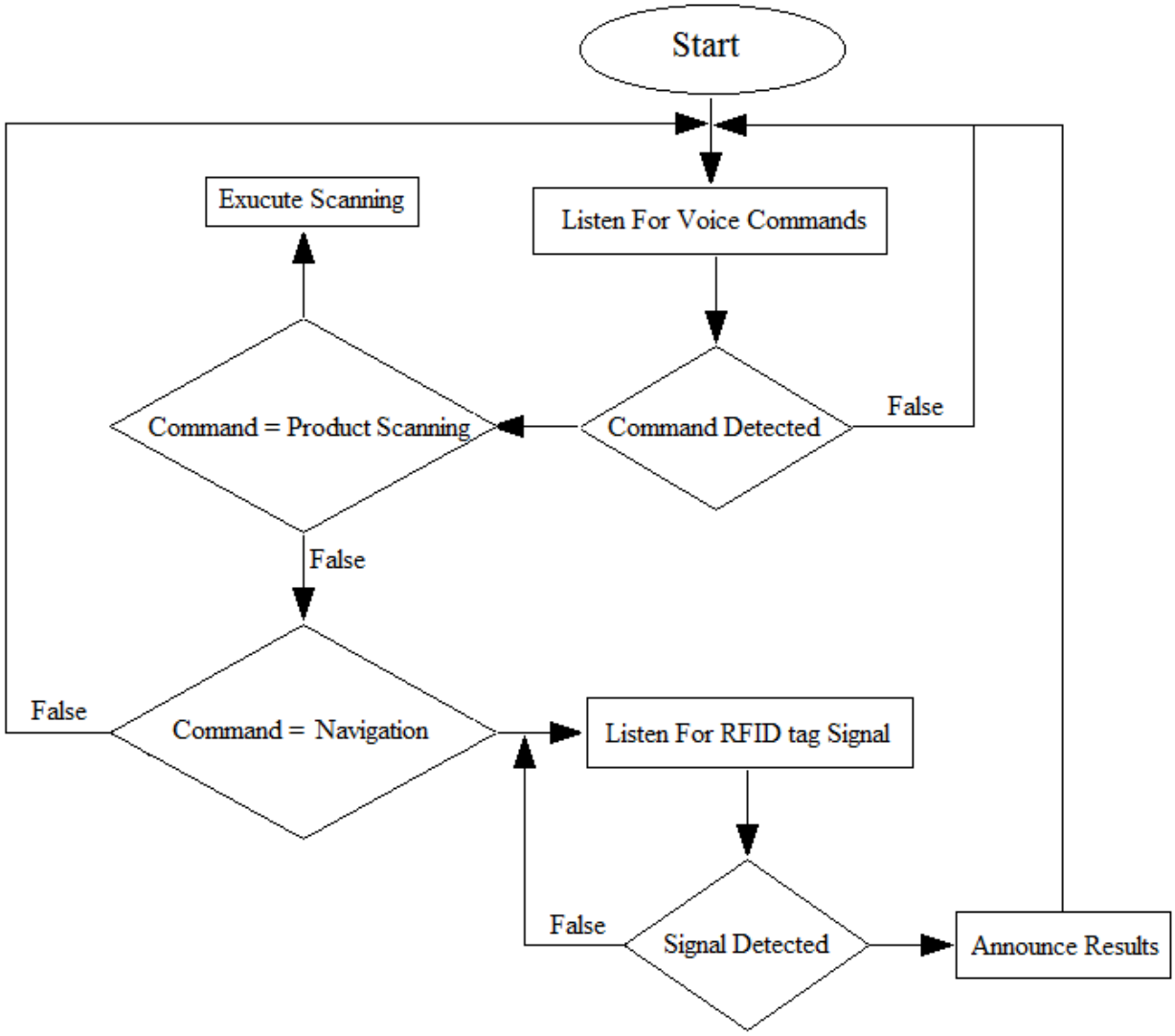

Figure 6.3: Methodology flow chart

7 Results

7.1 Experiment Methodology

After the deployment of this project, we had to test the effectiveness of the solution.
For this purpose, it was asked from a subject to cover his eyes and participate in the experimental procedure, by acting as a visually impaired man. The subject had no experience in navigating with a cane, so the comfort level of navigating in an indoor environment was significantly lower than this of a visually impaired.
The subject in the test was asked to reach a specific product, by searching the RFID tags that were placed on the floor, and scanning the barcodes on shelves. The layout was made to represent a supermarket environment in a smaller scale. This procedure was repeated ten times with different starting point and destination each time. The results were listed to measure the level of interaction between the subject and the system, and helped us evaluate the effectiveness of the solution.
The testing procedure took place in an indoor environment, large enough to have a distance of at least four meters between two RFID tags. The farthest distance between tags was fifteen meters and the average distance was about six meters. The area was cleared of obstacles before the execution of the procedure.

7.2 Experimental Results

The next stage was to test the system. This stage included three tasks:

- Navigating from point A to point B
- Finding and scanning the barcode on shelf
- Finding and scanning the barcode on product.

The locations were selected in a way that the navigating phase from the starting point to the destination required multi-turn directions and the coverage of a distance of minimum fifteen meters.
The next Table presents the results of the first task. The subject was asked to start from point A and reach point B, following any route he wanted. D, E, F and G represent locations of the layout, different with each other. The Table shows the route that the subject followed in each repetition of the test and the seconds required reaching a point from another. The total time needed to reach point B is presented in last column. The average time to complete the task is shown by the last row.

No. Times	A-D	D-E	E-F	F-G	G-B	Total Time
1	A to E -45		31	43	48	167
2	A to F - 52			49	51	152
3	23	52	47	33	39	194
4	A to E - 41		E to G - 56		38	135
5	19	37	E to G - 39		28	123
6	17	32	25	31	31	136
7	18	30	E to G - 41		25	114
8	15	25	19	27	29	115
9	A to F - 30			31	29	90
10	A to E - 32		25	24	27	108
Average Time						133.4

Table 7.1: Navigate from point A to point B in seconds

Table 7.2 displays the results of the second task. The subject was asked to find and scan the barcode of the specific category of products (like the products of the same shelf in a supermarket). Ten different barcodes were attached on ten different shelves. On each repetition, the locations of the barcodes were changing. The table shows the seconds required by the subject to scan the category he was asked for. The average time to complete the task is shown by the last row.

No. Times	Total Time
1	44
2	55
3	47
4	41
5	49
6	32
7	41
8	38
9	29
10	43
Average Time	41.9

Table 7.2: Find and scan the barcode on shelf in seconds

The next table demonstrates the results of the last task. The subject was asked to find and scan the barcode of a specific product. Ten different products were placed on a shelf, each one with a barcode embedded on its package. On each repetition, the locations of the products were changing. The table shows the seconds required by the subject to scan the product he was asked for. The average time to complete the task is shown by the last row.

No. Times	Total Time
1	81
2	78
3	83
4	92
5	74
6	65
7	79
8	77
9	48
10	67
Average Time	74.4

Table 7.3: Find and scan the barcode on product

The system can be evaluated by the time requires to complete all three stages mentioned before. Therefore, the total average time of the system can be calculated by the additions of the average time of each stage. This means that a visually impaired is expected to need 249.7 seconds on average, in order to find the product searching for, in an environment like a small supermarket.
In reviewing the results, we can notice that there is a learning curve in getting familiarity with the system, at least in the first stage, which refers to the navigation procedure in the environment. Furthermore, during testing, we noticed that the subject had to place the cane very close to the RFID tag in order to read. So, for better results, it is recommended the usage of a greater number of tags attached on the floor, in shorter distance between them.

7.3 Software Evaluation

The software's evaluation is a complicated task, so it was divided into four components:

- User input
- RFID reader input
- Speech output
- Database searching

User Input: at the system's menu, the user could choose between two voice commands. The first one is the command "Product", which enables the windows phone's camera in order to scan a barcode, and the second is the command "Navigate", which terminates the barcode scanning procedure and enters the RFD tag scanning procedure. Each time we chose the "Product" command, the system activated the camera for barcode scanning and each time we chose the "Navigation" command, the system deactivated the camera and was listening for RFID tags. So, all aspects, falling into the user input, performed accurately.
Reader Output: when the application initializes, the system enters a loop that constantly listens for RFID tags. As soon as the reader identifies a tag, it sends a message to the application, which contains the unique identifier of the tag. During the tests, this component of the software performed as it was expected.
Speech Output: this component corresponds to a windows phone 8 Text-To-Speech (TTS) method, which is incorporated into the project. This method is responsible for announcing the results of other procedures to the user, in physical language. This component performed very well in tests, supplying the user with clear directions.
Database Searching: the last component refers to the procedure of connecting via Wi-Fi with the server and search the database for the record with a specific ID. We could say that in tests, the required time for completing this procedure was satisfactorily short and did not cause any unpredictable delays on the execution of other procedures.

8 Future Work

Voice Assistant is a low-cost and easily implementable solution for visually impaired assisted shopping. Although the deployment was based on under the consideration of operating in a supermarket, the solution can be adopted to any scenario that meets the project's requirements (e.g. library environment). Experimental results on the field verified our technique in practice. However, while deploying the project, many considerations came off that could have improve the system's performance if a greater budget was available. For example, it would be useful to embed a navigation procedure into the solution. Such a procedure can be deployed to accept voice commands for the destination point. The system must be responsible to execute searching in the environment's database, which will contain a map of the environment.

Moreover, in conjunction with the addition mentioned before, the information that corresponds to the navigation procedure could be the result of an optional route algorithm, which will allow the user to reach the destination in the minimum distance. The optimal route algorithm can be embedded either on the mobile phone or the Server. Although it would be a useful addition to the system, it requires more processing power, either for the mobile phone, or for the server.

The capability of creating a shopping list appears in many already available solutions in visually impaired independent shopping market. In our solution, it could be achieved through a storage method in the windows phone application. The user must be able to record voice commands and store them in a database embedded into the application. The voice commands will be reproduced during the shopping procedure. This could make the shopping much easier for the visually impaired.

Although the solution is developed for windows phones, it can be performed in other operating systems without any tough effort. The Windows phone market is continuously growing, but the Android operating systems cover the biggest part of smart phone application market and the iPhone devices increase their number. The system can be expanded to cover all the available application markets.

Finally, a useful implementation would be the deployment of a method that executes additions of the products' prices. The user should inform the system with the products he adds to the shopping basket and the method will add the prices of the products and announce the total value of the basket, before reaching the cash desk.

9 Conclusions

The conclusions aroused from the tests of the project. The section is divided into hardware and software components. The conclusions are displayed under the consideration of supplying a system which can aid visually impaired in everyday procedures, like shopping in a supermarket. The current solution can be used as basis for more deployments of assisting systems.

9.1 Hardware

The hardware component managed to meet its objective, making it easier for the user to navigate in an environment and recognize the objects in it, by exploiting the advantages of the RFID technology. On the other hand, the tags that were used in our tests seem to be very limited. A tag of about 2 cm. contains a small antenna and it is difficult for the reader to recognize its signal. The reader should be on or extremely near the tag in order to identify the tags existence, so it was difficult for the user to keep the cane always on the floor in order to allow the RFID reader read the tag. In our tests, we used an NFC mobile phone to act as a reader and the accepted distance between the reader and the tag was about 3 cm. As a solution to this problem, the system can be equipped with a RFID reader which covers larger range. Moreover, it is highly recommended the use of a large number of tags on each checkpoint, in order to ensure the identification by the reader. Finally, as the tests demonstrated, the current server with a SQL database performed very well in small scale experiments. However, in the case of big supermarket with billions of products, tags and records in the database, it would be necessary to use a server with much more computational capabilities.

9.2 Software

An effective systems that aims the visually impaired people market, must be designed to facilitate the interaction between the system and the visually impaired. This can happen by adopting speech synthesis and speech recognition in the solution. Through earphones, the user can listen to the system’s directions and navigate into the environment. Furthermore, he can give voice commands and change the functionality of the system.

10 References

[1] Blasch, B. B., Wiener, W. R., and Welsh, R. L., Eds. Foundations of Orientation and Mobility, 2nd ed. American Foundation for the Blind (AFB) Press, 1997.
[2] Bruce, I., Mckennell, A., andWalker, E. Blind and Partially Sighted Adults in Britain: The RNIB Survey. Royal National Institute for the Blind, 1991
[3] Orr, A. L. The psychosocial aspects of aging and vision loss. In Vision and Aging: Issues in Social Work Practice, N. D. Miller, Ed. Haworth Press, 1991, pp. 1–14.
[4] Food Marketing Institute Research. The Food Retailing Industry Speaks 2006. Food Marketing Institute 2006.
[5] L. Atzori, A. Iera, G. Morabito, The Internet of Things: a survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[6] The Internet of Things, ITU Internet Reports, 2005. <http://www.itu.int/internetofthings/>.
[7] Paramvir Bahl and Venkata N. Padmanabhan, "RADAR: an In-building RF- based in Joint Conference of the IEEE Computer and Communications Societies, 2000
[8]International Telecommunication Union. Itu internet reports 2005: The internet of things 2005, 7th ed. Technical report, International Telecommunication Union, Geneva, Switzerland., November 2005.
[9] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier. The internet of things for ambient assisted living. In Proc. Seventh Int Information Technology: New Generations (ITNG) Conf, pages 804–809, 2010.
[10] S. Agrawal and M.L. Das. Internet of things — a paradigm shift of future internet applications. In Proc. Nirma University Int Engineering (NUiCONE) Conf, pages 1–7, 2011.
[11] P.C. Jain and K.P. Vijaygopalan. Rfid and wireless sensor networks. Proceedings of ASCNT–2010, CDAC, Noida, India, pages 1–11, 2010.
[12] F. Zhao. Sensors meet the cloud: Planetary-scale distributed sensing and decision making. In Proc. 9th IEEE Int Cognitive Informatics (ICCI) Conf, 2010.
[13] M. Zorzi, A. Gluhak, S. Lange, and A. Bassi. From today's intranet of things to a future internet of things: a wireless- and mobility-related view. 17(6): 44–51, 2010.
[14] M.M. Aung, Y.S. Chang, and J.U. Won. Emerging rfid/usn applications and challenges. International Journal of RFID Security and Cryptography, 1(1/2): 3–8, 2012.
[15] J. Al-Kassab and W.-C. Rumsch. Challenges for rfid cross-industry standardization in the light of diverging industry requirements. IEEE Systems Journal, 2(2):170–177, 2008.
[16] R. Want. Enabling ubiquitous sensing with rfid. *Computer*, 37(4):84–86, 2004.
[17] H. Liu, M. Bolic, A. Nayak, and I. Stojmenovic. Taxonomy and challenges of the integration of rfid and wireless sensor networks. IEEE Network, 22(6): 26–35, 2008.
[18] J.L. Volakis, G. Mumcu, K. Sertel, C.-C. Chen, M. Lee, B. Kramer, D. sychoudakis, and G. Kiziltas. Antenna miniaturization using magnetic-photonic and degenerate band-edge crystals. 48(5):12–28, 2006.
[19] T. Hassan and S. Chatterjee. A taxonomy for rfid. In Proc. 39th Annual Hawaii Int. Conf. System Sciences HICSS '06, volume 8, 2006.
[20] D.M. Dobkin. The RF in RFID: passive UHF RFID in practice. Newnes, 2007.
[21] P.V. Nikitin, K.V.S. Rao, and S. Lazar. An overview of near field uhf rfid. In Proc. IEEE Int RFID Conf, pages 167–174, 2007.
[22] Z. Tang, Y. He, Z. Hou, and B. Li. The effects of antenna properties on read distance in passive backscatter rfid systems. In Proc. Int. Conf. Networks

Security, Wireless Communications and Trusted Computing NSWCTC ’09, volume 1, pages 120–123, 2009.
[23] E. Nilsson, B. Nilsson, L. Bengtsson, B. Svensson, P. Wiberg, and U. Bilstrup. A low power-long range active rfid-system consisting of active rfid backscatter transponders. In Proc. IEEE Int RFID-Technology and Applications (RFIDTA) Conf, pages 26–30, 2010.
[24] L. Dong-Sheng, Z. Xue-Cheng, Z. Fan, and D. Min. Embeded eeprom memory achieving lower power - new design of eeprom memory for rfid tag ic. IEEE Circuits and Devices Magazine, 22(6):53–59, 2006.
[25] K.V.S. Rao, P.V. Nikitin, and S.F. Lam. Antenna design for uhf rfid tags: a review and a practical application. IEEE Transactions on Antennas and Propagation, 53(12):3870–3876, 2005.
[26]"Wi-Fi (wireless networking technology)", Encyclopedia Britannica.
[27] M. Hietanen, T. Alanko. "Occupational Exposure Related to Radiofrequency Fields from Wireless Communication Systems". XXVIIIth General Assembly of URSI – Proceedings. Union Radio-Scientifique Internationale, 2005.
[28] Ford-Long Wong, Frank Stajano, Jolyon Clulow. “Repairing the Bluetooth pairing protocol”. University of Cambridge Computer Laboratory, 2005.
[29] V. Stanford, “Pervasive Computing Goes the Last Hundred Feet with RFID Systems,” *IEEE Pervasive Computing*, April-June 2003, pp. 9-14.
[30] Kulyukin, V., Gharpure, C., Nicholson, J. 2005. RoboCart: Toward Robot-Assisted Navigation of Grocery Stores by the Visually Impaired. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), Edmonton, Canada, pp. 2845- 2850, IEEE Press. ISBN: 0-7803-8912-3.
[31] Kulyukin V, Gharpure C, Pentico C. (2007). Robots as interfaces to haptic and locomotor spaces (2007). Proceedings of the ACM/IEEE international conference on Human-robot interaction (HRI 2007), pp. 325-31, Washington, DC, USA, ACM.
[32] Nicholson, J. and Kulyukin, V. (2007). ShopTalk: Independent Blind Shopping = Verbal Route Directions + Barcode Scans. Proceedings of the 30-th Annual Conference of the Rehabilitation Engineering and Assistive Technology Society of North America (RESNA 2007), June 2007, Phoenix, Arizona, Avail. on-line
[33] Winlock T., Christiansen E., Belongie S. (2010). Toward real-time grocery detection for the visually impaired, Proceedings of the Computer Vision Applications for the Visually Impaired Workshop (CVAVI), San Francisco, CA, June, 2010.
[34] Krishna, S., Panchanathan, S., Hedgpeth, T., Juillard, C., Balasubramanian, V., Krishnan, N. C. (2008). A Wearable Wireless RFID System for Accessible Shopping Environments, 3rd Intl Conference on Body Area Networks (BodyNets’08), Tempe, AZ, March 2008.
[35] Krishna, S., Balasubramanian, V., Krishnan, N.C., and Hedgpeth, T. (2008). The iCARE Ambient Interactive Shopping Environment, California State University, Northridge, Center on Disabilities' 23rd Annual International Technology and Persons with Disabilities Conference (CSUN 2008), Los Angeles, CA, March 2008.
[36] Lanigan PE, Paulos AM, Williams AW, Rossi D, Narasimhan P. Trinetra: Assistive technologies for grocery shopping for the blind. In: International IEEE-BAIS Symposium on Research on Assistive Technologies (RAT). Dayton, OH 2007.
[37] Narasimhan P. Assistive Embedded technologies. IEEE Computer 2006; vol. 39: pp. 85-87.
[38] Janaswami K. ShopMobile: A mobile shopping aid for visually impaired individuals, M.S. Report, Department of Computer Science. Utah State University, Logan, Utah 2010.

[39] Conzola, V.C., Cox, A.R., Ortega, K.A., and Sluchak, T.J. (2002). Providing Location and Item Identification Data to Visually Impaired Shoppers in a Site Having Barcode Labels. U.S. Patent US 20020158133A1. Oct. 31, 2002.

11 Appendices

Windows Phone Application

Main Page

The functionality of the application's main page.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using VoiceAssistant.Resources;
using Windows.Phone.Speech.Synthesis;
using Windows.Phone.Speech.Recognition;
using Windows.Phone.Speech.VoiceCommands;
using Windows.Networking.Proximity;
using Windows.Networking.Sockets;
using System.Diagnostics;
using Windows.Storage.Streams;

namespace VoiceAssistant
{
  public partial class MainPage : PhoneApplicationPage
  {
    SpeechSynthesizer synth;
    //private ConnectionManager connectionManager = new ConnectionManager();
    // Constructor
    public MainPage()
    {
      InitializeComponent();

      //connectionManager.MessageReceived += connectionManager_MessageReceived;
      Listen();
      AppToDevice();
      //AppToApp();

    }

    async void connectionManager_MessageReceived(string message)
    {
      Debug.WriteLine("Message received:" + message);
      Speak(message);

    }

    private async void Speak(string message)
    {
      // Initialize the SpeechSynthesizer object.
      synth = new SpeechSynthesizer();
      await synth.SpeakTextAsync(message);
      /*
      // Query for a voice that speaks French.
      IEnumerable<VoiceInformation> frenchVoices = from voice in InstalledVoices.All
                                                   where voice.Language == "fr-FR"
```

```
                                    select voice;

            // Set the voice as identified by the query.
            synth.SetVoice(frenchVoices.ElementAt(0));

            // Count in French.
            await synth.SpeakTextAsync("un, deux, trois, quatre");
             */
        }

        private async void Listen()
        {
            try // try block recommended to detect compilation errors in VCD file
            {
                await VoiceCommandService.InstallCommandSetsFromFileAsync(
                      new Uri("ms-appx:///VoiceCommandDefinition.xml"));
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());// Handle exception
                Speak(ex.ToString());
            }
        }

        /// <summary>
        /// Finds the paired device
        /// </summary>
        private async void FindPaired()//not used anymore
        {
            // Search for all paired devices
            PeerFinder.AlternateIdentities["Bluetooth:Paired"] = "";
            try
            {
                var peers = await PeerFinder.FindAllPeersAsync();
                // Handle the result of the FindAllPeersAsync call
            }
            catch (Exception ex)
            {
                if ((uint)ex.HResult == 0x8007048F)
                {
                    MessageBox.Show("Bluetooth is turned off");
                }
            }
        }

        /// <summary>
        /// Method to connect the windows phone app with the app in another phone (e.g. NFC
enabled-NFC Reader) via bluetooth
        /// and process the the messages sent between phones
        /// </summary>
        private async void AppToApp()
        {
            // PeerFinder.Start() is used to advertise our presence so that peers can find us.
            // PeerFinder.Start must be called before FindAllPeersAsync to achieve its goal.
            PeerFinder.Start();
            try
            {
                var pairedDevices = await PeerFinder.FindAllPeersAsync();
                if (pairedDevices.Count == 0)
```

```
            {
                Debug.WriteLine("No paired devices were found.");
            }
            else
            {
                Speak("Connect with device " + pairedDevices[0].DisplayName);
                // The paired devices are added in a list. Select a paired device.
                PeerInformation selectedDevice = pairedDevices[0];
                // Try a connection between the paired devices.
                StreamSocket socket = new StreamSocket();
                // ID_CAP_NETWORKING must be enabled in WMAppManifest.xml, or the next
                // line will throw an Access Denied exception.
                // The second parameter of the call to ConnectAsync() is the RFCOMM port number,
//and can range
                // in value from 1 to 30.
                await socket.ConnectAsync(selectedDevice.HostName, "1");
                DataReader dataReader = new DataReader(socket.InputStream);
                uint sizeFieldCount = await dataReader.LoadAsync(1);
                if (sizeFieldCount != 1)
                {
                    // The underlying socket was closed before we were able to read the whole data.
                    // The application must return.
                    return;
                }
                // Read the message.
                uint messageLength = dataReader.ReadByte();
                uint actualMessageLength = await dataReader.LoadAsync(messageLength);
                if (messageLength != actualMessageLength)
                {
                    // The underlying socket was closed before we were able to read the whole data.
                    // The application must return.
                    return;
                }
                // Read the message and Announce it.
                string message = dataReader.ReadString(actualMessageLength);
                Speak(message);
            }
        }
        catch (Exception ex)
        {
            if ((uint)ex.HResult == 0x8007048F)
            {
                MessageBox.Show("Bluetooth is turned off");
                Speak("Bluetooth is turned off. You must enable Bluetooth to run the application.");
            }
        }

    }

    /// <summary>
    /// Method to connect the windows phone app with a bluetooth enabled device (e.g. NFC
enabled-NFC Reader)
    /// and process the the messages sent from the device to phone
    /// </summary>
    private async void AppToDevice()
    {
        // Configure PeerFinder to search for all paired devices.
        PeerFinder.AlternateIdentities["Bluetooth:Paired"] = "";
        try
```

```
        {
            var pairedDevices = await PeerFinder.FindAllPeersAsync();
            if (pairedDevices.Count == 0)
            {
                Debug.WriteLine("No paired devices were found.");
            }
            else
            {
                Speak("Connect with device " + pairedDevices[0].DisplayName);
                // Select a paired device. In this example, just pick the first one.
                PeerInformation selectedDevice = pairedDevices[0];
                // Attempt a connection
                StreamSocket socket = new StreamSocket();
                // ID_CAP_NETWORKING must be enabled in WMAppManifest.xml, or the next
                // line will throw an Access Denied exception.
                // The second parameter of the call to ConnectAsync() is the RFCOMM port number,
and can range
                // in value from 1 to 30.
                await socket.ConnectAsync(selectedDevice.HostName, "1");
                DataReader dataReader = new DataReader(socket.InputStream);
                uint sizeFieldCount = await dataReader.LoadAsync(1);
                if (sizeFieldCount != 1)
                {
                    // The underlying socket was closed before we were able to read the whole data.
                    return;
                }
                // Read the message.
                uint messageLength = dataReader.ReadByte();
                uint actualMessageLength = await dataReader.LoadAsync(messageLength);
                if (messageLength != actualMessageLength)
                {
                    // The underlying socket was closed before we were able to read the whole data.
                    return;
                }
                // Read the message and Announce it.
                string message = dataReader.ReadString(actualMessageLength);
                Speak(message);
            }
        }
        catch (Exception ex)
        {
            if ((uint)ex.HResult == 0x8007048F)
            {
                MessageBox.Show("Bluetooth is turned off");
            }
        }
    }


    private void PhoneApplicationPage_Loaded_1(object sender, RoutedEventArgs e)
    {
        PeerFinder.ConnectionRequested += PeerFinder_ConnectionRequested;
    }

    void PeerFinder_ConnectionRequested(object sender, ConnectionRequestedEventArgs args)
    {
        Connect(args.PeerInformation);
    }
```

```
    async void Connect(PeerInformation peerToConnect)
    {
      StreamSocket socket = await PeerFinder.ConnectAsync(peerToConnect);
    }


    protected override void OnNavigatedTo(NavigationEventArgs e)//does nothing
    {
      if (e.NavigationMode == System.Windows.Navigation.NavigationMode.New)
      {
        if (NavigationContext.QueryString.ContainsKey("voiceCommandName")) //voice
command in querystring
        {
          var command = NavigationContext.QueryString["voiceCommandName"]; //voice
command in querystring
          switch (command)
          {
            case "product": //The name of the VCD-command
              //var phrase = NavigationContext.QueryString["number"]; //get the phrase
              //DemoText.Text = string.Format("ShowMessage command {0} activated",
phrase);
              break;
            default:
              //DemoText.Text = "Couldn't find the command";
              break;
          }
        }
      }
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
      NavigationService.Navigate(new Uri("/ProductInfoPivotPage.xaml", UriKind.Relative));
    }
  }
}
```

The design of the page

```
<phone:PhoneApplicationPage
  x:Class="VoiceAssistant.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  FontFamily="{StaticResource PhoneFontFamilyNormal}"
  FontSize="{StaticResource PhoneFontSizeNormal}"
  Foreground="{StaticResource PhoneForegroundBrush}"
  SupportedOrientations="Portrait" Orientation="Portrait"
  shell:SystemTray.IsVisible="True">

  <!--LayoutRoot is the root grid where all page content is placed-->
  <Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
```

```
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>

    <!--TitlePanel contains the name of the application and page title-->
    <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
        <TextBlock Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle}"
Margin="12,0"/>
        <TextBlock Text="page name" Margin="9,-7,0,0" Style="{StaticResource
PhoneTextTitle1Style}"/>
    </StackPanel>

    <!--ContentPanel - place additional content here-->
    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <Button Content="Product Info" HorizontalAlignment="Left" Margin="147,141,0,0"
VerticalAlignment="Top" Click="Button_Click"/>

    </Grid>

    <!--Uncomment to see an alignment grid to help ensure your controls are
        aligned on common boundaries.  The image has a top margin of -32px to
        account for the System Tray. Set this to 0 (or remove the margin altogether)
        if the System Tray is hidden.

        Before shipping remove this XAML and the image itself.-->
    <!--<Image Source="/Assets/AlignmentGrid.png" VerticalAlignment="Top" Height="800"
Width="480" Margin="0,-32,0,0" Grid.Row="0" Grid.RowSpan="2" IsHitTestVisible="False" />-->
  </Grid>

</phone:PhoneApplicationPage>
```

Product Information Pivot Page

This page displays and announces the information that have been retrieved about the scanned product.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;

namespace VoiceAssistant
{
  public partial class ProductInfoPivotPage : PhoneApplicationPage
  {
    //http://192.168.1.65:54544/IProductService.cs
    public ProductInfoPivotPage()
    {
      InitializeComponent();

      ///*
      ProductServiceReference.ProductServiceClient productClient = new
ProductServiceReference.ProductServiceClient();
      //productClient.GetAllProductsCompleted += productClient_GetAllProductsCompleted;
      //productClient.GetAllProductsAsync();

      productClient.FindProductByIdCompleted += productClient_FindProductByIdCompleted;
      productClient.FindProductByIdAsync("COLA");
      //*/
    }

    void productClient_FindProductByIdCompleted(object sender,
ProductServiceReference.FindProductByIdCompletedEventArgs e)
    {
      Speak("The Product's Category is " + e.Result.ElementAt(0).CATEGORY.ToString());
      Speak("The Product's Name is " + e.Result.ElementAt(0).NAME.ToString());
      Speak("The Product's Description is " + e.Result.ElementAt(0).DESCRIPTION.ToString());
      Speak("The Product's Expiration Date is " +
e.Result.ElementAt(0).EXPIRATION_DATE.ToString());
      Speak("The Product's Ingrdients are " + e.Result.ElementAt(0).INGREDIENTS.ToString());
      Speak("The Product's Calories are " + e.Result.ElementAt(0).CALORIES.ToString());
      Speak("The Product's Weight is " + e.Result.ElementAt(0).WEIGHT.ToString());
    }

    private async void Speak(string message)
    {
      // Initialize the SpeechSynthesizer object.
      synth = new SpeechSynthesizer();
      await synth.SpeakTextAsync(message);
    }
```

The design of the page

```
<phone:PhoneApplicationPage
  x:Class="VoiceAssistant.ProductInfoPivotPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  FontFamily="{StaticResource PhoneFontFamilyNormal}"
  FontSize="{StaticResource PhoneFontSizeNormal}"
  Foreground="{StaticResource PhoneForegroundBrush}"
  SupportedOrientations="Portrait"  Orientation="Portrait"
  shell:SystemTray.IsVisible="True"
  xmlns:local="clr-namespace:VoiceAssistant">

  <phone:PhoneApplicationPage.Resources>
    <DataTemplate x:Key="ProductDataTemplate">
      <StackPanel Orientation="Horizontal">
        <TextBlock Margin="10" Text="{Binding CALORIES}"/>
        <TextBlock Margin="10" Text="{Binding WEIGHT}"/>
      </StackPanel>
    </DataTemplate>
    <DataTemplate x:Key="ProductImageTemplate">
      <StackPanel Orientation="Horizontal">
        <Image Source="{Binding IMAGE}" Width="150" Stretch="Uniform"
HorizontalAlignment="Center" />
      </StackPanel>
    </DataTemplate>
    <DataTemplate x:Key="ProductImageTemplate2">
      <Grid HorizontalAlignment="Center">
        <Grid.Resources>
          <local:ImageConverter x:Key="imgConverter"></local:ImageConverter>
        </Grid.Resources>
        <Grid.ColumnDefinitions>
          <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Image Stretch="None" Source="{Binding IMAGE.Bytes, Converter={StaticResource
imgConverter}}" />
      </Grid>
    </DataTemplate>
    <DataTemplate x:Key="ProductImageTemplate3">
      <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <Grid.Resources>
          <local:ImageConverter x:Key="imgConverter"></local:ImageConverter>
        </Grid.Resources>
        <Image
        Height="350"
        HorizontalAlignment="Left"
        Margin="15,15,0,0"
        Name="image1"
        Stretch="Fill"
        VerticalAlignment="Top"
        Width="450"
        Source="{Binding IMAGE.Bytes, Converter={StaticResource imgConverter}}"    />
        <TextBlock
        Text="{Binding NAME}"
```

```
            TextWrapping="Wrap"
            Style="{StaticResource PhoneTextAccentStyle}"
            Margin="15,454,10,20" />

            <!-- RenderTransformOrigin="0.496,0.803" -->
        </Grid>
    </DataTemplate>
  </phone:PhoneApplicationPage.Resources>

  <!--LayoutRoot is the root grid where all page content is placed-->
  <Grid x:Name="LayoutRoot" Background="Transparent">
    <!--Pivot Control-->
    <phone:Pivot Title="MY APPLICATION" Loaded="Pivot_Loaded">

      <!--Pivot item one-->
      <phone:PivotItem Header="Product Image">
        <Grid x:Name="ContentPanel1" Grid.Row="1" Margin="0,0,0,0">
          <phone:LongListSelector
        x:Name="LLS1" HorizontalAlignment="Left" Height="548" Margin="39,31,0,0"
VerticalAlignment="Top" Width="383"
        ItemsSource="{Binding}" ItemTemplate="{StaticResource ProductImageTemplate3}"
SelectionChanged="LLS1_SelectionChanged"/>
        </Grid>
      </phone:PivotItem>

      <!--Pivot item two-->
      <phone:PivotItem Header="Product Info">
        <Grid x:Name="ContentPanel2" Grid.Row="1" Margin="-250,200,0,250">
          <phone:LongListSelector
        x:Name="LLS2" HorizontalAlignment="Left" Height="548" Margin="363,-141,-314,0"
VerticalAlignment="Top" Width="383"
        ItemsSource="{Binding}" ItemTemplate="{StaticResource ProductDataTemplate}"/>
        </Grid>
      </phone:PivotItem>
    </phone:Pivot>
  </Grid>

</phone:PhoneApplicationPage>
```

Scan Page

This is the page that scans the barcode.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using Microsoft.Devices;
using ZXing;
using System.Windows.Media.Imaging;
using System.Windows.Input;
using System.Windows.Threading;
using Windows.Phone.Speech.Synthesis;
using Windows.Phone.Speech.VoiceCommands;

namespace VoiceAssistant
{
  public partial class Scan : PhoneApplicationPage
  {
    SpeechSynthesizer synth;

    private PhotoCamera _phoneCamera;
    private IBarcodeReader _barcodeReader;
    private DispatcherTimer _scanTimer;
    private WriteableBitmap _previewBuffer;

    public Scan()
    {
      InitializeComponent();

      Listen();
    }

    protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs
e)
    {
      // Initialize the camera object
      _phoneCamera = new PhotoCamera();
      _phoneCamera.Initialized += cam_Initialized;
      _phoneCamera.AutoFocusCompleted += _phoneCamera_AutoFocusCompleted;

      CameraButtons.ShutterKeyHalfPressed += CameraButtons_ShutterKeyHalfPressed;

      //Display the camera feed in the UI
      viewfinderBrush.SetSource(_phoneCamera);


      // This timer will be used to scan the camera buffer every 250ms and scan for any
barcodes
      _scanTimer = new DispatcherTimer();
      _scanTimer.Interval = TimeSpan.FromMilliseconds(250);
      _scanTimer.Tick += (o, arg) => ScanForBarcode();
```

```
        viewfinderCanvas.Tap += new EventHandler<GestureEventArgs>(focus_Tapped);

        base.OnNavigatedTo(e);
    }

    void _phoneCamera_AutoFocusCompleted(object sender,
CameraOperationCompletedEventArgs e)
    {
        Deployment.Current.Dispatcher.BeginInvoke(delegate()
        {
            focusBrackets.Visibility = Visibility.Collapsed;
        });
    }

    void focus_Tapped(object sender, GestureEventArgs e)
    {
        if (_phoneCamera != null)
        {
            if (_phoneCamera.IsFocusAtPointSupported == true)
            {
                // Determine the location of the tap.
                Point tapLocation = e.GetPosition(viewfinderCanvas);

                // Position the focus brackets with the estimated offsets.
                focusBrackets.SetValue(Canvas.LeftProperty, tapLocation.X - 30);
                focusBrackets.SetValue(Canvas.TopProperty, tapLocation.Y - 28);

                // Determine the focus point.
                double focusXPercentage = tapLocation.X / viewfinderCanvas.ActualWidth;
                double focusYPercentage = tapLocation.Y / viewfinderCanvas.ActualHeight;

                // Show the focus brackets and focus at point.
                focusBrackets.Visibility = Visibility.Visible;
                _phoneCamera.FocusAtPoint(focusXPercentage, focusYPercentage);
            }
        }
    }

    void CameraButtons_ShutterKeyHalfPressed(object sender, EventArgs e)
    {
        _phoneCamera.Focus();
    }

    protected override void
OnNavigatingFrom(System.Windows.Navigation.NavigatingCancelEventArgs e)
    {
        //we're navigating away from this page, we won't be scanning any barcodes
        _scanTimer.Stop();

        if (_phoneCamera != null)
        {
            // Cleanup
            _phoneCamera.Dispose();
            _phoneCamera.Initialized -= cam_Initialized;
            CameraButtons.ShutterKeyHalfPressed -= CameraButtons_ShutterKeyHalfPressed;
        }
    }
```

```
    void cam_Initialized(object sender, Microsoft.Devices.CameraOperationCompletedEventArgs
e)
    {
      if (e.Succeeded)
      {
        this.Dispatcher.BeginInvoke(delegate()
        {
          _phoneCamera.FlashMode = FlashMode.Off;
          _previewBuffer = new WriteableBitmap((int)_phoneCamera.PreviewResolution.Width,
(int)_phoneCamera.PreviewResolution.Height);

          _barcodeReader = new BarcodeReader();

          // By default, BarcodeReader will scan every supported barcode type
          // If we want to limit the type of barcodes our app can read,
          // we can do it by adding each format to this list object

          //var supportedBarcodeFormats = new List<BarcodeFormat>();
          //supportedBarcodeFormats.Add(BarcodeFormat.QR_CODE);
          //supportedBarcodeFormats.Add(BarcodeFormat.DATA_MATRIX);
          //_bcReader.PossibleFormats = supportedBarcodeFormats;

          _barcodeReader.TryHarder = true;

          _barcodeReader.ResultFound += _bcReader_ResultFound;
          _scanTimer.Start();
        });
      }
      else
      {
        Dispatcher.BeginInvoke(() =>
        {
          MessageBox.Show("Unable to initialize the camera");
        });
      }
    }

    void _bcReader_ResultFound(Result obj)
    {
      // If a new barcode is found, vibrate the device and display the barcode details in the UI
      if (!obj.Text.Equals(tbBarcodeData.Text))
      {
        VibrateController.Default.Start(TimeSpan.FromMilliseconds(100));
        tbBarcodeType.Text = obj.BarcodeFormat.ToString();
        tbBarcodeData.Text = obj.Text;
        Speak(obj.Text);
      }
      else
      {
        Speak(obj.Text);
      }
    }

    private void ScanForBarcode()
    {
      //grab a camera snapshot
      _phoneCamera.GetPreviewBufferArgb32(_previewBuffer.Pixels);
      _previewBuffer.Invalidate();
```

```
        //scan the captured snapshot for barcodes
        //if a barcode is found, the ResultFound event will fire
        _barcodeReader.Decode(_previewBuffer);

    }

    private async void Speak(string message)
    {
        // Initialize the SpeechSynthesizer object.
        synth = new SpeechSynthesizer();
        await synth.SpeakTextAsync(message);

    }

    private async void Listen()
    {
        try // try block recommended to detect compilation errors in VCD file
        {
            await VoiceCommandService.InstallCommandSetsFromFileAsync(
                  new Uri("ms-appx:///VoiceCommandDefinition.xml"));
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());// Handle exception
        }
    }
  }
}
```

The design of the page

```
<phone:PhoneApplicationPage

  x:Class="VoiceAssistant.Scan"

  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"

  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"

  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

  FontFamily="{StaticResource PhoneFontFamilyNormal}"

  FontSize="{StaticResource PhoneFontSizeNormal}"

  Foreground="{StaticResource PhoneForegroundBrush}"

  SupportedOrientations="Portrait" Orientation="Portrait"

  mc:Ignorable="d"

  shell:SystemTray.IsVisible="True">
```

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="*" />
    <RowDefinition Height="100" />
  </Grid.RowDefinitions>

  <Canvas x:Name="viewfinderCanvas">

    <!--Camera viewfinder -->
    <Canvas.Background>

      <VideoBrush x:Name="viewfinderBrush">
        <VideoBrush.RelativeTransform>
          <CompositeTransform
          x:Name="viewfinderTransform"
          CenterX="0.5"
          CenterY="0.5"
          Rotation="90"/>
        </VideoBrush.RelativeTransform>
      </VideoBrush>
    </Canvas.Background>
    <TextBlock
    x:Name="focusBrackets"
    Text="[  ]"
    FontSize="40"
    Visibility="Collapsed"/>
  </Canvas>
  <!--Used for debugging >-->
```

```
      <StackPanel Grid.Row="1" Margin="20, 0">
        <TextBlock x:Name="tbBarcodeType" FontWeight="ExtraBold" />
        <TextBlock x:Name="tbBarcodeData" FontWeight="ExtraBold" TextWrapping="Wrap" />
      </StackPanel>
    </Grid>

</phone:PhoneApplicationPage>
```

Connection Manager

This class manages the connection and the messages sent and received through Bluetooth.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Networking;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

namespace VoiceAssistant
{
  /// <summary>
  /// Class to control the bluetooth connection.
  /// </summary>
  public class ConnectionManager
  {
    /// <summary>
    /// Socket used to communicate with.
    /// </summary>
    private StreamSocket socket;

    /// <summary>
    /// DataWriter used to send commands easily.
    /// </summary>
    private DataWriter dataWriter;

    /// <summary>
    /// DataReader used to receive messages easily.
    /// </summary>
    private DataReader dataReader;

    /// <summary>
    /// Thread used to keep reading data from socket.
    /// </summary>
    private BackgroundWorker dataReadWorker;

    /// <summary>
    /// Delegate used by event handler.
    /// </summary>
    /// <param name="message">The message received.</param>
    public delegate void MessageReceivedHandler(string message);

    /// <summary>
    /// Event fired when a new message is received.
    /// </summary>
    public event MessageReceivedHandler MessageReceived;

    /// <summary>
    /// Initialize the manager, should be called in OnNavigatedTo of main page.
    /// </summary>
    public void Initialize()
    {
```

```
        socket = new StreamSocket();
        dataReadWorker = new BackgroundWorker();
        dataReadWorker.WorkerSupportsCancellation = true;
        dataReadWorker.DoWork += new DoWorkEventHandler(ReceiveMessages);
    }

    /// <summary>
    /// Finalize the connection manager, should be called in OnNavigatedFrom of main page.
    /// </summary>
    public void Terminate()
    {
        if (socket != null)
        {
            socket.Dispose();
        }
        if (dataReadWorker != null)
        {
            dataReadWorker.CancelAsync();
        }
    }

    /// <summary>
    /// Connect to the given host device.
    /// </summary>
    /// <param name="deviceHostName">The host device name.</param>
    public async void Connect(HostName deviceHostName)
    {
        if (socket != null)
        {
            await socket.ConnectAsync(deviceHostName, "1");
            dataReader = new DataReader(socket.InputStream);
            dataReadWorker.RunWorkerAsync();
            dataWriter = new DataWriter(socket.OutputStream);
        }
    }

    /// <summary>
    /// Receive messages through bluetooth.
    /// </summary>
    private async void ReceiveMessages(object sender, DoWorkEventArgs e)
    {
        try
        {
            while (true)
            {
                // Read first byte (length of the subsequent message, 255 or less).
                uint sizeFieldCount = await dataReader.LoadAsync(1);
                if (sizeFieldCount != 1)
                {
                    // The underlying socket was closed before we were able to read the whole data.
                    return;
                }

                // Read the message.
                uint messageLength = dataReader.ReadByte();
                uint actualMessageLength = await dataReader.LoadAsync(messageLength);
                if (messageLength != actualMessageLength)
                {
                    // The underlying socket was closed before we were able to read the whole data.
```

```
                return;
            }
            // Read the message and process it.
            string message = dataReader.ReadString(actualMessageLength);
            MessageReceived(message);
          }
        }
        catch (Exception ex)
        {
          Debug.WriteLine(ex.Message);
        }

      }

      /// <summary>
      /// Send command through bluetooth.
      /// </summary>
      /// <param name="command">The sent command.</param>
      /// <returns>The number of bytes sent</returns>
      public async Task<uint> SendCommand(string command)
      {
        uint sentCommandSize = 0;
        if (dataWriter != null)
        {
          uint commandSize = dataWriter.MeasureString(command);
          dataWriter.WriteByte((byte)commandSize);
          sentCommandSize = dataWriter.WriteString(command);
          await dataWriter.StoreAsync();
        }
        return sentCommandSize;
      }
    }
}
```

Convertor

This class converts the information in database (even images) to a form that can be displayed and announced.

```
using System;
using System.Windows.Data;
using System.IO;
using System.Windows.Media.Imaging;
using System.Globalization;
using System.Windows.Controls;
using System.Windows.Media;
using Microsoft.Phone;

namespace VoiceAssistant
{
   public class ImageConverter : IValueConverter
   {
      public object Convert(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
      {
         MemoryStream memStream = new MemoryStream((byte[])value);
         memStream.Seek(0, SeekOrigin.Begin);
         BitmapImage image = new BitmapImage();
         image.SetSource(memStream);
         return image;
      }

      public object ConvertBack(object value, Type targetType, object parameter,
System.Globalization.CultureInfo culture)
      {
         throw new NotImplementedException();
      }
   }
}
```

Voice Command Definition

This class contains the valid voice commands for the application.

```
<?xml version="1.0" encoding="utf-8"?>

<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.0">
 <CommandSet xml:lang="en-US">
  <CommandPrefix>Go</CommandPrefix>
  <Example> product </Example>

  <Command Name="Product">
   <Example> product </Example>
   <ListenFor> [and] go [to] product </ListenFor>
   <ListenFor> [and] product </ListenFor>
   <!-- Listens for VoiceAssistant Product -->
   <Feedback> Going to product scaning... </Feedback>
   <Navigate Target="Scan.xaml" />
  </Command>

  <Command Name="Navigate">
   <Example> navigate </Example>
   <ListenFor> navigate </ListenFor>
   <!-- Listens for VoiceAssistant Navigate -->
   <Feedback> Going to navigate... </Feedback>
   <Navigate Target="MainPage.xaml" />
  </Command>

  <Command Name="Info">
   <Example> Info </Example>
   <ListenFor> [and] info </ListenFor>
   <Feedback> Going to product info... </Feedback>
   <Navigate Target="ProductInfoPivotPage.xaml" />
  </Command>

 </CommandSet>
</VoiceCommands>
```

Database Service

Service Methods

These are the methods of the service that are used to retrieve data from database.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace VoiceAssistantService
{
   // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class
name "ProductsService" in code, svc and config file together.
   // NOTE: In order to launch WCF Test Client for testing this service, please select
ProductsService.svc or ProductsService.svc.cs at the Solution Explorer and start debugging.
   public class ProductService : IProductService
   {
      //port 54544
      public string FindProductNameById(string id)
      {
         ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
         var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.NAME;
         return res.ToString();
      }

      public string FindProductDescriptionById(string id)
      {
         ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
         var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.DESCRIPTION;
         return res.ToString();
      }

      public object FindProductImageById(string id)
      {
         ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
         var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.IMAGE;
         return res;
      }

      public string FindProductIngredientsById(string id)
      {
         ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
         var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.INGREDIENTS;
         return res.ToString();
      }

      public string FindProductWeightById(string id)
      {
         ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
         var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.WEIGHT;
         return res.ToString();
```

```
    }

    public string FindProductExpirationDateById(string id)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.EXPIRATION_DATE;
        return res.ToString();
    }

    public string FindProductCaloriesById(string id)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.CALORIES;
        return res.ToString();
    }

    public string FindProductCategoryById(string id)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select
r.CATEGORY;
        return res.ToString();
    }

    public List<ProductsCollection> FindProductById(string id)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.BARCODE_GUID == id select r;
        return res.ToList();
    }

    public List<ProductsCollection> GetProductsByName(string name)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.NAME == name select r;
        return res.ToList();
    }

    public List<ProductsCollection> GetProductsByExpirationDate(string expirationDate)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.EXPIRATION_DATE ==
expirationDate select r;
        return res.ToList();
    }

    public List<ProductsCollection> GetProductsByCategory(string category)
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections where r.CATEGORY == category select r;
        return res.ToList();
    }

    public List<ProductsCollection> GetAllProducts()
    {
        ProductsDataClassesDataContext context = new ProductsDataClassesDataContext();
        var res = from r in context.ProductsCollections
```

```
                select r;
        return res.ToList();
    }
  }
}
```

Service Declaration

The declaration of service methods

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace VoiceAssistantService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the interface
name "IProductsService" in both code and config file together.
    [ServiceContract]
    public interface IProductService
    {
        [OperationContract]
        string FindProductNameById(string id);

        [OperationContract]
        string FindProductDescriptionById(string id);

        [OperationContract]
        object FindProductImageById(string id);

        [OperationContract]
        string FindProductIngredientsById(string id);

        [OperationContract]
        string FindProductWeightById(string id);

        [OperationContract]
        string FindProductExpirationDateById(string id);

        [OperationContract]
        string FindProductCaloriesById(string id);

        [OperationContract]
        string FindProductCategoryById(string id);

        [OperationContract]
        List<ProductsCollection> GetProductsByName(string name);

        [OperationContract]
        List<ProductsCollection> GetProductsByExpirationDate(string expirationDate);

        [OperationContract]
        List<ProductsCollection> GetProductsByCategory(string category);

        [OperationContract]
        List<ProductsCollection> FindProductById(string id);

        [OperationContract]
        List<ProductsCollection> GetAllProducts();
    }
}
```

Database Instances

This is a copy of the database declaration

```
<?xml version="1.0" encoding="utf-8"?><Database Name="Product"
Class="ProductsDataClassesDataContext"
xmlns="http://schemas.microsoft.com/linqtosql/dbml/2007">
 <Connection Mode="WebSettings" ConnectionString="Data Source=PANOS64-PC;Initial
Catalog=Product;Integrated Security=True"
SettingsObjectName="System.Configuration.ConfigurationManager.ConnectionStrings"
SettingsPropertyName="ProductConnectionString" Provider="System.Data.SqlClient" />
 <Table Name="dbo.ProductsCollection" Member="ProductsCollections">
  <Type Name="ProductsCollection">
   <Column Name="BARCODE_GUID" Type="System.String" DbType="NVarChar(100) NOT
NULL" CanBeNull="false" />
   <Column Name="IMAGE" Type="System.Data.Linq.Binary" DbType="Image"
CanBeNull="true" UpdateCheck="Never" />
   <Column Name="NAME" Type="System.String" DbType="NVarChar(30)" CanBeNull="true"
/>
   <Column Name="DESCRIPTION" Type="System.String" DbType="NVarChar(150)"
CanBeNull="true" />
   <Column Name="CALORIES" Type="System.String" DbType="NVarChar(20)"
CanBeNull="true" />
   <Column Name="WEIGHT" Type="System.String" DbType="NVarChar(50)" CanBeNull="true"
/>
   <Column Name="INGREDIENTS" Type="System.String" DbType="NVarChar(200)"
CanBeNull="true" />
   <Column Name="EXPIRATION_DATE" Type="System.String" DbType="NVarChar(12) NOT
NULL" CanBeNull="false" />
   <Column Name="CATEGORY" Type="System.String" DbType="NVarChar(50)"
CanBeNull="true" />
  </Type>
 </Table>
</Database>
```

NFC tag Scanner and Sender

This class identifies the NFC tag and sends a message via Bluetooth

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using ReadNFC_SendBt_PhoneApp.Resources;
using Windows.Networking.Proximity;
using System.Runtime.InteropServices.WindowsRuntime;
using System.Text;
using System.Diagnostics;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;

namespace ReadNFC_SendBt_PhoneApp
{
  public partial class MainPage : PhoneApplicationPage
  {
    // Constructor
    public MainPage()
    {
      InitializeComponent();

      ProximityDevice device = ProximityDevice.GetDefault();

      // Make sure NFC is supported
      if (device != null)
      {
        MessageBox.Show("NFC present");
        long Id = device.SubscribeForMessage("Windows.SampleMessageType",
messageReceived);
        MessageBox.Show("Published Message. ID is: " + Id);

        // Store the unique message Id so that it
        // can be used to stop subscribing for this message type
      }
      else
        MessageBox.Show("Your phone has no NFC or NFC is disabled");
    }

    private void messageReceived(ProximityDevice sender, ProximityMessage message)
    {
      MessageBox.Show("Received from: " + sender.DeviceId + Environment.NewLine +
message.DataAsString);

      AppToApp(message.DataAsString);

    }

    private async void AppToApp(string message)
    {
```

```
        // PeerFinder.Start() is used to advertise our presence so that peers can find us.
        // It must always be called before FindAllPeersAsync.
        PeerFinder.Start();

        try
        {
            var pairedDevices = await PeerFinder.FindAllPeersAsync();

            if (pairedDevices.Count == 0)
            {
                Debug.WriteLine("No paired devices were found.");
            }
            else
            {
                MessageBox.Show("Connect with device " + pairedDevices[0].DisplayName);

                // Select a paired device. In this example, just pick the first one.
                PeerInformation selectedDevice = pairedDevices[0];
                // Attempt a connection
                StreamSocket socket = new StreamSocket();
                // ID_CAP_NETWORKING must be enabled in WMAppManifest.xml, or the next
                // line will throw an Access Denied exception.
                // In this example, the second parameter of the call to ConnectAsync() is the RFCOMM
port number, and can range
                // in value from 1 to 30.
                await socket.ConnectAsync(selectedDevice.HostName, "1");
                DataWriter dataWriter = new DataWriter(socket.OutputStream);

                dataWriter.WriteString(message);
                await dataWriter.StoreAsync();
            }
        }
        catch (Exception ex)
        {
            if ((uint)ex.HResult == 0x8007048F)
            {
                MessageBox.Show("Bluetooth is turned off");
            }
        }

    }
  }
}
```