TRABAJO DE FIN DE GRADO

Título:	Desarrollo de Sistema de entrega de contenido habilitado con Beacons para pacientes con Alzheimer equipados con Google Glass
Título (inglés):	Development of a Beacon enabled Content Delivery System for Alzheimer's patients equipped with Google Glass
Autor:	Daniel Paniagua Caro
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Mercedes Garijo Ayestarán
Vocal:	Carlos Ángel Iglesias Fernández
Secretario:	Álvaro Carrera Barroso
Suplente:	Juan Fernando Sánchez Rada

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE GRADO

DEVELOPMENT OF A BEACON ENABLED CONTENT DELIVERY SYSTEM FOR ALZHEIMER'S PATIENTS EQUIPPED WITH GOOGLE GLASS

Daniel Paniagua Caro

Enero de 2016

Resumen

Esta memoria es el resultado de un proyecto cuyo objetivo es desarrollar e implementar un sistema de distribución inteligente de contenidos para pacientes de Alzheimer, que les ayudará a ser más independientes y mejorar su memoria. Para lograr esto vamos a utilizar dos técnicas de estimulación cognitiva imágenes mentales y orientación de la realidad.

El escenario se desarrolla en un hogar inteligente, que se basa en la proximidad proporcionada por Estimote beacons y donde el paciente debe estar equipado con Google Glass. En función de su ubicación, así como su perfil semántico el sistema seleccionará y entregará el contenido a los portadores de las Google Glass. Por otra parte, los cuidadores y familiares pueden gestionar la información que se le muestra a los pacientes a través de un servidor de administración de contenido. Además, se pueden definir reglas adaptables que utilizan tecnologías semánticas en este servidor.

El sistema ha sido validado en varios escenarios de e-Health para pacientes con Alzheimer. En esos escenarios el sistema permite al paciente tener tanto una orientación temporal como espacial, el paciente podrá detectar objetos cercanos y las gafas le mostrará información sobre los familiares que se encuentren cerca. En este caso las gafas también usará imágenes autorreferenciales donde se le mostrarán fotos antiguas en las que aparezca junto a ese miembro de la familia. Otro escenario evaluado es aquel en el que la aplicación ayuda al paciente a recordar tareas diarias y proporcionar advertencias sobre posibles riesgos potenciales. El último escenario evaluado es el caso de incontinencia, donde el paciente recibirá un mensaje avisando que debe ir al baño si no ha ido durante un período de tiempo que los cuidadores o familiares previamente han indicado en el servidor de automatización de tareas utilizando reglas semánticas, y la familia recibirá un correo electrónico notificando la situación.

Se presentan finalmente las conclusiones extraídas de este trabajo así como las posibles líneas de acción para mejorar el trabajo en el futuro.

Palabras clave: Google Glass, Alzheimer, Beacons, GDK, Live Card

Abstract

This thesis collects the result of a project whose objective is to develop and deploy a smart context aware content delivery system for Alzheimer's patients, which helps them increase their independence and improve their memory. To achieve this we will use two cognitive stimulation techniques reality orientation board and mental imagery.

The scenario unfolds in a intelligent home, which is based on proximity aware Estimote beacons and where the patient should be equipped with Google Glass. Based on their location as well as their semantic profile, they system will select and deliver content to their Google Glass wearables. Furthermore, the caregivers and family can manage the information to be shown to Alzheimer's patients through a Content Management Server. In addition, they can define adaptable rules using semantic technologies from Task Automation Server.

The system has be validated in various e-Health scenarios for Alzheimer's patients. In these the system enables temporal and spatial orientation, the patient can detect nearby items and the Glass displays information about their relatives if they are in the proximities. Furthermore, it shows self-referential imagery using old pictures where the patient is next to the family member. The application also help the patient to remember routine task and provide warning about potential risks. The last scenario evaluated is incontinence case, where the patient receives message that should go to the bathroom if he has not gone during the period of time indicated by caregivers using semantic rules in the server, and family receive an email notifying the situation.

Finally, we present the conclusions of the work and the possible future work that could be done in order to improve the project.

Keywords: Google Glass, Alzheimer, Beacons, GDK, Live Card

Agradecimientos

A mis padres

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

Alzheimer's disease (AD) is a progressive neurodegenerative disorder that gradually destroys one's ability to learn, reason, and carry out daily activities. Most people who develop AD do so after the age of 65, but people under this age can also develop it. In 2015, 46.8 million people worldwide suffer from AD [17], and that number is expected to increase drastically in the coming years, reaching 74.7 million in 2030 and 131.5 million in 2050. The total worldwide cost of dementia in 2015 are estimated at $818 billion, an increase of 35.4% compared with $604 billion in 2010. The ability to live independently can be compromised in people with dementia, and it a reason commonly for nursing home placement. This is a high cost to families, delaying nursing home placement only one month as a result $4 billion savings, according to a study [6]. Therefore, it is necessary to find ways to improve memory or to develop strategies that will help people compensate for memory loss and live at home or in the community longer.

In our project we wanted to develop a system to help Alzheimer's patients. First of all, we have keep it in mind that AD has different stages, it is important to plan appropriate care. Although, doctors can diagnose a disease with five, six or seven levels, we will provide an overview of the the three stage AD model:

Mild or Early stage, in this stage a person may function independently. They have frequent recent memory loss, such as forgetting familiar words or the location of everyday objects. They need reminders for daily activities.

Moderate or Middle stage, it is typically the longest stage. They suffer change in the personality and behavioral. Moreover, the memory loss is pervasive and persistent, including forgetfulness of events or about personal history and inability to recognize friends and family. They need structure, reminders, and assistance with the activities of daily living.

Severe or Late stage, individuals lose the ability to remember, communicate, or process information. Generally they may still say words or phrases, but communicating pain becomes difficult. Physical abilities also continue to worsen. Behavior changes, hallucinations, and delirium are increased. In this stage, the person will need round the clock intensive support and care.

We will follow two different strategies that are focused on the first two states. On the one hand, increasing the independence of patients with simple reminders or practices that could not be done without the help of relatives or nurses. We have relied on the guidance for the care of Alzheimer's patients [26]. On the other hand, we wanted to improve memory and one possible strategy that has shown benefit to healthy adults in the cognitive aging literature is using mental imagery at the time of encoding. Mental imagery and cognitive training can lead to considerable improvements in memory, which they are reliable and can be maintained over time. Furthermore, researches [6] have demonstrated that self-referential imagery is a technique which has particular benefit. However, there are no studies showing improved memory encoding in patients with AD using mental imagery. Given that healthy older adults show such robust improvement, these techniques could achieve a slight improvement in the memory of Alzheimer's patients. In our project we use mental images to try to achieve this improvement.

1.2 Project goals

The main goal of this project is to develop and deploy a smart context aware content delivery system for Google Glass users, based on their location as well as their semantic profile, they system will select and deliver content to their Google Glass wearables.

This main goal includes some tasks such as:

- Development of the Google Glass's application to display information using beacons.
- Define adaptable rules for users based on semantic technologies.
- Design and develop a content manager that allows users to manage the content of Google Glass patient's.
- Evaluate the architecture in a practical case of e-Health.

1.3 Structure of this document

In this section we will provide a brief overview of the chapters of this document. It has been structured as follows:

Chapter 1 presents the problem that we aim to solve, and describes the structure of this document. *Chapter 2* describes the available technologies that we have used during the progress of the project. *Chapter 3* explains the complete architecture, built it in different modules that are deeply explained in the chapter. *Chapter 4* provides an overview of the use cases. *Chapter 5* sums up the conclusions obtained as a result of the project and gives some future work that could be done to improve the features.

CHAPTER 2

Enabling Technologies

In this chapter we explain the different technologies that we have used during the progress of the project. Starting from a description the Google Glass device and its development features in section 2.1. To finish, the Beacon device and its development tools in section 2.2.

2.1 Google Glass

Google Glass [22]: is a wearable computer developed by Google, its operating system is based on a version of Android and it support as well as Bluetooth and Wi-Fi connectivity. The device looks like a pair of eyeglasses but it offers an augmented reality experience. To achieve this, the Glass is supported by some of the following components: speakers, GPS, small screen to display information using an optical head-mounted, front camera to take photos or record a video for scholars and allows users to interact with Glass, wearers can utilize natural language voice commands using the microphone or via gestures using the touchpad located on the right side of the Google Glass.

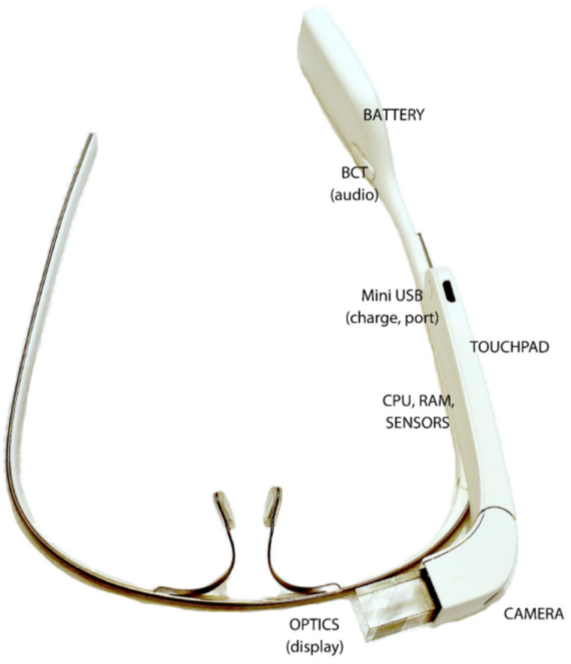


Figure 2.1: An overhead view of Glass.

In spite of the Google Glass is similar to a smartphone, its user experience is dramatically different. Consequently Glass user interface has to change too and Google has designed the Timeline, which is the main user interface that is comprised of 640x360 pixel cards that you can swipe forward and backward to view chronological list of cards because timeline and the home card, which is a default card that the Glass shows you when your turn it on.

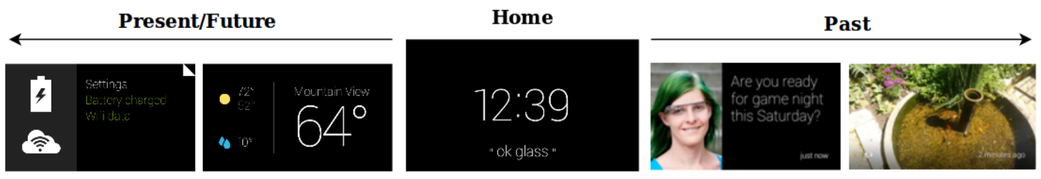


Figure 2.2: Timeline

In other words, timeline is a line of cards and we can design different types of cards essentially static cards, live cards and immersions. These cards are used by different types of applications, which on Glass are called Glassware and we found 3 types: Mirror API, Glass Development Kit and Hybrid that we will be explained in the following sections in more detail.

2.1.1 Mirror API

The Mirror API [21] allows you to build web-based services that interact with Google Glass. Mirror API provides functionality over a cloud-based API and does not require running code on Glass and you can write the code in a variety of programming languages, such as Java, PHP and Python. Furthermore, Mirror API has many services fall into a few categories of API. Mirror API features allow us to insert, modify and delete static cards. Moreover, it also allow you to observe the user's location in timeline items, request their last known location. In addition, you can insert, modify and delete static cards, or push it to Google Glass [12].

2.1.1.1 Static Cards

Static cards are used to display information relevant to the user at time of delivery, this information does not change at all or does not change frequently. The static cards reside to the right of the Glass clock by default, when Glassware inserts it into the timeline all previous static cards shift to the right too, in the next figure we can see [14].

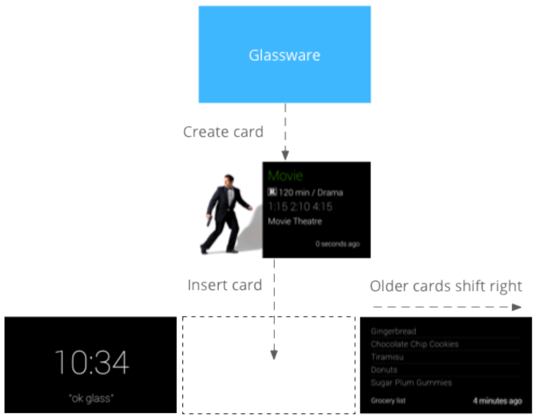


Figure 2.3: Static card timeline

2.1.2 Glass Development Kit (GDK)

Another option to build Glassware is Glass Development Kit (GDK), which is an add-on to Android SDK that runs directly on Glass as independent apps. It's worth mentioning that the GDK is currently in an early release that is subject to changes, withdrawals, and additions at any time. GDK give us two ways to design Glassware that we will be explained in the next section.

2.1.2.1 Immersions

Immersions are Glassware that are independent of the timeline experience, which run on Glass as independent apps. When the application runs and takes over the current screen, you can not continue to navigate the timeline, but rather, giving you complete control over the user experience from the time Glass launches the immersion [10].

In contrast to Mirror API, you have to write the code in the native programming language. However, Immersions give you more ways to consume user input and create user interfaces. This allows you to create the most custom experience, but involves the most work.

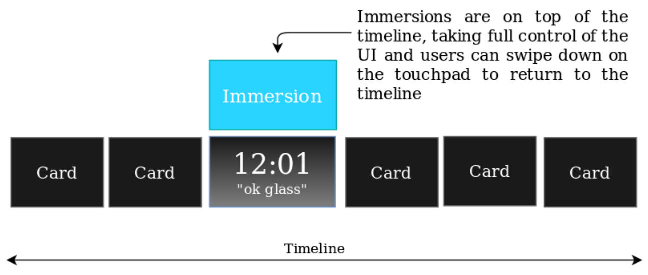

Immersions are on top of the timeline, taking full control of the UI and users can swipe down on the touchpad to return to the timeline
Immersion
Card
Card
12:01
"ok glass"
Card
Card
Card
Timeline

Figure 2.4: Immersion timeline

2.1.2.2 Live Cards

Live Cards are akin to static card, except they show real-time information that are relevant at current time or at a future time rather than a point in the past. Moreover, live cards can change in real time. In the timeline they appear in the present section, left of the Glass clock. Unlike immersions you can always swipe through the timeline to other cards. Live cards are great for when users are actively engaged in a task, but want to periodically check Glass for supplemental information, such as a display that shows the running status of an action, an animated map during navigation, or a music player [11]. In addition, you have again two options to render these cards:

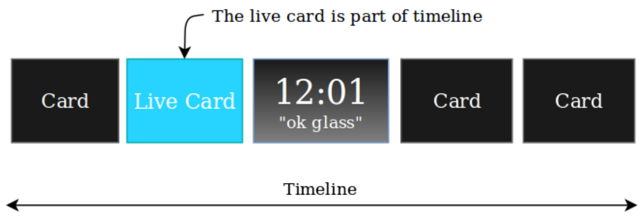


Figure 2.5: Live card timeline

2.1.2.2.1 Low Frequency

Low frequency is a simple way to create live cards with simple content, the card is rendered using Remote View, which is limited to a small set of Android views and can only update the display once every few seconds. A background service is responsible for this updating.

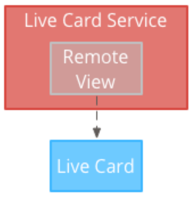


Figure 2.6: Live card low frequency process

2.1.2.2.2 High Frequency

High frequency rendering lets you draw directly on the backing Surface of the live card. You can draw anything and you can update the card many times a second. The system gives you the actual backing surface of the live card that you draw directly onto using 2D views and layouts or even complex 3D graphics with OpenGL.

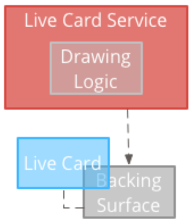


Figure 2.7: Live card high frequency process

2.1.3 Hybrid Glassware

Hybrid Glassware uses both Mirror API and GDK. Mirror API Glassware can invoke GDK Glassware through a menu item. This model can be used to leverage existing web properties that you can launch fuller experiences that you can run directly on your Glass [13].

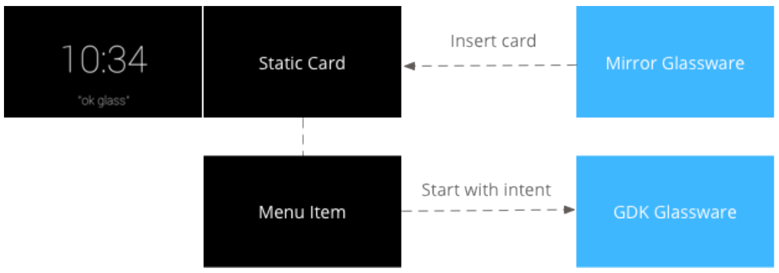


Figure 2.8: Hybrid glassware timeline

2.2 Beacon

Beacon is a small computer which transmits radio signals via Bluetooth low energy (BLE). In the market we can found different types of beacons, in our case we have used Estimote Beacons [7]. Smart devices use beacon's signal to estimate the distance by measuring received signal strength. Estimote Beacons have a range of up to 70 meters radio waves that smart devices can pick up it. Nevertheless, in real world conditions the signal can be diffracted, interfered with, or absorbed by obstacles. Metal and water will have the strongest effect, moreover the human body if made of water, conversely wood, synthetic materials and glass will have the lowest effect, for all these reasons the real range is of about 40-50 meters. All I have to add is that beacon is blinking, it is not broadcasting continuously. The signal detection is more reliable when the more frequent blinks.

2.2.1 Beacon Hardware

Estimote Beacons have installed a piece of low level software called Estimote firmware, that handles tasks like processing sensor data or encrypting a beacon's ID for improved security.

Under the silicone covering, beacons have 32-bit ARM Cortex M0 CPU with 256kB of flash with a built-in 2.4GHz radio supporting both BLE as well as 2.4GHz operation [24]. Current version of Estimote Beacons have 1000mAh CR2477 coin battery, which can last more than 3 years on default settings and about 6 months on maximum power settings.

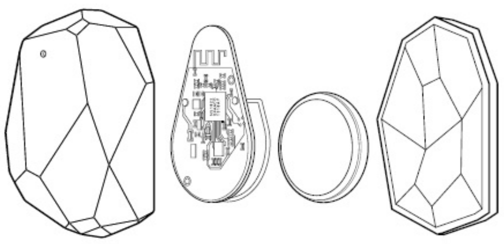


Figure 2.9: Estimote Beacon

Beacons include both a temperature sensor and an accelerometer. Moreover, we can found the antenna on the thinner side of the beacon, which is a short wire sticking out of the CPU. It's twisted and looks like a zigzag [2]. Moreover, the antenna broadcasting electromagnetic waves. Where the radio wave's center frequency is 2.4 GHz and the electromagnetic field around a straight wire is shaped like a donut. However, the best field would be a perfectly spherical field, because the waves propagate in every direction with the same strength, unlike our field that leaving out sort of blank areas. But in reality this field is not possible in real world conditions. Consequently, beacon placement and orientation play an important role for optimal signal propagation. Estimote experiments have determined that the most effective placement is vertical, with the little dot facing upwards [25].

2.2.2 How do beacon work?

As we have explained before, beacons use BLE, meaning they enable broadcasting only small amounts of data. The maximum payload of a Bluetooth 4.2 packet is 257 bytes. It's not enough to embed media content and that's why beacons only broadcast their identifiers and information about signal power, essential for a nearby smartphone to calculate proximity.

Smart devices do not scan continuously for beacons. As mentioned above, beacons are blinking. Advertising interval describes the amount between each blink. For this reason, smart devices scan for beacons with a given frequency as well, and it may depend on the smart device state. For instance, if its state is active, scans will be happening very often, while if its state is locked for a few minutes, it will start preserving its battery by limiting the number of Bluetooth scans.

Advertising interval has the value ranges between 100 and 2000 ms, and It is set to 950 ms by default. This interval can be adjusted with Estimote SDK, Cloud or app.

Another important feature of the beacons is the broadcasting power. It is the power with which the beacon broadcast its signal. A beacon's range derives directly from the broadcasting power. It is measured in dBm and the value ranges between -30 and +4 dBm. Just as with Advertising Interval, Broadcasting Power on beacons can be changed with Estimote SDK, Cloud or app.

Calculating proximity

The predictor of distance is based on Received Signal Strength Indicator (RSSI), It is the strength of the beacon's signal as seen on the receiving device. Furthermore, If we want to calculate the distance, we need to know the Measured Power, which indicates what is the average RSSI received at 1 meter distance. Therefore, smart devices can estimate the distance to the beacon by comparing the actual signal level with the expected signal level at 1 meter.

2.2.3 Protocols and SDKs

Beacons can use different protocols and SDKs that they help us to achieve bigger yield in beacon applications. We will explain what Estimote Beacons can use below.

2.2.3.1 iBeacon

iBeacon is a Bluetooth 4.0 communication protocol, which has developed by Apple and It has official support for iOS devices only [9]. Estimote Beacon broadcast packets of data, which contains information about signal strength and universally unique identifier, iBeacon ID, it is 20 bytes long and slits into three parts.

- **UUID** (16 bytes) is standard identifying system which allows a unique number to be generated for a device. The purpose of the ID is to distinguish iBeacons in your network, from all other beacons in networks outside your control. most commonly represented as a string.
- **Major** (2 bytes) and **Minor** (2 bytes) are number assigned to your iBeacons, in order to identify them with higher accuracy than using UUID alone. They are unsigned integer values between 1 and 65535, 0 is a reserved value.

We can define different beacon regions [3] using a specific combination of UUID, Major or Minor. These values can be changed with Estimote SDK, Cloud or app, just as with Advertising Interval and Broadcasting Power. There are two methods provided by iBeacon to interact with regions and beacons within these regions.

- **Monitoring** is the action triggered on entering or exiting region's range of one or more beacons. It works in the foreground, background, and even if the app is not running.
- **Ranging** actively scans for any nearby beacons and provides proximity estimations every second. It works only when the app is active in the foreground.

2.2.3.2 Eddystone

Eddystone is an open Bluetooth 4.0 communication protocol. It allows developers to create applications for iOS and Android devices [8]. Eddystone supports multiple data packet types although Estimote Beacons can only broadcast one type of packet at a time. We will describe them below.

- **Eddystone-UID** packet contains an identifier of a beacon, which is 16 bytes long and is divided into two parts:
  - Namespace (10 bytes) is intended to ensure ID uniqueness across multiple Eddystone implementers and may be used to filter on-device scanning for beacons. It has a default namespace set to: *EDD1EBEAC04E5DEFA017*. If you want to use a different ID, the Eddystone specification recommends taking the first 10 bytes of an SHA-1 hash of your domain name, or generate a version 4 UUID then remove bytes 5 to 10.
  - Instance (6 bytes) is unique within the namespace and represented as a string up to 12 characters long. It is used to differentiate between your individual beacons.
- **Eddystone-URL** frames a broadcast a single field that contains a URL. The size of the field depends on the length of the URL compressed encoding format used. The URL can be used to providing access to a web page, which contains relevant information.
- **Eddystone-TLM** contains the telemetry data about beacons operation. These data are useful for monitoring the health and operation of a fleet of beacons.
  - Battery voltage, which is the current battery charge in millivolts.
  - Beacon temperature is the temperature in degrees Celsius sensed by the beacon.
  - Number of packets sent since the beacon was last powered-up or rebooted.
  - Beacon uptime, for example, time since last power-up or reboot.

2.2.3.3 Estimote Indoor Location SDK

Estimote Indoor Location SDK [23] is a set of tools, which enables real-time beacon-based mapping and indoor location. The way to create the application is attaching at least one Estimote Beacons per wall, then the location's map is automatically uploaded to Estimote Cloud. It allows users for:

- Embed a built-in Location View, you can create a map of your location with your current position marked on it.
- The Location View is configurable, which allows one to customize its appearance, show and hide labels or leave a trace of your movement.
- Set up a location manually, you can use the coordinates to draw your own room.

- Get raw positioning data , your coordinates and orientation.
- Save and load locations to and from Estimote Cloud.

2.2.3.4 Estimote Android SDK

In our project we have used the Estimote SDK for Android [19]. It is a library that allows users to interact with our Estimote beacons. Android 4.3 or above and device with BLE are required to use this library. The SDK enables you to:

- Beacon ranging, your device is being triggered continuously and gets you a list of beacons discovered.
- Beacon monitoring, your device is triggered whenever you cross the boundary of the previously defined region in your beacons.
- Eddystone scanning, as explained above there are different types Eddystone packages which can be scanned.
- Nearables discovery, it allows using stickers which is akin to tiny beacon, but we have not used in this project.
- Beacon characteristic reading and writing: broadcasting power, advertising interval, proximity UUID, major and minor values.
- Easy way to meet all requirements for beacon detection: runtime permissions or acquiring all rights.

2.2.4 Estimote Cloud

Estimote Cloud [20] is a web-based platform that helps you better control your beacon network. It allows you to remote management options and geolocation. Moreover, Estimote Cloud have specifications as App Templates which enables users to generate themselves apps powered by the Estimote stack from predefined templates. Estimote Cloud is also the backend responsible for:

- Cloud API, which enables you accessing your beacon fleet data in a programmatic, machine-friendly way.

- Cloud-Based authentication protecting beacons from unauthorized access. You need to be logged in to your Estimote Account and Estimote Cloud needs to authenticate you as the owner.
- Segure UUID, preventing piggybacking on your beacon network. It causes rotation of the beacon's ID so it's broadcasting unpredictable, encrypted values.
- Analytics help track beacon activity across your apps. You can measure interaction with beacon regions or individual beacons, number of visits and unique visitors.

2.3 Rule Automation

In this project, we are going to use a Semantic Rule Task Automation Engine that uses the EWE ontology [5] for rule specification, and has been implemented by Sergio Muñoz López [16]. In the following sections we are going to provide an introduction to the main technologies of the semantic task automation engine used in this project: N3 (Sect. 2.3.1), EYE (Sect. 2.3.2 and EWE (Sect. 2.3.3).

2.3.1 N3

This is a language which is a compact and readable alternative to RDF's XML syntax, but also is extended to allow greater expressiveness. It has subsets, one of which is RDF 1.0 equivalent, and one of which is RDF plus a form of RDF rules. This language has been developed in the context of the Semantic Web Interest Group.

The aims of the language are:

- To optimize expression of data and logic in the same language.
- To allow RDF to be expressed.
- To allow rules to be integrated smoothly with RDF.
- To allow quoting so that statements about statements can be made.
- To be as readable, natural, and symmetrical as possible.

The language achieves these with the following features:

- URI abbreviation using prefixes which are bound to a namespace (using @prefix) a bit like in XML.

- Repetition of another object for the same subject and predicate using a comma ",".
- Repetition of another predicate for the same subject using a semicolon ";".
- Bnode syntax with a certain properties just put the properties between [and].
- Formulae allowing N3 graphs to be quoted within N3 graphs using and .
- Variables and quantification to allow rules, etc to be expressed.
- A simple and consistent grammar.

2.3.2 EYE

Euler Yet another proof Engine (EYE) [15] is is a high-performance reasoning engine that uses an optimized resolution principle, supporting forward and backward reasoning and Euler path detection to avoid loops in an inference graph. It is written in Prolog and supports, among others, all built-in predicates defined in the Prolog ISO standard. Backward reasoning with new variables in the head of a rule and list predicates are a useful plus when dealing with OWL ontologies, so is more expressive than RDFox or FuXi, whilst being more performant than other N3 reasoners.

Internally, EYE [16] translates the supported rule language, N3, to Prolog Coherent Logic intermediate code and runs it on YAP (Yet Another Prolog) engine, a high performance Prolog compiler for demand-driven indexing. The inference engine supports monotonic abduction-deduction-induction reasoning cycle. EYE can be configured with many options of reasoning, such as not providing false model, output filtering, and can also provide useful information of reasoning, for example, proof explanation, debugging logs, and warning logs. The inference engine can be added new features by using user-defined plugins.

2.3.3 EWE Ontology

Evented Web Ontology (EWE) is a standardized data schema (also referred as “ontology” or ”vocabulary”) designed to describe elements within Task Automation Services [4] enabling rule interoperability. Referring to the EWE definition, the goals of the EWE ontology to achieve are:

- Provide a common model to represent TAS’s rules so that it enables rule interoperability.
- Enable to publish raw data from Task Automation Services (Rules and Channels) online and in compliance with current and future Internet trends.
- Provide a base vocabulary for building domain specific vocabularies e.g. Twitter Task Ontology or Evernote Task Ontology.

2.3.3.1 EWE Ontology main classes

The ontology has four main classes: Channel, Event, Action and Rule. They are briefly explained following.

- Channel: It defines individuals which either generates Events, provide Actions, or both. In this context, Channel can define a Web service or a device thus these last ones provides Events and Actions [1].
- Event: This class defines a particular occurrence of a process. Events are instantaneous: they have no duration over time. Event individuals are generated by a certain Channel, and they are triggered by the occurrence of the process which defines them. Events usually provide further details that can be used within Rules to customize Actions: they are modelled as output-parameters. Events also let users describe under which conditions should they be triggered. These are the configuration parameters, modelled as input-parameters. Event definitions are not bound to certain Channels since different services may generate the same events.
- Action: This class defines an operation or process provided by a Channel. Actions provides effects whose nature depend on itself. These include producing logs, modifying states on a server or even switching on a light in a physical location. By means of input-parameters actions can be configured to react according to the data collected from an Event. These data are the output-parameters.

- Rule: The class Rule defines an "Event-Condition-Action" (ECA) rule. This rule is triggered, and means the execution of an Action. Rules defines particular interconnections between instances of the Event and Action classes; those include the configuration parameters set for both of them: output from Events to input of Actions.

2.3.4 MySQL

MySQL [18] is an open source relational database management system (RDBMS) based on Structured Query Language (SQL). MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language.

CHAPTER 3

Architecture

In this section, we will explain the architecture of this project. In the first place, we will evaluate several alternative design architectures. In the second place, we will explain an overview of the full architecture of the project itself, structured into different modules that are deeply explained at the end.

3.1 Alternative architectures

In the previous chapter we described the different ways to build applications on Google Glass. On the whole, we can build web-based services that interact with Google Glass or Glassware that run directly on them as independent apps.

On the one hand, we can use the Mirror API. A number of applications are installed and operate exclusively on a remote server. It can be used if you need:

- Platform Independence, develop outside the Glass on a server in any supported language.
- Built-in functionality.
- Common infrastructure.

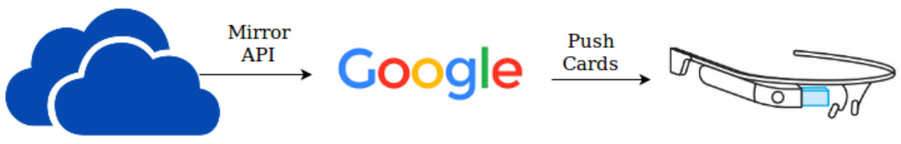


Figure 3.1: Mirror API architecture.

On the other hand, GDK Glassware applications are installed on the Glass device itself. Since Mirror API apps can only update the timeline when an Internet connection is established, the GDK allows you to create applications that don't require wireless access.

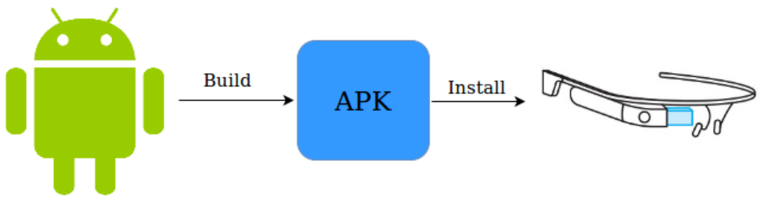


Figure 3.2: GDK architecture.

Since we want to create an application that does not require continuous Internet connection, we need to use GDK to create applications. The benefits of this is a greater independence, the application does not need neither network connection or google services to run. In addition, content delivery is faster than Mirror API.

As we explained in Chapter 2, GDK gives us two ways to design Glassware: immersion and live cards. In our project we want real-time interaction with users and free navigation through the timeline while the application works. Therefore, live cards are well suited for these characteristics, because when using them, the timeline has still control over the user experience, so swiping forward or backward on a live card navigates the timeline instead acting on the live card itself. In addition, they allow real-time updates to the user interface.

There are two types of live cards, specifically we will use high frequency rendering instead of low-frequency rendering, because we want the highest speed warning alerts falls and it allows updating the live card many times a second.

Since users are caregivers in our scenario, it is desirable that they could customize the content delivered by the application to the patients. Nevertheless, the proposed architecture only enables them to change that information directly on the glasses. Consequently, we will design and implement an content management server to manage contents through a web interface.

In addition, we extend the described functionality with a model for task automation, which allows users to define adaptive rules based on semantic technologies.

3.2 Overview

The overall architecture of this project is composed of the following modules:

- **Content Management Server:** We need software module to manage the content of Google Glass users.
- **Task Automation Server:** [**16**] This server will execute tasks on behalf of a user, based on the defined automation rules.
- **GDK app:** The Google Glass native application that the patient will use as a memory tool. This module is split into 4 sub-modules:
  - Ranging
  - Content personalization
  - Controller
  - Live Card

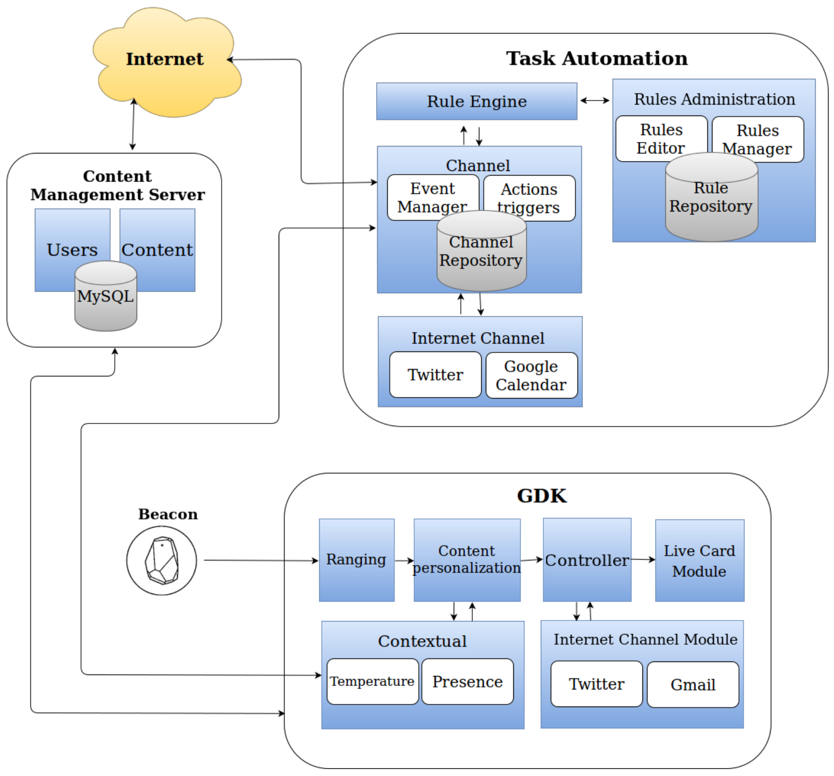


Figure 3.3: Architecture.

3.3 Content Management Server

One of the applications of beacons is to provide contextual information about the object the user is next to. For this purpose, a number of content management systems (CMS) have been developed for providing information about products in a retail scenario. In our case, we are going to develop a CMS for managing the information to be shown to Alzheimer's patients with the aim of improving its memory about relatives and objects or providing warning about potential risks. CMS has been implemented using different technologies: HTML5, CSS, JavaScript and PHP.

3.3.1 Database

Google Glass does not provide an interface for storing images, so we have decided to use MySQL to store and access images. MySQL allows us to use blob data type, consequently we can work with the images. The caregivers or family access to the CMS where they create the data which the patient uses in the Glass. The data in the database are downloaded by the Glass using JSON, as shown below.

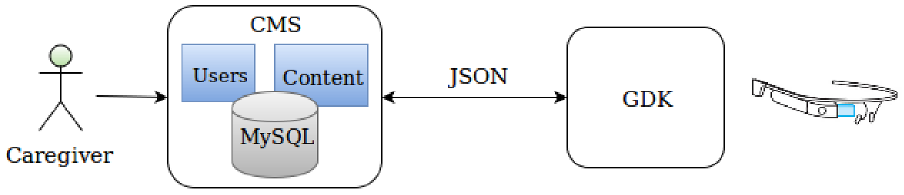


Figure 3.4: CMS interconnection with GDK

A schema representing the structure of the different tables inside this database is shown in Table 3.1, Table 3.2 and Table 3.3.

Field	Type	Null	Key	Default	Extra
minor	int(11) unsigned	No	Pri	Null	Auto increment
major	int(11)	No		1	
description	varchar(50)	No		null	
activatedWhen	varchar(50)	No		null	
image	blob	No		null	
pathImage	varchar(100)	No		noPicture	

Table 3.1: Warnings table.

Field	Type	Null	Key	Default	Extra
minor	int(11) unsigned	No	Pri	Null	Auto increment
major	int(11)	No		2	
name	varchar(50)	No		null	
relationship	varchar(50)	No		null	
profilePicture	blob	No		null	
pathProfilePicture	varchar(100)	No		noPicture	
oldPicture1	blob	No		null	
pathOldPicture1	varchar(100)	No		noPicture	
description1	varchar(50)	No		null	
oldPicture2	blob	No		null	
pathOldPicture2	varchar(100)	No		noPicture	
description2	varchar(50)	No		null	
oldPicture3	blob	No		null	
pathOldPicture3	varchar(100)	No		noPicture	
description3	varchar(50)	No		null	

Table 3.2: Relatives table.

Field	Type	Null	Key	Default	Extra
minor	int(11) unsigned	No	Pri	Null	Auto increment
major	int(11)	No		3	
name	varchar(50)	No		null	
image	blob	No		null	
pathImage	varchar(100)	No		noPicture	

Table 3.3: Items table.

3.3.2 Graphic Interface

In the following photos we can see the user interface from the beginning section, where the user has to enter their credentials to the different sections where the user manages the content.

This is the section where the user need to login to access the web content.

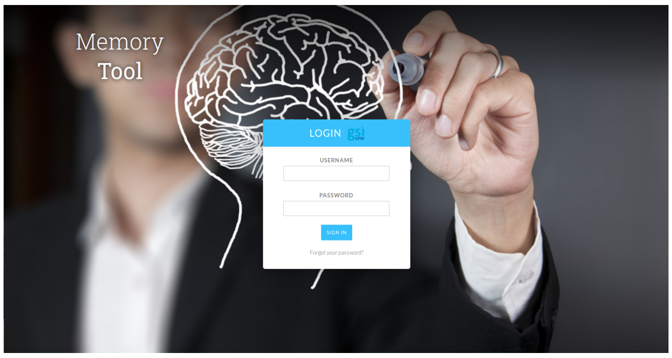


Figure 3.5: Login web page.

When the user has logged in, he will be redirected to homepage and there he can access all sections. It is shown in Figure 3.6.

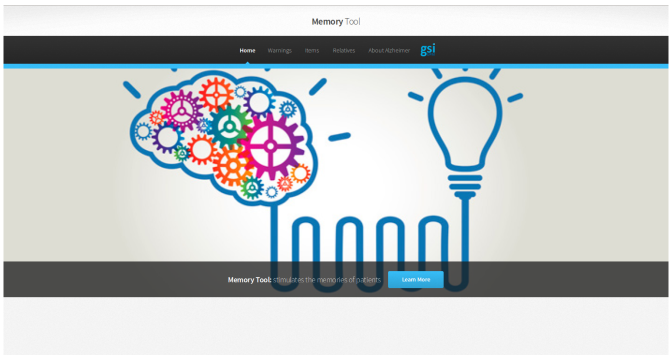


Figure 3.6: Home page.

Then, users can register items as shown in Figure 3.7. Each registered item is assigned a beacon, then it should be placed in the item. He must fill the form where he needs indicate the name and attach a photo of the item.

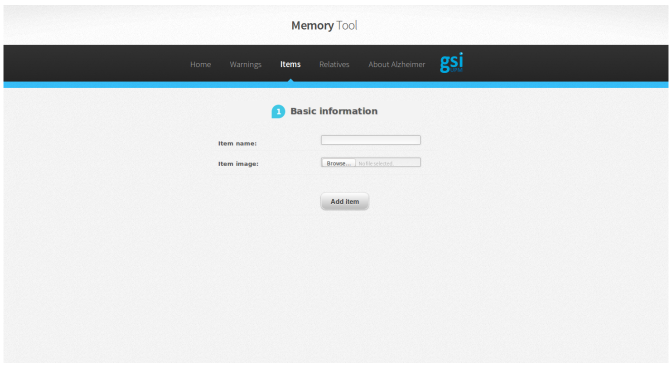
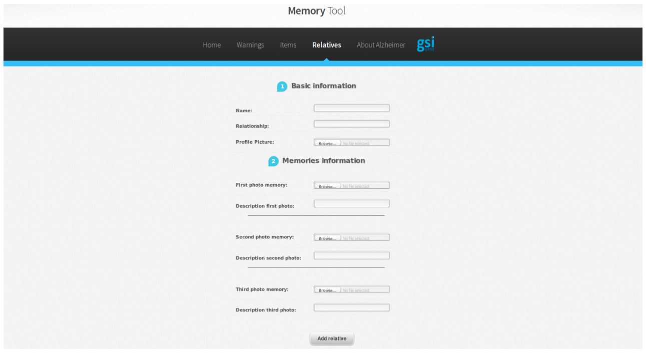


Figure 3.7: Form items.

Another interesting feature of the application is its ability to improve the ability of patients to remember their relatives. With this purpose, cards of relatives can be created so that they are shown when they are with their relatives. For this, their relatives should wear a personalized beacon. The form to edit relative information is shown in Figure 3.8.



1 Basic information

Name:
Relationship:
Profile Picture: Browse... No file selected.

2 Memories information

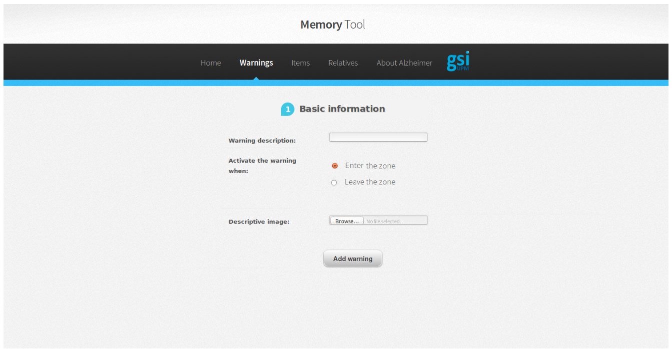
First photo memory: Browse... No file selected.
Description first photo:

Second photo memory: Browse... No file selected.
Description second photo:

Third photo memory: Browse... No file selected.
Description third photo:

Figure 3.8: Form relatives.

Finally, patients can also receive warnings in case of risk. This risk can be detected in some cases such as falls based on Google Glass sensors. The following figure shows in Fig. 3.9.



1 Basic information

Warning description:
Activate the warning when: [✓ Enter the zone] Leave the zone
Descriptive image: Browse... No file selected.

Figure 3.9: Form warnings.

3.4 Task Automation Server

The task automation server [16] provides a web service to create rules. The first step to create it is the authentication. In this page the user can edit a created rule or create a new. In the following picture we can see the interface to create rules.

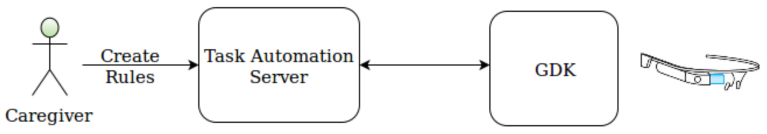
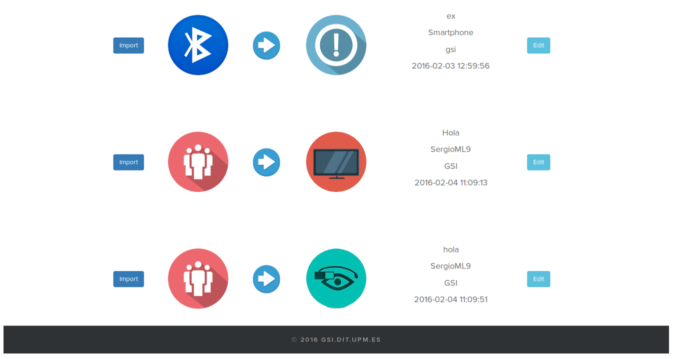


Figure 3.10: Task automation server interaction with GDK



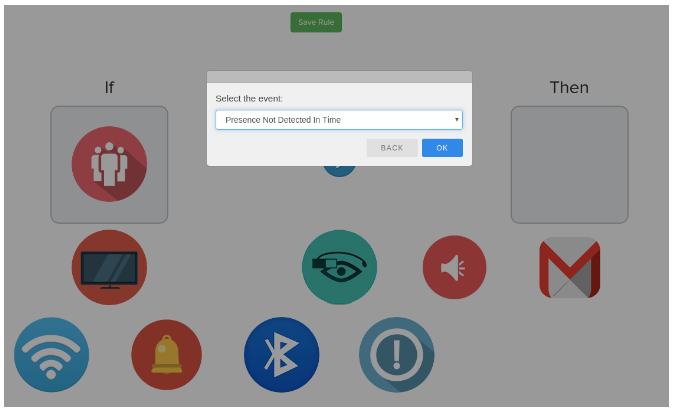
ex
Smartphone
gsi
2016-02-03 12:59:56

Hola
SergioML9
GSI
2016-02-04 11:09:13

hola
SergioML9
GSI
2016-02-04 11:09:51

Figure 3.11: Task automation web

The web allows the user to select between different available channels. This selection is made by dragging the box representing the corresponding channel and dropping it into the available container, then he must chose the action. This simple interface allows users to create their own rules without knowing anything about programming.



Select the event: Presence Not Detected In Time

Figure 3.12: Create rule

We will describe how to create a rule programing, where we send an email to the family if the patient does not go to the bathroom in 180 minutes. In the first place, we have to differentiate between event and action. The event is “When the patient does not go to the bathroom in 180 minutes” and its channel is the presence detection. In the second place, the action is “send email to the family” and its channel is Gmail. We can see this example below.

Listing 3.1: Rule example

```
 @prefix math: <http://www.w3.org/2000/10/swap/math#>.
 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
 @prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/#> .
 @prefix ewe-gmail: <http://gsi.dit.upm.es/ontologies/ewe-gmail/ns/#> .
 @prefix ewe-presence: <http://gsi.dit.upm.es/ontologies/ewe-connected-
     home-presence/ns/#> .

  {
?event rdf:type ewe-presence:PresenceNoDetetedAtTime
?event ewe:sensorID  ?sensorID
?sensorID string:equalIgnoringCase  '1a2b3c' .
?event!ewe:time math:greaterThan 180
  }
  =>
  {
ewe-gmail:Gmail rdf:type ewe-gmail:SendMsg ;
ov:message ``I have not gone to the bathroom in 180 minutes ''.
  }.
```

3.5 GDK app

3.5.1 Structure

In the following section I will explain how the app is structured and what are the modules, methods and algorithms used in the Glass to achieve our goals.

3.5.1.1 Ranging

This module is responsible for detecting all the beacon in range. The library that we use to achieve this is Estimote SDK for Android. Specifically we use beacon ranging, and the startRanging method of the BeaconManager class to determine relative proximity of beacons in the region. In addition, we use the setRangingListener method of the BeaconManager class because we want update every second a list of currently found beacons with an estimated proximity to each of them. It works in the foreground.

Listing 3.2: Code fragment example

```
    beaconManager.setRangingListener(new BeaconManager.RangingListener()
        {
@Override
public void onBeaconsDiscovered(Region region, final List<Beacon> beacons
    ) {
    mapEstimoteBeacons(beacons);
  }
```

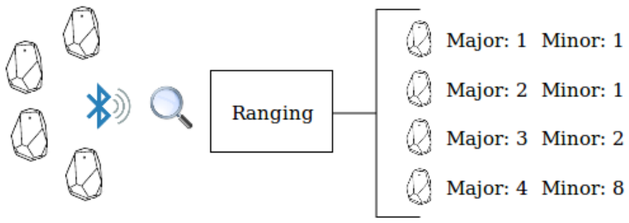


Figure 3.13: Ranging module.

3.5.1.2 Content personalization

This module is divided in two parts. On the one hand, we manage the content to be displayed depending on beacon identifiers. Ranging module provides us list of beacon identifiers. As we explained in the previous chapter, beacons have three identifiers but we only will use two: major and minor. Major will be used to identify the type of beacon and the minor will be used to specify exactly what it is. The table below clarify this.

Major		Minor	
1	Warnings	1	stove
		2	fall
2	Relatives	1	son
		2	wife
3	Items	1	shoes
		2	shirt
4	Places	1	kitchen
		2	living room
		3	dining room
		4	bathroom
		5	hall
		6	yard
		7	garage
		8	bedroom

We have defined four types of beacons: warnings, items, people and places. The first three can be updated by adding new elements in those categories, but in the category places we have predefined values that are shown in the table. This category is small enough to be defined with the elements we have described.

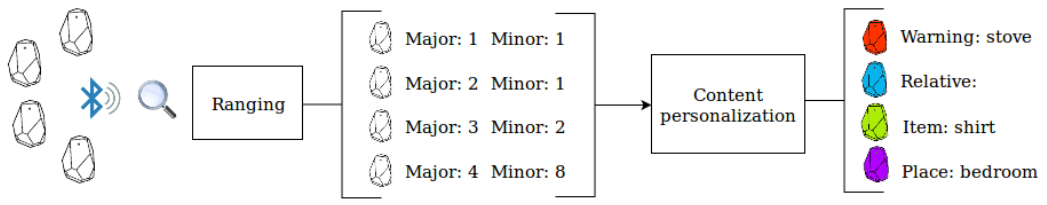


Figure 3.14: Content personalization module.

On the other hand, this module is responsible for sending the request with the parameters required by the rule that we have defined on the task automation server. The content personalization module saves in a global variable the time that has elapsed since no detected the beacon of specific place. In the following example we can see one of the defined rules, this rule has as parameter this elapsed time since no presence is detected in the place that is also sent as a parameter.

Listing 3.3: Input example

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ewe: <http://gsi.dit.upm.es/ontologies/ewe/ns/#> .
@prefix ewe-presence: <http://gsi.dit.upm.es/ontologies/ewe-connected
    -home-presence/ns/#> .

ewe-presence:PresenceSensor rdf:type ewe-presence:
    PresenceNoDetetedAtTime;
  ewe-presence:PresenceSensor ewe:sensorID #SensorPlaceID# ;
  ewe-presence:elapsedTime #Time# .
```

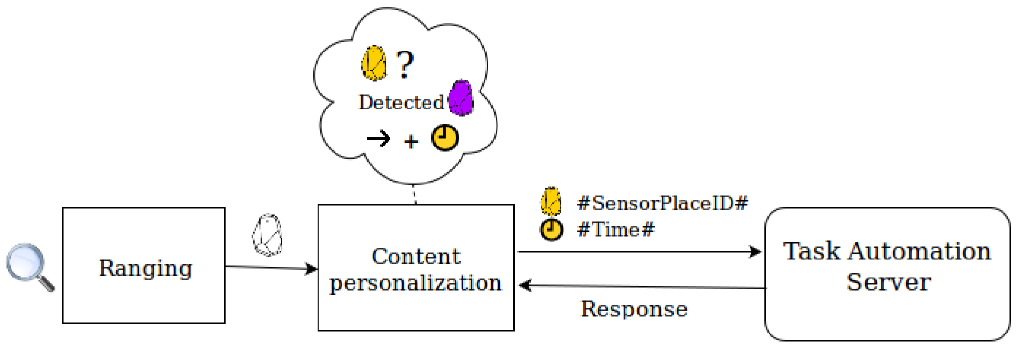


Figure 3.15: Content personalization interaction with Task Automation Server.

Task automation server sends a JSON response. That Gmail and Live Card are the used channels, these were previously selected in the server.

```
[caption={Response}, label={lst: response}],
      {
  "success": 1,
  "actions": [
    {
      "channel":"Live card",
      "action":"Show",
      "parameter":"null"
        },
    {
      "channel": "Gmail",
      "action": "Send",
      "parameter": "Example of message "
    }
  ]
}
```

3.5.1.3 Controller

This module is responsible for receiving the personalized list of currently found beacons, which is sent by the content personalization module.Controller module sends the appropriate view to the live card. The view that we show to the user depends of the priority of beacons found at this moment. In the following table we can see the assigned priorities.

Type view	Priority
Warning	Maximum
Relatives	Medium
Items	Low
Greeting	None

Table 3.4: Items table.

Therefore, if our glass detects a beacon type warning, it will ignore other types of beacons that can be found at this time. Consequently we can only see the warning view in that scenario. But we should not forget that in our scenario more than one beacon of the same type can coexist. In this case, the device displays the data of the beacon that has the highest priority and is closer to the user.

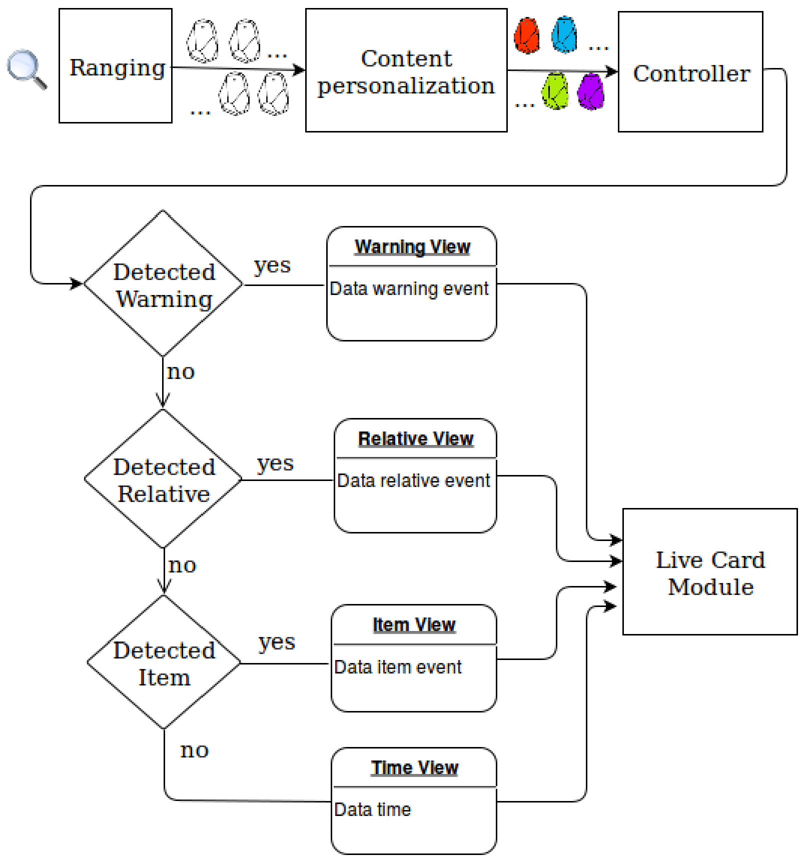


Figure 3.16: Controller module.

We have observed that fluctuations in the beacons occur much frequently, the distance is constantly changing. If we are in a place where there are several beacon of the same type, as a result of fluctuations the view changes so quickly that it is inconvenient for the user. To achieve a stable view, we have established a safety time in each priority level. In this small safety time, the controller checks whether the user's position has changed with respect to the beacon or otherwise if there was a fluctuation.

This module is also responsible for responding to possible events derived from the use of rules, the content personalization module sends us the data channels will manage. In our case we will use three channels: gmail, twitter and live card. The content personalization module also send us the action to perform. These can be display view on the screen, send message or tweet. When the live card is used and we have to display view, it has the same priority that warning beacons.

3.5.1.4 Live Card

We render live cards with high frequency, the system gives us the actual backing surface of the live card that we draw directly onto using 2D views. Firstly we have created the class LiveCardRenderer that implements DirectRenderingCallback, it allows to create a background thread to update the card in response to beacon detection. Secondly, we display graphics with canvas library. For this purpose, we use several methods such as drawLine, drawColor, drawTex and drawBitmap.

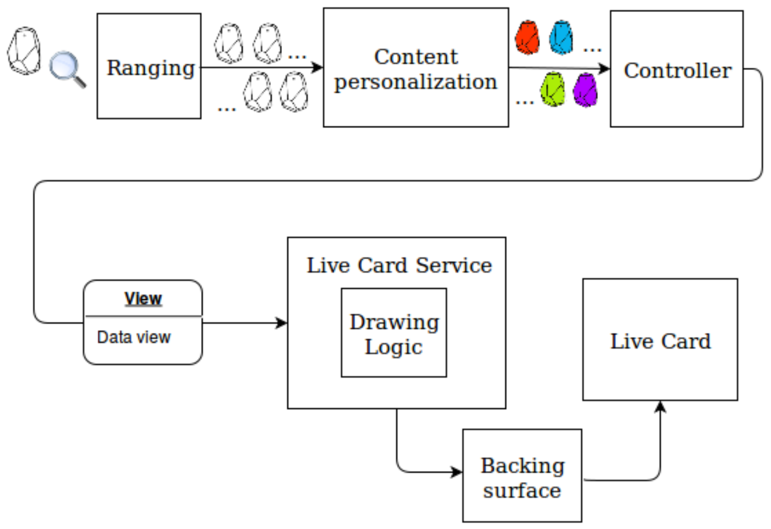


Figure 3.17: Live card module.

3.5.2 Accessing the app

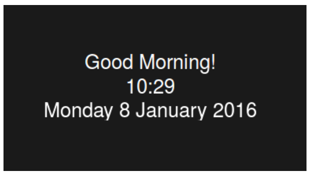
The user can access the app in two different ways:

- **Voice menu:** Google Glass counts with a built-in voice menu that will be displayed if the user says ”ok glass”. Inside that menu our app is shown and is accessible by saying ”Memory tool”.

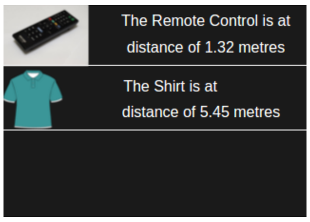
- **Tactile menu:** A menu will appear if the user taps on the touchpad while being in the home card, our app will be one of the options.

3.5.3 User interface and user control

In order to make easy to use the patient, once the application is launched, it displays the information automatically based on proximity aware beacons. The wearers can open the menu tap on the card, but the unique action that they can use is finish the activity. In the following pictures we can see examples the different views that we can see depending on the context.

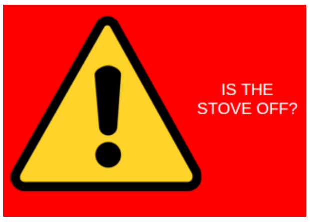

Good Morning!
10:29
Monday 8 January 2016

(a) Time view.

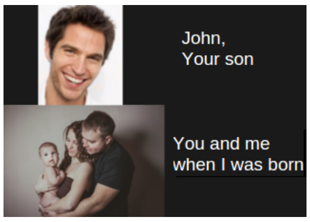

The Remote Control is at distance of 1.32 metres

The Shirt is at distance of 5.45 metres

(b) Items view.


IS THE STOVE OFF?

(c) Warning view.


John,
Your son

You and me when I was born

(d) Relative view.

Figure 3.18: Interface Google Glass.

3.6 Summary

In this chapter we have explained the features and objects that this project has along with their communication and relationship.

To sum up, this project counts with a Google Glass application that is able to receive contextual events coming from beacons and to generate accordingly actions. Furthermore, we have a Content Management Service where the caregivers and family are be able to manage the content of Google Glass users. In addition, they can use Task Automation Server to define rules based on semantic technologies that can be adapted.

CHAPTER 4

Case study

In this chapter we are going to describe different use cases. This description will cover the main Memory Tool features, and its main purpose is to completely understand the functionalities of this project, and how to use it.

4.1 Problem and scenario

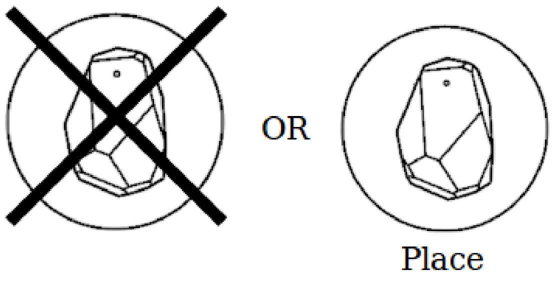
Alzheimer's is a brain disease that causes a slow decline in memory, thinking and reasoning skills. As we explained in the chapter 1, symptoms may also vary depending on the stage of the illness. There is currently no cure for Alzheimer's disease, or treatment to stop its progression or reverse the symptoms. Medications may help on a short-term basis to slow cognitive decline and non drug treatments can help with some symptoms.

With our application we want to increase the independence of patients and improve their memory. To achieve this we will use two cognitive stimulation techniques, on the one hand we will employ reality orientation board, it is used to display both personal and orientation information, the therapy focuses on repeatedly reminding patients of information in order to create continuity between different bits of information mental imagery. On the other hand, we will use mental imagery to improve memory. In the following sections, we will describe all the use cases which cover all this techniques.

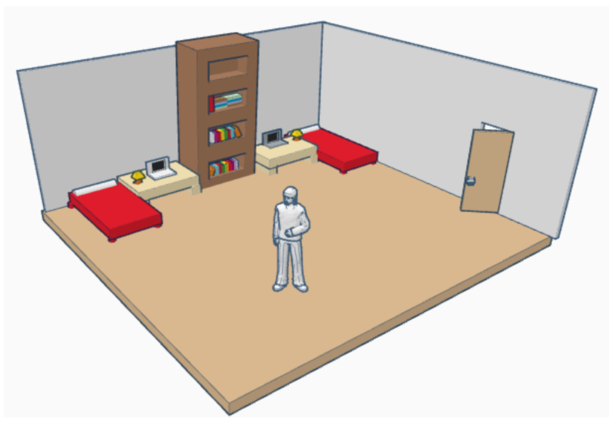
4.2 Temporal orientation case

Reality orientation is based upon the belief that continual, repetitive reminders will keep the patient stimulated and lead to an increase in temporal orientation.

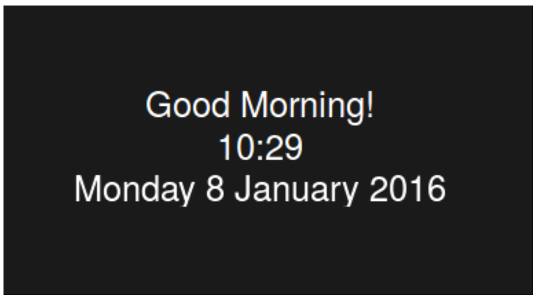
In temporal orientation case, the patient turns on Google Glass and it will display on screen information of the hour, the day of the week, number, month and year so that the patient can see it without the need to ask. This information will show the patient when the device does not detect any beacon or only detects place beacons.



(a) Trigger when.



(b) Scenario.



Good Morning!
10:29
Monday 8 January 2016

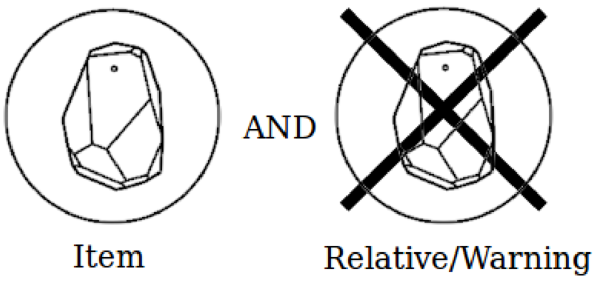
(c) Temporal orientation view.

Figure 4.1: Temporal orientation case.

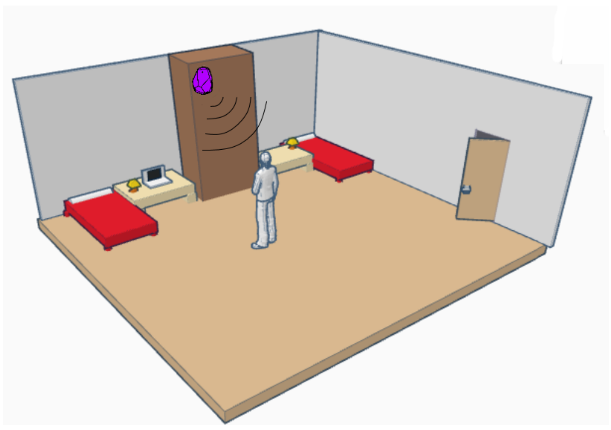
4.3 Spatial orientation case

We want to help the patient to find the objects needed to perform daily activities. For example, he could find clothes to wear.

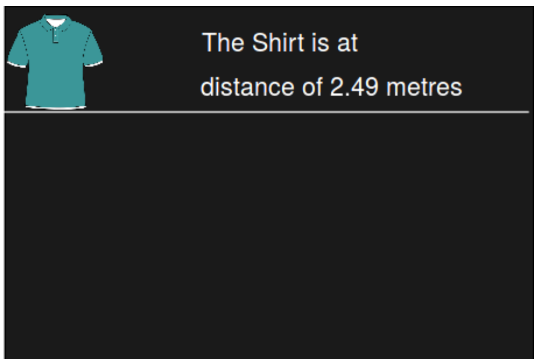
n this case the caregiver or family member must register items on the server. The first step to take is the authentication on the web. Then he must go to the item section and fill the form shown in figure 3.7. Each item has associated a beacon that should be placed on the item. When all objects are registered and placed in the appropriate places, and the information has been updated by the Glass, it is ready to use. The patient receives the location information of nearby items, this will happen as long as a higher priority beacon is not detected.



(a) Trigger when.



(b) Scenario.



The Shirt is at distance of 2.49 metres

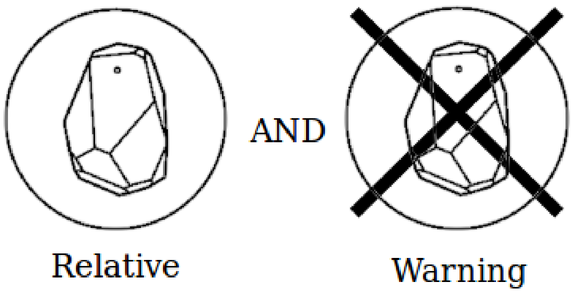
(c) Temporal orientation view.

Figure 4.2: Spatial orientation case.

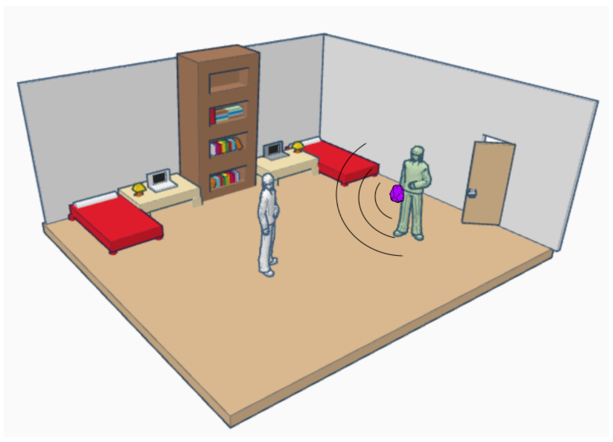
4.4 Relatives case

We want to display information about nearby relatives. In addition, we will stimulate the memory through mental imagery, showing photos of relative and events from the patient's past.

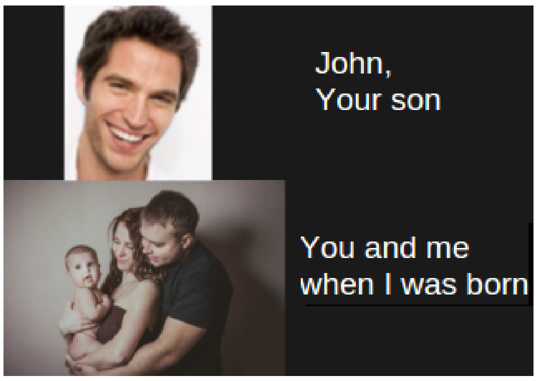
As we explained in the previous case, the patient need a caregiver or family member has registered previous information, that case is information from relatives and he will must fill the form shown in figure 3.8. He must fill the name, relationship, attach a profile photo and several old photos where he appears next to the patient and indicates a description of them. Then the family member must have the beacon, that regularly transmits the number that identifies him. When the patient meets a relative and the Glass does not detect warning beacon, it will show information relative who previously filling in the web. Old photos will not be shown all at once, the Glass will display one by one with their description and it will change after a few seconds.



(a) Trigger when.



(b) Scenario.



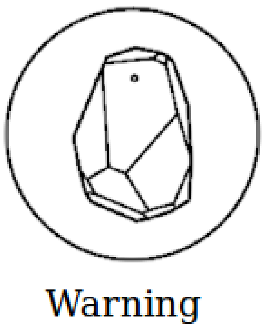
(c) Temporal orientation view.

Figure 4.3: Relatives case.

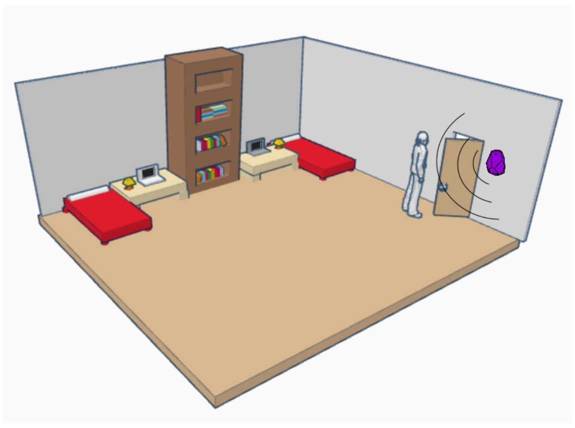
4.5 Warnings case

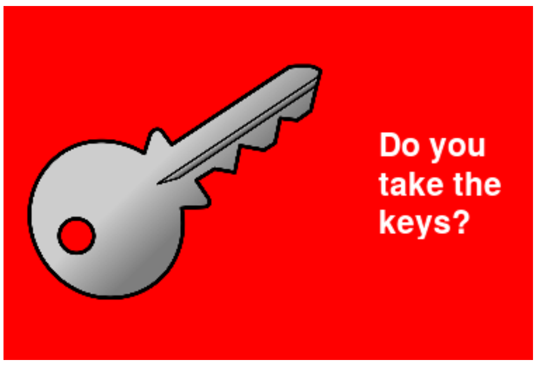
In this case we want to help the patient to remember routine task and provide warning about potential risks. For instance, turn off the stove, lock the door or notify some uneven area.

Firstly, the caregiver or family member has to register this warning in the web. In the warning section he must to fill the form shown in figure 3.9. In warning form he must fill the description, attach a descriptive photo and indicates when it show a notification (enter or leave area). Secondly, he has to be placed in the appropriate places the beacons. Finally, when the Glass enters or leaves an area of a warning beacon, it will display the associated warning.



(a) Trigger when.





(b) Scenario.

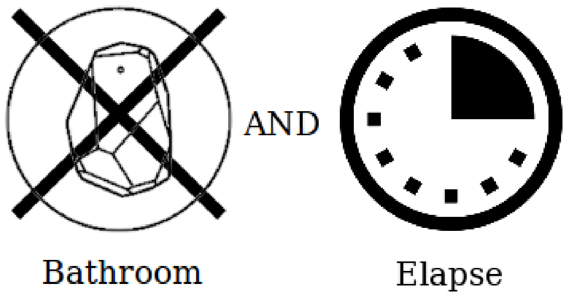
(c) Temporal orientation view.

Figure 4.4: Warnings case.

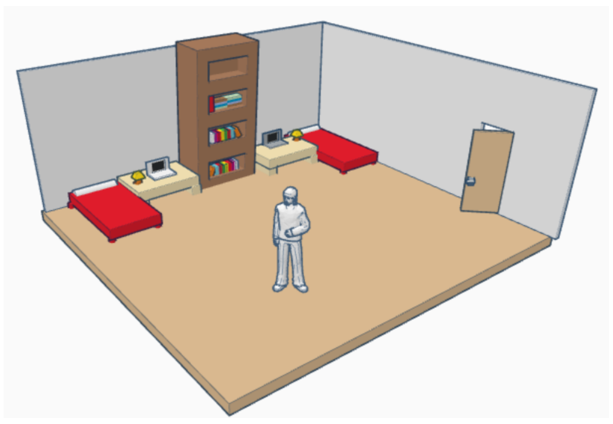
4.6 Incontinence case

Incontinence is generally primary reason why many caregivers decide to seek nursing home placement. Urinary incontinence may be controlled for some time by trying to monitor times of urinating. Once a schedule has been established, the patient may be able to anticipate incontinent episodes using our application.

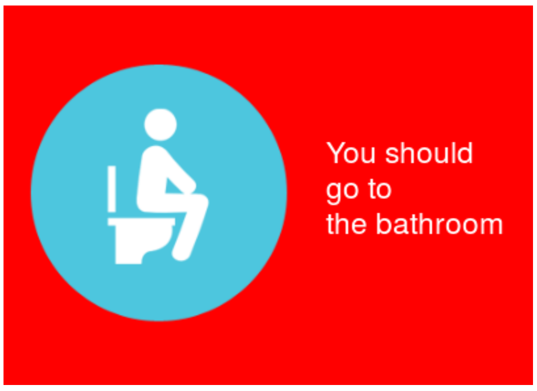
In this case the caregiver or family member must fill the form shown in figure 3.12 to be authenticated in the task automation web. Once authenticated, he may navigate to the Rules page. This page shows a list of the rules created by the user. In this page he can edit a created rule or create a new one. In this case he must to create a rule, that when in the bathroom no presence is detected during the time he considers appropriate, a web service must to trigger two actions: send emails and display live card. Thus, the patient receives the message that should go to the bathroom if he has not gone during the period of time indicated on the web. Furthermore, the family will receive an email notifying the situation.



(a) Trigger when.



(b) Scenario.



(c) Temporal orientation view.

Figure 4.5: Incontinence case.

CHAPTER 5

Conclusions and future work

In this chapter we will gather the conclusions obtained as a result of the project, as well as possible future work that can be done for further development of this project.

5.1 Conclusions

All in all, in this project we have developed and deployed a system to help people compensate for memory loss and improve it. We have used and to develop reality orientation board and mental imagery techniques. Alzheimer's patients should be equipped with Google Glass and the system is based on proximity aware beacons. The application enables temporal and spatial orientation, the patient can detect nearby items and the Glass displays information about their relatives if they are in the proximities. Furthermore, it shows self-referential imagery using old pictures where the patient is next to the family member. The application also help the patient to remember routine task and provide warning about potential risks.

We have developed a Content Management Server, it has been implemented using different technologies: HTML5, CSS, JavaScript, PHP and MySQL. In this server the caregivers or the family can manage these data they can register items, warnings and relative profiles. Each them have associated a beacon that should be placed on the right place.

The system also allows the user to automate tasks using a Task Automation Server [16], where he is able to create and edit semantic rules. Our system has implemented rules for them it uses semantic technologies (N3, EYE and RDF). One example rule is when the system sends an email to the family if the patient have not gone to the bathroom after a certain time, the family can chose this time in the server.

5.2 Achieved goals

Developing a Google Glass application to display information using beacons, this was the main goal of the project, the system is able to show contextual information based on proximity aware Estimote Beacons.

Define adaptable rules for users based on semantic technologies, as we have described in the previous section, these semantic technologies have been used to define adaptable rules for the caregivers and family.

Design and develop a content manager that allows users to manage the content of Google Glass patient's, we have developed a Content Management Server for managing the information to be shown to Alzheimer's patients with the aim of improving its memory about relatives and objects or providing warning about potential risks.

Evaluate the architecture in a practical case of e-Health, we have evaluated five different practical scenarios that have used different techniques increasing the independence of patients and improve their memory.

5.3 Problems faced

Google Glass device: this device is still under development, this fact means that it is still a limited hardware, therefore problems such as overheating or processing delays are unavoidable.

High frequency live card: the interface design using high frequency prevents us from using html to design the cards, as a result the design has been much slower and hard to having to use the canvas library.

Beacons accuracy: We have observed that fluctuations in the beacons occur much frequently, the distance is constantly changing. We have faced this problem creating a small safety time, where the system checks whether the user's position has changed with respect to the beacon or otherwise if there was a fluctuation.

5.4 Future work

Here we expose some recommendations that could be made in order to improve the features and characteristics of the work:

- We can use nearables and stickers which are much smaller beacons. While original beacons were designed for venues and static locations, stickers are great for making individual objects smart.
- The Google Glass interface could be improved by using animations or including videos. In addition, we can add the use of voice to communicate notifications.
- Create a map using Estimote Indoor Location SDK, to improve the spatial orientation of the patient.
- Implement monitoring to improve detection when the patient comes in and out of the zones.
- Adapt the application for use in smart watches, these would be more comfortable to wear for patients.

Bibliography

[1] Oscar Araque. Design and implementation of an event rules web editor, July 2014.

[2] Wojtek Borowicz. How do beacons work? the physics of beacon tech. `http://blog.estimote.com/post/106913675010/how-do-beacons-work-the-physics-of-beacon-tech`, 2015. Accessed December 13, 2015.

[3] Wojtek Borowicz. What is a beacon region? `https://community.estimote.com/hc/en-us/articles/203776266-What-is-a-beacon-region-`, 2015. Accessed December 15, 2015.

[4] Miguel Coronado and Carlos Angel Iglesias. Task automation services: Automation for the masses. *IEEE Internet Computing*, 20(1):52–58, 2016.

[5] Miguel Coronado, Carlos Angel Iglesias, and Emilio Serrano. Modelling rules for automating the evented web by semantic technologies. *Expert Syst. Appl.*, 42(21):7979–7990, 2015.

[6] Irene Piryatinsky Andrew E. Budson-Brandon A. Ally Erin Hussey, John G. Smolinsky. Using mental imagery to improve memory in patients with Alzheimer's disease: Trouble generating or remembering the mind's eye? 2012.

[7] Estimote. Beacon tech overview. `http://developer.estimote.com/`, 2015. Accessed December 13, 2015.

[8] Estimote. What is Eddystone? `http://developer.estimote.com/eddystone/`, 2015. Accessed December 15, 2015.

[9] Estimote. What is iBeacon? `http://developer.estimote.com/ibeacon/`, 2015. Accessed December 15, 2015.

[10] Google. Immersions. `https://developers.google.com/glass/develop/gdk/immersions`, 2015. Accessed December 11, 2015.

[11] Google. Live cards. `https://developers.google.com/glass/develop/gdk/live-cards`, 2015. Accessed December 12, 2015.

[12] Google. The Mirror API. `https://developers.google.com/glass/develop/mirror/`, 2015. Accessed December 11, 2015.

[13] Google. Platform overview. `https://developers.google.com/glass/develop/overview`, 2015. Accessed December 17, 2015.

[14] Google. Static cards. `https://developers.google.com/glass/develop/mirror/static-cards`, 2015. Accessed December 11, 2015.

[15] J.DeRoo. Euler yet another proof engine. `http://eulersharp.sourceforge.net`, 2013. Accessed December 26, 2015.

[16] Sergio Muñoz López. Development of a task automation platform for beacon enabled smart homes. 2016.

[17] Maëlenn Guerchet Gemma-Claire Ali Yu-Tzu Wu Matthew Prina Martin Prince, Anders Wimo. World Alzheimer report 2015 the global impact of dementia. 2015.

[18] MySQL AB. *MySQL 5.0 Reference Manual*, 2006.

[19] Lukasz Pobereżnik. Estimote SDK for Android. `https://github.com/Estimote/Android-SDK`, 2015. Accessed December 17, 2015.

[20] Ola Puchta. Estimote cloud. `https://community.estimote.com/hc/en-us/articles/203854516-What-is-Estimote-Cloud-`, 2015. Accessed December 14, 2015.

[21] Eric Redmond. *Programming Google Glass*. Pragmatic Bookshelf, 2013.

[22] Eric Redmond. *Programming Google Glass: Build Great Glassware Apps with the Mirror API and GDK*. Second edition, 2015.

[23] Witek Socha. Indoor location. `https://community.estimote.com/hc/en-us/articles/203493626-What-s-Estimote-Indoor-Location-`, 2015. Accessed December 17, 2015.

[24] Witek Socha. Technical specification of Estimote beacons and stickers. `https://community.estimote.com/hc/en-us/articles/204092986-Technical-specification-of-Estimote-Beacons-and-Stickers`, 2015. Accessed December 11, 2015.

[25] Agnieszka Steczkiewicz. Best practices for installing Estimote beacons. `https://community.estimote.com/hc/en-us/articles/202041266-Best-practices-for-installing-Estimote-Beacons`, 2015. Accessed December 13, 2015.

[26] Vanesa Sánchez-Valladares Jaramillo-Ana Balbás Repila Víctor Isidro Carretero, Cynthia Pérez Muñano. Guía práctica para familiares de enfermos de alzheimer. 2011.