ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE UN SISTEMA DE RECONFIGURACIÓN Y PROGRAMACIÓN DE OBJETOS INTELIGENTES USANDO REALIDAD AUMENTADA"

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

FERNANDO EMMANUEL CAMPAÑA ROJAS

VICTOR ALFONSO MENOSCAL GARCIA

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Mis más sinceros agradecimientos a Dios primeramente por darme fuerza y salud durante toda mi vida académica, también le agradezco a mis padres Víctor y Consuelo por guiarme en el camino del bien y superación. A mi familia, hermanos y amigos que me han apoyado directa o indirectamente por su tiempo estoy muy agradecido, y por todos mis profesores que me han brindado su granito de arena del conocimiento, muchísimas gracias.

Víctor Menoscal

Le agradezco a mis familiares, amigos y profesores que de alguna u otra manera contribuyeron a brindarme el conocimiento y ánimos para culminar con éxito este trabajo. Adicionalmente le agradezco al Centro de Tecnologías de Información (CTI), por brindarme el uso de sus recursos e instalaciones. Especial agradecimiento a mi abuelita Bélgica, por enseñarme que hasta el más mínimo esfuerzo vale la pena, motivarme a estar en constante aprendizaje y encaminarme por el camino del bien, y a mi madre Blanca por cuidarme, enseñarme el valor de los libros y el trabajo duro. Eternamente agradecido con todos.

Fernando Campaña

TRIBUNAL DE EVALUACIÓN

PhD. Boris Vintimilla	**PhD. Federico Domínguez**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

Víctor Alfonso Menoscal García

.....

Fernando Emmanuel Campaña Rojas

RESUMEN

Cada día los objetos cotidianos se están convirtiendo e incrementando en objetos inteligentes incorporando circuitos integrados para conectarse a *Internet*, así mismo se incrementa el número de aplicaciones a recordar para usar tales objetos, y es así que la memorización en el uso de los objetos se dificulta al tener muchos objetos inteligentes de varios proveedores ya que la memoria y el tiempo está limitada para otras tareas del hogar.

Nuestra solución consiste en crear una aplicación de realidad aumentada que permite reconocer al objeto inteligente mediante un marcador. Este marcador es una imagen con detalles significativos como es el contraste y el brillo de la imagen que se utiliza para obtener acceso a un contenido. En nuestro caso mostrar una interfaz de usuario en el teléfono móvil y así activar circuitos en el hogar y hacer uso de las funcionalidades del objeto que no se encontraban muy visibles al usuario. Además el sistema de reconfiguración y programación de objetos inteligentes permite conectar varios objetos para realizar una acción específica, así mismo un administrador web para administrar los objetos e interfaces.

Se utilizó *SCRUM* como la metodología de desarrollo por la facilidad de gestionar proyectos de manera rápida y hacer entregables en cada etapa. El sistema consta de tres módulos el administrador web, la aplicación móvil y el *Back-End*.

El administrador web y el *Back-End* fue implementado en una plataforma de desarrollo por bloques llamado *Node-Red*, así mismo la aplicación móvil se desarrolló utilizando el *Browser Argon* de Realidad Aumentada para mostrar contenido multimedia en el teléfono móvil.

Por lo tanto, obtuvimos como resultado un sistema gráfico de realidad aumentada para identificar y accionar objetos inteligentes, objetos como encender una cafetera o variar la intensidad de luz sin necesidad de memorizar su funcionamiento o su difícil acceso a los botones manuales del objeto.

Así mismo resolvemos el inconveniente de controlar varios objetos de diferentes proveedores con una sola aplicación móvil para que todos los objetos del hogar

funcionen como un sistema total en toda la casa, permitiendo conectar objetos inteligentes con otros objetos inteligentes para reprogramar las funcionalidades para las que originalmente fueron propuestos.

Cabe mencionar que para este proyecto se empleó conocimiento adquirido en las materias de la carrera, tales como Aplicaciones Multimedia Interactivas, Interacción Humano Computador, Desarrollo de Aplicaciones Web, Gráficos por Computadora y Programación de Sistemas.

ÍNDICE GENERAL

AGRADECIMIENTOS ii
TRIBUNAL DE EVALUACIÓN iii
DECLARACIÓN EXPRESA iv
RESUMEN v
ÍNDICE GENERAL vii
CAPÍTULO 1 1
1. ANÁLISIS DE LA PROBLEMÁTICA. 1
1.1 Introducción 1
1.2 Descripción del problema 1
1.3 Objetivos 2
1.3.1 Objetivo General 2
1.3.2 Objetivos Específicos 2
1.4 Revisión de literatura 3
1.4.1 Conceptos claves 3
1.4.2 Soluciones existentes 4
1.4.3 Frameworks de Realidad Aumentada 7
1.5 Solución propuesta 9
1.5.1 Creación de concepto 9
CAPÍTULO 2 12
2. METODOLOGÍA DE DESARROLLO. 12
2.1 Metodología de desarrollo del software 12
2.2 Arquitectura del Sistema de Realidad Aumentada 13
2.2.1 Módulo: Aplicación Móvil 13
2.2.2 Módulo: Back-End 15
2.2.3 Módulo: Administrador Web 16
2.3 Prototipo 18
2.4 Herramientas tecnológicas empleadas 20

2.4.1 Desarrollo 21
2.4.2 Debugging 21
2.4.3 Control de versiones 21
2.4.4 Multipropósito 21
CAPÍTULO 3 23
3. SOLUCIÓN PROPUESTA. 23
3.1 Desarrollo móvil 23
3.1.1 Conectar objetos 25
3.1.2 Desconectar objetos 28
3.1.3 Realidad Aumentada 29
3.1.4 Prueba del Módulo Móvil 32
3.1.5 Dispositivos 33
3.2 Desarrollo Web 36
3.2.1 Estructura de datos 36
3.2.2 Implementación 37
3.3 Back-End 44
3.3.1 Estructura del Administrador Web 44
3.3.2 Flujos del Back-End 45
CAPÍTULO 4 49
4. ANÁLISIS DE RESULTADOS. 49
4.1 Casos de Uso 49
4.1.1 Usuario Administrador 49
4.1.2 Usuario Final 50
CONCLUSIONES Y RECOMENDACIONES 57
BIBLIOGRAFÍA 60

CAPÍTULO 1

1. ANÁLISIS DE LA PROBLEMÁTICA.

1.1 Introducción

En el Centro de Tecnologías de Información (CTI), se ha desarrollado un sistema básico de reconfiguración y programación de objetos inteligentes usando interfaces táctiles como parte de la línea de investigación existente y un proyecto anterior de materia integradora. Ahora se busca extender el alcance de este proyecto agregando una interfaz gráfica de realidad aumentada usando Vuforia, *Three.js* y expandir el tipo de objetos y acciones a implementar.

La realidad aumentada es un área en crecimiento debido a los avances en el procesamiento de los chips y la incorporación de estos en tecnologías móviles como celulares y tabletas. Su uso potencial se espera en áreas como construcción, ayuda para discapacitados, reparación de maquinarias, autoaprendizaje, automatización, transporte y campañas publicitarias.

Los objetos inteligentes nos están llevando a la época de la computación ubicua. Cada vez se incorporan más objetos a nuestros hogares, objetos diseñados para facilitar nuestras vidas. Tenemos objetos para transmitir contenido al televisor, encender luces, abrir puertas, alterar la temperatura, reproducir música, entre otras. Y se están creando más objetos con usos aún no previstos que afectarán todas las áreas del conocimiento, como por ejemplo, deporte, industria, publicidad, medicina, entre otras.

1.2 Descripción del problema

Como se ha indicado, los objetos inteligentes fueron creados para mejorar la calidad de vida de las personas. Generalmente automatizar tareas, por lo cual la tendencia es la de convertir cada objeto cotidiano en un objeto inteligente, así como se convirtieron objetos cotidianos en software, como por ejemplo el correo en *e-mail*. Los objetos cotidianos se transforman en objetos inteligentes incorporando circuitos integrados para conectarse a *Internet*, y así recibir y enviar órdenes a distancia mediante la web o un dispositivo móvil.

Los objetos inteligentes generalmente cuentan con una aplicación para su configuración y uso. Pero a medida que el número de objetos inteligentes se incrementa, se incrementa el número de aplicaciones a recordar para usar [1]. Según *Business Insider*, en el 2020 habrá más de 24 billones de objetos inteligentes en el mundo, lo que equivale a 4 objetos por cada ser humano en la Tierra [2]. Será tal la cantidad de objetos y aplicaciones en el hogar que su memorización será difícil. La memoria y el tiempo está limitada para otras tareas del hogar, por lo que la búsqueda de la aplicación termina convirtiéndose en una molestia para el usuario.

Por otro lado, no existe una forma sencilla de conectar objetos de distintos proveedores para que funcionen como un sistema. Sistema en el cual, un evento de un dispositivo dispare varios eventos de otros objetos en cadena, como por ejemplo que las luces de la casa se enciendan al abrir la puerta del garaje. Se necesitaría de dos aplicaciones para que la interconexión funcione correctamente, en lugar de una única aplicación [3].

Por lo tanto, el problema a resolver es la dificultad de reconocer el objeto inteligente y hacer uso de sus funcionalidades, así como encadenar eventos entre objetos para reprogramar sus interacciones y lograr combinar la funcionalidad del mundo virtual con el real. Dado que el mundo físico es menos adaptable y configurable que el mundo digital que es prácticamente infinito [4].

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar un sistema de reconfiguración y programación de objetos inteligentes usando realidad aumentada en dispositivos móviles.

1.3.2 Objetivos Específicos

- Investigar y familiarizarse con las herramientas y *frameworks* para realizar proyectos de realidad aumentada, por ejemplo: Vuforia, *Unity* y *Three.js.*

- Familiarizarse con el sistema existente Intgio de programación y configuración de objetos inteligentes. El mismo que fue creado en la materia integradora pasada.
- Diseñar e implementar un sistema gráfico usando la herramienta de realidad aumentada seleccionada.
- Integrar la herramienta diseñada al *Back-End* existente de control de objetos inteligentes.
- Pruebas de integración con dispositivos para prototipado de objetos inteligentes como Arduinos, *Photons*, etc.

1.4 Revisión de literatura

1.4.1 Conceptos claves

La realidad aumentada es el siguiente paso a la realidad virtual. La realidad virtual es un sistema que interactúa con el usuario simulando un entorno real en un entorno ficticio [5].

Realidad Aumentada

La realidad aumentada es la superposición de contenidos digitales sobre los objetos reales a través del video de la cámara de dispositivos electrónicos como *Smartphones*, Tabletas, entre otros [6][7]. La realidad aumentada permite la superposición de contenido informativo sobre imágenes del mundo real al instante con la intención de mejorar la experiencia de usuario al momento de interactuar con el mundo real.

Objetos Inteligentes

Un objeto inteligente es un dispositivo electrónico que es capaz de conectarse a una red e interactuar con el usuario o con otros objetos inteligentes mediante una aplicación [4]. Su meta es brindar un servicio o realizar una acción a distancia a través de *Internet*. Por ejemplo el *Nest Thermostat*, registra los patrones de uso del usuario para cambiar automáticamente la temperatura.

1.4.2 Soluciones existentes

Existen varias soluciones que tratan de solucionar el problema de seleccionar la aplicación del dispositivo a usar, minimizando el tiempo de búsqueda y la carga mental del usuario.

Hubs y Apps

Como primera opción tenemos empresas que diseñan un hub, que es una caja con circuitos que integra la mayoría de tecnologías inalámbricas para conectar diferentes dispositivos, e integrarlos mediante una *App*. El usuario puede ver la lista de todos sus dispositivos de manera rápida y realizar acciones de inmediato. En esta categoría tenemos a *AT&T Digital Life* [8], *Revolv*, comprado por *Nest*, *Smarthings* [9], Vera [10] y *Nest*. *Nest* se destaca porque cualquier producto de la familia *Nest* puede actuar como *hub* sin necesidad de un dispositivo adicional como en los demás [11]. El problema reside en que si se tienen muchos dispositivos de un mismo tipo, y la aplicación los guarda con nombres o ids, entonces se vuelve un problema identificarlos al momento de querer usarlos.

Accionadas por voz

Los avances en inteligencia artificial y audio, han dado lugar a una nueva era de dispositivos controlados por voz, siendo la voz la interfaz de comunicación para usarlos. *Google Home* [12] y *Amazon Echo* [13] son plataformas que permiten la interconexión de dispositivos inteligentes y el control de dichos dispositivos mediante voz a través del *Google Assistant* y *Alexa* respectivamente. *Google* posee una tasa de error de palabras de 4.9% [14] y mejorando. Aunque la adquisición de estos dispositivos adicionales representa un costo significativo para el usuario, $129 por *Google Home* y $179.99 por *Amazon Echo*. Ambos dispositivos soportan el uso de *IFTTT (If This Then That)*. *IFTTT* es un servicio que permite crear recetas en base a eventos para ejecutar una acción cuando ocurra un evento. Por medio de la integración con *IFTTT*, se puede encadenar eventos entre dispositivos y servicios en la nube.

Apple Homekit permite añadir dispositivos a la *App* para *iOS* llamada *Home*, y luego accionar cada dispositivo con comandos de voz a *Siri* o a través de la aplicación [15].

Control remoto más un Hub

Existe un control remoto que permite configurar sus botones para controlar los objetos inteligentes, también permite la configuración de actividades para encadenar eventos. Se conecta con una *App* en el teléfono, que muestra acciones adicionales de acuerdo al dispositivo a usar [16].

Open Hybrid

Es una plataforma para la interacción con objetos inteligentes mediante una interfaz de realidad aumentada. Es de código abierto y fue desarrollada por el *MIT Media Lab*. Consta de dos partes: una aplicación móvil para *iOS* y un servidor de *Node.js*. La aplicación móvil llamada *Reality Editor* permite la interacción y reconfiguración de los objetos para encadenar eventos. Y el servidor permite la subida de imágenes para ser añadidas como marcadores, los cuales serán detectados por la *App* móvil para que se coloque la interfaz sobre el objeto, dicha interfaz es una página web que se posiciona por medio de *Three.js*. *Reality Editor* presenta un mapeo directo que permite interactuar con los dispositivos directamente en el lugar [17].

Como la interfaz se puede crear por medio de código HTML, el usuario puede crear sus propias interfaces de Realidad Aumentada sin conocimientos 3D. También permite conectar la funcionalidad de varios dispositivos por medio de arrastrar y soltar sobre marcadores superpuestos en el mundo real sobre la pantalla del celular. Permite agregar todo tipo de dispositivos, mediante la programación de las interfaces de *hardware* para el dispositivo en cuestión [18].

Open Hybrid es la principal inspiración del proyecto y se usó parte del código y contenido multimedia para implementar nuestro propio sistema

en *Android*.

IntgIO

IntgIO es un sistema para interconectar dispositivos inteligentes y reconfigurar sus acciones mediante una interfaz web que representa a los dispositivos como grafos. Cada dispositivo está formado por uno o más objetos, que a su vez tienen una o más acciones disponibles. La interfaz permite conectar un objeto con otro por medio de arrastrar y soltar entre los nodos de tipo objeto, para que al momento que se realice una acción en un objeto su estado se notifique al otro objeto y realice una acción acorde al evento. La comunicación entre dispositivos puede ser mediante MQTT o TCP.

Existen varios tipos de objetos: híbrido, emisor y receptor. El híbrido es un receptor y emisor. El emisor envía datos a ser procesados por el receptor. Por lo que las conexiones se hacen de un objeto emisor a un receptor, también se puede hacer de un emisor a un híbrido.

Las acciones son de tipo incremental o tipo binario (ON/OFF), la cual es una de las principales diferencias con *Open Hybrid* donde las acciones son siempre un número flotante entre 0.0 y 1.0.

Los dispositivos son identificados con un código, por lo que implementaron una opción para identificar dispositivos por medio de código QR. Lo que facilita la identificación de dispositivos para el usuario [19].

Otros trabajos relacionados

Entre otros trabajos encontramos a *Remote Touch* [20] que es un sistema diseñado para personas con algún problema físico de movimiento como discapacitados o ancianos. En este sistema la comunicación entre dispositivos se realiza mediante *X10*, los datos se guardan en un archivo XML y el reconocimiento del objeto se realiza mediante código QR. Luego tenemos *exTouch* [21] que usa realidad aumentada para manipular un objeto a través del espacio mediante

gestos sobre la interfaz de realidad aumentada o movimientos de la tablet. El movimiento está dado por la diferencia entre la posición del marcador en la interfaz gráfica y la posición del objeto real. También tenemos *Deus EM Machina* [22] que utiliza un sensor conectado al celular, para detectar el dispositivo en base a la señal electromagnética emitida. Al detectar la señal y reconocer el dispositivo, el celular abre automáticamente la aplicación del mismo o muestra acciones que llaman "*charms*" para ejecutar acciones adicionales con el dispositivo en base a la aplicación actual, por ejemplo, imprimir un documento en una impresora. Por último, encontramos a *SmartX Virtuality* [23] que utiliza procesamiento digital de imágenes para reconocer los objetos, y mostrar una interfaz aumentada. Lo interesante del sistema es que usa un microcontrolador *Texas Instruments* MSP430G2553 como servidor y puede correr en dispositivos *Android* 2.2 y superior.

1.4.3 Frameworks de Realidad Aumentada

Vuforia

Vuforia es una plataforma de software para desarrollar aplicaciones de Realidad Aumentada (AR). Se puede agregar fácilmente la funcionalidad de visión por computadora permitiendo reconocer marcadores y objetos, o mapear superficies en el mundo real.

Las capacidades de reconocimiento y tracking de Vuforia pueden ser usadas en diferentes tipos de objetos, tales como:

- ***Image Targets:*** Imágenes planas, sin necesidad de código QR o marcadores específicos.
- ***VuMarks:*** Imágenes personalizadas que permiten codificar información como identificadores de objetos o marcas. Presenta ventajas con respecto a las imágenes comunes en que se pueden crear millones de instancias únicas [24].

Componentes de la Plataforma

- **Herramientas:**

  - o ***Vuforia Object Scanner:*** Para *Android*, permite escanear objetos 3D y convertirlos en un formato compatible con Vuforia.
  - o ***Target Manager:*** Para crear base de datos de los marcadores.
  - o ***License Manager:*** Para crear y administrar las *License Keys*.
- ***Cloud Recognition Service:*** Servicio de reconocimiento de imágenes en la nube.

Funcionamiento

El dispositivo en el cual se ejecuta la aplicación debe contar con una cámara para enfocar el marcador y realizar los cálculos respectivos para mostrar el contenido multimedia en el marcador. Vuforia detecta los "*feature points*" del marcador (*Image Target*) y luego usa los datos de las características de la imagen y el *frame* recibido de la cámara [25].

Para el proceso de reconocimiento en Vuforia se sube el marcador en la aplicación de Vuforia como se muestra en la Figura 1.1.

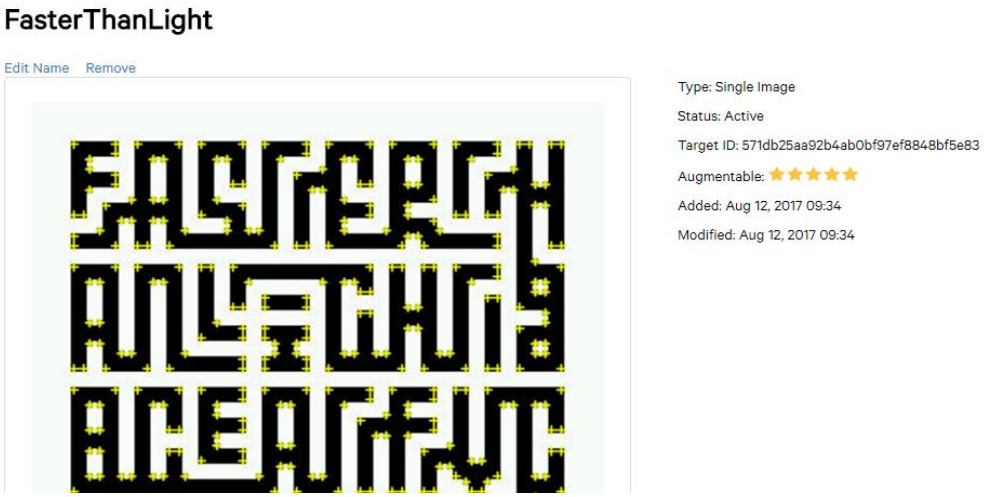

Figura 1.1: Imagen en Vuforia

La puntuación de las estrellas representa la calidad de la imagen para reconocer el marcador. Los archivos descargados son el XML que contiene el nombre del ID del marcador y el tamaño que ha sido

seleccionado antes de descargar en la base de datos.

El segundo archivo es un binario en el que se almacenan los descriptores del marcador. Ambos archivos son necesarios para el proceso de reconocimiento y *tracking* de Vuforia.

1.5 Solución propuesta

A continuación se listan varias opciones que pueden considerarse para resolver partes del problema planteado.

Para reconocer objetos, se puede utilizar la cámara, la proximidad con el objeto, el contacto con el objeto y la posición del objeto. Por medio de la cámara se puede reconocer imágenes y códigos. Las imágenes a reconocer pueden ser fotos del objeto, imágenes con características detalladas (marcadores) o una marca fiduciaria. Los códigos a reconocer pueden ser QR o de barras. La proximidad puede detectarse mediante *Bluetooth*, RFID o NFC.

La comunicación del sistema con los dispositivos puede ser a través de redes inalámbricas de área local que permite establecer conexiones a Internet (*Wi-Fi* del inglés, *Wireless Fidelity*). Y el usuario puede interactuar con el mismo mediante gestos o clicks.

El servidor puede ser una *Raspberry Pi*, un Arduino Yun o cualquier computadora con Linux. También se puede emplear un microcontrolador como el MSP430G2553.

La interfaz de realidad aumentada puede ser desarrollada usando Web, *Unity* o *Android* y otra forma es utilizando archivos ARAF.

1.5.1 Creación de concepto

La idea es desarrollar una aplicación para *Android* que interactúe con un servidor Linux. El servidor correrá una instancia de *Node-red* y *Neo4j*. La aplicación analizará el marcador que corresponde al objeto, este marcador es una imagen con bastante detalle similar a la imagen del código QR que se utiliza para obtener acceso a un contenido en nuestro caso mostrar una interfaz de usuario en el teléfono móvil y así activar

circuitos en el hogar. La interfaz de usuario constará de conectores circulares y acciones por ejemplo *switches*, *sliders*, etc. Los conectores permitirán conectar la funcionalidad de un objeto con otro. Los objetos podrán usarse de manera física y a través de la interfaz aumentada como se muestra en la Figura 1.2.

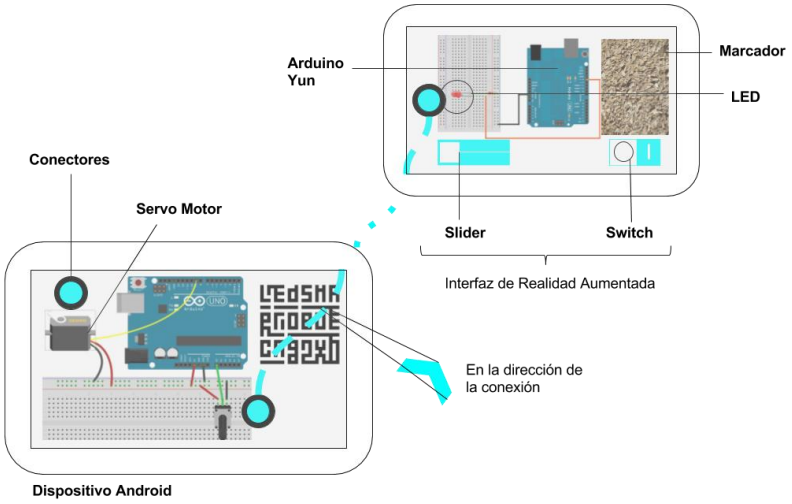

Figura 1.2: Idea general

Para facilitar la conexión de objetos lejanos entre sí, se podrá dar tap sobre el conector, lo que provocará que la pantalla se muestre con un marco rojo alrededor, aparezca un botón de cancelar en la parte inferior izquierda, y el conector seleccionado se tornará rojo lo que significa que está listo para ser conectado con otro conector. Al realizar la conexión aparecerá una línea que indique la dirección de la conexión y la pantalla volverá a su estado original. Al tocar el botón cancelar, se borrará la selección y la pantalla regresará a su estado original como se muestra en la Figura 1.3.

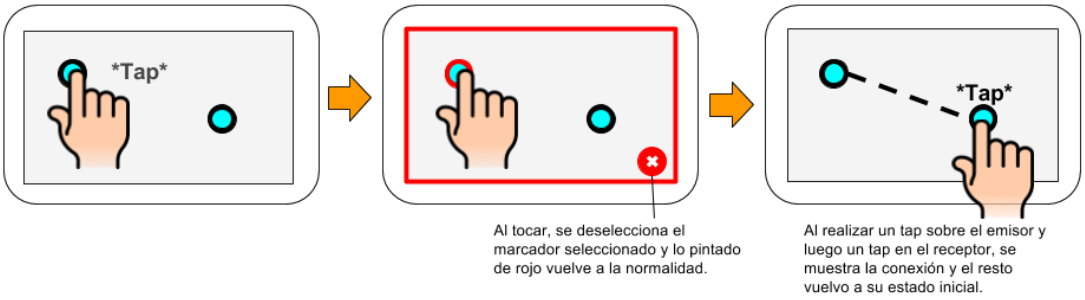

Figura 1.3: Conexión

Para evitar que el usuario intente conectar objetos que no son compatibles en base al tipo (emisor, receptor), cuando seleccione un conector de un tipo, los conectores que no sean compatibles para ser conectados se deshabilitarán haciéndose translúcidos como se muestra en la Figura 1.4.

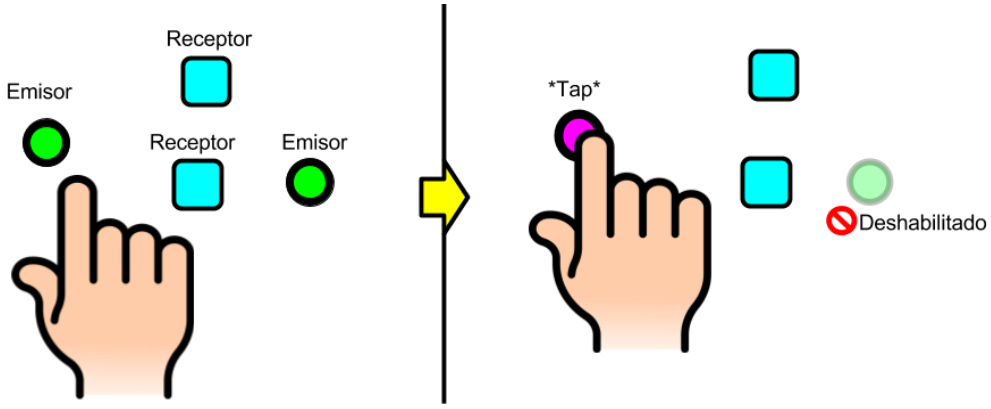

Figura 1.4: Interacción para evitar errores de conexión

Además la comunicación entre móvil-servidor y servidor-objetos, se realizará mediante *Wi-Fi*.

CAPÍTULO 2

2. METODOLOGÍA DE DESARROLLO.

En este capítulo se describe la metodología, la arquitectura y las herramientas de desarrollo del presente proyecto, así mismo una breve explicación del funcionamiento del sistema.

Se utilizó para el presente desarrollo una metodología ágil que se basa en el desarrollo de software iterativo e incremental (*SCRUM* del inglés, *Agile*), también se utilizó para el control de versiones *Git* y *Github* para el almacenamiento de repositorio del código fuente.

El sistema consta de tres módulos el administrador web, la aplicación móvil y el *Back-End* encargado de procesar, validar y guardar la información del usuario final y del usuario administrador, entonces al iniciar el sistema se deberá primeramente registrar el objeto inteligente en el administrador web, así mismo se deberá subir al servidor: el marcador, la base de datos de Vuforia y la interfaz de usuario que se mostrará en la aplicación móvil. Una vez registrado el objeto inteligente, el usuario imprimirá su marcador para ser detectado por la aplicación móvil, sobre el marcador se mostrará la interfaz que el usuario haya subido en el administrador web, además el usuario podrá reconfigurar y programar objetos por medio de conexiones virtuales.

2.1 Metodología de desarrollo del software

La metodología de desarrollo del software se la puede definir como un conjunto de procedimientos y técnicas que guiará al equipo de desarrollo en todo el ciclo del software. En la cual utilizamos la metodología *SCRUM* porque esta nos brinda entregables en cada iteración, cada iteración consta de 4 semanas además nos proporciona flexibilidad al cambio obteniendo retroalimentación del usuario para el mejoramiento del sistema en cada etapa.

Además se usó un cronograma de actividades en cada semana para agilizar el desarrollo del software.

2.2 Arquitectura del Sistema de Realidad Aumentada

Como se mencionó anteriormente el sistema está dividido en 3 módulos: aplicación móvil, *Back-End* y administrador web, tal como se muestra en la Figura 2.1.

Figura 2.1: Módulos del Sistema

2.2.1 Módulo: Aplicación Móvil

La aplicación móvil permite conectar y desconectar conexiones entre objetos e interactuar con la interfaz de cada objeto. Lo que se detalla en el caso de uso de la Figura 2.2.

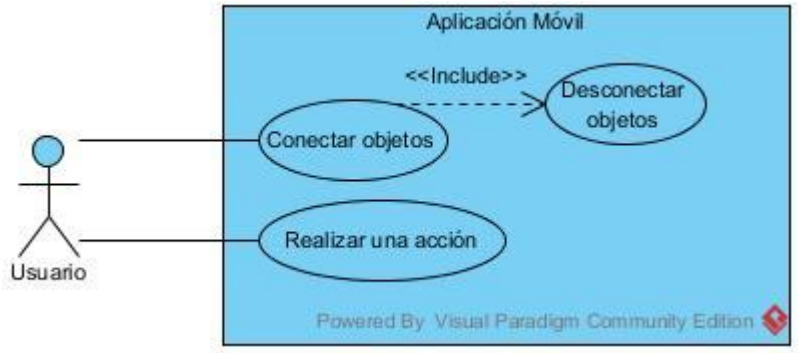

Figura 2.2: Casos de uso de la aplicación móvil

La aplicación móvil está soportada para *Android* 6.0+ (API 23). Desarrollada mediante *Nativescript*, el cual es un *framework* para realizar aplicaciones multiplataforma por medio de tecnologías web como *Javascript*, CSS, *Typescript*, *Angular*, entre otros.

En la Figura 2.3 se muestra la estructura básica de la aplicación, compuesta por un *WebView* en el cual se carga el código de la interfaz de usuario de la aplicación, así como la vista de la cámara. Dentro de dicha interfaz se encuentra un *Div* en el cual se cargan las interfaces

aumentadas de los objetos conforme son detectados. La interfaz de usuario está basada en el trabajo realizado por el MIT Lab en su *Reality Editor*.

Figura 2.3: Estructura de la aplicación móvil

La aplicación utiliza Vuforia para el reconocimiento de marcadores. Además, para mostrar el contenido de realidad aumentada se utiliza el *Browser* de *Argon.js*.

Argon.js utiliza la librería *Three.js* para el procesamiento de contenido 3D y los objetos están representados en coordenadas geoespaciales, utilizando *Cesium.js*

Argon.js proporciona soporte para tecnologías web estándar como HTML, CSS3, *JavaScript*, *WebGL*, tal como se muestra en la Figura 2.4.

Figura 2.4: *Browser* de *Argon.js* [26]

Para la comunicación y la actualización de las interfaces en tiempo real se planteó usar *Socket.io*, pero no fue necesario. *Socket.io* es una librería de *Websockets* que permite mantener conexiones activas y realizar broadcasts entre los clientes. Fue suficiente la comunicación a través del protocolo de transferencia de hipertexto (HTTP del inglés, *Hypertext Transfer Protocol*) y del protocolo de transporte de mensajes (MQTT del inglés, *Message Queuing Telemetry Transport*). Una vista general de los componentes se muestra en la Figura 2.5.

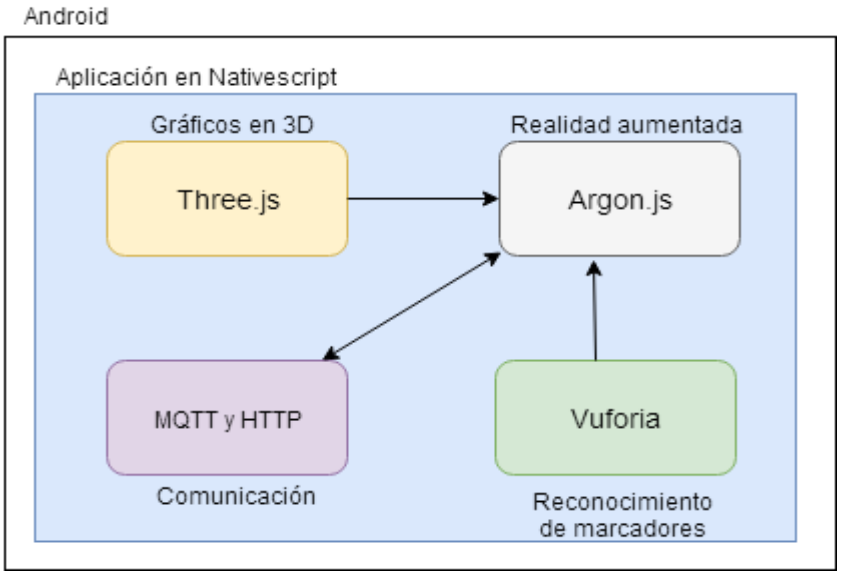

Figura 2.5: Arquitectura de la Aplicación Móvil

2.2.2 Módulo: Back-End

El *Back-End* contiene la información del estado de los objetos y permite acceder a ellos mediante un *Webservice* implementado en *Node-red*. *Node-red* es un entorno gráfico para programar en *Node.js*, provee de nodos y conexiones para formar flujos que implementan funcionalidades.

La base de datos no relacional *Neo4j* basada en grafos, lo cual es una metáfora directa del problema a resolver. Se utilizó porque nos brinda menor tiempo de desarrollo, visualización de los datos, facilidad de conectar nodos para formar relaciones y mantenimiento del *Back-End.*

En esta base se crearon entidades como: Objeto, Dispositivo y Acción.

Un dispositivo puede tener uno o más objetos, y un objeto puede tener una o más acciones. Para encadenar acciones, se realiza una relación entre varios objetos.

Luego la funcionalidad de comunicación entre objetos se implementa en *Node-red*. Cada objeto tiene asociado un nombre de marcador de Vuforia y una ruta de una página web, la cual será servida al reconocer el marcador del objeto.

2.2.3 Módulo: Administrador Web

El sistema cuenta con un administrador web para poder agregar contenido de realidad aumentada y administrar objetos.

El administrador web está implementado en *Node-red*, *Bootstrap* y *Dropzone*.

El administrador web permite crear nuevos dispositivos, agregar objetos a los dispositivos, agregar acciones a cada objeto, agregar marcadores y agregar interfaces.

Este administrador web usa *Webservices* implementados en el *Back-End* para actualizar la información de los objetos, dispositivos y acciones. Actualizaciones tales como: crear, eliminar y conectar elementos.

Para añadir marcadores a los objetos se muestra la página de Vuforia, útil para subir y descargar bases de datos de marcadores. Esta base de datos se agrega al objeto inteligente para que al reconocer un marcador, el objeto inteligente muestre su interfaz en el dispositivo móvil.

La comunicación a través del servidor es por medio de *Wi-Fi*. Si un objeto realiza una acción mediante algún control físico, ese cambio de estado es notificado al servidor para que actualice la base de datos. A continuación, modifica el estado de los objetos conectados y les envía un requerimiento para que realicen una acción físicamente y se muestre el cambio en el control de la aplicación de realidad aumentada como se muestra en la Figura 2.6.

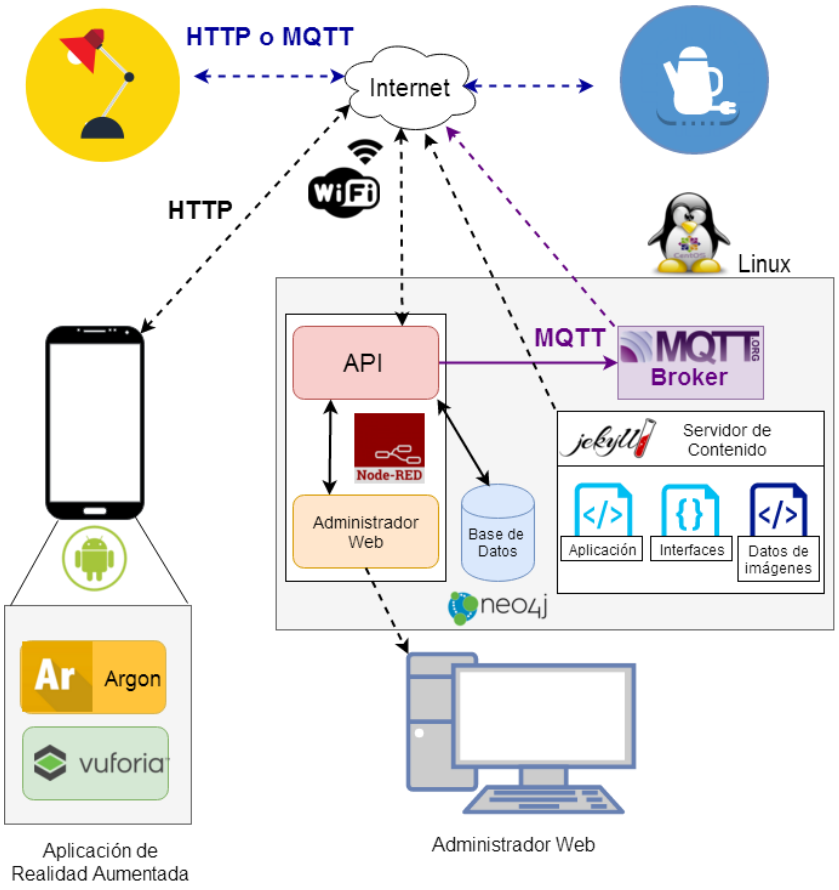

Figura 2.6: Arquitectura del sistema

La interacción de los elementos que conforman el sistema es por medio de una conexión virtual, para que al presionar el botón de un objeto inteligente se encienda el diodo emisor de luz (*LED* del inglés, *Light Emitting Diode*) de otro objeto inteligente como se muestra en la Figura 2.7.

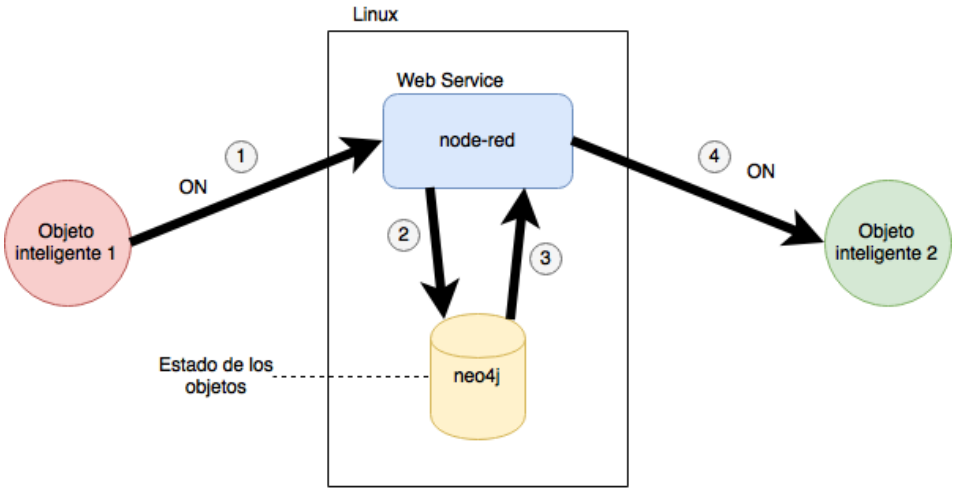

Figura 2.7: Comunicación entre dos objetos

2.3 Prototipo

Para el prototipo del administrador web utilizamos *Draw.io*. *Draw.io* es una aplicación web que permite crear diagramas de manera intuitiva y rápida. Cuenta con un módulo para agregar componentes de *Bootstrap*, el cual se utilizó para crear las interfaces de usuario.

En la página principal del administrador se muestran los dispositivos que forman parte del sistema, ver Figura 2.8.

Figura 2.8: Página principal del administrador

Los modales presentan la información básica de cada elemento. Estos modales permiten la edición y creación de dispositivos, objetos o acciones. La interfaz de usuario se adapta dependiendo si se selecciona MQTT o TCP, ver Figura 2.9.

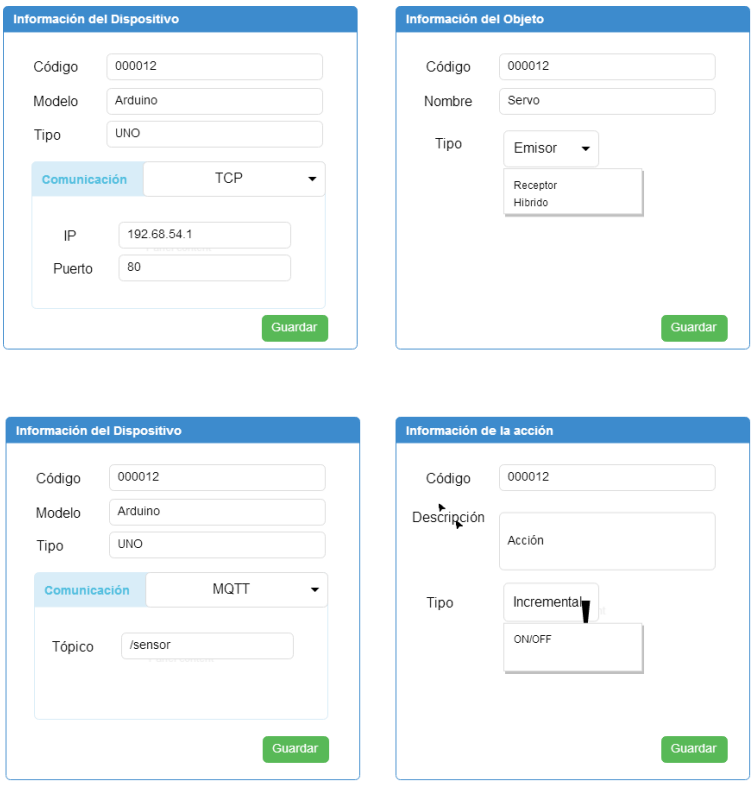

Figura 2.9: Modales para editar y crear elementos

Para el prototipo de la aplicación móvil se utilizó *Proto.io*, en la cual cuando un objeto inteligente es mostrado, la cámara detectará los marcadores y añadirá en los marcadores los nodos de conexión, estos nodos de conexión representan al objeto que consiste en emisores que envían señales o receptores que reciben señales para realizar una acción, como se muestra en la Figura 2.10.

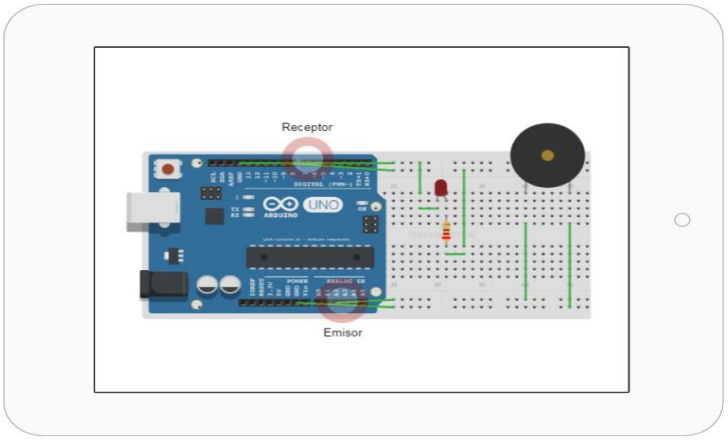

Figura 2.10: Detector de marcadores

La conexión virtual permite la comunicación entre emisores y receptores de varios objetos inteligentes, como se muestra en la Figura 2.11, así mismo se podrá eliminar la conexión virtual cuando el usuario realice un gesto de corte (swipe) en la línea de la conexión virtual.

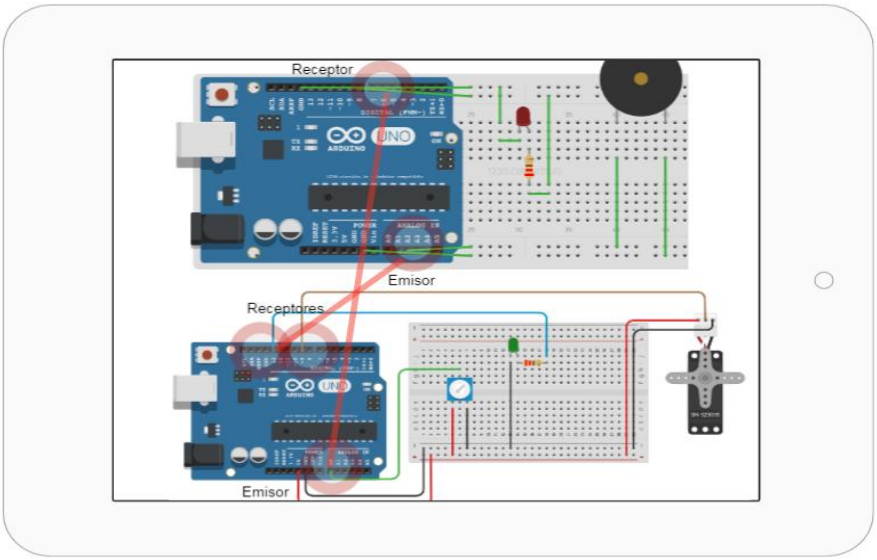

Figura 2.11: Conexión virtual

2.4 Herramientas tecnológicas empleadas

Para el éxito de nuestro proyecto, se buscó utilizar las mejores herramientas actuales para agilizar el proceso de desarrollo, poseer mayor eficiencia y efectividad.

2.4.1 Desarrollo

Para el desarrollo de la aplicación móvil se utilizó *Visual Studio Code*, que cuenta con soporte para *Typescript*. Así mismo, cuenta con plugins para *Nativescript* e integración con *Git*.

En la parte del administrador web y el *Back-End* se utilizó *Node-red* en la cual se instalaron más nodos de los que vienen por defecto en *Node-red*.

2.4.2 Debugging

Para el *debugging* del servidor, se utilizaron nodos de *debug* de *Node-red*, que permiten imprimir valores como si fuera una consola.

Para el *debugging* de la aplicación móvil se utilizó la misma herramienta *Visual Studio Code* para facilitar la depuración de las aplicaciones web. Como *Nativescript* compila aplicaciones para *Android* nativo, se necesita asociarlo con las herramientas del SDK de *Android* y el JDK de *Java*. Así mismo en la aplicación de *Android* se mostraba una consola como si fuera un navegador para visualizar errores en la aplicación móvil.

En la consola se imprimieron mensajes para saber hasta donde se ejecutaba el código, o se imprimió información sobre los objetos. La consola del navegador permite alterar objetos, por lo que se modificaron valores o ejecutaron funciones directamente para ver cómo se comportaba.

Para hacer *debugging* de los objetos se imprimieron mensajes en el *Serial* de Arduino.

2.4.3 Control de versiones

Para el control de versiones se utilizó *Git*, y a su vez se utilizó *Github* para almacenar el repositorio remoto y evitar pérdidas en caso de daños en el repositorio local.

2.4.4 Multipropósito

Se utilizó *Vagrant* para crear máquinas virtuales y poder compartirlas fácilmente mediante la nube, por ejemplo se creó una máquina virtual

para practicar la instalación del servidor de *Argon* para luego poder descargarla y usarla, hasta que se tengan todos los componentes integrados en un solo servidor.

El sistema operativo utilizado para el desarrollo fue *Windows* así que se utilizó *WinSCP* para trasladar archivos a servidores en Linux y viceversa.

CAPÍTULO 3

3. SOLUCIÓN PROPUESTA.

En este capítulo se describe en detalle cada módulo implementado para el sistema de realidad aumentada y la solución del problema planteado descrito anteriormente. Nuestra solución consiste en crear una aplicación de realidad aumentada que permite reconocer al objeto inteligente mediante un marcador y posicionar sus interfaces relativos a ella, facilitar el encadenamiento de eventos entre varios objetos inteligentes e interactuar con los objetos.

3.1 Desarrollo móvil

La aplicación móvil permite conectar objetos entre sí para encadenar acciones entre los objetos. Se empezó desarrollando el módulo de la aplicación de realidad aumentada debido a que posee mayor grado de dificultad y es el más importante para el éxito del proyecto. Inicialmente se intentó crear una aplicación de *Android* nativo colocando un *Webview* y el código para cargar una página con realidad aumentada pero la cámara no se mostraba, de igual manera no se mostraba en los navegadores de *Android*. Por lo tanto, se decidió utilizar el *Browser* de *Argon.js*.

El *Browser* de *Argon.js* se compiló y se ejecutó en el celular para usarlo en la implementación de la aplicación de realidad aumentada. Se eliminó ciertos elementos gráficos y se procedió a cargar la interfaz gráfica de *Reality Editor* combinada con *Argon.js* dentro de un *Webview* en la aplicación, tal como se muestra en la Figura 3.1. Se crearon varias carpetas en la carpeta "*components"*, para la compilación en *Android*. Se creó una carpeta llamada "*userinterface*" donde reside el código de la interfaz de *Reality Editor* y se creó una carpeta "*resources*", donde residen las librerías y elementos necesarios para *Argon.js*.

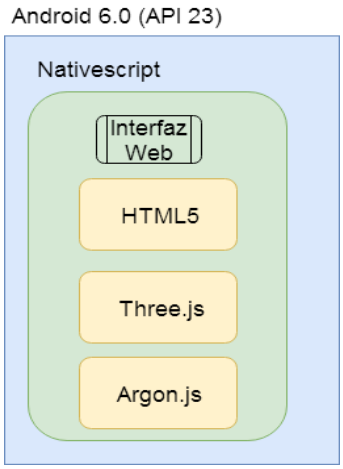

Figura 3.1: Versión inicial

Por seguridad no se puede cargar localmente los marcadores y assets de *Three.js* dentro del celular, por lo tanto se utilizó el servidor *Argon.js* para servir los contenidos, incluyendo el código de la aplicación que se había colocado previamente en la carpeta "*components*". Dicho servidor está escrito en *Node.js* y se levanta sobre *Jekyll*. *Jekyll* está diseñado para ejecutarse en Linux, por lo que se creó una máquina virtual (VM) de Ubuntu 12.04 mediante *Vagrant* y *Virtualbox*. *Vagrant* ofrece ventajas para hacer SSH, configurar la VM y subir a la nube. Se instaló y se resolvieron problemas de instalación y configuración relacionados a *Jekyll* y *Ruby*. Para que el servidor de la VM sea accesible, se realizó un "*port forwarding*" para el puerto 1337 y así poder acceder mediante la dirección IP del Host, como se muestra en la Figura 3.2.

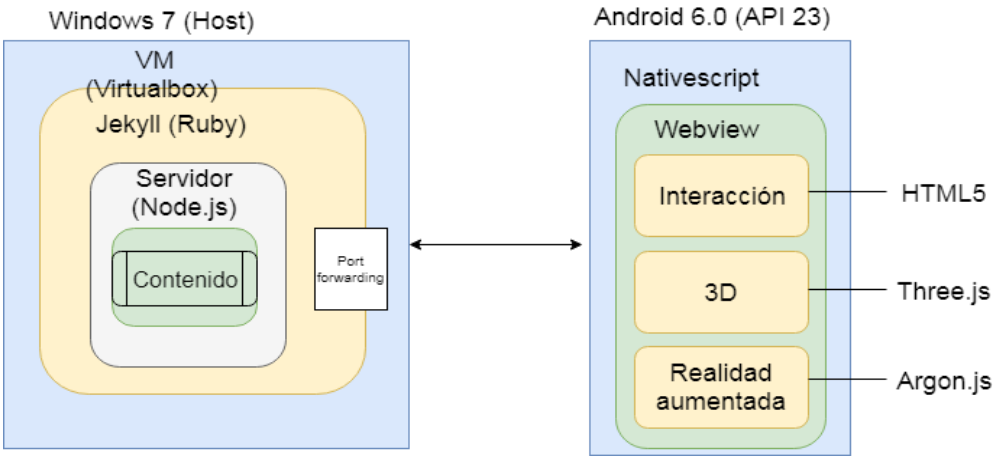

Figura 3.2: Versión con servidor

Además, se probó la detección de varios marcadores simultáneamente, mostrando contenido en 3D sobre los marcadores. Contenido como letras en 3D, e interfaces de usuario. Las interfaces de usuario son páginas web y se cargan dentro de un *Webview*. Para colocarlo en 3D, el *Webview* es transformado usando transformaciones de CSS. Así mismo se colocaron nodos de conexión sobre cada marcador, estos nodos representan a los objetos como se mencionó anteriormente.

Los dispositivos contienen varios objetos, los objetos se representan como nodos de conexión y sus acciones se representan con interfaces gráficas. Los dispositivos, objetos y acciones son identificados por un código.

3.1.1 Conectar objetos

Para la conexión entre líneas se superpuso un Canvas sobre la interfaz que muestra el video de la cámara en tiempo real y que permite colocar objetos en 3D. En dicho *Canvas* las líneas se dibujan en 2D, lo que permite dibujar y animar las líneas en el recuadro que forma la ventana de la *App*. Las líneas en 2D permiten dibujar cualquier forma en 2 dimensiones, lo que las hace más personalizables que las de 3D que usan una clase de *Three.js*. Sobre el *Canvas* se colocó un elemento HTML (*Div*) adicional para mostrar la interfaz de usuario y obtener los eventos de tocar la pantalla. La Figura 3.3 muestra dicha estructura.

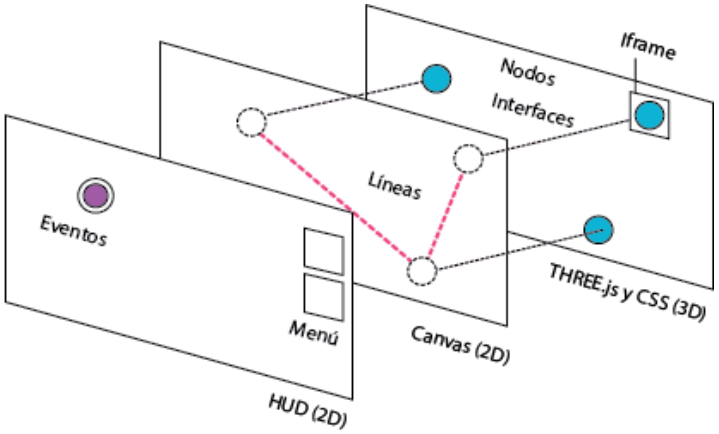

Figura 3.3: Versión con servidor y móvil

Al momento de conectar objetos, se muestran los nodos de conexión que representan a los objetos. Dichos nodos son páginas HTML cargadas dentro de un *iframe* y poseen un título indicando el nombre del dispositivo y el objeto (dispositivo-objeto). Cuando se selecciona el primer objeto que se desea conectar, este se torna de color fucsia, esperando a que se seleccione el segundo objeto para volver a su estado normal. Cuando se selecciona el segundo objeto, se crea una conexión entre ambos, en la base de datos y en la aplicación móvil. Esta conexión se representa por una línea que conecta los dos objetos seleccionados. La secuencia indicada se muestra en la Figura 3.4. Para deseleccionar un objeto, se puede tocar el objeto que está en fucsia o tocar un botón fucsia que aparece en la parte inferior derecha de la pantalla.

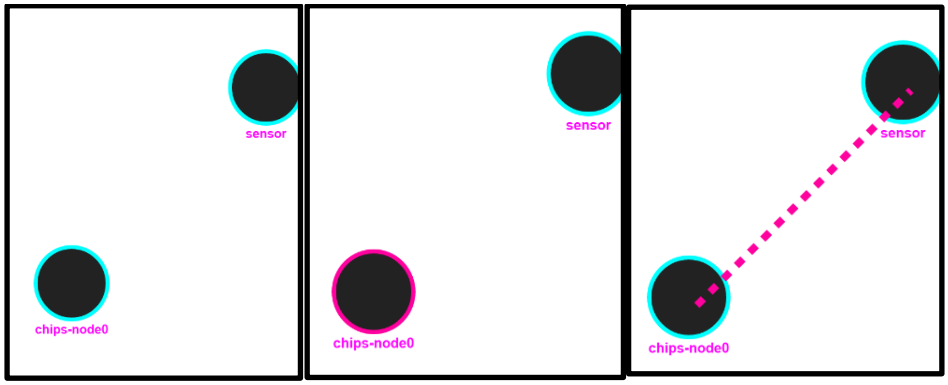

Figura 3.4: Interacción de conexión

Los objetos o nodos de conexión pueden ser de tipo emisor para enviar datos o receptor para recibir datos. Por lo tanto, para representar al emisor se utiliza un círculo y para representar al receptor se utiliza un cuadrado. Como se muestra en la Figura 3.5.

Figura 3.5: Interacción de conexión

Debido a que los datos viajan en un sentido, de un emisor a un receptor, para representar esa dirección se usaron las líneas animadas de *Open Hybrid* las cuales se mueven en la dirección en la cual se establece la conexión. Pero debido a problemas de rendimiento, se cambió a una versión en la cual hay una especie de flecha con colores indicando la dirección de la conexión, dicha línea se muestra a la derecha en la Figura 3.6.

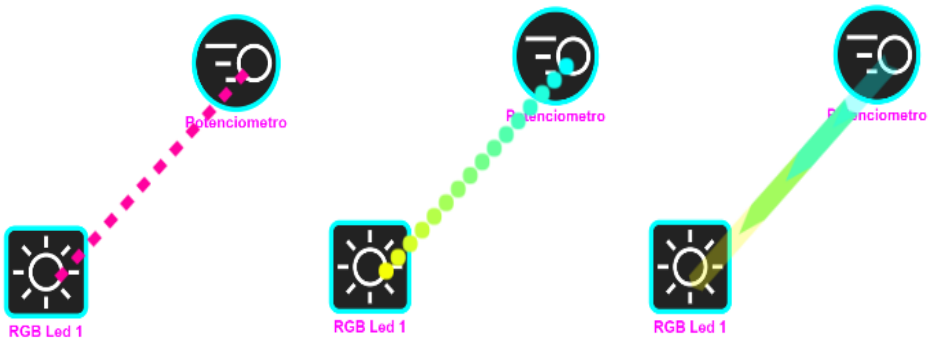

Figura 3.6: Evolución de las líneas de conexión

Como los nodos de conexión son páginas web cargadas dentro de un *iframe*, por razones de seguridad de comunicación entre diferentes orígenes, el *window* donde se ejecuta la aplicación no puede alterar directamente el estado de las páginas cargadas dentro de él. Por lo tanto, se debe programar el *window* y el *iframe* para que se envíen mensajes entre sí, cuando suceda un evento o se ejecute una función. En la Figura 3.7, se muestra un ejemplo del código necesario para habilitar la comunicación, el código del *iframe* a la derecha y a la izquierda el del *window*.

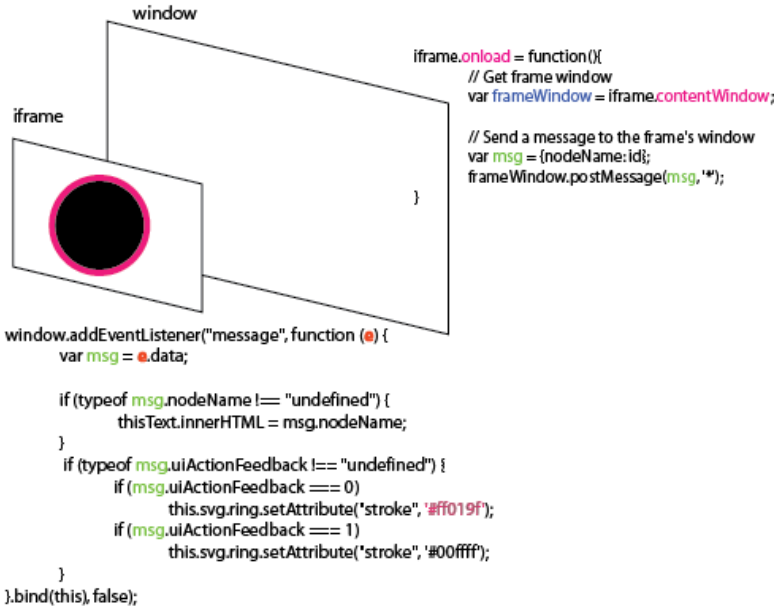

Figura 3.7: Desconexión de líneas

Al enviar mensajes a las páginas web se puede modificar partes del HTML al momento de ser cargado como por ejemplo el nombre del nodo, el ícono, el color del borde, la forma, etc. Y así es como se hizo para activar los íconos previamente definidos por *Open Hybrid*. Algunos íconos se muestran en la Figura 3.8.

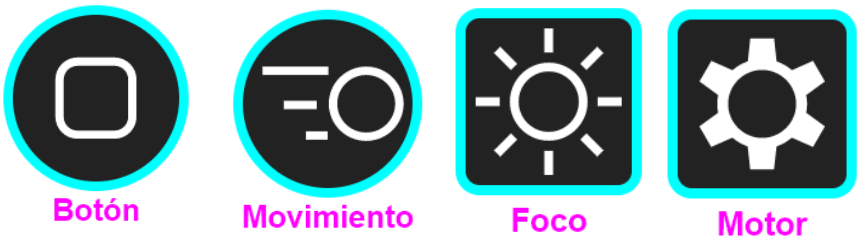

Figura 3.8: Iconos representando actuadores y sensores

3.1.2 Desconectar objetos

Se desconecta objetos mediante el corte de la línea que los conecta, el corte se muestra como una línea roja en la Figura 3.9. El corte se ejecuta tocando la pantalla, arrastrando el dedo a través de la línea para finalmente soltarlo. El punto inicial y final del movimiento es registrado para usarlo en la eliminación. Para dicho corte se utiliza una función que

detecta si dos líneas se cruzan, y si se cruzan entonces se elimina la línea de conexión. Está detección recorre todas las conexiones de todos los objetos en cada frame, tomando como una línea la conexión entre dos objetos y la otra línea está dada por el movimiento del dedo.

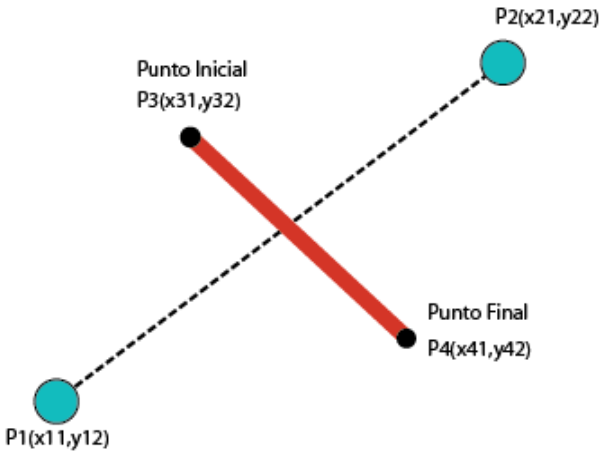

Figura 3.9: Desconexión de líneas

3.1.3 Realidad Aumentada

La aplicación utiliza Vuforia para detectar los marcadores. La aplicación detecta el marcador asignado a cada objeto y muestra las acciones disponibles para el objeto. El usuario puede seleccionar la ubicación de las acciones y objetos respecto al marcador, alterando los campos X y Y en el administrador web. El objetivo es colocar el objeto virtual cerca del objeto real en el vídeo. La aplicación posee un menú para elegir entre mostrar los objetos (Figura 3.10) o mostrar las interfaces de las acciones (Figura 3.11).

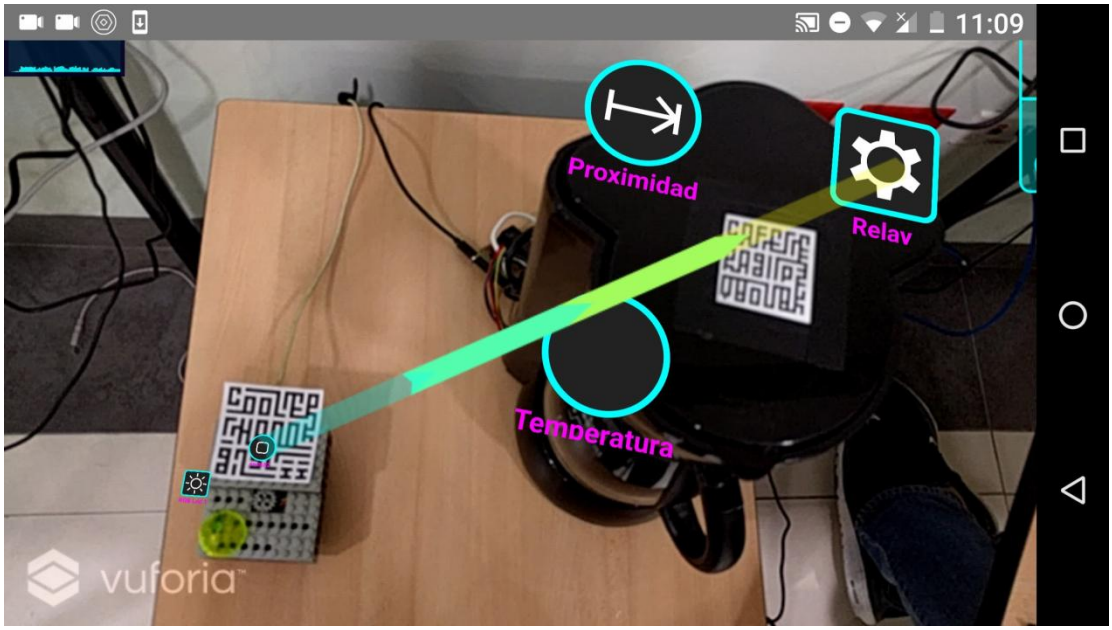

Figura 3.10: Objetos

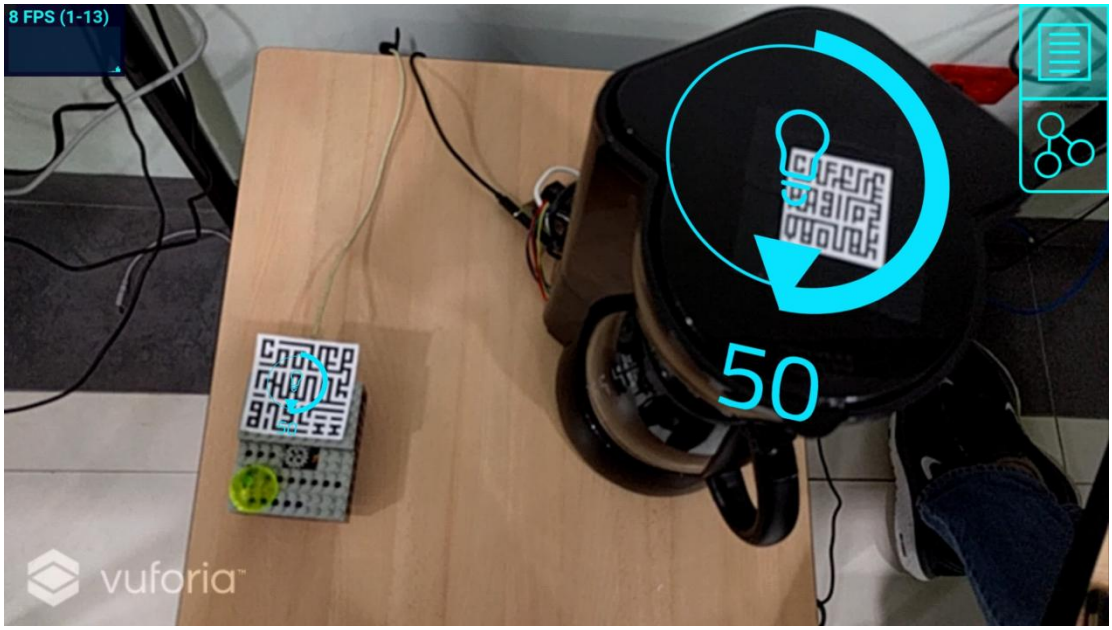

Figura 3.11: Acciones

Se comenzó el desarrollo de la aplicación usando datasets de prueba que contenían marcadores recomendados por Vuforia. Cuando se logró una detección estable de los marcadores, se procedió a seleccionar marcadores para ser usados en el sistema de los cuales se tomó en

consideración *ARMarker* [27], *Augmented Reality Generator* [28] y *Human Readable Quick Response Code* (HQRQ) [29] (Figura 3.12). Se seleccionó el último porque se puede escribir el código del dispositivo y ser reconocido por un humano fácilmente, al igual que por la aplicación ya que los marcadores generadores tienen puntuación de 5 estrellas, que es la máxima otorgada por la página de Vuforia.

Figura 3.12: Posibles marcadores

Al realizar las conexiones entre objetos, las acciones pueden ser ejecutadas mediante la interfaz aumentada o mediante el mundo real. Por ejemplo, un interruptor puede ser accionado mediante el interruptor físico o mediante un *checkbox* en la *App*, como se muestra en la Figura 3.13.

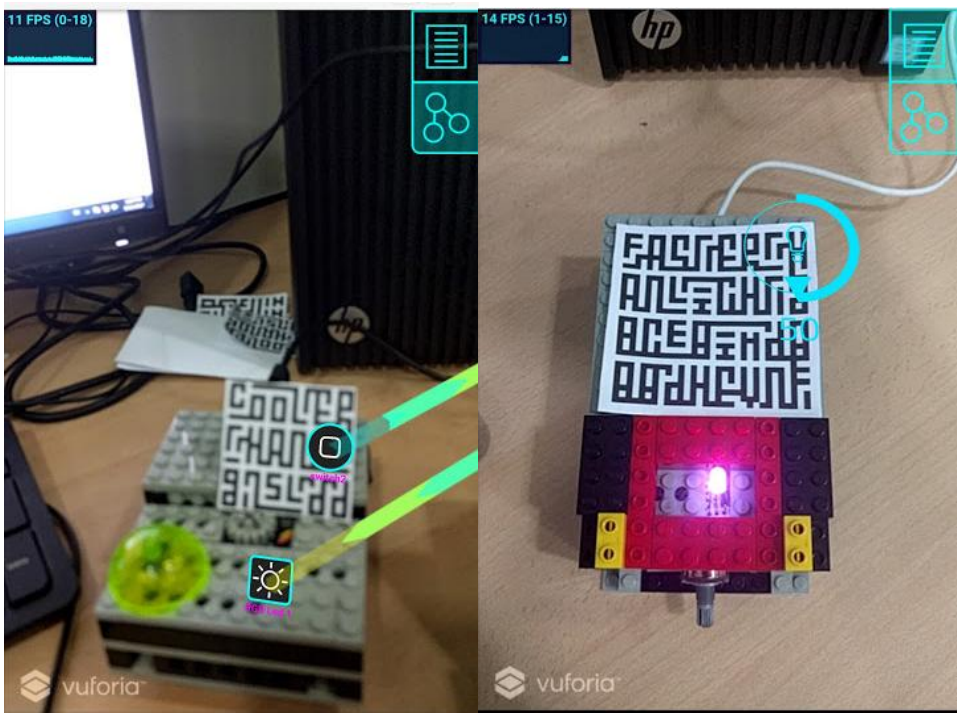

Figura 3.13: Versión Final de la aplicación

3.1.4 Prueba del Módulo Móvil

Se realizó pruebas de la plataforma de Vuforia para el reconocimiento de marcadores en la cual los marcadores deben ser reconocidos al interior del edificio con iluminación moderadamente brillante y difusa, se recomienda que no tenga sombra de otros objetos o personas.

El dispositivo móvil debe estar en paralelo al plano de la superficie del marcador para minimizar el ruido y así obtener un correcto reconocimiento y rastreo del marcador. La plataforma también permite detectar cambios en el posicionamiento de los marcadores o la cámara del dispositivo en tiempo real.

Además los marcadores que serán definidos por el usuario y enviados a la plataforma deben tener muchos detalles, un buen contraste y no deben tener patrones repetidos.

3.1.5 Dispositivos

Para probar el sistema, se programó un par de dispositivos. Dichos dispositivos consisten en varios *Particle Photon*. El *Particle Photon* es un dispositivo con *Wi-Fi* integrado que se programa como Arduino en C++.

El primer dispositivo es el “photon1”, mostrado en la Figura 3.14, que contiene un componente electrónico que en su interior está formado por tres diodos de color rojo, verde y azul (*LED* RGB del inglés, *Light Emitting Diode Red-Green-Blue*) que es un receptor para la acción de encender y apagar (si recibe un 1.0, los colores del RGB se mapean todos a 255 y el *LED* emite luz blanca, y si recibe 0.0 los colores se mapean a 0 y el *LED* se apaga), y un *push button* que es un emisor con la acción de encender (enviar el valor de 1.0).

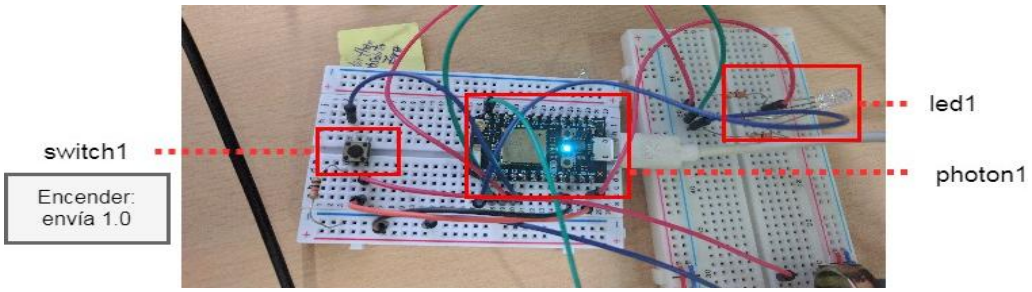

Figura 3.14: Dispositivo conocido como “photon1”

El segundo dispositivo es el “photon2”, mostrado en la Figura 3.15, que contiene un *LED* Verde que es un receptor con la acción de encender y apagar, y un *push button* que es un emisor con la acción de apagar (enviar el valor de 0.0).

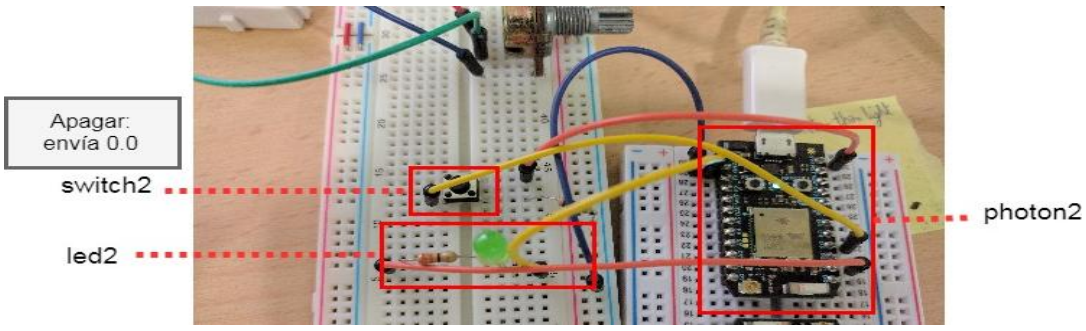

Figura 3.15: Dispositivo conocido como “photon2”

Para mejorar la apariencia de los dispositivos y mostrar su uso en aplicaciones reales, se construyeron tres objetos, el primero representa una lámpara con su respectivo interruptor para encender y apagar como se muestra en la Figura 3.16, el segundo representa una lámpara con variación de color o intensidad, y el tercero representa un motor, el cuál puede ser colocado en una puerta para abrir y cerrar o para subir y bajar una cortina, ver Figura 3.17.

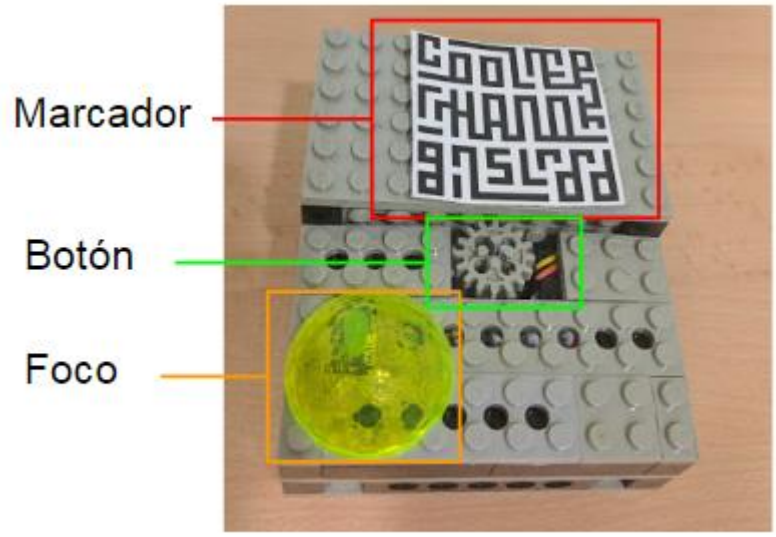

Figura 3.16: Objeto que representa una lámpara

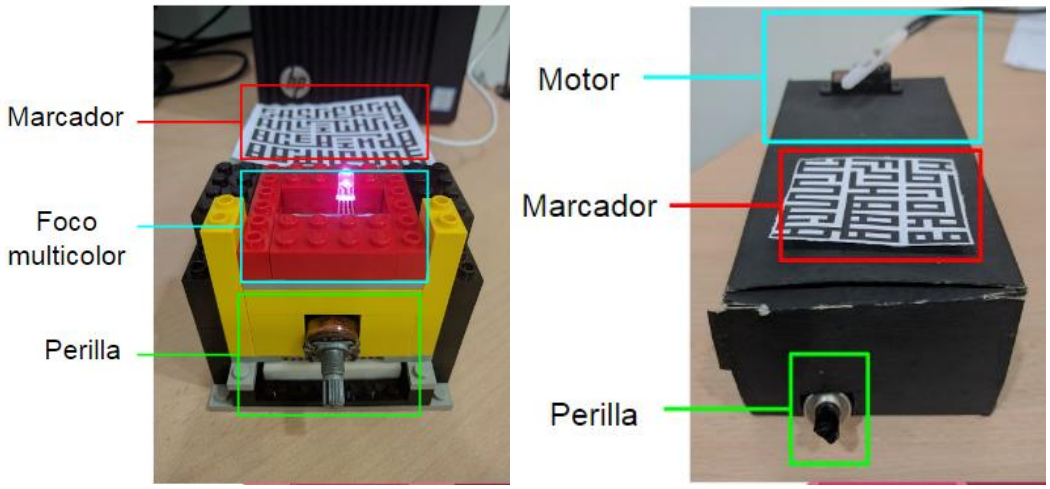

Figura 3.17: Objetos que representan una lámpara con variación de color y una puerta o una cortina accionada por una perilla.

Los dispositivos poseen un servidor TCP, un HTTP Client para recibir y enviar los datos y un cliente MQTT. MQTT es un protocolo ligero optimizado para redes no confiables y con alta latencia, generalmente usado por sensores y dispositivos móviles [30]. Cuando los dispositivos se conectan a *Internet*, envían un HTTP GET *request* con la IP obtenida al conectarse, el código del dispositivo y el puerto en el cuál se va a crear un servidor TCP que recibirá los mensajes enviados hacia el dispositivo. El servidor le responde con las acciones del dispositivo junto con su estado, para que el dispositivo pueda actualizarlo. También utiliza los datos recibidos para suscribirse mediante MQTT al canal de las acciones del dispositivo, procedimiento indicado en la Figura 3.18.

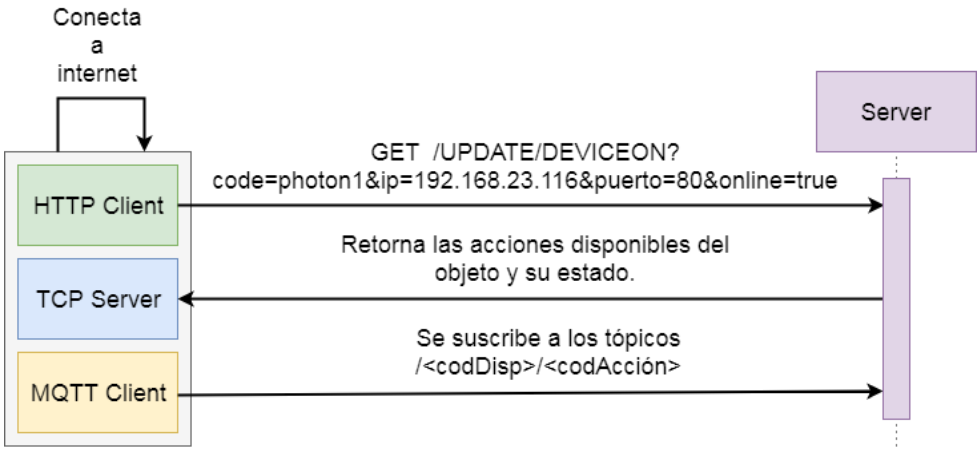

Figura 3.18: Al conectarse el dispositivo

Cuando se ejecuta una acción de un objeto, como por ejemplo presionar un botón, el dispositivo accionado enviará un HTTP GET *request* con el código de la acción ejecutada y el valor de la acción. Luego en el servidor se recorre todos los objetos conectados al objeto que realizó la acción y se envía a dichos objetos el valor de la acción ejecutada junto con el código de la acción de cada dispositivo. Dependiendo del protocolo del dispositivo, si es TCP se envía un mensaje TCP y se cierra la conexión enviando un “\n”, o si es MQTT se publica en el canal de la acción correspondiente, que el dispositivo leerá ya que se suscribió al conectarse, proceso que se muestra en la Figura 3.19.

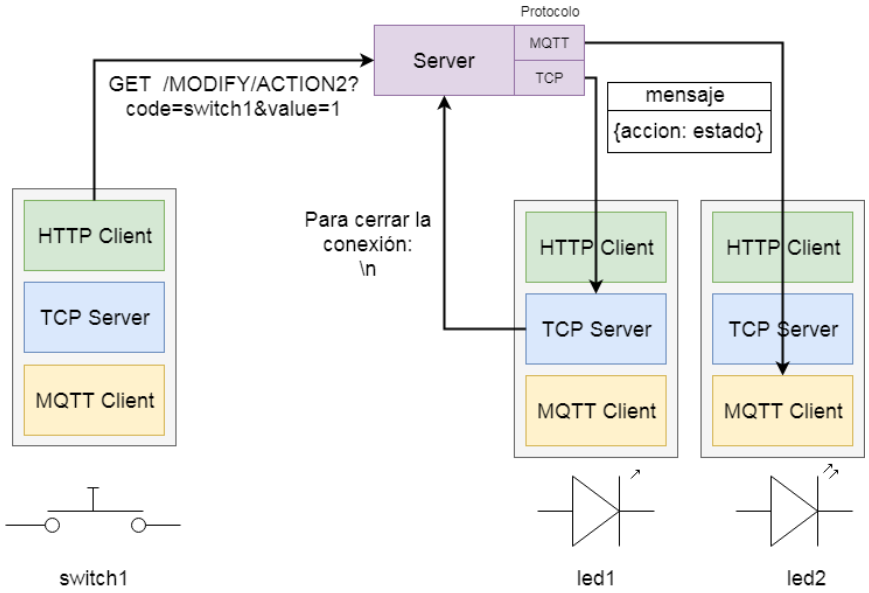

Figura 3.19: Al ejecutar una acción

3.2 Desarrollo Web

El administrador web permite al usuario ingresar objetos inteligentes para luego realizar conexiones virtuales en la aplicación móvil sin tener ningún contacto físico entre ellos.

3.2.1 Estructura de datos

La estructura de datos para el administrador web se detalla a continuación.

Dispositivo

Este contiene información del dispositivo como es el modelo, código, tipo de comunicación: TCP el cual utiliza la IP y el puerto; otro tipo de comunicación es el MQTT que utiliza un tópico, así mismo el dispositivo puede contener uno o varios objetos.

Objeto

Este representa información del objeto y se divide en dos categorías: emisor y receptor, a su vez el objeto puede poseer una o varias acciones.

Acción

Este está representado por el estado del objeto que puede ser de tipo incremental o binario, así mismo contiene el estado de habilitado que permite cambiar el estado del dispositivo tanto en la base de datos como en el dispositivo real.

En la Figura 3.20 se muestra la estructura de datos de forma visual en *Neo4j*, en la cual el nodo azul representa el dispositivo, el nodo verde el objeto y el nodo rojo la acción también se observa la vinculación entre dispositivo al objeto y del objeto con la acción.

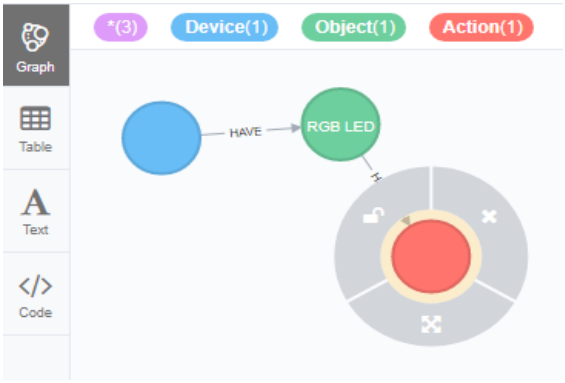

Figura 3.20: Estructura de datos

3.2.2 Implementación

La comunicación del administrador web con el *Back-End* se realiza mediante peticiones *AJAX*, en la cual se permite la creación, edición y eliminación de la estructura de datos y las relaciones que existen entre los dispositivos, objetos y acciones de los objetos inteligentes.

La interfaz de usuario del administrador web consta de varios botones para el registro, edición y eliminación de objetos inteligentes también la importación y descarga del mismo, además se incluye la opción para consultar los objetos inteligentes, añadir un marcador y una interfaz al objeto como se muestra en la Figura 3.21.

Figura 3.21: Página Principal

El diseño del administrador web se realizó con un *framework* llamado *Bootstrap* para cada uno de los modales en la creación, edición y eliminación de los objetos inteligentes. En el modal de registro del dispositivo se proporciona la información del mismo, como se muestra en la Figura 3.22.

En los modales de registro del objeto y la acción se proporciona información del mismo y hace referencia a los dispositivos o a los objetos que se encuentran en la base de datos para su vinculación como se muestra en la Figura 3.23 y en la Figura 3.24.

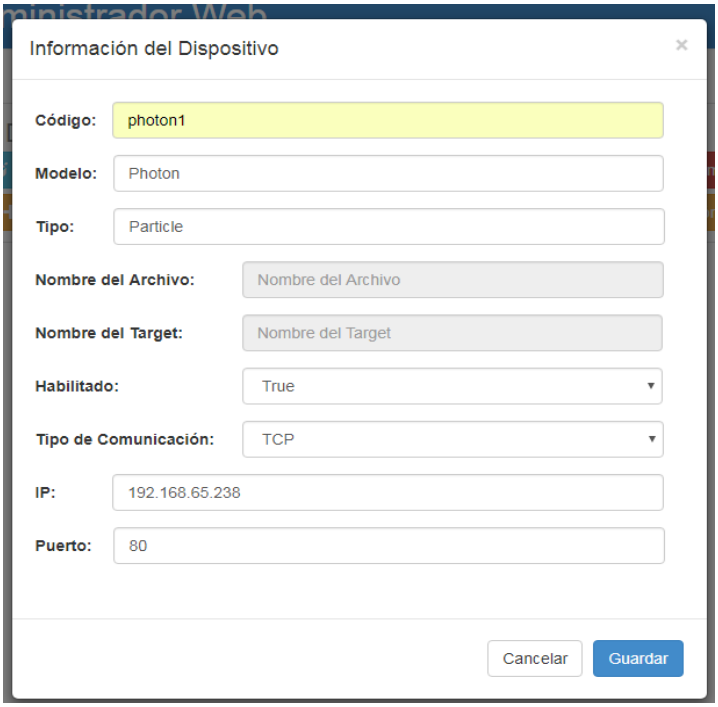

Figura 3.22: Registro del Dispositivo

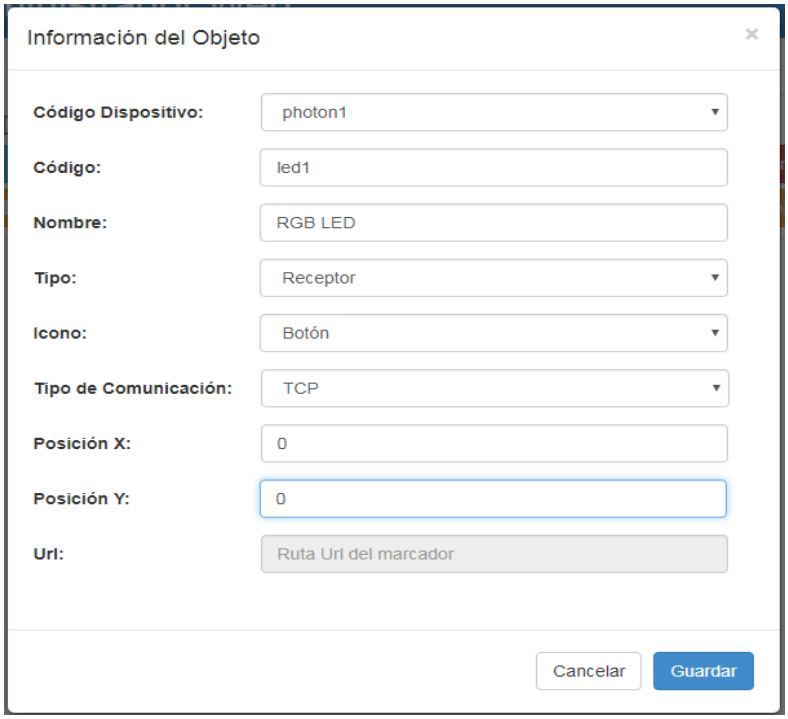

Figura 3.23: Registro del Objeto

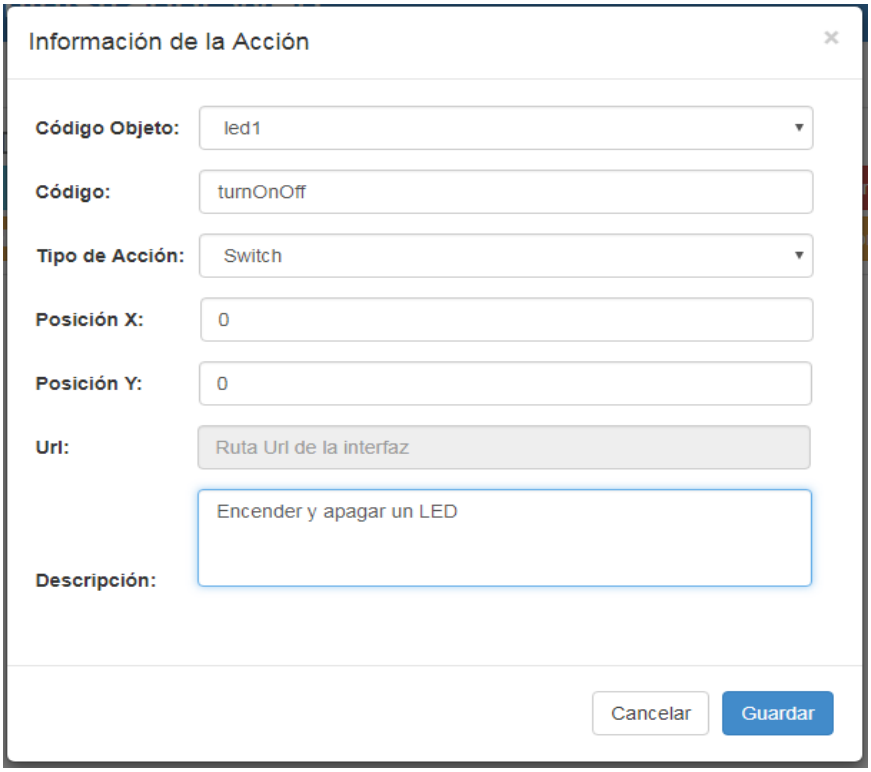

Figura 3.24: Registro de la Acción

Para editar cada objeto inteligente se proporcionará un *dropdown* mostrando los objetos inteligentes que se encuentran en la base de datos y al seleccionar uno de ellos se muestran los datos en el formulario para su edición, como se muestra en la Figura 3.25.

En la parte de eliminación se proporcionará el código del dispositivo, objeto u acción y este elimina el nodo y sus conexiones, también en añadir marcadores e interfaces y consulta se proporcionará el código del dispositivo como se muestra en la Figura 3.26.

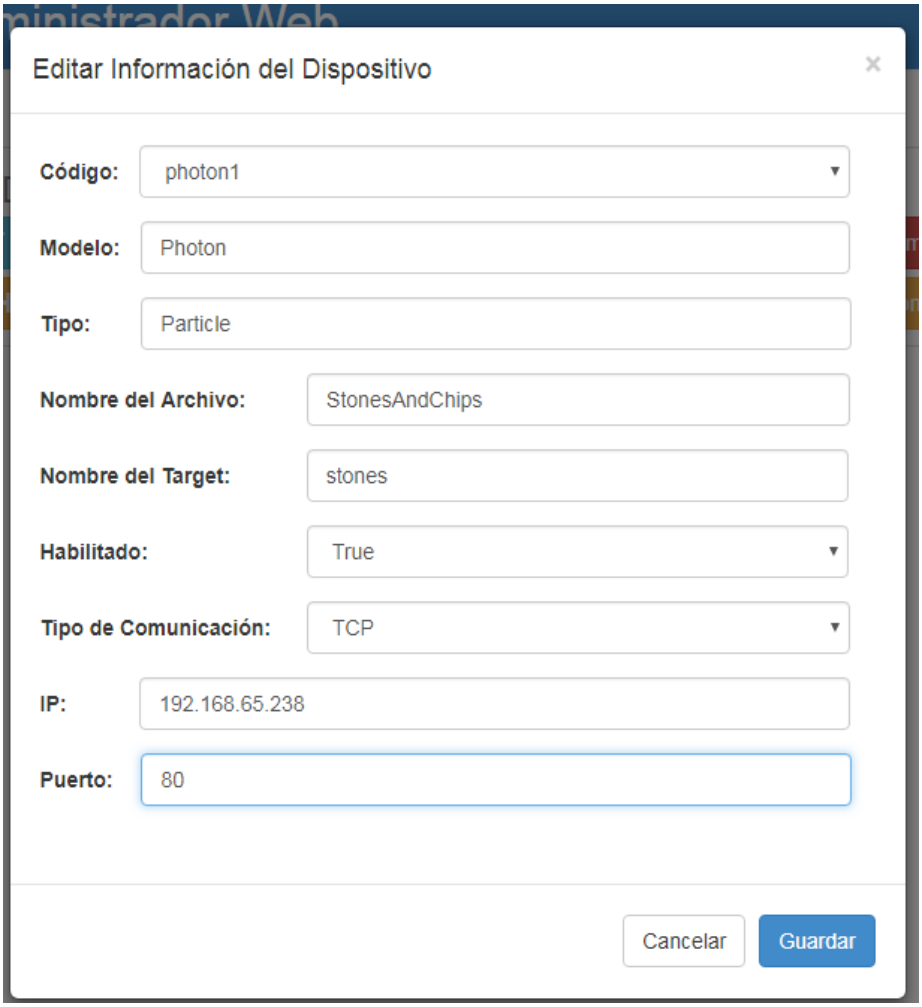

Figura 3.25: Editar Dispositivo

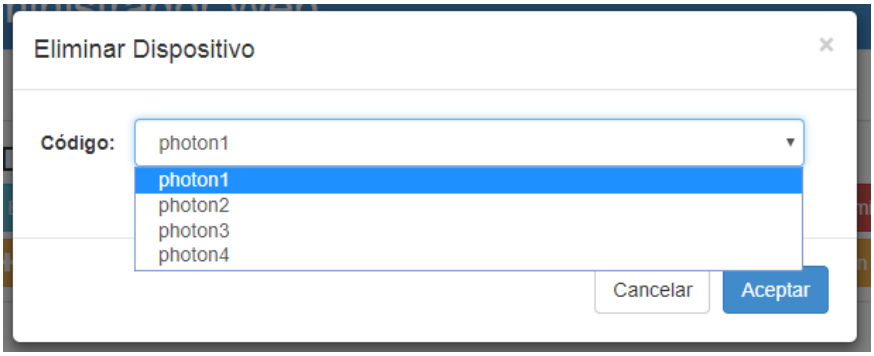

Figura 3.26: Eliminar Dispositivo

En la sección de *Target* o marcadores como se muestra en la Figura 3.27 se podrá utilizar la página de Vuforia directamente en el

administrador web, subir la base de datos y el marcador con un arrastrar y soltar al servidor con la ayuda de la librería *Dropzone*.

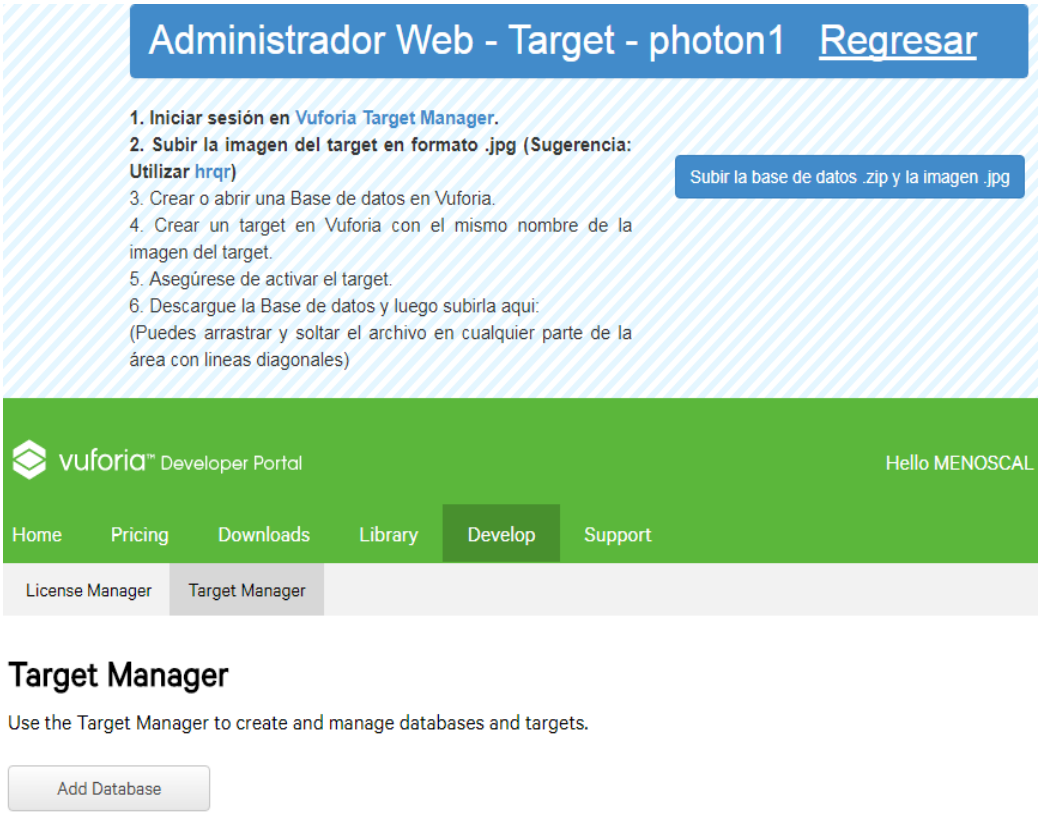

Figura 3.27: Página Target

En la sección de Interfaz en la Figura 3.28 se muestra el contenido de las carpetas de interfaz y *target*, estas carpetas se incluyen en cada directorio del objeto inteligente mostrado en la parte superior de la página y estas carpetas se añaden automáticamente al subir archivos al servidor en la que además podrá visualizar o eliminar según las necesidades del usuario, cabe señalar que para añadir archivos a la carpeta *target* deberá ir a la página target del administrador web.

Figura 3.28: Página Interfaz

En la sección de descarga e importación de la base de datos se las realiza en formato CSV con la cabecera al inicio del archivo y con separación por coma para cada atributo.

En la parte de visualización de los objetos inteligentes se utilizó la librería *Neovis.js* y se encuentra en la sección de consulta que por medio del código del dispositivo se visualizan las conexiones de los nodos, similar a la base de datos *Neo4j* como se muestra en la Figura 3.29.

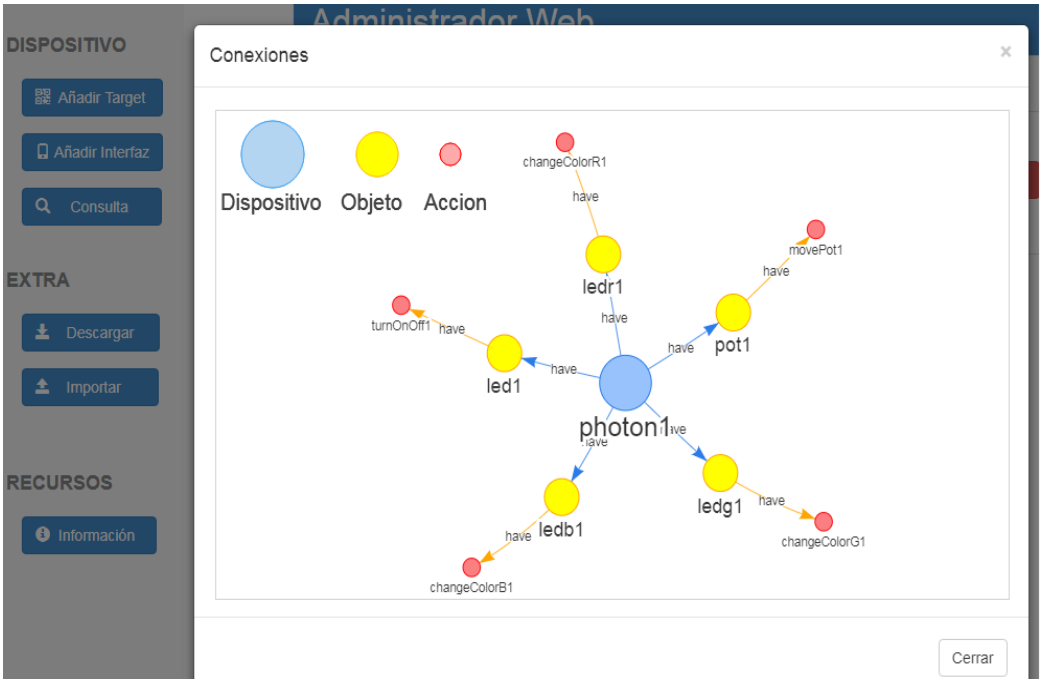

Figura 3.29: Visualización de los objetos inteligentes

3.3 Back-End

El *Back-End* está desarrollado en *Node-red* este contiene la estructura del administrador web, así mismo contiene las validaciones para conectarse a la base datos.

Cada acceso a la base de datos lo hace a través del *Back-End* por medio de *Cypher query* que envía peticiones HTTP a la base de datos *Neo4j* para su actualización.

3.3.1 Estructura del Administrador Web

Cada vez que el usuario accede al administrador web se mostrará el nodo *index.html* que contiene el template de la página principal, la parte del css y *Javascript* se encuentran en nodos separados así mismo la parte de la comunicación *AJAX*, como se muestra en la Figura 3.30.

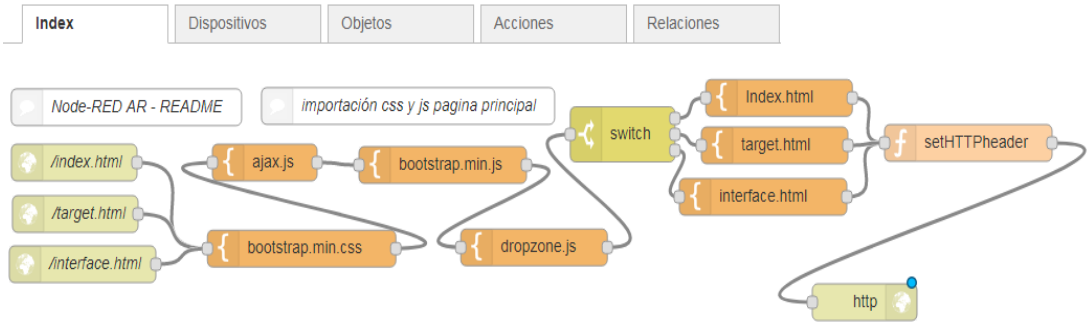

Figura 3.30: Flujo del Administrador Web

En la Figura 3.31 se muestra el flujo de directorios para la creación de carpetas de interfaz y *target* en el servidor.

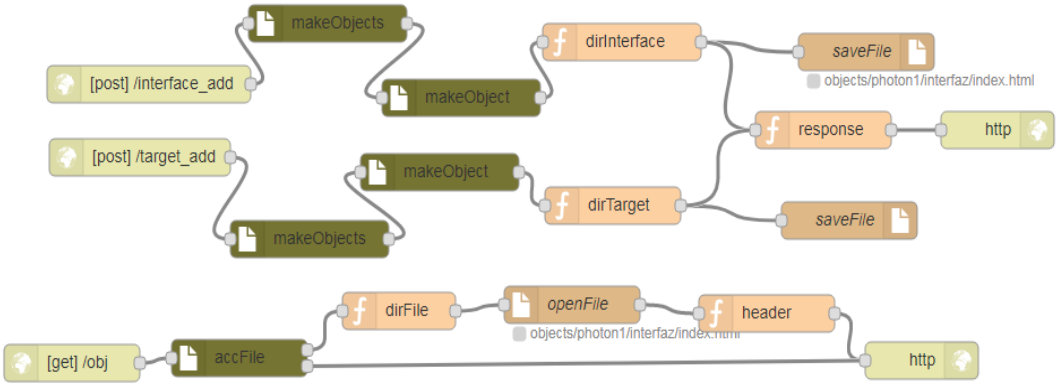

Figura 3.31: Flujo de Directorios

3.3.2 Flujos del Back-End

Node-red utiliza flujos las cuales se utiliza para separar cada uno de las validaciones y acceso a la base de datos, en el flujo de Dispositivos mostrado en la Figura 3.32, se recepta las peticiones AJAX por medio de *Webservice* y de métodos POST en cada creación, modificación, consulta y eliminación de la base de datos y así mismo en cada flujo de objetos, acciones y relaciones.

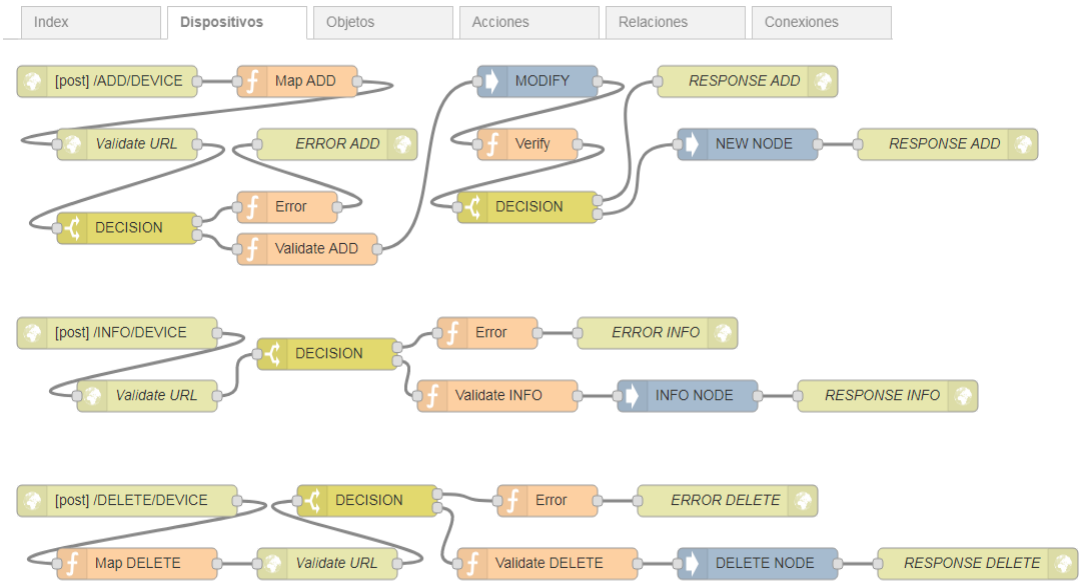

Figura 3.32: Flujo de los Dispositivos

En el flujo de relaciones se validará y se creará el vínculo entre dispositivos, objetos y acciones, como se muestra en la Figura 3.33.

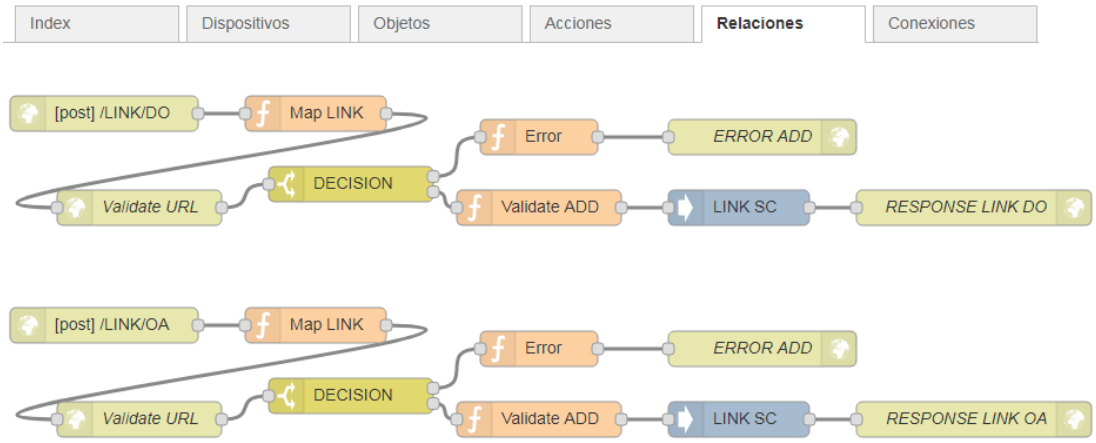

Figura 3.33: Flujo de las Relaciones

En la Figura 3.34 se muestra el flujo que permite actualizar la IP del dispositivo cuando se conecta a *Internet*. Primero se valida que el dispositivo exista, si existe se modifica con la información enviada, caso contrario se envía un mensaje de error. Luego se obtienen todos los objetos del dispositivo y se envía un *JSON* que contenga todas las acciones del dispositivo y su estado actual.

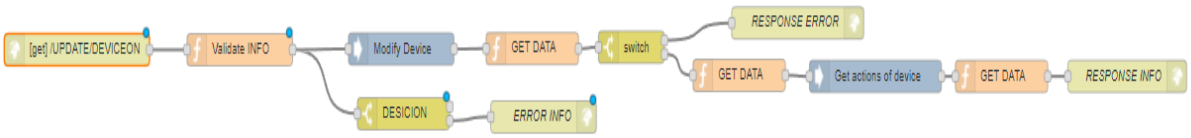

Figura 3.34: Flujo de Actualizar IP

Para cargar la información de la base a la aplicación móvil, sobre dispositivos, objetos, acciones y sus conexiones, se elaboró un flujo (Figura 3.35), en el cual se obtienen todos los dispositivos, por cada dispositivo se obtienen sus objetos y los objetos a los que está conectado, para en base a esos objetos generar las conexiones y colocarlas en un formato que la aplicación móvil pueda entender. Luego se une toda la información para crear un *JSON* que contenga dispositivos, donde cada dispositivo contiene varios objetos y cada acción contiene varias acciones, también el *JSON* contiene un campo para las conexiones.

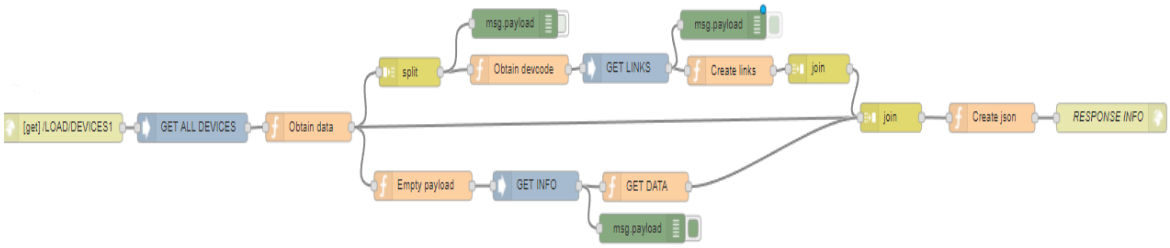

Figura 3.35: Obtener toda la información del servidor

El flujo de ejecutar una acción, primero valida que la acción exista, si existe cambia su estado, sino envía un mensaje de error. Luego obtiene el objeto correspondiente a la acción y busca los objetos conectados al mismo, para actualizar el estado de sus acciones con el mismo estado y así sucesivamente hasta haber recorrido todos los objetos conectados. Por cada acción ejecutada, se obtiene el dispositivo y se envía un mensaje TCP o MQTT, según el protocolo utilizado por el dispositivo; mensaje que contiene el código y estado de la acción. Un extracto del flujo se puede ver en la Figura 3.36.

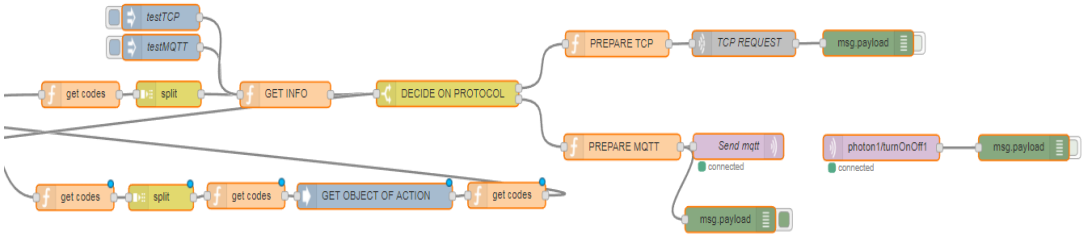

Figura 3.36: Actualizar acciones dependiendo del protocolo

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS.

En este capítulo se describe un análisis detallado de los resultados y casos de uso del sistema desarrollado.

4.1 Casos de Uso

En este capítulo hemos analizado dos tipos de usuario, el usuario administrador y el usuario final. A continuación se presentan los casos de uso del sistema de acuerdo al usuario.

4.1.1 Usuario Administrador

Para realizar las pruebas del usuario administrador se realizaron los siguientes casos de uso:

Añadir dispositivos al sistema

El usuario crea un dispositivo en el administrador web con sus objetos y acciones, luego sube el marcador y la base de datos de Vuforia en la página *target* en la cual la ruta de los archivos quedan guardadas en la base de datos para ser accedida por el usuario final en la aplicación móvil, también se añade interfaces en la página interfaz como se mencionó en el capítulo 3, con estas recomendaciones el usuario procedió a ejecutar el administrador web.

Al finalizar el usuario sugirió que ciertos campos en el registro del administrador web se deshabiliten y se coloquen solo en la parte de edición, también cambiar la parte del registro de coordenadas para la colocación de nodos y de interfaz por medio de listas desplegables que indiquen la posición como por ejemplo en el centro o arriba del marcador.

Configurar dispositivos

Para configurar los dispositivos el usuario programa el dispositivo colocando los códigos del dispositivo, objetos y acciones, tal cual se

nombraron en el administrador web. Y usa funciones para actualizar el estado, recibir un mensaje de acción y ejecutar la acción, y comunicarse por MQTT y HTTP. Está parte es pensada para los desarrolladores de los dispositivos y no para el usuario final. Ya que la idea del sistema es usar el estándar creado por *Open Hybrid* para conectar dispositivos en *Android*.

4.1.2 Usuario Final

Para realizar las pruebas del usuario final se realizaron los siguientes casos de uso:

Accionar objetos directamente

En la Figura 4.1 se muestra al usuario utilizando la aplicación móvil, la aplicación detecta el marcador que identifica al objeto inteligente y le muestra las acciones disponibles para el dispositivo mediante interfaces en realidad aumentada. En este caso una cafetera inteligente que cuenta con acceso a *Internet* y su propia aplicación, pero que puede ser encendida y apagada mediante nuestra aplicación.

Figura 4.1: Detección del marcador

Una vez que el usuario realiza una acción mediante la interfaz del móvil, como encender la cafetera; la retroalimentación se muestra al cambiar el estado del botón virtual y encender el botón del *switch* de la cafetera,

indicando que está calentando, como se muestra en la Figura 4.2.

Figura 4.2: Encender cafetera

Conectar objetos

El usuario también conectó varios objetos inteligentes entre sí, conectando un interruptor independiente que representa el encendido/apagado de la cafetera. La conexión se realiza mediante una conexión virtual en el móvil como se muestra en la Figura 4.3.

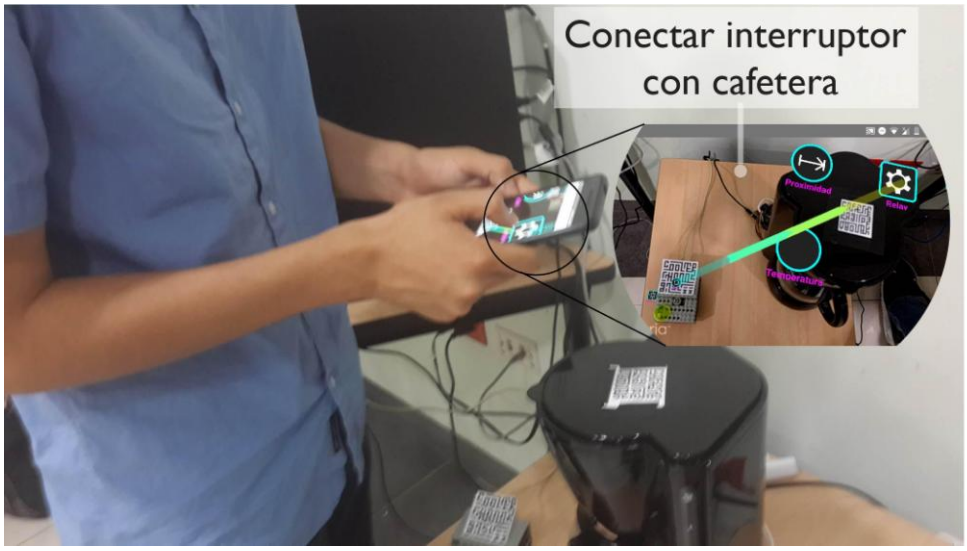

Figura 4.3: Conectar cafetera con el interruptor mediante la aplicación móvil

En la Figura 4.4 se muestra cuando el usuario terminó de programar la cafetera para que pueda ser encendida por otro objeto inteligente, un interruptor, permitiendo encender la cafetera al presionar el interruptor.

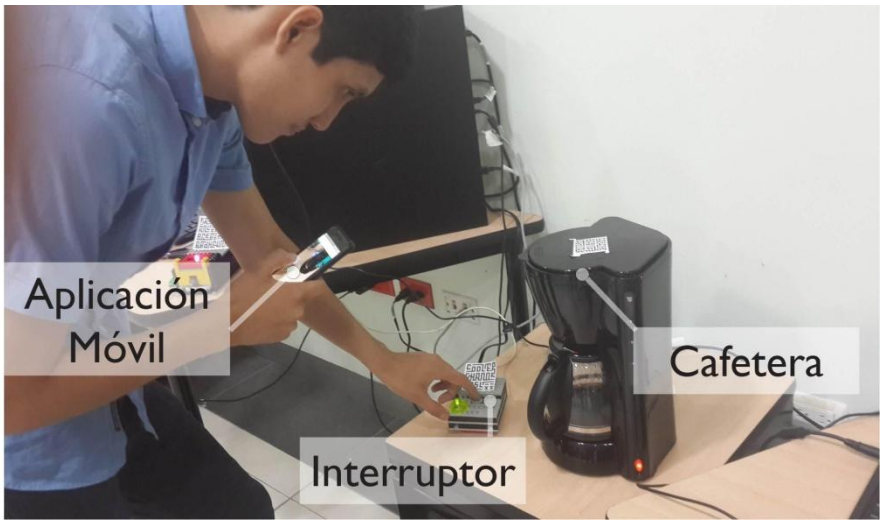

Figura 4.4: Encender cafetera con el interruptor (conectados previamente mediante la aplicación móvil)

En las pruebas realizadas el usuario se mostró satisfecho con el sistema de reconfiguración de objetos inteligentes en la detección de marcadores

y la de unir objetos para realizar nuevas acciones.

La retroalimentación recibida de parte del usuario indica que el flujo de información en las conexiones es bidireccional, cuando en realidad tiene un sentido definido por unas flechas. Lo que indica que debemos hacer más notoria la dirección o buscar otra forma de indicar la dirección. En base a esto, para ayudar al usuario a seleccionar el emisor y el receptor, se hizo que cuando todavía no se ha seleccionado el emisor, todos los receptores queden deshabilitados y de color gris (como se muestra en la figura 4.5), para que el usuario conozca que solo puede tocar los emisores de color celeste, y cuando lo selecciona, los emisores se desactivan y se activan los receptores.

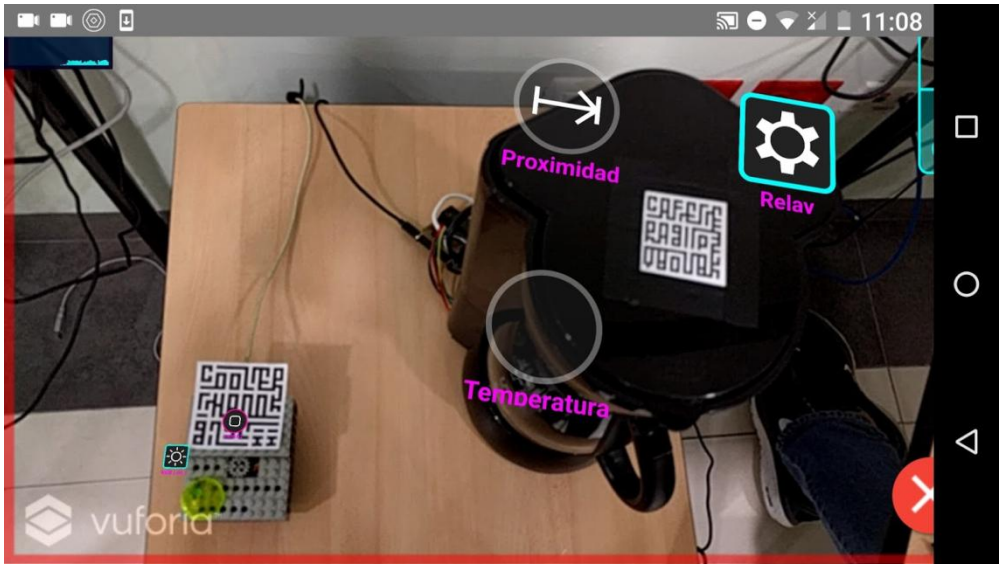

Figura 4.5: Visualización de objetos a conectar

El usuario sugirió que los nodos de conexión deberían aparecer en la parte donde no se obstaculice la visión del objeto, como por ejemplo arriba del objeto o encima de los marcadores, tal como se muestra en la esquina inferior izquierda de la Figura 4.5 (*Switch*). También sugirió mejoras en los iconos del menú del móvil, independientemente de qué ícono sería el adecuado, y propuso hacer los nodos de conexión más pequeños.

Además los usuarios recomendaron que el tamaño de los dispositivos deben tener concordancia con la acción que ejecutan; por ejemplo, sobre el dispositivo con el *LED* multicolor un usuario dijo que “solo hace eso” refiriéndose a cambiar el color y encender el *LED*, ya que debido a su tamaño creía que accionaba un motor o algún movimiento. Algunos dispositivos eran grandes porque en su interior tenían *protoboards* con sus respectivas resistencias, cables, sensores y actuadores, lo cual se puede simplificar mediante una placa de circuitos integrados. Por motivos de tiempo y para pruebas se realizó con *protoboards*, legos y algunos objetos cotidianos. Sin embargo, esta mejora puede ser considerada como parte de un trabajo futuro.

También se realizaron pruebas de conectar un servo con otro dispositivo con perilla, para mover el servo a la distancia o cambiar el color de un *LED* multicolor mediante la rotación de la perilla. Adicionalmente se usó la perilla para apagar y prender un foco, y se usó un botón para mover el servo directamente al final o al inicio, el servo representa una puerta o cortina inteligente.

El sistema posee variedad de usos, uno de ellos es la automatización usando sensores, un ejemplo es colocar un sensor de luminosidad afuera de la casa y por medio de la aplicación, conectar el sensor a los focos de la casa de tal manera que en el día se apaguen, y en la noche se enciendan automáticamente, pudiendo ser uno, varios o todos a la vez. Si el usuario cuenta con cortinas puede hacer que se sincronicen igual que los focos. Existen dificultades para encender un foco cuando se apaga la luz y el cuarto es oscuro, ya que el marcador no es lo suficientemente visible para ser detectado y mostrar la interfaz para accionar el foco, a menos de que estuviera conectado previamente a un interruptor. Una alternativa para iluminar el marcador es la de activar el *flash*, o pintar la pantalla de blanco y reconocer con la cámara frontal como lo hace la aplicación de *Photos* de *Google* para fotos con poca luz.

Accionar dispositivos a la distancia

Usando un marcador lo suficientemente grande se podría accionar y conectar dispositivos a la distancia, útil para personas discapacitadas o ancianos. Se imprimó un marcador cuyos lados son del ancho de una hoja A4, el cual se pudo detectar desde 2 metros, la detección permite mostrar las acciones y objetos mediante la interfaz aumentada, lo que permite accionar dispositivos a distancias cortas como se muestra en la Figura 4.6. Se estima que un mayor tamaño del marcador permitiría alcanzar mayores distancias, aunque no es viable crear marcadores grandes. Se recomienda buscar otra forma de detectar dispositivos a la distancia, como futuras mejoras.

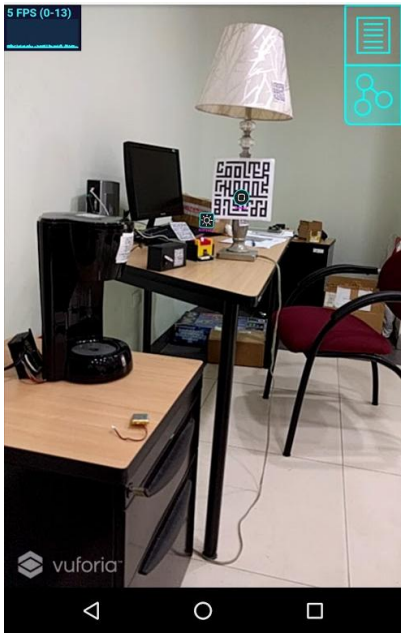

Figura 4.6: Accionar dispositivo a la distancia

Rendimiento

En un dispositivo *Nexus 6P* con *Android N*, la aplicación se mantiene estable entre 3 a 15 fotogramas por segundo (FPS del inglés, *Frames Per Second*), con caídas ocasionales que llegan a 0-3 FPS cuando hay varios marcadores a detectar, varios objetos, o varias interfaces; debido probablemente a que los *iframes* que contienen los objetos y las

interfaces no se actualizan en el *update listener* que provee Argon porque son páginas separadas, lo cual puede ser implementado como futura mejora. Cuando suceden estas caídas la aplicación tarda en reaccionar o se inhibe. Aunque entre 3 a 15 FPS, funciona correctamente y se puede interactuar sin problemas. Cuando se inicia la aplicación en modo horizontal, la aplicación se cae, lo cual es un *bug* del *branch* del *Browser* de *Argon* en el que se basa la aplicación.

Se probó con marcadores de diferentes tamaños, 3, 6 y 10 cm. Los marcadores menores de 3 cm no son detectados, de 3 cm en adelante si los detecta. Aunque esto depende de la resolución de la cámara, distancia cámara-marcador, imagen del marcador (generados por HRQR [29]). Quizás se deba al enfoque de la cámara, que al acercarse, el marcador se ve borroso y los *features* de los marcadores no son reconocibles. La aplicación está diseñada para detectar dos marcadores simultáneamente y conectar dispositivos, aunque se puede aumentar el número, mediante una variable que indica el número de marcadores a detectar simultáneamente.

CONCLUSIONES Y RECOMENDACIONES

Se implementó un sistema gráfico de realidad aumentada para identificar y accionar objetos inteligentes. Así mismo permite conectar objetos inteligentes con otros objetos inteligentes para reprogramar las funcionalidades para las que originalmente fueron propuestos y emplearse en otras configuraciones. Para lo cual se integró los dispositivos y la aplicación con el *Back-End* por medio de un *API REST*.

El potencial del sistema desarrollado puede ser utilizado en distintos entornos como se mencionó anteriormente, uno de ellos es en el hogar para que los objetos estén controlados como un sistema total para mejorar la eficiencia de las tareas en el hogar.

Otro uso potencial es la capacidad del sistema en ayudar a personas ancianas o con alguna discapacidad para que ellos puedan realizar actividades básicas rápidamente como encender una lámpara de su cuarto con el teléfono móvil, sin necesidad de pedir ayuda a un familiar o en caso de emergencia encender una alarma para que el familiar, vecino o médico se alerte de algún problema que existe con la persona discapacitada.

Como fortalezas del sistema tenemos que conectar objetos se torna una cuestión de números, ya que se transmiten números entre 0.0 y 1.0 para realizar acciones. Depende de cada dispositivo configurar las acciones a realizar cuando el objeto reciba el mensaje con el número. También la aplicación permite cargar contenido web, permitiendo crear contenido de manera rápida o usar contenido de aplicaciones web existentes. Como debilidades de la aplicación tenemos las líneas y el rendimiento, ya que a veces los puntos finales de las líneas se mueven del lugar del objeto, arruinando la inmersión y experiencia del usuario.

La metodología inicial tuvo sus falencias, pero se la corrigió sobre la marcha. El plan inicial era usar todo el proyecto de *Open Hybrid*, servidor e interfaz web, pero debido a problemas al migrar a *Android* se procedió a seleccionar partes del código para usarlas en nuestra aplicación y se mejoró el *Back-End* de intgIO

para integrar el administrador web, el móvil y los dispositivos. Otro problema fue no haber tomado en cuenta la programación y el diseño de los dispositivos, lo que se debió considerar como un módulo adicional.

Las herramientas utilizadas fueron las adecuadas. *Nativescript* permite compilar aplicaciones para *Android* y *iOS*, aunque solo fue probada en *Android*, también tendríamos una aplicación de *iOS*. Argon resultó ser un *framework* muy útil para la creación de la aplicación de realidad aumentada ya que permite usar tecnologías web, y posee funcionalidades de Vuforia, así mismo el proyecto de *Open Hybrid* nos permitió reutilizar sus interfaces y código.

Para usar la aplicación se recomienda utilizar un celular *Android* 6.0 (API 23) para que soporte la realidad aumentada.

Como trabajos futuros se plantea la mejora del rendimiento de la aplicación para que pueda contar con líneas de conexión animadas en la dirección del flujo de información, usadas en el proyecto pero descartadas por bajo rendimiento. También se plantea la implementación de filtros para mostrar solo los dispositivos de un mismo propósito como focos, motores, etc, o para mostrar los del mismo tipo como receptor o emisor. Una mejora adicional es guardar fotos de los objetos con sus interfaces, para la operación de los dispositivos a distancia. Se podría implementar una interacción similar al *Smart Remote* de *Sevenhugs* [31].

Otra recomendación sería la de usar otro tipo de identificación de dispositivos, se puede usar *Bluetooth* para detectar cuando se está cerca del dispositivo a accionar, pero no se usó porque no todos los dispositivos cuentan con *Bluetooth*. Otras formas podrían ser RFID, Código QR, las ondas electromagnéticas del dispositivo, reconocimiento de imágenes, etc.

También se podría considerar cambiar el *Back-End* por uno más general y diseñado para la conexión de dispositivos, como lo es *IoTivity* [32], lo que no se consideró debido a que el proyecto consistía en usar el sistema implementado en la materia integradora anterior y probar que la aplicación móvil funcione.

Otro trabajo futuro podría ser la integración de la aplicación móvil con gafas de

realidad aumentada como por ejemplo *EPSON Moverio BT-200* que permite ejecutar aplicaciones *Android* [33].

BIBLIOGRAFÍA

[1] Open Hybrid, (2017, Mayo 13). Direct Mapping: The need for new visual Paradigms

for the Operation of the Internet of Things [Online]. Disponible en: http://openhybrid.org/direct-mapping.html

[2] Business Insider, (2016, Septiembre 8). There will be 24 billion IoT devices installed on Earth by 2020. Disponible en: http://www.businessinsider.com/there-will-be-34-billion-iot-devices-installed-on-earth-by-2020-2016-5

[3] P. Moorhead, (2013, Septiembre 26). The Problem With Home Automation's Internet Of Things (IoT) [Online]. Disponible en: https://www.forbes.com/sites/patrickmoorhead/2013/09/26/the-problem-with-home-automations-iot/#4e94e33d70ec

[4] Valentin Heun, Shunichi Kasahara, Pattie Maes. "Smarter Objects: Using AR technology to Program Physical Objects and their Interactions" ACM Intl. Conf. Human Factors in Computing (CHI 2013).

[5] J. A. Somolinos Sánchez, “Avances en robótica y visión por computador”, 2002.

[6] O. Bimber, R. Rakar, “Spatial Augmented Reality. Merging Real and Virtual Worlds”, 2005.

[7] S. Cawood, M. Fiala, “Augmented Reality: A practical guide”, 2008.

[8] AT&T Digital Life, (2017, Mayo 13). Explore Smart Home Security [Online]. Disponible en: https://my-digitallife.att.com/learn/explore-home-automation

[9] Smartthings, (2017, Mayo 13). Getting Started [Online]. Disponible en: https://www.smartthings.com/getting-started

[10] Vera, (2017, Mayo 13). Smarter Home Control [Online]. Disponible en: http://getvera.com/

[11] Nest, (2017, Mayo 13). Works with Nest [Online]. Disponible en: https://nest.com/works-with-nest/

[12] Google, (2017, Mayo 13). Google Home [Online]. Disponible en:

https://madeby.google.com/home/

[13] Amazon, (2017, Mayo 13). Amazon Echo - Black [Online]. Disponible en: https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E

[14] E. Protalisnki, (2017, Mayo 13). Google's speech recognition technology now has a 4.9% word error rate. Disponible en: https://venturebeat.com/2017/05/17/googles-speech-recognition-technology-now-has-a-4-9-word-error-rate/

[15] Apple, (2017, Mayo 13). Apple Homekit [Online]. Disponible en: https://www.apple.com/es/ios/home/

[16] W. Greenwald, (2017, Mayo 13). Logitech Harmony Elite [Online]. Disponible en: http://www.pcmag.com/review/349593/logitech-harmony-elite

[17] V. Heun, J Hobin, P Maes, "Reality Editor: Programming Smarter Objects", en UbiComp'13, 2013, Zurich, Switzerland 307,

[18] OpenHybrid, (2017, Mayo 13). Learn, Setup, Operate [Online]. Disponible en: http://openhybrid.org/learn%2c-setup%2c-operate.html

[19] J. A. Aguilar, K. G. Campuzano "SISTEMA DE INTEGRACIÓN PARA OBJETOS

INTELIGENTES" Informe de Materia Integradora, Facultad de Ingeniería Eléctrica y Computación, FIEC. Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, 2017.

[20] A. M. Ullah, M. R. Islam, S. F. Aktar y S. A. Hossain, "Remote-touch: Augmented reality based marker tracking for smart home control", Computer and Information Technology (ICCIT), 15th International Conference, IEEE, pp. 473-477, Diciembre, 2012.

[21] S. Kasahara, R. Niiyama, V. Heun, y H. Ishii, "exTouch: spatially-aware embodied manipulation of actuated objects mediated by augmented reality", Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, ACM, pp. 223-228, Febrero, 2013.

[22] R. Xiao, G. Laput, Y. Zhang, y C. Harrison, “Deus EM Machina: On-Touch Contextual Functionality for Smart IoT Appliances”, Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 4000-4008, ACM, Mayo, 2017.

[23] A. K. Singh, S. Roy, S. Sodhi, D. Goel, y S. M. Arora, “SmartX Virtuality: A Smarter way to Interact Virtually with Physical Objects”, International Journal of Scientific and Research Publications, Vol. 4, Issue 6, Junio 2014.

[24] Vuforia. (2016, Octubre 30) VuMark. [Online]. Disponible en: https://library.vuforia.com/articles/Training/VuMark

[25] S. Rad. (2016, Octubre 30). How does Vuforia image recognition work? Disponible en: http://stackoverflow.com/questions/12253664/how-does-vuforia-image-recognition-work

[26] H. Rouzati. (2012, Mayo 11). Argon Augmented Reality Browser - WebGL & Vuforia Disponible en: https://www.youtube.com/watch?v=naZQzDGGWEw

[27] Shawn Lehner. (2017, Septiembre 18). ARMaker. Disponible en: https://shawnlehner.github.io/ARMaker/

[28] Brosvision. (2017, Septiembre 18). AR Marker Generator. Disponible en: http://www.brosvision.com/ar-marker-generator/

[29] Valentin Heun. (2017, Septiembre 18). Human Readable Quick Response Code. Disponible en: http://www.valentinheun.com/portfolio/hrqr/

[30] A. Stanford-Clark y A. Nipper. (2017, Septiembre 18). MQTT. Disponible en: http://mqtt.org/

[31] Sevenhugs. (2017, Septiembre 18). Sevenhugs Smart Remote: The Remote for Everything. Disponible en: https://www.indiegogo.com/projects/sevenhugs-smart-remote-the-remote-for-everything#/

[32] Iotivity. (2017, Septiembre 18). Iotivity. Disponible en: https://www.iotivity.org/

[33] Epson. (2017, Septiembre 18). Epson Moverio BT-200 Smart Glasses. Disponible en: https://epson.com/For-Work/Wearables/Smart-Glasses/Moverio-BT-200-Smart-Glasses-%28Developer-Version-Only%29/p/V11H560020