UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Kit basado en sensores para personas con disminución visual y ciegas

AUTORES: Altolaguirre María Paula – Torales Rodrigo

DIRECTOR: Lic. Fava Laura - Lic. Harari Ivana

CODIRECTOR:

ASESOR PROFESIONAL: APU Durante Mateo

CARRERA: Licenciatura en Sistemas

Resumen

Confección de kit formado por un bastón, lente y aplicación para Android. Se puede apreciar que el bastón y el lente pueden ser configurados desde la aplicación la cual permite además consultar servicios web posicionando ómnibus y recorridos para la ciudad de La Plata.

Palabras Clave

Tecnologías Asistivas, Accesibilidad, Kit basado en sensores para personas con disminución visual y ciegas, App Mobile Asistiva.

Conclusiones

El kit alcanzado brinda una solucion integral entre el bastón, el lente y el software para el recorrido de ómnibus que mejore la calidad de vida de las personas ciegas y disminuidas visuales para un tránsito autónomo e independiente. Es una solución innovadora y superadora desarrollada teniendo en cuenta la participación y opinión de personas ciegas como así también teniendo en cuenta otros antecedentes y desarrollos previos para lograr una propuesta mejoradora.

Trabajos Realizados

- Diseño y desarrollo de una aplicación asistiva para ciegos.
- Diseño y construcción de bastón y lentes basado en sensores de proximidad y vibración para asistir a personas con disminución visual y ciegas.
- Pruebas de los sucesivos prototipos con personas con disminución visual y ciegas.

Trabajos Futuros

- Almacenar los resultados del procesamiento de los datos obtenidos del webservice de los recorridos de las lineas de ómnibus dentro de algún repositorio persistente y así poder consultarlo más rápido.
- Incorporar algún módulo (cámaras en el bastón) con reconocimiento de imágenes, que detecte los tipos de objetos que se encuentran en el recorrido del invidente.
- Incorporar algún tipo de aviso que alerte al usuario ciego que las baterías del lente y/o el bastón están por acabarse.

Fecha de la presentación: Septiembre 2020

Kit basado en sensores para personas con disminución visual y ciegas

AUTORES
Maria Paula Altolaguirre
Rodrigo Torales

DIRECTORAS
Lic. Ivana Harari
Lic. Laura Fava

ASESOR PROFESIONAL
APU Mateo Durante

Tesina de Licenciatura en Sistemas

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

Índice General

1

1. Introducción

El avance de la tecnología en los últimos años marcó un cambio sustancial en el papel que cumple en los distintos ámbitos. Uno de los ámbitos más novedosos donde se incorpora la tecnología es la accesibilidad, incorporando dispositivos y aplicaciones para facilitar cualquier actividad que se desarrolle de manera cotidiana, como los lectores de texto, reconocimiento de imágenes, chats especializados, entre otros.

Según los últimos datos de la Organización Mundial de la Salud, el número estimado de personas con alguna discapacidad visual oscila entre los 285 millones, de las cuales 39 millones son ciegas y 246 millones presentan baja visión. El 90% de las personas con esta discapacidad proviene de países subdesarrollados, haciendo que el acceso a servicios de prevención, reducción, tratamiento y rehabilitación sea más difícil [1].

Asimismo en nuestro país, según lo informado por el INDEC, se estima una tasa de prevalencia de población con dificultad o limitación permanente que oscila entre 9,0% y 10,2%, es decir, una estimación menor a la observada en el Censo 2010 (12,9%).

En este sentido, de la experiencia en Argentina en la medición de discapacidad a través de los censos se puede observar que existiría un conjunto de población con dificultad severa que es captada por la mayoría de las definiciones conceptuales y operacionales de discapacidad. Sin embargo, hay grupos de población donde la medición se vuelve mucho más compleja, principalmente porque la dificultad es más difícil de captar por las particularidades que esta presenta, por ejemplo, en el caso del dominio visual que tiende a ser sobreestimado y el caso de los niños [2].

Estas personas con discapacidad visual se encuentran con diferentes barreras, tanto físicas, de actitud (estigma, prejuicio y discriminación), de comunicación (no disponibilidad de Braille por ejemplo) y de transporte (deficiente difusión de transportes adaptados a personas con discapacidad visual) [3].

El presente trabajo tendrá como objetivo principal desarrollar un kit de movilidad, compuesto por un bastón, lentes y una aplicación móvil, con la finalidad de que las personas con disminución visual puedan apoyarse en la tecnología para moverse y utilizar los medios de transporte públicos de la ciudad de La Plata.

1.1.Motivación

Desde principios del Siglo XIX las personas ciegas han llevado bastones para facilitar su desplazamiento. Con la aparición de los primeros automóviles en las ciudades (teniendo en cuenta la inexistencia de infraestructura vial, como semáforos y sendas peatonales), las personas con discapacidad visual empezaron a enfrentarse con nuevos retos. Los bastones convencionales servían para caminar por la vereda, pero no servían como señal de advertencia para sortear obstáculos de proximidad como los vehículos.

Frente a dicha problemática se tomaron diferentes medidas en el mundo. En París, “el bastón blanco” pegó un salto gracias a la campaña de Madame Guilly d’Herbemontl, quién escribió una carta en Noviembre de 1930, dirigida a un periódico local para concientizar sobre el uso de los bastones blancos, además de donar 5.000 bastones a los residentes ciegos de la ciudad. Por otra parte, en Estado Unidos, George A. Bonham logró la colaboración de miles de socios para la introducción del bastón blanco. En Diciembre de 1930 se aprobó la primera ley de seguridad del bastón blanco de EE. UU. en el Ayuntamiento de Peoria, dicha ley brindaba a los ciudadanos ciegos el derecho de paso y de protección cuando llevaban el

bastón blanco. En 1959 mediante una exhaustiva campaña de sensibilización dentro de todos los estados de EE. UU., se promulgaron las leyes de seguridad del bastón blanco [4].

El bastón blanco de orígenes allá por 1921, fue protagonista de diferentes procesos evolutivos para adaptarse a los nuevos obstáculos que amenazaban su valor como herramienta de apoyo. Es así que en la actualidad en algunos países europeos y en Estados Unidos se pueden encontrar bastones inteligentes con capacidad de detección de obstáculos por medio de sensores de proximidad, módulo de GPS integrado que faciliten la localización y desplazamiento, apoyados en avances científicos referentes a la miniaturización, tecnología láser, ultrasónica, bluetooth, sonidos y vibraciones. Los bastones inteligentes utilizan distintos tipos de software para poder brindar estas funcionalidades ya sean open source o comerciales.

En la Argentina, los bastones inteligentes no se programan ni se comercializan. Estos productos deben ser importados y adquiridos mediante cupos y por montos muy elevados, complicando su adquisición. Teniendo en cuenta lo mencionado anteriormente, este trabajo de tesina pretende una reingeniería de varios prototipos de bastones inteligentes, actualmente comercializados en Europa, agregando funcionalidades que faciliten el tránsito del peatón, proporcionando un mejor sentido de orientación y mayor autonomía. Estará compuesto básicamente por microcontroladores capaces de ser ajustados dependiendo de las características fisonómicas del portador, velocidad y densidad de ocupación de las calles, de interiores o exteriores, baterías recargables de alta capacidad para un uso continuo y prolongado, y sensores ultrasónicos para una mayor exactitud en la detección de obstáculos. Este kit podría brindar a las personas con discapacidad visual mayor autonomía en el desplazamiento por diferentes terrenos inclusive con obstáculos suspendidos en el aire como ser carteles, ramas de árboles, extintores, entre otros. Los bastones inteligentes tradicionales tienen la característica de detectar obstáculos por debajo de la

cintura del usuario dependiendo del ángulo de inclinación del sensor, dejándolos expuestos a objetos suspendidos.

1.2. Objetivos y metodología a emplear

El sentido de la vista es una de las principales capacidades sensoriales de los seres humanos utilizada para desenvolverse en cualquier actividad cotidiana. La discapacidad visual puede limitar a las personas a realizar estas tareas, afectando su movilidad y su calidad de vida, así como la posibilidad de interactuar con su entorno. Como ya se mencionó, según el estudio de la OMS, el número estimado de personas con alguna discapacidad visual oscilaba entre los 285 millones. El 90% de estas personas proviene de países subdesarrollados, esto influye directamente en las posibilidades de acceso a servicios de prevención, reducción, tratamiento y rehabilitación.

Para atacar la problemática referente a la movilidad existen modalidades como la utilización del bastón blanco o los perros guía. El bastón blanco es un medio que identifica a las personas con discapacidad visual además de ser una herramienta guía para desplazarse de manera autónoma al caminar. Existen varios tipos de bastones de acuerdo a la fisionomía y la comodidad del individuo. Otra alternativa es la de incorporar al mejor amigo del hombre como guía aprovechando su astucia y habilidades, el “perro”. Dicho perro es entrenado para guiar a las personas con discapacidad visual en su vida cotidiana. Es uno de los tipos de perros de asistencia y es el único reconocido legalmente hoy en día por tener la capacidad de percibir los peligros eventuales, lo que le permite algunos derechos y privilegios como la aceptación obligatoria en los transportes públicos [5].

Con la finalidad de aportar a las personas que sufren algún tipo de discapacidad visual, el objetivo de esta tesina consiste en crear un kit basado en sensores para mejorar su movilidad. El presente kit estará

compuesto por un bastón y un par de lentes que tendrán sensores de proximidad los cuales podrán ser configurados mediante una aplicación móvil. Dicha aplicación gestiona los niveles de sensado y sirve como guía para abordar los ómnibus en la ciudad de La Plata.

Para ello se realizarán las siguientes actividades:

- Realizar un relevamiento sobre las herramientas tecnológicas de apoyo para personas ciegas a nivel nacional, desempeño y demanda.
- Analizar las características de los bastones inteligentes de UMH (Universidad Miguel Hernández).
  Establecer las características/funcionalidades del bastón inteligente a desarrollar. Determinar las distintas configuraciones del bastón.
- Elaborar lentes inteligentes para determinar/detectar e informar sobre los obstáculos suspendidos, próximos a las personas que sostienen el bastón. Determinar las distintas configuraciones del lente.
- Analizar los programas existentes para obtener la geolocalización de los ómnibus.
- Desarrollar los algoritmos necesarios acordes a los requerimientos funcionales y no funcionales preestablecidos.
- Determinar el funcionamiento del programa seleccionado realizando pruebas de humo.
- Desarrollar la aplicación que gestione el bastón, el lente y sus distintas configuraciones junto con los recorridos.

1.3. Estructura de la tesina

En el segundo capítulo se expondrán los temas fundamentales a abordar a lo largo de la tesina; el bastón, los lentes y la aplicación de gestión, haciendo un recabado de información (dentro de la Argentina y del exterior) sobre prototipos existentes e introduciendo la posibilidad de una

solución TIC que permita la gestión del kit y la circulación del individuo por la ciudad de La Plata.

En el tercer capítulo resaltaremos las leyes que amparan a las personas con discapacidad visual, cuáles aspectos se encuentran estipulados dentro de dichas leyes y cuáles quedan fuera referentes a la circulación y acceso a los espacios públicos. La opinión de la APANOVI (Asociación Pro Ayuda Al Invidente) sobre la postura del gobierno frente a la discapacidad visual y además expondremos por todo lo antes mencionado, la importancia de desarrollar un kit de movilidad en la Argentina.

En el cuarto capítulo se detallarán los lineamientos seguidos para el desarrollo. Cuáles capacidades integradas se utilizaron y cómo se determinaron para el kit, las tecnologías utilizadas (hardware y software) y el procedimiento realizado para la integración.

2

2. Estado del arte o de la cuestión

Actualmente en Argentina existe la Ley N° 26.858, que regula el derecho de acceso, deambulación y permanencia de las personas con discapacidad acompañadas con perros guías . A pesar de que el artículo 10 de dicha ley, establece que la persona acompañada de un perro guía debe tener un asiento adecuado, el acceso de los perros dentro del transporte público sigue representando una dificultad [6].

Teniendo en cuenta lo anteriormente dicho, este trabajo consideró importante desarrollar una herramienta que facilite el acceso de las personas con discapacidad visual a cualquier tipo de transporte público, sin depender de un animal, que por obvias razones podría representar mayores dificultades tanto para los dueños del perro como para los usuarios del transporte público.

Si bien en Argentina se han desarrollado prototipos de bastones inteligentes, no tienen todas las características reunidas en un solo prototipo como el kit que desarrolla este trabajo. Los prototipos desarrollados en Argentina no están orientados a la comercialización de los mismos dentro del país. Para acceder a un bastón inteligente se requiere de la compra a través de internet en el mercado exterior, los cuales representan altos costos.

Para una mejor organización, este trabajo se divide en tres temas centrales los cuales hacen referencia al estado del arte: bastones inteligentes, lentes inteligentes y tecnologías de apoyo evaluadas para el desarrollo.

2.1. Bastones Inteligentes

Existen muchas iniciativas relacionadas a dispositivos para asistir a las personas ciegas. Los prototipos visualizados en esta sección fueron tomados como base para el desarrollo del kit de movilidad. Se estudiaron las funcionalidades de cada uno de los prototipos y con ellos la viabilidad de implementación. A continuación describiremos algunos de ellos:

- Universidad Miguel Hernández (UMH) de Elche, 2 de Julio de 2013.
  Desarrollaron un bastón que ayuda a los ciegos a detectar obstáculos en altura, que no pueden ser detectados por los bastones normales. El aparato se puede adaptar según las características físicas de la persona y de la vía que esté atravesando.
  La detección de los objetos se realiza mediante un conjunto de sensores, que se adaptan a un bastón blanco tradicional y mejoran su funcionalidad.
  Emite un aviso cuando detecta objetos que supongan un peligro para la integridad de la persona, a través de un innovador sistema de vibración colocado en la muñeca.

Figura 1: Una persona con discapacidad visual detecta una barra de metal gracias al bastón, en la presentación del aparato.

.

- Vasileios Tsormpatzoudis, Universidad de Manchester en Reino Unido, 23 de Noviembre de 2016.

  Desarrolló MySmartCane, un bastón inteligente que está compuesto por un bastón blanco tradicional al que se le incorpora en el extremo inferior un módulo esférico parecido a una pelota, que contiene los sensores de ultrasonido y componentes electrónicos.

  El aparato mide de manera inalámbrica la distancia a la que se encuentran los obstáculos y traduce la información a señales de audio.

Figura 2: Una persona con discapacidad visual detecta los objetos que hay en su camino gracias al sonido del bastón.

- Alumnos de la Escuela Técnica 1 de la ciudad bonaerense de Bragado, 15 de Diciembre de 2017.

  Desarrollaron un bastón inteligente que cuenta con un detector de humedad para evitar que las personas ciegas pisen los charcos, un sensor que les advierte si hay ramas y una rueda giratoria en la parte inferior para evitar que se clave en el piso, haciendo que sea más fácil maniobrar hacia adelante o a los costados.

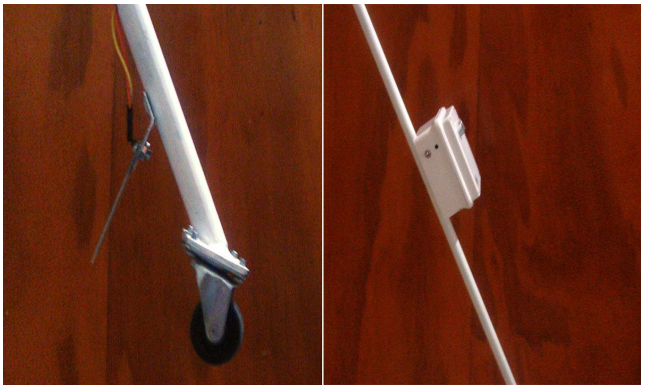

Figura 3: bastón inteligente para ciegos, con detector de humedad.

2.2. Anteojos Inteligentes

- Alumnos del colegio José Hernández de la provincia de Jujuy, 14 de Junio de 2018.

  Desarrollaron unos anteojos que integran sensores, emitiendo un alerta cuando la persona se acerca a un obstáculo.

Figura 4: SiVeo son unos anteojos de asistencia para personas con ceguera o discapacidad visual.

- Rodrigo Facundo Ñañez, 12 de Enero de 2019

  Desarrolló un prototipo de lentes inteligentes. Estos alertan sobre la presencia de un objeto a una distancia de un metro. Cuando algo interrumpe la visión del sensor ultrasonido, comienza una vibración suave y a medida que la distancia se acorta, vibra aún más.

Los lentes utilizan energía y deben ser recargados a través de un cable USB. Trabajan con cargador portátil de celular, pero tienen un consumo bajo y se calcula que con una carga de una hora sirve para todo un día.

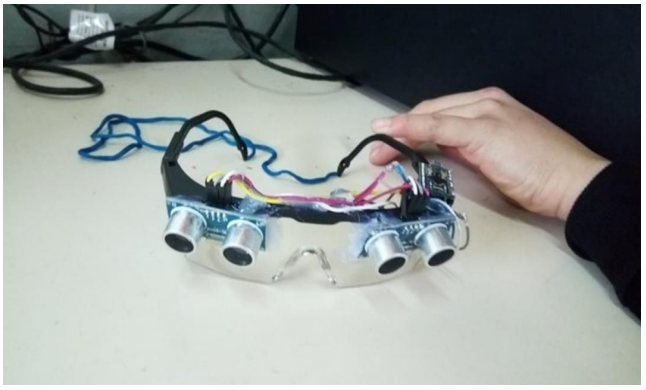

Figura 5: Anteojos diseñados por Rodrigo Facundo Ñañez, para las personas con discapacidad visual.

2.3. Tecnologías de apoyo

Las tecnologías de apoyo son un conjunto de productos, dispositivos, equipos e instrumentos de hardware y software que se usan para mantener, incrementar y mejorar la capacidades funcionales de las personas con discapacidad. La referente a la discapacidad visual es la tiflotecnología. Se encuentra reglamentada por la normas ISO 9999: 2007.

Las Tecnologías de la Información y la Comunicación (TIC) son un conjunto de tecnologías desarrolladas para una información y comunicación más eficiente. Han transformado los parámetros de obtención de información por medio de las tecnologías de la comunicación (diario, radio y televisión), a través del desarrollo de Internet y de los nuevos dispositivos tecnológicos como la computadora, la tablet y el *smartphone*, así como las plataformas y *softwares* disponibles [7].

Las **AT** (del inglés *assistive technologies)* junto con las **TIC** pueden ser de gran aporte para las personas con discapacidad, favoreciendo su autonomía personal a la hora de solucionar problemas y trámites cotidianos.

Un buen uso de recursos tecnológicos podría facilitar la comunicación y el contacto directo con asociaciones o personas con discapacidad en su misma situación, facilitando la comunicación, el acceso/proceso de la información, el desarrollo cognitivo, la realización de todo tipo de aprendizajes, la adaptación y autonomía ante el entorno, el ocio, instrumentos de trabajo, posibilidades de realizar actividades laborales, mejoran la autoestima, entre otros.

Algunas tecnologías de apoyo existentes son:

- NonVisual Desktop: Es un programa para Windows Open Source que funciona como lector de pantalla proporcionando información al usuario por voz sintética y braille.
- Lector de pantalla Orca: Es un programa para Linux Open Source que funciona como lector de pantalla extensible proporcionando acceso al escritorio gráfico a través de voz y braille.
- Window-Eyes: Es un programa distribuido bajo licencia para Windows, es el lector de pantalla más estable disponible en el mercado hoy en día.
- Google Talkback: Es un servicio de accesibilidad. Se trata de una audioguía del sistema, con comentarios hablados de cada menú y vibración para navegar por Android. Está preinstalado en la mayoría de dispositivos Android y se puede activar desde el menú de Accesibilidad.
- BeMyEyes: Es un sistema colaborativo que permite a las personas con discapacidad visual apuntar con la cámara un objeto a reconocer y los usuarios conectados lo describen.
- Lazarillo GPS: Es una aplicación para Android que utiliza el GPS e informa detalladamente las rutas, entornos, tiendas y servicios cercanos. Una guía auditiva en español, inglés e indonesio compatible con Talkback.

- TTS (TextToSpeech): Librería de Android que permite la conversión de texto en voz sintética, por ejemplo este artículo sea trasladado a la voz.

El conjunto de tecnologías citado tiene principal enfoque en brindar servicios de asistencia para personas con discapacidad visual. Dichas tecnologías no se encuentran aisladas y utilizan otras para la obtención de información. En el siguiente listado mencionamos tecnologías utilizadas en nuestro desarrollo.

- Google Maps: Es uno de los GPS más utilizados por los usuarios. Fácil de usar. Permite calcular el camino más corto y los distintos transportes a utilizar para llegar al destino [8].
- Web Service SOAP: Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Para esta tesina se tiene acceso a web services con información de líneas de colectivos de La Plata [9].
- SoapIU: Es una aplicación muy versátil que nos permite probar, simular y generar código de servicios web de forma ágil, partiendo del contrato de los mismos en formato WSDL y con vínculo SOAP sobre HTTP [10].
- XML: Es un metalenguaje de definición de documentos, estructurado mediante etiquetas o marcas. Su nombre viene de las palabras inglesas eXtensible Markup Language (Lenguaje de marcado ampliable o extensible). Fue desarrollado por la World Wide Web Consortium (W3C), con el objetivo de lograr páginas web mucho más semánticas, separando la estructura del contenido web y ofreciéndole al

desarrollador la capacidad de crear vocabularios modulares personalizados [11].

- API: Aplication Program Interface, Interfaz de protocolo de aplicación entre un cliente y un servidor simplificando el desarrollo del código del lado del cliente. Es un contrato entre el cliente y el servidor, si el cliente realiza una solicitud en un formato específico siempre obtendrá una respuesta en un formato específico [12].

Finalmente después de estudiar todas las alternativas mencionadas, podemos destacar algunas características comunes entre los distintos prototipos. La utilización de sensores para medir obstáculos, la incorporación de un sistema de alertas ya sea sonoro o por medio de vibraciones y la posibilidad de configurar los niveles de medición. También se destacan las funcionalidades de las aplicaciones de asistencia como la localización en tiempo real del individuo frente a un lugar físico determinado, trazabilidad de ruta y duración de recorridos de transportes públicos entre otros. Estas funcionalidades pueden ser agrupadas en un kit, desarrolladas con tecnología barata y de fácil acceso.

3

3. Descripción del problema. Hipótesis de trabajo

Según la OMS el 10% de las personas en el mundo padecen de alguna discapacidad, este porcentaje equivale a 650 millones de personas. Dentro de ellas el 4,34% sufren discapacidad visual. En Argentina por su parte hay 2.176.123 con alguna discapacidad, que significa un 7.07% de la población total, de los cuales 567.005 padecen de las discapacidades más frecuentes, el 25% de ellas padecen de discapacidad Visual-motora [13].

La legislación Argentina establece un conjunto de leyes para las personas con discapacidad visual, con el objetivo de garantizar su protección integral como: atención médica, educación y seguridad social, entre otros. Existe una guía de legislación sobre discapacidad de la República Argentina que contiene treinta y seis leyes nacionales y doce leyes provinciales. Así también la APANOVI (Asociación Pro Ayuda Al Invidente) incorporó reglamentaciones adicionales.

Cabe mencionar que el artículo 20 de la Ley Nacional N° 24.314, tiene como finalidad la eliminación de barreras físicas urbanas que impidan una cómoda movilidad de la persona con discapacidad. Estas barreras comprenden la construcción y caracterización de los itinerarios peatonales, escaleras y rampas, espacios libres, obras públicas, señales verticales y elementos urbanos varios suspendidos [14].

Según la legislación, la discapacidad está amparada, pero se centra en la discapacidad motora, no así en la discapacidad visual ya que se encuentran excluidas por las leyes, por ejemplo: la inexistencia de semáforos sonoros, falta de infraestructura para el acceso a espacios públicos con perros de asistencia, la inexistencia de mapas con relieves (braille). Además del incumplimiento a cabalidad de las leyes expuestas anteriormente, no se registra el cumplimiento de lo que Amengual (1983,

p.16) propuso en el diseño de las infraestructuras viales, como señales acústicas y táctiles, señalizaciones, andamios, carteles, suministros electrógenos, entre otros.

Desde el punto de vista de las ONG, la APANOVI, a través de los años, expuso la falta de interés de los organismos gubernamentales, la burocracia existente y falta de compromiso de las áreas destinadas a ayudar, pero agradece a las demás ONG y a las personas solidarias que aportan de manera voluntaria con la intención de resolver esta problemática [15].

Frente a esta situación y a la falta de herramientas tecnológicas para una efectiva movilización en los espacios públicos, ya sean estos: mapas con relieves, información accesible referente al recorrido de transportes públicos, infraestructura adecuada en los espacios públicos, señalización, etc., para personas con discapacidad visual, se pretende incorporar materiales y recursos adaptados para facilitar la movilidad de los discapacitados visuales.

Actualmente, la Facultad de Informática cuenta con alumnos con discapacidad visual que recurren frecuentemente a la misma para el dictado de cursos, haciendo uso de transportes públicos, por medio de la asistencia del bastón y generalmente apelando a la solidaridad de las personas para identificar las paradas y puntos referenciales. No obstante, no siempre son asistidos y en muchas ocasiones quedan expuestos a obstáculos y peligros al abordar el transporte público, esquivar obstáculos que se encuentran en el piso o los suspendidos, entre otras.

Con el objetivo de responder a esta demanda detectada, este trabajo de tesina consiste en desarrollar un kit de movilidad que se encuentre disponible para las personas con discapacidad visual y que así puedan mejorar sustancialmente su calidad de vida, abarcando funcionalidades de ubicación y detección de objetos configurable con un sistema de alarmas amigable.

La hipótesis propuesta consiste en la siguiente idea: se comprueba empíricamente que esta solución tecnológica satisface las características detectadas de la población objetivo.

4

4. Propuesta de solución

El desarrollo del kit está compuesto de cuatro componentes tecnológicos que se presentarán a continuación. Dichos enfoques se encuentran separados de acuerdo a sus características obvias y grado de complejidad.

4.1 Recabado de información

El puntapié inicial que dió origen al desarrollo de la tesina se describe en los siguientes párrafos. La inquietud que nos surge en el ámbito del programa de tutorías cada año respecto al bienestar y adaptación de los nuevos ingresantes al mundo universitario. Dicho programa de tutorías tiene como objetivo hacer un acompañamiento a los ingresantes durante el primer año guiándolos, de manera que puedan entender el funcionamiento institucional de una universidad, los trámites que necesitan realizar como ingresantes, los plazos y fechas límite de entrega, metodologías y técnicas de estudio utilizadas desde la experiencia personal del tutor para la aprobación de materias, cómo obtener información de cada cátedra, en dónde se dictan las clases, información relacionada a los servicios expuestos por la universidad como ser comedores universitarios, recorridos de ómnibus universitarios, entre otros. Esta información en gran medida es entregada a los ingresantes con gran aceptación.

En el caso de que el individuo lo requiera y se identifique como persona con necesidad de asistencia especializada, el programa de tutorías asigna a personas capacitadas como sus tutores, los cuales hacen utilidad

de herramientas y metodologías alternativas a las utilizadas convencionalmente.

Durantes los dos años que participamos del programa de tutorías, hemos interactuado con chicos con distintas discapacidades, buscando herramientas y metodologías que faciliten su transición por la facultad, una de las discapacidades que requería un cierto nivel de creatividad era la visual, se utilizaban distintos tipos de materiales para hacerles llegar la información y así ellos puedan entender y razonar los problemas que se les presentaban. Frente y gracias a esta iniciativa del programa de tutorías, con un sentido de sociedad más colaborativo y por el espacio que nos brindaron, pudimos establecer contacto con los chicos con discapacidad visual y determinar de manera más específica las necesidades que debían ser cubiertas según sus vivencias cotidianas enfocando nuestro trabajo final de tesina en el bienestar y la inclusión de más personas al ámbito universitario.

El proceso de recabado de información constó de varias etapas.

En un proceso de relevamiento de información, citamos a dos chicos con diferentes niveles de disminución visual para comentarles el interés de realizar una solución tecnológica que sirva de apoyo en sus actividades cotidianas. En dicha reunión y para determinar las áreas que podíamos abarcar, los posibles puntos de nexo entre los dos chicos utilizamos la técnica de recabado de información denominada *Brainstorming*. Dicha técnica se aplica a grupos de individuos estimulando la aparición de nuevas ideas sobre un problema concreto, en este caso la discapacidad visual y cómo la informática aplicada puede mejorar la vida cotidiana. Durante el tiempo que llevó la “lluvia de ideas” los chicos citaron y describieron soluciones tecnológicas en el mercado, funcionalidades adicionales a las existentes, desde reconocimiento de imágenes, lectores de documentos con voz sintética, programas colaborativos de detección de objetos, impresiones de documentos en braille, lectores con inteligencia artificial capaces de leer libros, entre otros.

Al finalizar el plazo de tiempo especificado para incorporar nuevas ideas, se leyeron y discutieron cada una de estas para eliminar redundancias y temas fuera del alcance comprendido en un tesina que tenga relevancia para una licenciatura en sistemas. Se discutieron las ventajas y desventajas, caracterización de dichas ideas y soluciones, ámbito de aplicación, *skills* necesarios, facilidades de acceso a la tecnología, precios, tiempo de desarrollo requerido, necesidad del involucramiento de estudiantes de otras carreras para el desarrollo, entre otras.

Posterior a nuestro primer contacto con los chicos, estudiamos las factibilidades de implementar las ideas propuestas junto con nuestras directoras de tesina. Preparamos una segunda reunión centrándonos en el desarrollo de herramientas para asistencia a la movilidad y/o software colaborativos para la detección de obstáculos. En la reunión se decidió el desarrollo de una herramienta de asistencia a la movilidad ya que los chicos expusieron experiencias al movilizarse por las calles de La Plata, las irregularidades de la superficie al caminar junto a alguna obra en construcción, los medidores de energía colocados provisoriamente en las veredas, ramas de árboles, extintores suspendidos ubicados en los pasillo y varios obstáculos más. Se tocaron temas como el sensado, modo de notificación al detectar obstáculos, rango de mediciones, tipos de objetos a detectar, posibles sensores a utilizar, tecnologías de posicionamiento para determinar ubicaciones.

He aquí el despertar de nuestro interés por contribuir con la comunidad que necesita algún tipo de asistencia especial confeccionando este kit de movilidad. Lo propuesto en la tesina es el resultado de un trabajo consensuado donde las personas ciegas fueron parte de la elaboración, escuchando sus necesidades y demandas y se hallaron soluciones integrales a las mismas en forma conjunta lo más inclusivo posible.

4.2 Descripción funcional del kit

La solución tecnológica propuesta consiste en un dispositivo de detección de objetos u obstáculos y una aplicación que permite la movilidad urbana del usuario: acceso y traslado en ómnibus. El resultado consistió en:

- Un bastón con sensores de proximidad que permite detectar obstáculos cercanos de una altura del piso a los hombros configurables mediante una aplicación en el teléfono, avisando al portador cada vez que tenga un objeto dentro del rango de configuración especificado.
- Un par de anteojos con sensores de proximidad que detecta objetos cercanos a la altura de la cabeza, alertando al portador la presencia de algún objeto dentro del rango especificado para el lente.
- Una aplicación móvil para la gestión de configuración del bastón y los lentes.
- Una funcionalidad adicional que brinda información en formato de audio referente a los horarios de los ómnibus y ubicación de los mismos en tiempo real.

5. Diseño y desarrollo del kit

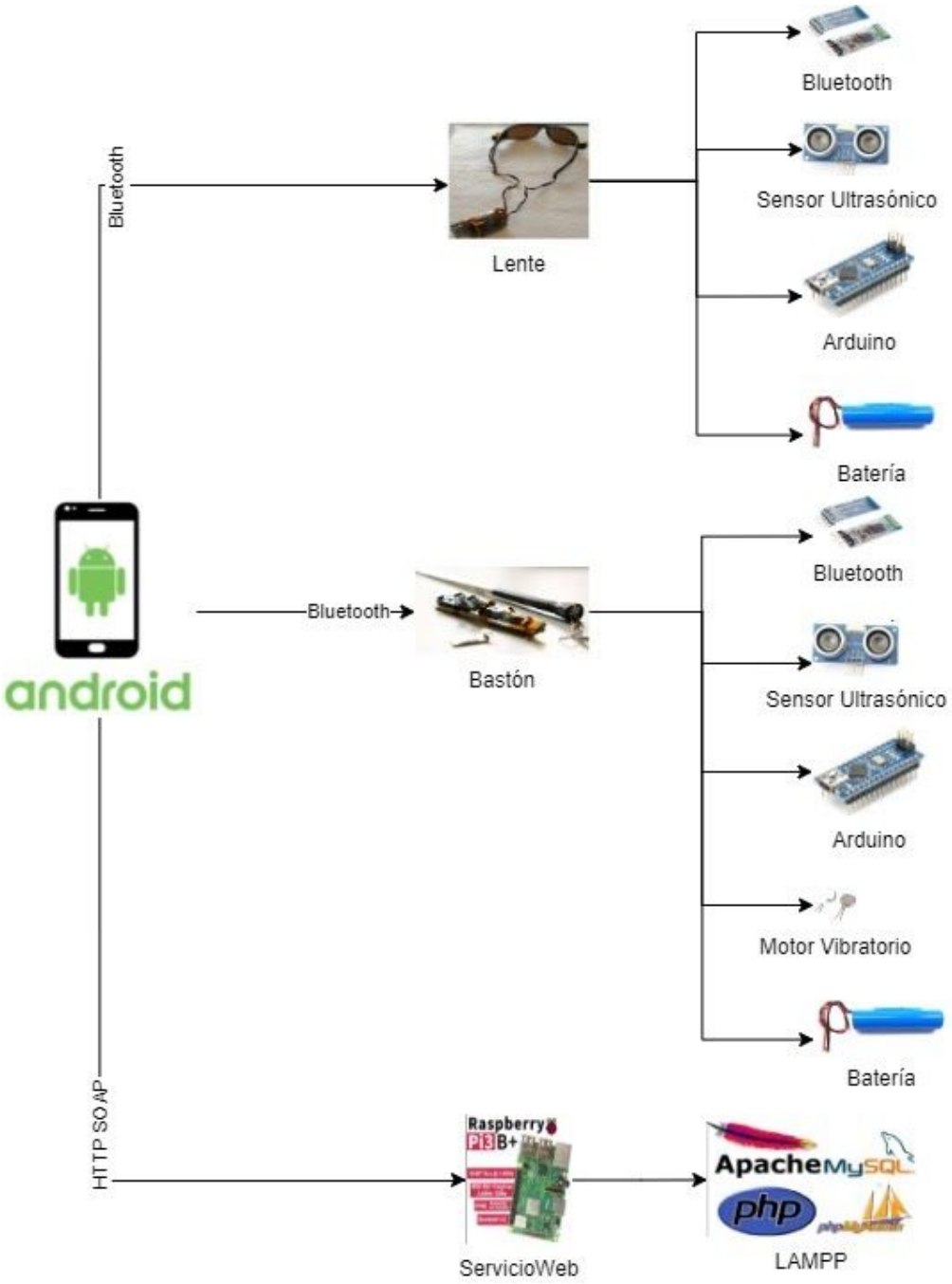

Figura 6: Interconexión de las componentes utilizadas.

En la imagen anterior se puede visualizar las componentes e interconexiones que conforman el kit. Este producto es el resultado de un desarrollo incremental donde a medida que se avanza en las versiones del hardware/software, se incorporan funcionalidades adicionales para soportar la infraestructura mencionada.

A continuación se describen los procesos de desarrollo de cada módulo que componen el kit.

5.1 Diseño y desarrollo del bastón

5.1.1 Conectando componentes del bastón - Primera versión del hardware

Una vez definido el kit a desarrollar, obtuvimos información de diferentes fuentes y extrajimos las funcionalidades más importantes de los diferentes prototipos. Entre ellas se encuentra la capacidad de detectar objetos en un rango determinado de medición, alarmas que avisen la detección de objetos, configuración de medición, entre otras.

Para el primer prototipo del bastón seleccionamos y utilizamos materiales y componentes que se usan en la mayoría de los proyectos de robótica:

- Placa de desarrollo Arduino Nano 3.0 Atmel Atmega328p.
- Módulo Bluetooth Hc05 Maestro Esclavo.
- Sensor Ultrasonido Hc-sr04.
- Protoboard de 400 puntos.
- Cables para protoboard macho-macho.
- Estaño, soldador.

A continuación explicaremos en detalle los materiales más relevantes utilizados.

5.1.1.1 Placa de desarrollo Arduino Nano 3.0 Atmel Atmega328p

La placa de desarrollo Nano 3.0 es una placa de tamaño compacto, completa y compatible con protoboards basada en el microcontrolador ATmega328P [16]. La misma posee 14 pines de E/S (entrada, salida) digitales 6 de los cuales pueden ser usados como PWM (*pulse-width modulation*), 6 entradas analógicas, un cristal de 16mhz [17]. En la Figura 7 se muestra el diagrama de pines de la placa de desarrollo Arduino NANO.

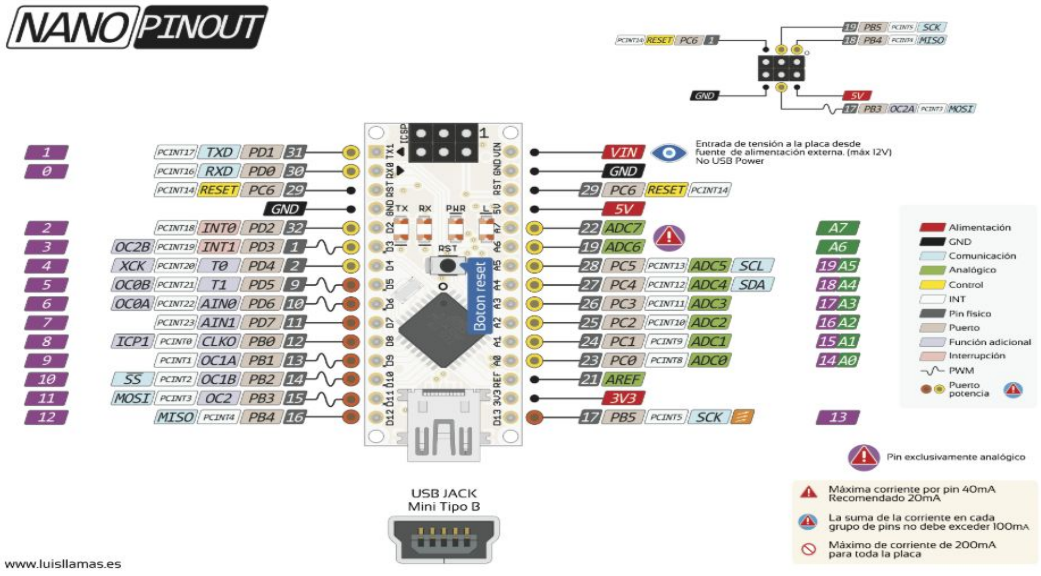

Figura 7: Diagrama de pines de la placa Arduino Nano.

5.1.1.2 Módulo Bluetooth Hc05 Maestro Esclavo

El Módulo Bluetooth Hc05 es un dispositivo utilizado para la comunicación inalámbrica entre dispositivos con Bluetooth, donde la comunicación de sus microcontroladores se gestiona mediante el protocolo de transmisión USART [18]. Dicho módulo posee una configuración estándar que puede cambiarse por medio de los comandos AT (Attention,

son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un Terminal) [19] a otros modos de trabajo como Maestro y Esclavo. Algunas de las aplicaciones inalámbricas como auriculares y controlador de dispositivos tienen un alcance de hasta 100 metros dependiendo de las condiciones atmosféricas y de las características geográficas y urbanas. En la imagen de la Figura 8, se puede apreciar el diagrama de pines del Módulo Bluetooth Hc05.

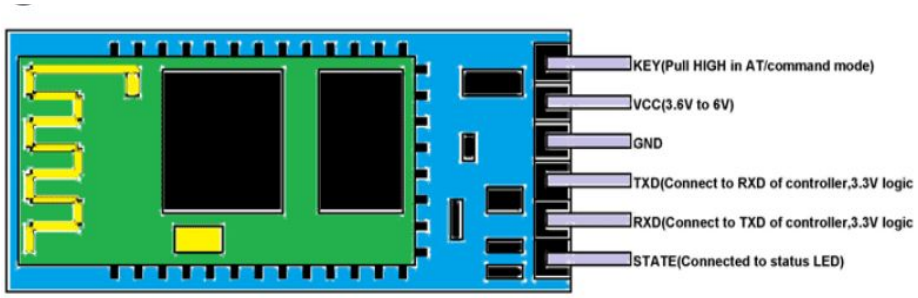

Figura 8: Diagrama de Pines del Módulo Bluetooth Hc05.

5.1.1.3 Sensor Ultrasonido

El sensor Ultrasonido Hc-sr04 es un sensor de distancias por ultrasonido capaz de detectar y calcular a qué distancia se encuentra un objeto de dicho sensor, en un rango de 2 a 450 centímetros. Su funcionamiento consiste en enviar pulsos de arranque y medir la anchura de los pulsos de retorno, este sensor se caracteriza por su bajo consumo y precio. En la imagen de la Figura 9 se puede apreciar el diagrama de pines del módulo ultrasónico Hc-sr04.

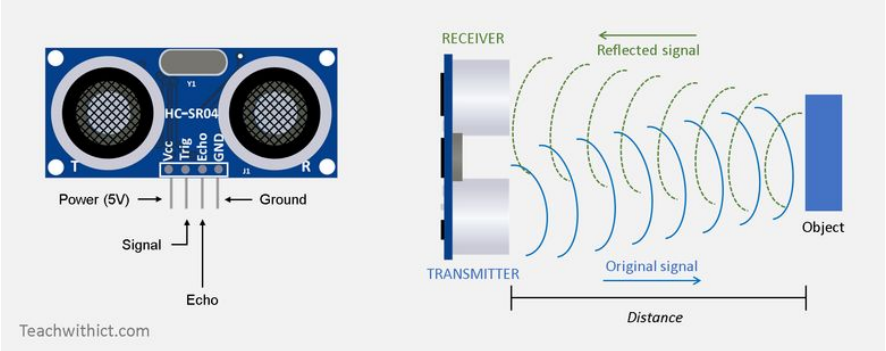

Figura 9: Diagrama de pines del Sensor Ultrasonido Hc-sr04.

5.1.1.4 Protoboards y cables

Las Protoboards son una especie de tablero con orificios en la cual se pueden acoplar componentes electrónicos y cables para armar circuitos electrónicos sin necesidad de utilizar una placa de circuito impreso -PCB-. Sirve para experimentar con circuitos electrónicos asegurando su buen funcionamiento, de fácil armado, sin necesidad de soldar componentes. Compuesto de un material aislante generalmente de plástico, internamente presenta laminillas metálicas para la conexión eléctrica. Estas laminillas forman bloques para permitir la conexión. Estos bloques son uno central y dos a los costados. Las separaciones entre orificios es de 1/10” (2.54 mm) la cual es una medida estándar en la separación de pines de los circuitos integrados y otros componentes electrónicos. Para realizar conexiones entre la protoboard y los diferentes circuitos electrónicos, utilizamos cables sin necesidad de realizar soldaduras pudiendo organizar las conexiones de acuerdo a colores y posiciones, muy práctico para realizar test. En la imagen de la Figura 10, se puede observar una imagen de una protoboard de 400 puntos que utilizamos en nuestro proyecto (izquierda) y los cables macho-macho usados para las conexiones (derecha).

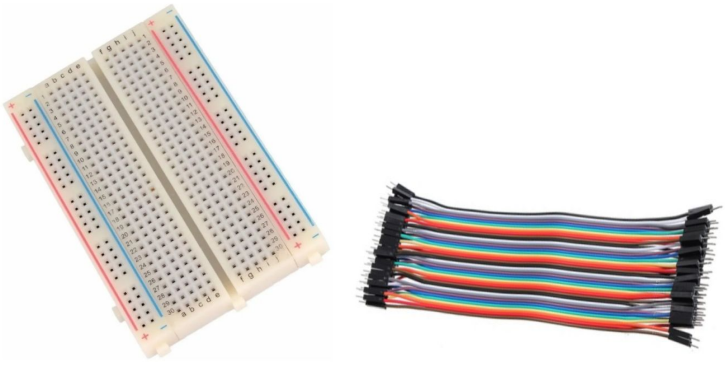

Figura 10: Protoboard de 400 puntos y cables

5.1.1.5 Esquema de componentes integrado

A continuación se expone el esquema de interconexión realizado para las componentes descritas.

En la parte central de la imagen de la Figura 11, se puede observar la placa de desarrollo Arduino que se configuró de la siguiente manera. El pin 4 es utilizado como pin receptor del módulo Hc-sr04 para incorporar información al microcontrolador y éste pueda realizar los cálculos necesarios, está configurado en modo entrada de datos. El pin 5 es utilizado como transmisor al módulo Hc-sr04, esto requiere su configuración en modo salida de datos para que por medio de éste se envíe las señales ultrasónicas. Además de los pines 4 y 5 configurados en Arduino para el módulo Hc-sr04, este módulo como cualquier otro requiere de fuente de alimentación y conexión a tierra, VCC (*voltage at the common collector*) y GND (*ground*) respectivamente. Dichos pines también presentes en el Arduino se encuentran conectados por medio de cables macho a la protoboard y a las líneas destinadas para alimentación y tierra, para brindar la energía necesaria a todos los componentes conectados a la protoboard. Es decir, que cada componente mencionada a partir de ahora estará conectada a dichas líneas por medio de la protoboard y los cables macho, a menos que se especifique lo contrario. Cabe mencionar que la placa de desarrollo se encuentra alimentada por entrada USB pudiendo ser la fuente un ordenador, un tomacorriente con una fuente de alimentación de 5V o una celda de batería.

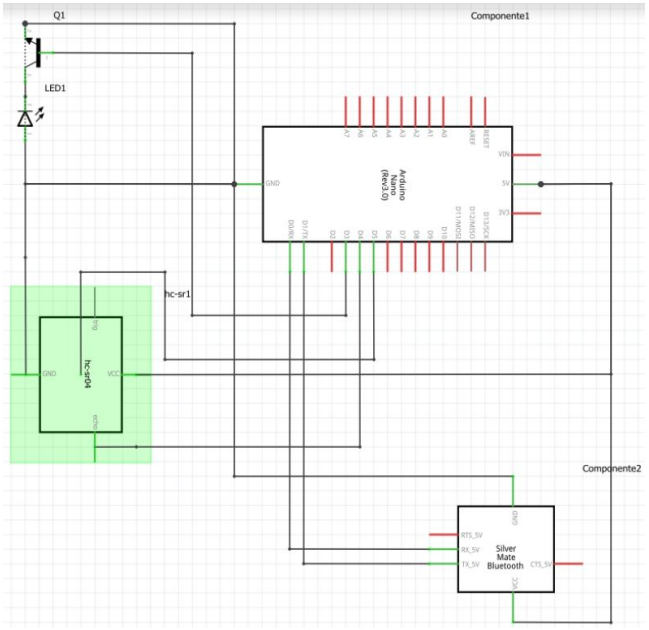

Figura 11: Esquema de interconexión.

El pin 13 es un pin led de control utilizado para realizar debugs al configurar la placa de desarrollo, puede utilizarse de distintas maneras, en nuestro caso lo configuramos en modo salida de datos digital, el cual nos da otra manera de testear los objetos detectados por el sensor ultrasónico.

El pin 3 del Arduino según el esquema está denotado con la siglas (PWM) por sus siglas en inglés de “Pulse Width Modulation”, como su nombre lo dice es un tipo de señal de voltaje utilizada para enviar información o para modificar la cantidad de energía que se envía a una carga. Este tipo de señales es muy utilizada en circuitos digitales que necesitan emular una señal analógica. En éste esquema el pin 3 está configurado en modo salida de datos analógico, dicho modo permite generar cargas de entre 0~5 volts con una corriente máxima de 40 mA en sus salidas y con ésto alimentar otros circuitos, en nuestro caso y en esta primer versión de interconexión el pin 3 se encontraba alimentando un led verde que se prendía en diferentes intensidades de acuerdo a la cantidad de voltaje que se le suministra.

Los pines Tx(transmit) y Rx(receive) del Arduino son los pines de entrada y salida estándar serializada respectivamente que posee el microcontrolador, dichos pines son utilizados para que el microcontrolador

se comunique con el mundo exterior recibiendo y enviando órdenes. En nuestro esquema, estos pines se encuentran conectados con el módulo bluetooth Hc05 el cual también posee sus pines Tx y Rx para transmisión y recepción de órdenes. Esta configuración se realiza de la siguiente manera, el pin Tx del Arduino se configura con el pin Rx del módulo bluetooth, es decir que las órdenes salientes del microcontrolador son recibidas por el pin Rx del módulo Hc05 y posteriormente enviadas para su ejecución al dispositivo con el que se encuentre conectado.

De igual manera el módulo Hc05 recibe información y transmite lo recibido por su pin Tx, dicho pin está configurado con el pin Rx del Arduino, por medio del cual recupera la información transmitida por bluetooth desde el dispositivo al cual se encuentra conectado. Con esta configuración tenemos comunicación con la placa de desarrollo y el mundo exterior.

Lo que se explicó en los párrafos anteriores comprende la configuración del hardware inicial del prototipo del bastón, el cual no sufrió mayores modificaciones en las posteriores versiones.

5.1.2 Desarrollando la estética del bastón - Segunda versión del Hardware

Luego de las pruebas de integración de las componentes básicas, abordamos la estética o aspecto ergonométrico del bastón, buscando colocar las componentes en un formato más compacto para que puedan adherirse al mango del bastón sin provocar incomodidad en su uso. Es decir, el trabajo realizado fue el de presentar un producto final actual de las funcionalidades del bastión, asistidos por Emiliano Albarracín, quien forma parte de este equipo de desarrollo. Este prototipo se conecta, al igual que el anterior, con una aplicación Android, dicha aplicación estaba mejor prestada y agregaba la funcionalidad de aumentar y disminuir la distancia de sensado. A nivel código de Arduino no se realizaron modificaciones significativas. La imagen de la Figura 12 muestra el aspecto alcanzado en el segundo prototipo.

Figura 12: Segundo prototipo realizado.

5.1.3 Agregando componentes para comunicación entre bastón/lente - Tercera versión del Hardware

Durante la mayor parte del desarrollo, el hardware se mantuvo sin muchos cambios, de hecho, para esta versión se utiliza el mismo hardware que al inicio con algunos agregados. Estos agregados son indispensables para una funcionalidad adicional al kit que lo hace más tolerante a fallos y más adaptable. Como ya se mencionó, en la primera versión se debía configurar desde la aplicación del teléfono, usando a este como puente de información entre el bastón y el lente, esto requería que el teléfono esté encendido de manera permanente para que el lente pueda pasar la información de sensado al bastón. En este modelo, si el teléfono se desconectaba del lente, se apagaba el bluetooth o el teléfono se quedaba sin carga, se perdía la capacidad de sensado del lente. Por este motivo decidimos buscar una alternativa para que el bastón y el lente se sigan comunicado frente a los casos mencionados.

El primer intento consistió en que el bluetooth entre en modo AT abasteciendo y desabasteciendo de corriente el pin EN/KEY (*Enable*) y VCC del módulo en una secuencia determinada. Esto al realizarse de manera manual funcionó correctamente, el pin VCC del módulo bluetooth

era alimentado directamente del pin 5V del Arduino sin límite de corriente (500mA), es decir la limitación real estaba dada por la cantidad total suministrada por la fuente. El pin EN/KEY del módulo alimentado por el pin 3.3V del Arduino también limitado por la capacidad máxima de corriente suministrada por la fuente. Es un dato importante a tener en cuenta puesto que el Arduino puede entregar un total de 200 mA distribuidos en sus pines de entrada salida, y cada pin tiene una entrega máxima de 40 mA, dato importante a la hora de utilizar los pines como fuente de alimentación a otros dispositivos.

Una vez realizada dicha prueba, y sabiendo que dos módulos bluetooth en configuración esclavo no pueden comunicarse directamente sin alguien que establezca la conexión, decidimos automatizar el procedimiento manual de configuración maestro esclavo. Para realizar esto utilizamos el pin 2 digital y el pin 3 analógico del Arduino de la siguiente manera. El pin 2 se programó en modo salida de dato digital, es decir los valores a tomar por este pin son 0-5 volts si está prendido o apagado, este ping suministrará al módulo bluetooth de corriente para operar, el pin 3 fué configurado en modo salida de datos analógico, de tal manera de poder regular el voltaje a suministrar. Los valores posibles a tomar son 0-255 representado en ese rango 0-5 volts, el pin EN/KEY (que será suministrado de corriente por el pin 3) soporta un máximo de 3.3 volts, realizando una regla de tres simple (255x3.3V)/5V=168,3 redondeando para arriba tenemos el valor 169 necesario para suministrar por el pin un voltaje aproximado a 3.3 volts.

Una vez desarrollado el programa necesario para prender y apagar los pines como lo hacíamos manualmente, lo corrimos en un prototipo mucho más simple, sin el módulo de sensado (Hc-sr04) y sin ningún tipo de led o motor conectado al Arduino, con un Arduino Nano.

Las primeras pruebas no arrojaron el resultado esperado, el módulo entraba en una especie de estado AT pero sin reconocer los comandos propios del mismo, en otros casos directamente el módulo bluetooth no encendía, tras realizar varios cambios en los pines para descartar que el

problema sea el Arduino o el bluetooth decidimos pasar a una placa más robusta, el Arduino Uno. Montamos nuevamente el circuito básico para la configuración del bluetooth pero esta vez la prueba fue exitosa, podíamos entrar en modo AT para realizar la configuración del módulo de manera programática. Configurando de manera adecuada pudimos entrar y enviar los comandos al módulo por medio del puerto serie, un paso importante para realizar la configuración automática. Si bien aún estábamos enviando los comandos de configuración manualmente ya habíamos superado la conexión manual necesaria para poder hacer la configuración.

¿Por qué no podíamos configurar los módulos una única vez y dejar todo listo para utilizarlo al encender los dispositivos?.

La respuesta a esto es que no se puede de manera normal, se realizó el intento pero los teléfonos de fábrica vienen configurados en modo maestro. Para cambiar el modo de operación del teléfono es necesario entrar en modo ROOT y la manera de entrar a este modo es diferente para cada dispositivo, algo muy tedioso.

En este punto, necesitábamos el teléfono para poder activar, desactivar y cambiar los parámetros de medición de ambos dispositivos y como no podíamos cambiar su modo de operación asumimos el desafío. El teléfono en el estado inicial y activo se debe conectar a los dos módulos esclavos, configurarlos y pasar información entre ellos. En caso de que la conexión con el teléfono se pierda uno de los dispositivos en modo esclavo debe asumir el rol de maestro y manejar la conexión para que se sigan comunicando.

Se realizó la escritura del código para automatizar el proceso, configuramos el puerto serie, el código de intermitencia para entrar en modo AT y los comandos AT necesarios. El resultado fue exitoso, se logró conectar el dispositivo con el lente sin problemas, previa configuración por medio de parámetros, luego al cortar la energía y encender el dispositivo, éste se conectaba automáticamente. La fase final consistía en hacer la misma conexión con todos los dispositivos conectados al bastón.

Se realizó dicho procedimiento y se probó la ejecución, el resultado fue desalentador. El dispositivo no entraba en modo AT en algunos casos, en otros entraba pero al intentar cambiarse al modo normal para realizar o recibir conexiones quedaba tildado. Pasamos del Arduino Uno al Arduino Nano nuevamente pero el resultado era el mismo o peor por lo que regresamos con el Arduino Uno. Se agregaron un par de resistencias (se visualizarán en la conexión final) en la conexión RX del bluetooth, dicho pin sólo soporta 3.6 volts. Al realizar este ajuste el Arduino dejó de tener problemas al entrar en modo AT aunque no era capaz de conectarse al modo normal. Se realizaron las investigaciones correspondientes y encontramos información sobre la capacidad máxima de suministro de corriente por cada pin de E/S (explicado anteriormente). En las especificaciones técnicas del bluetooth HC-05 mencionan que el dispositivo para funcionar correctamente utiliza una corriente de 50 mA (miliamperio). He aquí el problema con el bluetooth, tanto el Arduino Nano como el Uno no puede entregar esta corriente por los pines de E/S haciendo que el mismo no pueda controlarlo de manera correcta, esta es la causa del problema de intermitencia. Para poder solucionar este problema buscamos posibles alternativas para de alguna manera, pasar más corriente al bluetooth controlándolo por medio del Arduino, una de ellas es la utilización de un transistor. Un transistor es un dispositivo que regula el flujo de corriente o de tensión sobre un circuito actuando como un interruptor y/o amplificador para tensiones y corrientes.

Hay dos tipos de transistores, NPN y PNP ambos de unión bipolar (BJTs). Son controlados por corriente permitiendo la amplificación de corriente. Con una corriente en la base del transistor permitimos corrientes mucho mayores en los pines del emisor y colector. Ambos transistores tienen este fin, la diferencia entre estos está en cómo la energía debe ser suministrada a los pernos terminales para que proporcionen esta funcionalidad de amplificación o traspuesta. Los NPN reciben voltaje positivo en el colector y la base para su correcto funcionamiento mientras que los PNP reciben un voltaje positivo en la terminal del emisor y un voltaje

negativo o mejor dicho, más bajo (que la suministrada en la terminal del emisor) para la terminal de la base.

El dispositivo utilizado es un transistor de propósito general denominado 2N2222 del tipo NPN. En las imágenes de la Figura 13 y 14 se puede observar su composición y forma de funcionamiento [20].

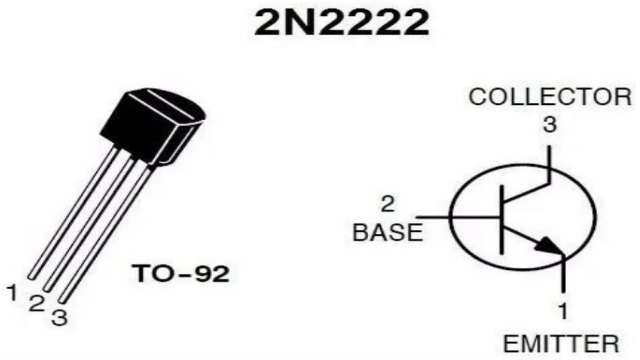

Figura 13: Segundo prototipo realizado. Transistor NPN 2N2222.

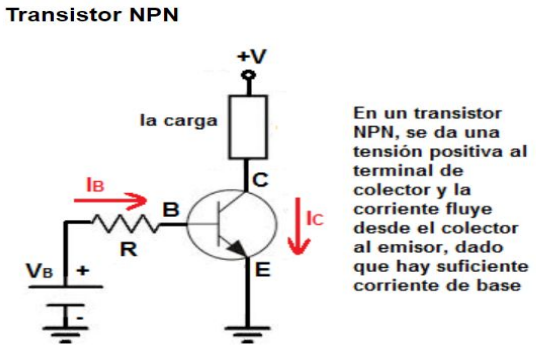

Figura 14: Segundo prototipo realizado. Funcionamiento transistor NPN.

Gracias a la utilización del transistor, se pudo suministrar al módulo bluetooth la corriente directamente desde la fuente, con una aplicación mínima de corriente en la base, alcanzó con 10 nA para su corte o activación. Se actualizó el software para lograr el funcionamiento pretendido. Luego de un par de ajustes, las pruebas fueron exitosas.

El hardware a nivel general no presentó grandes cambios, los dispositivos utilizados en las diferentes etapas son los mismos con la

diferencia de la distribución lógica asociada a los pines y la incorporación de componentes adicionales para corregir los problemas de corriente como las resistencias y el transistor. Del software asociado al hardware no se puede decir lo mismo, pues este fue modificado en varias etapas ajustándose y agregando nuevas funcionalidades. A continuación realizamos una caracterización general del software en sus diferentes etapas.

5.1.4 Conectando el bastón con el teléfono - Primera versión del Software

Habiendo explicado el ensamblado de las componentes, en este punto, comenzaremos a describir las etapas del desarrollo del software descargado en la placa de desarrollo. En la primera parte del código a nivel de comentario, explicamos la configuración de los pines y la utilización de cada uno de ellos. La segunda sección comprende la declaración de variables donde definimos las variables globales y asociamos identificadores a los pines para referenciarlos dentro del código.

La sección de setup o configuración como su nombre lo sugiere, configura los pines en sus diferentes modos de operación, también en ella se configura la velocidad de comunicación de los puertos serie y funcionamiento del dispositivo bluetooth. Continuando con la estructuración del código lo siguiente en cuestión son la definición de las funciones invocadas, acciones a realizar en determinados momentos que se encuentran encapsuladas y referenciadas como verbos. Es en esta parte donde se presentaron la mayoría de cambios entre las diferentes versiones del código.

Funciones:

- La función de apagado es la encargada de apagar los leds de sensado (pin 13) y el led verde provisional que representa al motor de vibración (el cual es referenciado por el pin 3).
- La función de vibrar es la encargada de simular la vibración del prototipo pasándole una intensidad de vibración, para los fines

prácticos utilizamos una luz led verde conectado al pin 3 en la etapas iniciales, antes de tener el motor de vibración.

- La función de distancia es la que obtiene la respuesta de rebote del sensor Hc-sr04 y realiza el cálculo de distancia entre el tiempo de emisión de la onda y la recepción del rebote.
- Por último el programa principal que orquesta las funcionalidades mencionadas, se encarga de ejecutar las funciones constantemente para el correcto funcionamiento del bastón.

A continuación se presenta el código Arduino del primer prototipo funcional desarrollado:

```
/*
Firmware para Bastón no videntes con sensores y actuadores sensitivos. v_0.5 (viene de v_0.4)
Autor: Albarracin Emiliano (emilianoalbarracin.ea@gmail.com)

Hardware implementado en esta versión:
  -Arduino NANO
  -Vibrador con electronica basica de contol de potencia, conectado al PIN3 (PWM).
  -Blue Thooth HC05, conectados a RX0 y TX1 del Arduino.
  -Sensor de disrancia por Ultrasonido HC-SR04, conectado a los pines 4 y 5.
En esta versión probaremos agregar la funcionalidad de encender el vibrador con diferentes intensidades según la distancia del objeto (cuanto más cerca mas intenso).
Para activar dicha opción deberá recibir por el serial la Orden = 2.
 Orden = 0 (apagado)
 Orden = 1 (On/Off a 100cm)
 Orden = 2 (intensidad variable según distancia a partir de los 200 cm)

ACLARACION: Al cargar el firmware en el arduino, por usar el BlueTooth conectado a los pines 0 y 1,
            es probable que sea necesario desenergizar o desenchufar la placa BlueTooth para que no interfiera con el serial del USB.

*/
//--------------------------------------------------------------------

int vibrador = 3;    //Conexión control vibrador (PWM).
int led = 13;        //Conexión del led testigo (viene incorporado en la placa arduino).
int eco = 4;         //Conexión del pin receptor del Ultrasonido.
int trig = 5;        //Conexión del pin emisor del Ultrasonido.
char orden;          //lo usamos para almacenar el dato del bluetooth.
int distancia;       //lo usamos para almacenar el valor de la medición tomada.
int intensidad;      //lo usamos para setear la intensidad de la vibración (PWM).

//--------------------------------------------------------------------

void setup() {
  Serial.begin(9600);           // el BlueToth trabaja como puerto serial.
  pinMode (vibrador, OUTPUT);   //indicamos que el PIN donde está conectado el vibrador es una salida.
  pinMode (led, OUTPUT);        //activamos el pin 13 como salida, vamos a usar el led que ya trae el arduino como testigo.
  pinMode(trig, OUTPUT);        //configuramos el pin del Trig como salida.
  pinMode(eco, INPUT);          //configuramos el pin del Edo como entrada.
  analogWrite (vibrador,0);     //configuramos en "0" la salida PWM del vibrador para asegurarnos de que comience apagado.
  digitalWrite(led, LOW);       //configuramos que el led del pin 13 inicie apagado.

}
```

```
//--------------------------------------------------------------------

void apagar(){                  //escribimos el comportamiento de vibración como una función aparte para futuras modificaciones.
  analogWrite(vibrador, 0);     //el vibrador está conectado a 5v, con lo que 0PWM es equivalente a 0v y 255PWM es equivalente a 5v (vibra mas suave o mas fuerte).
  digitalWrite(led, LOW);       //apagamos led del pin 13.
}
//--------------------------------------------------------------------

void vibrar( int aux){                //escribimos el comportamiento de vibración como una función aparte para futuras modificaciones.
  analogWrite(vibrador, aux);     //el vibrador está conectado a 5v, con lo que 0PWM es equivalente a 0v y 255PWM es equivalente a 5v (vibra mas suave o mas fuerte).
  if (aux>0){
    digitalWrite(led, HIGH);      //encendemos led del pin 13 como testigo.
  }else{
    digitalWrite(led, LOW);       //encendemos led del pin 13 como testigo.
  }

}

//--------------------------------------------------------------------

int calcularDistancia(){        //Creamos una funcion para realizar la medición cada vez que queramos.

int duracion;                   //Variable donde almacenar la medición en tiempo (lo que tarda en regresar el Eco).
digitalWrite(trig, LOW);        //Nos aseguramos de tener apagado el emisor de ultrasonido.
delayMicroseconds(2);           //esperamos dos microsegundos y luego comienza la secuencia de medición.
digitalWrite(trig, HIGH);       //encendemos el emisor de ultrasonido.
delayMicroseconds(10);          //dejamos emitiendo durante 10 microsegundos (duración del pulso).
duracion = pulseIn(eco, HIGH);  //tomamos la medida del tiempo transcurrido hasta que el receptor detecte el pulso del eco.
distancia= duracion*0.034/2;    //hacemos un ajuste matemático para pasar de tiempo transcurrido a distancia recorrida, teniendo como dato la velocidad del sonido.

if (distancia<1 || distancia>500){    //aplicamos un filtro para eliminar las mediciones con valores menores a 1cm o mayores a 5 metros (fuera del rango), los convierte en
      distancia=0;
      }
Serial.print("Distancia:");           //mostramos en el monitor del puerto serial la medida tomada.
Serial.println(distancia);
Serial.print("Intensidad:");          //mostramos en el monitor del puerto serial la intensidad.
Serial.println(intensidad);

return distancia;               //devuelve el valor de la medición.

}
//--------------------------------------------------------------------------------
```

```
int  orden_actual=1;
int  orden_anterior;
bool falg;
char numStr[4];
long  cotaSuperior;
int index;
void loop() {
   if (Serial.available()==1) {
      orden_anterior=orden_actual;
      orden_actual=Serial.read();                                   //si recibe algo el blueTooth, guarda la orden.
      falg=false;
      cotaSuperior=10;
   }
   else if(Serial.available()>1){
      falg=true;
      index=0;
      orden_anterior=orden_actual;
      orden_actual=Serial.read();
      while(Serial.available()){
           numStr[index]=Serial.read();
           index+=1;
      }
      numStr[index]='\0';
      cotaSuperior= atol(numStr);

   }
   switch (orden_actual) {                                          //segun la orden del bluetooth sera la funcion que se ejecuta
      case '0':                                                     //con el mensaje "0" apagamos el vibrador.
        if (orden_anterior=0){
          apagar();                                               //ejecutamos la funcion para "apagar()".
          Serial.println("Estado: APAGADO");                      //mostramos en el monitor del puerto serial el estado.
        }
        break;
      case '1':                                                 //con el mensaje "1" encendemos el vibrador.
        calcularDistancia();                                    //toma una medición de distancia con el Ultrasonido.
        if (distancia >1 && distancia < cotaSuperior) {               //si hay una distancia menor a 50 centimetros vibra fuerte.
             intensidad = 255;
        }
        else  {                                                 //si no apaga el vibrador.
             intensidad = 0;
        }
        vibrar(intensidad);
        Serial.print("Parametro:");
        Serial.println(cotaSuperior);
        break;
      /*case '2':                                                   //con el mensaje "1" encendemos el vibrador.
        calcularDistancia();                                    //toma una medición de distancia con el Ultrasonido.



        if (distancia >1 && distancia < 200) {                //si hay una distancia entre 1 y 200 centimetros ajusta el valor de intensidad.
            intensidad = map(distancia, 1 , 200, 254, 0);
        }
        vibrar();*/                                             //activa la vibración a la intensidad configurada.
        break;

   } //FIN SWITCH

}    //FIN LOOP
```

Figura 15: Primer prototipo código Arduino.

5.1.4.1 Testeo de la primer versión del Software con el Hardware

Esta versión desarrollada fue probada con una aplicación genérica descargada de la PlayStore de Android llamada Android Bluetooth Controller. Dicha aplicación permite enviar órdenes a un dispositivo bluetooth en concreto con el cual el teléfono debe emparejarse previamente. Las órdenes programadas en los botones son simples, '1' para encender el dispositivo y empezar a realizar el sensado y '0' para finalizar y apagar el bastón.

A continuación se visualizan los estados posibles del bastón.

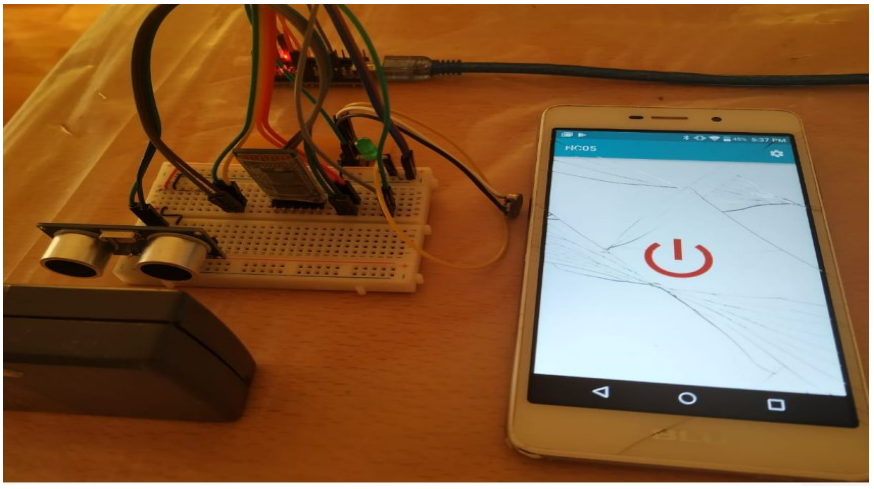

Figura 16: Versión de software probado con la aplicación Android Bluetooth Controller (botón apagado).

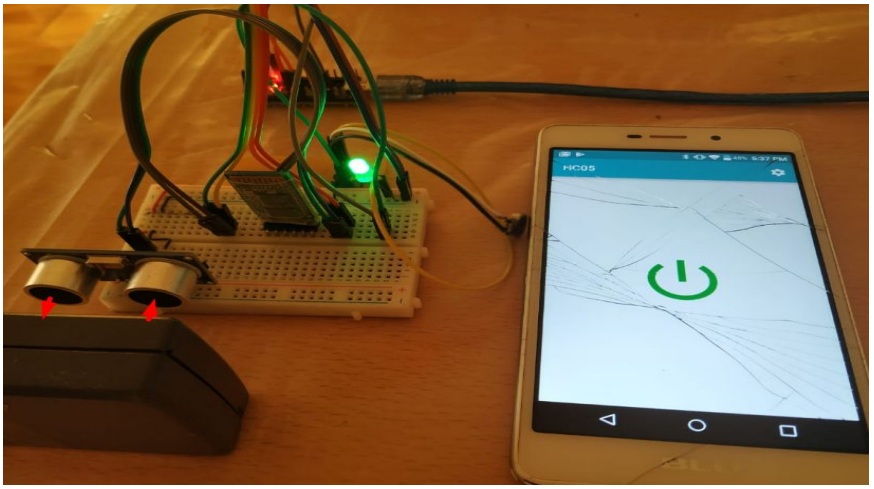

Figura 17: Versión de software probado con la aplicación Android Bluetooth Controller (botón encendido).

En las imágenes anteriores se puede apreciar el funcionamiento del prototipo. La imagen de la Figura 16 muestra el dispositivo montado

encendido ya emparejado con la aplicación móvil, la misma se muestra con el botón de color rojo expresando que lo que sea que esté conectado al teléfono se encuentra apagado, la aplicación del teléfono envía un “0” al presionar el botón (suponiendo que el botón de la aplicación estaba en verde y al presionarlo se pasó a rojo). La orden “0” enviada al bastón según el código visto más arriba hace que el dispositivo deje de sensar y se apague el led de detección. Como podemos apreciar en la imagen de la Figura 17, al presionar el botón nuevamente este se coloca en verde enviando la orden “1” al bastón que según el código, comienza con el sensado y verifica por defecto si hay algún obstáculo a 10 centímetros del sensor ultrasónico. Para esta prueba colocamos un objeto dentro del rango y se aprecia que el led de notificación, el que está asociado al pin 3 (led verde) se encuentra encendido indicando que hay un objeto dentro del rango de medición.

En la siguiente imagen se puede apreciar como al alejar el dispositivo con el módulo encendido “1” el led verde se hace más tenue hasta que el objeto sale del rango de medición y el led se apaga.

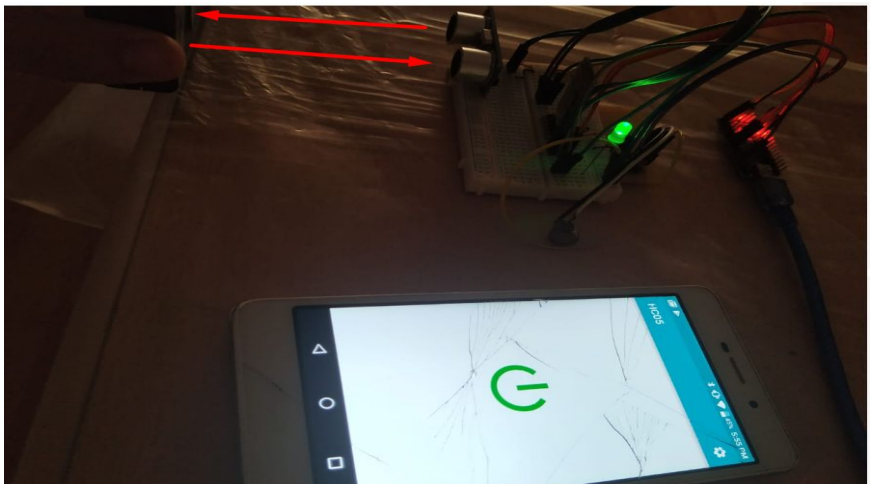

Figura 18: Prueba de sensores, luz prendida

Figura 19: Prueba de sensores, luz apagada.

5.1.5 Configurando distancias desde el teléfono - Segunda versión del Software

Para el segundo prototipo funcional del bastón se desarrolló un nuevo firmware montado sobre el Arduino capaz de recibir órdenes adicionales y cambiar la distancia de detección de obstáculos del módulo ultrasónico, la idea de esta funcionalidad es que los usuarios puedan hacer un ajuste de medición en base a sus características ergonómicas, las de la superficie y el ámbito cotidiano en donde se desplazan. Junto con esta nueva funcionalidad, se incorporó en el hardware un motor vibratorio con el circuito asociado para controlar el voltaje que a éste se le suministra. El motor funciona con el mismo principio que el led (pin 3). Al encenderse el bastón, este automáticamente configura la detección de obstáculos en 10 centímetros, pero el módulo si bien está encendido, no realiza mediciones hasta que el comando de inicio “1” sea enviado a través del puerto serie (9600 baudios) del Arduino, una vez que la orden de sensado activado es enviada, el prototipo entra en estado de medición y permite mediante una serie de códigos especiales que recibe por el mismo puerto, cambiar la distancia de detección.

En la siguiente imagen se muestra el código Arduino que permite realizar dicha actividad:

```
void loop() {
  if (Serial.available() == 1) {
    orden_anterior = orden_actual;
    orden_actual = Serial.read();                                  //si recibe algo el blueTooth, guarda la orden.
    falg = false;
    cotaSuperior = 10;
    Serial.println(orden_actual);
  }
  else if (Serial.available() > 1) {
    falg = true;
    index = 0;
    orden_anterior = orden_actual;
    orden_actual = Serial.read();
    Serial.println(orden_actual);
    while (Serial.available()) {
      numStr[index] = Serial.read();
      index += 1;
      Serial.println(numStr[index]);
    }
    numStr[index] = '\0';
    if (atol(numStr) % 3 == 0)
    {
      if (atol(numStr) == '15' or atol(numStr) == '30' or atol(numStr) == '90' or atol(numStr) == '120' ) {
        cotaSuperior = atol(numStr);
      }
    } else {
      cotaSuperior = atol(numStr);
    }
  }
  switch (orden_actual) {                                           //segun la orden del bluetooth sera la funcion que se ejecuta
    case '0':                                                       //con el mensaje "0" apagamos el vibrador.
      if (orden_anterior = 0) {
        apagar();                                                 //ejecutamos la funcion para "apagar()".
        //Serial.println("Estado: APAGADO");                        //mostramos en el monitor del puerto serial el estado.
      }
      break;
    case '1':                                                     //con el mensaje "1" encendemos el vibrador.
      calcularDistancia();                                        //toma una medición de distancia con el Ultrasonido.
      if (distancia > 1 && distancia < cotaSuperior) {              //si hay una distancia menor a 50 centimetros vibra fuerte.
          intensidad = map(round((distancia*255)/cotaSuperior), 1 , 255, 255, 1);
      }
      else  {                                                     //si no apaga el vibrador.
```

Figura 20: Nuevo código para recibir órdenes adicionales.

Se puede apreciar en el código un comportamiento distinto al algoritmo del primer prototipo al recuperar las órdenes obtenidas del puerto serie. El algoritmo consta de una condición que verifica el estado del bastón, si está activado o desactivado, esta verificación se realiza en el primer "*if*", luego en el segundo "*if*" se verifica si la orden entrante es mayor en longitud a la de un carácter. En el caso de que la orden posea más de un carácter se extraen los datos serializados y se los transforma a números enteros, estos números enteros representan la nueva distancia de medición actualizando la cota superior de detección. Adicional al algoritmo se desarrolla un aplicación Android que permite la conexión con el prototipo y modificación mediante botones de las distancias de medición que se muestra en la imágen de la Figura 21.

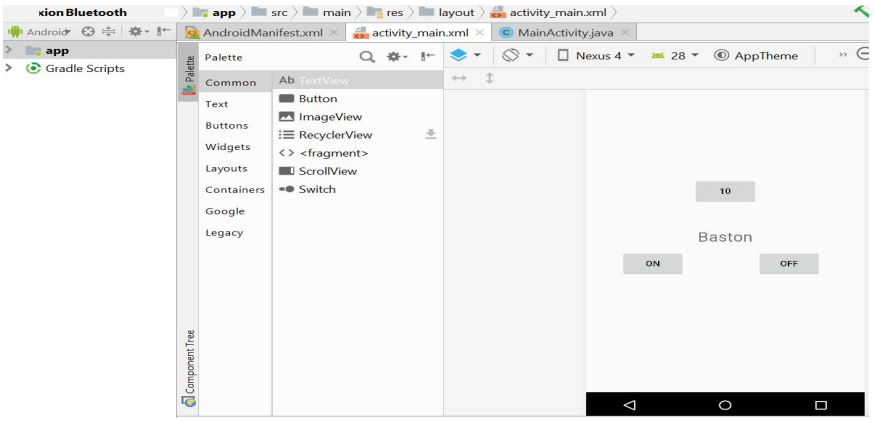

Figura 21: Segundo Prototipo Software Bastón.

5.1.6 Actualizando interfaz del teléfono para una configuración más amigable del bastón - Tercera versión del Software

Para la tercera versión del software se realizaron modificaciones al software asociado al teléfono, las modificaciones comprenden la posibilidad mediante botones de ajustar los rangos de medición del bastón por medio de un interfaz muy sencilla.

En la siguiente imagen se visualiza el menú de configuración del bastón:

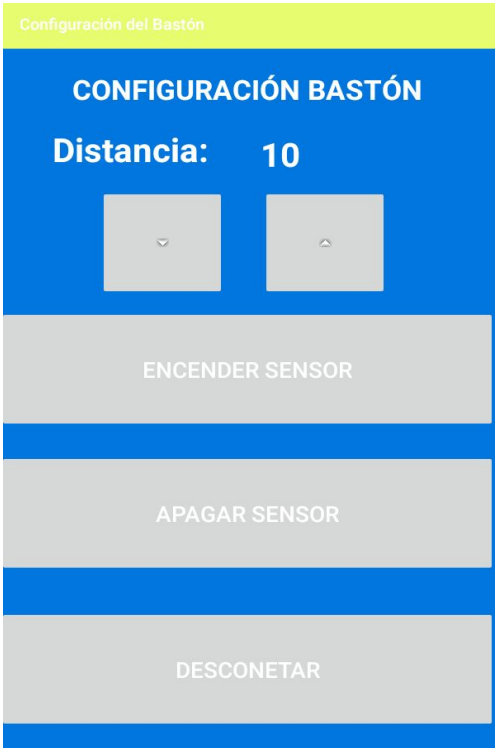

Figura 22: Tercer Prototipo Software Bastón.

Fue probado por personas con discapacidad visual durante la exposición de ciencias de la Facultad de Informática. Dichas personas expusieron su percepción a partir de su experiencia como usuarios: estas percepciones se tradujeron en sugerencias para modificar la interfaz y hacerla más amigable. También, mediante las pruebas, se comprobó empíricamente la insuficiencia de un solo dispositivo para detectar los obstáculos, puesto que el sensor ultrasónico, al estar adherido al bastón, y al estar éste en constante movimiento, suspendía la medición de objetos al aparecer y desaparecer estos de su rango y ángulo de percepción. Por este motivo se decidió incorporar lentes al kit de movilidad: los lentes, al estar en la cabeza, la cual tiene rango y ángulo de movimiento inferior al bastón, permitió tener una segunda fuente con la cual se obtuvo mayor certeza de medición de los objetos que constituyan obstáculos, además de servir como alerta ante objetos que se encuentren suspendidos. Los lentes contarían, al igual que el bastón, con prestaciones muy similares.

5.1.7 Incorporando conexión múltiple bastón /lente por medio del teléfono - Cuarta versión del software

En esta versión se incorpora la funcionalidad de conexión múltiple entre los dispositivos. Como se ha mencionado anteriormente (apartado más arriba sobre la decisión tomada para realizar dicha conexión) había una necesidad de cubrir la detección de obstáculos por encima de la cintura, lo que requería de un segundo módulo capaz de realizar esta funcionalidad. Para ello se debía incorporar la lógica necesaria para controlar el nuevo dispositivo, esto más bien se logra duplicando el código anterior e incorporando en la aplicación Android los botones necesarios para este nuevo módulo. La forma tradicional de conexión de la versión anterior cambia, anteriormente al elegir la opción en el menú de “Camino Seguro”, uno debía de seleccionar el dispositivo desde una lista de dispositivos vinculados previamente al teléfono, el dispositivo elegido actuaría de bastón en cambio en esta nueva versión en cada botón de prendido (ON) asociado a los dos módulos había una acción que se disparaba realizando la vinculación entre el teléfono y el módulo. Las acciones asociadas que realizaban la vinculación utilizan la MAC (*Media Access Control*) del dispositivo al cual se requiere conexión, la MAC estaba hardcodeada en las funciones. Esta versión no se pensó como una definitiva si no más bien una versión beta para realizar pruebas de conexión y funcionamiento, ajustar detalles.

Vista del menú de configuración de los módulos (Android)

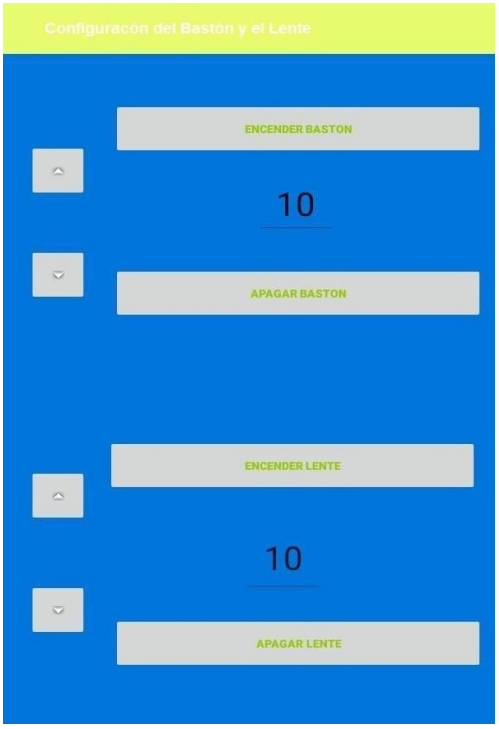

Figura 23: Prototipo conexión Bastón y Lente.

Función para conectar el bastón (Android)

```
private void conectarBaston(){
  btAdapter = BluetoothAdapter.getDefaultAdapter();
        //Direccion mac del dispositivo a conectar
  BluetoothDevice device = btAdapter.getRemoteDevice(MACBaston);
  try
  {
          //Crea el socket sino esta conectado
    if(!estado)
    {
      btSocket = createBluetoothSocket(device);
      estado = btSocket.isConnected();
    }

  }
  catch (IOException e)
  {
    Toast.makeText(getBaseContext(), "La creacción del Socket fallo", Toast.LENGTH_LONG).show();
  }

        // Establece la conexión con el socket Bluetooth.
  try
  {
          //Realiza la conexion si no se a hecho
    if(!estado)
    {
      btSocket.connect();
      estado = true;
      MyConexionBT = new ConexionThread(btSocket);
      MyConexionBT.start();
      MyConexionBT.write("+.");
      Toast.makeText(MainActivity.this, "Conexion del bastón realizada exitosamente", Toast.LENGTH_SHORT).show();
    }

    else{
      Toast.makeText(MainActivity.this, "El bastón ya esta vinculado", Toast.LENGTH_SHORT).show();

    }
  }
  catch (IOException e)
  {
    try {
      Toast.makeText(MainActivity.this, "Error:", Toast.LENGTH_SHORT).show();
      Toast.makeText(MainActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
      btSocket.close();
    }
    catch (IOException e2) {}
  }
}
```

Figura 24: Función para conectar el bastón.

Función para conectar el lente (Android)

```
private void conectarLente(){
    btAdapterL = BluetoothAdapter.getDefaultAdapter();

        //Direccion mac del dispositivo a conectar
    BluetoothDevice device = btAdapterL.getRemoteDevice(MACLente);
    try
    {
            //Crea el socket sino esta conectado
        if(!estadoL)
        {
            btSocketL = createBluetoothSocket(device);

            estadoL = btSocketL.isConnected();
        }

    }
    catch (IOException e)
    {
        Toast.makeText(getBaseContext(), "La creacción del Socket fallo", Toast.LENGTH_LONG).show();
    }

        // Establece la conexión con el socket Bluetooth.
    try
    {
            //Realiza la conexion si no se a hecho
        if(!estadoL)
        {
            btSocketL.connect();
            estadoL = true;
            MyConexionBTL = new ConexionThread(btSocketL);
            MyConexionBTL.start();
            MyConexionBTL.write("+");
            Toast.makeText(MainActivity.this, "Conexion del lente realizada exitosamente", Toast.LENGTH_SHORT).show();
        }

        else{
            Toast.makeText(MainActivity.this, "El lente ya esta vinculado", Toast.LENGTH_SHORT).show();

        }

    }
}
```

Figura 25: Función para conectar el lente.

En las dos versiones del código se puede visualizar la utilización de variables globales, MACBaston y MACLente que se encuentran definidas previamente al inicio del programa.

5.1.8 Ultimando detalles, versión final del software - Quinta versión del Software

Esta versión comprende modificaciones tanto en el software del teléfono como del bastón. Aquí se da soporte a la **Tercera versión del Hardware** del bastón por lo que se realizaron varios cambios en la codificación. Este código posee una sección de declaración de variables mucho más extensa que se visualiza en la siguiente imagen.

Baston_V0.9_Integrador.ino

```
*/
//---------------------------------------------------------------------

int vibrador = 9;    //Conexión control vibrador (PWM).
int enBT = 8;        //Pin EN BlueToth
int powerBT = 2;     //Pin power BlueToth
int led = 13;        //Conexión del led testigo (viene incorporado en la placa arduino).
int eco = 4;         //Conexión del pin receptor del Ultrasonido.
int trig = 5;        //Conexión del pin emisor del Ultrasonido.
char orden;          //lo usamos para almacenar el dato del bluetooth.
int distancia;       //lo usamos para almacenar el valor de la medición tomada.
int intensidad;      //lo usamos para setear la intensidad de la vibración (PWM).
int inicio = 0;
bool comandoInicial = false, desconectado;
char aux;
unsigned long startMillis = startMillis = millis();
unsigned long currentMillis;
const unsigned long period = 5000;
String resultado1, resultado2, command, buff;
char   orden_actual = '+';
char   orden_anterior;
bool   flag;
long   cotaSuperior = 0;


//---------------------------------------------------------------------

void setup() {
  BT.begin(38400);              // el BlueToth trabaja como puerto serial.
  Serial.begin(38400);             // se utiliza para hacer debugg.
  pinMode (vibrador, OUTPUT);   //indicamos que el PIN donde está conectado el vibrador es una salida.
  pinMode (led, OUTPUT);        //activamos el pin 13 como salida, vamos a usar el led que ya trae el arduino como testigo.
  pinMode(trig, OUTPUT);        //configuramos el pin del Trig como salida.
  pinMode(eco, INPUT);          //configuramos el pin del Edo como entrada.
  analogWrite (vibrador, 0);    //configuramos en "0" la salida PWM del vibrador para asegurarnos de que comience apagado.
  digitalWrite(led, LOW);       //configuramos que el led del pin 13 inicie apagado.
  Serial.println("Encendido");
  pinMode(enBT, OUTPUT);        //Salida EN
  pinMode(powerBT , OUTPUT);    //Definir pin como salida
  digitalWrite (powerBT, HIGH); //Enciende el modulo
  digitalWrite(enBT , LOW);     // poner el Pin en LOW

`
```

Figura 26: Versión cinco del código.

Se declaran variables referentes al motor vibratorio, a la alimentación del bluetooth (pins EN/KEY y PWD), pin led de notificación, pines para utilizar el módulo ultrasónico, distancias, intensidad de vibrado, flags utilizados como banderas y parámetros de tiempo de ejecución de inicio y fin para una acción determinada (más adelante se explica su utilización). Se puede observar en la sección setup la asociación de estos pines.

En la siguiente sección se visualiza el programa principal y explicamos el funcionamiento de las partes generales.

```
void loop() {
  flag = false;
  inicializar();
  if (desconectado == true) {
    cambiarAMaestro();
    BT.write("+.");
    delay(1000);
    BT.write("+10.");
    desconectado = false;
  }
  if (BT.available()) {
    buff=BT.readString();
    command=buff;
    Serial.println("El comando enviado es: " + buff + " de longitud" + buff.length());
    if (buff.length() >=2) {
      if(command!="+." && command!="-." &&  command.charAt(0)!= '/'){
        command = substrbefore(buff, '_');
        String check = substrafter(buff, '_');
        if (checkSum(command) == check.toInt()) {
          flag = true;
        } else {
          BT.write("=*");
        }
      }else{
        flag = true;
      }
      if (flag == true) {
        if (command.charAt(0) == '+' || command.charAt(0) == '-') {
          orden_anterior = orden_actual;
          orden_actual = command.charAt(0);
          if (command.length() > 3 && !command.equals("+0.\n")) {
            comandoInicial = true;
            cotaSuperior = command.substring(1, command.length() - 1).toInt();
            Serial.println("Se actualiza cota superiro");
            Serial.println(command.length());
          }
        } else {
          if (command.charAt(0) == '/') {
            orden_anterior = orden_actual;
            orden_actual = command.charAt(0);
          }
        }
```

Figura 27: Programa principal.

El programa comienza con la función **inicializar** (se detalla el código más abajo). Esta función realiza la intermitencia energética necesaria en los pines EN/KEY y PWD del bluetooth para ingresar en modo AT. Envía el comando "***AT+CMODE=1***" para cambiar el modo de operación del bluetooth a esclavo y así éste permita la comunicación con el teléfono, cambia la velocidad de *chanel* de comunicación nuevamente a ***9600*** baudios. (para entrar en modo AT es necesario que la velocidad del puerto serie esté configurado en ***38400*** baudios). Se realiza una especie de ping para determinar la conexión con el teléfono, como se puede observar esta sección de código solo se ejecuta una vez, cuando el módulo del bastón arranca por primera vez.

En la siguiente sección verifica si hay alguna orden proveniente del teléfono o del lente, en caso de haber una, rearma la orden original, recordemos que el modo de comunicación entre los dispositivos se realiza de manera serial, es decir caracter por caracter. En esta versión del código mejoramos la comunicación entre los puestos que pueden hacer que haya interferencias entre los sistemas al transmitir y recibir señales, más con estos dispositivos baratos y de fácil acceso. Un comportamiento que ocurre frecuentemente entre la comunicación del bastón y el teléfono es la interferencia en las señales, es decir apretamos el botón para aumentar y/o disminuir la distancia de medición, el teléfono registraba dicha configuración, enviaba los datos pero en la recepción de la orden del lado del bastón dicho mensaje estaba corrupto, intentaba realizar la modificación del parámetro de medición fallando. Ejemplo: al apretar el botón de aumentar distancia, el contador interno del teléfono aumenta de 10 centímetro a 15 centímetro pero el bastón al recibir la señal distorsionada no podía terminar de configurar la nueva distancia generando discrepancia entre la variable interna del teléfono (15 centímetros) y la variable interna real del módulo (10 centímetros). Para detectar el caso de la distorsión del mensaje generamos una función tanto del lado del bastón como del teléfono denominada checkSum muy común en redes informáticas.

El **checkSum** o suma de comprobación es un dato pequeño derivado de un bloque de datos digital con el fin de detectar errores que pueden haberse producido durante la transmisión, se utiliza para verificar la integridad de los datos pero no se cuenta con ella para verificar la autenticidad de los mismos. El procedimiento resultante se denomina suma de verificación [21]. Si la suma de verificación calculada para la entrada de datos coincide con el valor de la suma de verificación calculada previamente, hay una alta posibilidad de que los datos no estén corruptos por la transmisión. Hay varias formas de calcular esta suma de comprobación, en nuestro caso desarrollamos una manera muy sencilla para determinarla, todas las órdenes que impliquen un cambio de configuración emitidas por el teléfono, ejemplo (+. ó +10.) tienen un juego

de caracteres al final que representa la suma total de los caracteres ascii que componen la orden. Es decir que una orden como las de +10. que simboliza setear la variable de medición interna del bastón o del lente en 10 centímetros, tiene adjunta la suma de sus caracteres (+10._186) esto se interpreta de la siguiente manera +=43;1=49;0=48;.=46. Se obtienen de la entrada los caracteres antes del "_" (substring before "_") que representan la orden y los caracteres después del "_" (substring after "_") que representa la suma de comprobación. Se aplica la función checkSum definida por nosotros al primer segmento de la entrada (correspondiente a la orden) y ésto retorna la suma de los caracteres que componen la orden, en caso de que la función checkSum aplicada al primer segmento y el número resultante del segundo segmento (de aplicar el substring after "_" a la entrada) sean iguales podemos asegurar que la orden no está corrupta. Continuando con la explicación del código seguimos con el siguiente paso para determinar la consistencia de la orden, si la orden no corresponde con la salida de la función checkSum se emite una orden especial al emisor, "=*" que simboliza el reenvío de la orden por parte del mismo. Uno podría decir que la orden de retransmisión no posee ninguna suma de comprobación y esto es verdad pero para las pruebas realizadas la distorsión de los datos enviados se realizaba en un solo sentido, del teléfono al bastón y/o lente, en cambio en el sentido contrario no se encontró evidencia suficiente en las pruebas realizadas para implementar la misma funcionalidad en el software del teléfono. Esto lo dejamos abierto para futuras mejoras.

Una vez obtenida la orden consistente se procede a clasificarla para determinar si es una orden de encendido o apagado, es una orden para aumentar o disminuir la distancia de configuración del bastón o es una orden proveniente del módulo del lente para procesar una petición de vibrado por detección de algún obstáculo. En caso de ser una orden para modificar las mediciones, se guarda el nuevo valor.

En la siguiente imágen se puede observar la función ejecutarSentencia desarrollada en el arduino.

```
  ejecutarSentencia();
  currentMillis = millis();
  if (currentMillis - startMillis >= period && comandoInicial == true && flag == false) {
    if (!BT.available()) {
      BT.write("?*");
      Serial.println("Enviar Pin");
      bool aux = hayConexionV2();
      Serial.println(aux);
      if (!aux) {
        desconectado = true;
        Serial.println("Desconectado");
        comandoInicial = false;
      }
    }
    startMillis = millis();
  }

}    //FIN LOOP
```

Figura 28: Función ejecutarSentencia.

Seguimos con la función **ejecutarSentencia**, dicha función encapsula el procesamiento de la orden en sí, como en las versiones anteriores. Contiene un "*Case*" que realiza una determinada acción en base a la orden recibida, prender o apagar el módulo, aumentar o disminuir la distancia de medición, calcular la distancia de medición o ejecutar la orden directa de vibración con una intensidad específica enviada por el lente.

Lo siguiente es parte de una verificación de conectividad entre el bastón y el teléfono agregada al bastón que viene de la mano con la incorporación del cambio de modo de operación en el dispositivo bluetooth que éste incorpora. Lo que realiza este segmento de código es verificar conectividad mediante el envío y recepción de un comando "ping" entre ambos y medir la tasa de tiempo en que el ping es enviado al teléfono y en que éste responda al bastón. Si al cabo de 5 segundos desde que se envía una petición de testeo de conectividad, el módulo no recibe una respuesta por parte del teléfono, éste se configura en modo AT ejecutando una función denominada **cambiarAMaestro** que se encuentra al inicio de la imagen controlada por una variable bandera que se activa cuando se pierde conectividad con el teléfono.

Esta función como su nombre lo indica realiza el proceso inverso a la función inicializar, configurar el bluetooth para entrar en AT y cambiar el modo de operación a maestro, con ello setea una MAC determinada (en este caso la MAC del bluetooth montado en el lente) para realizar una

conexión directa con dicho dispositivo. Una vez seteada la MAC y el modo de operación regresa a la velocidad predeterminada (9600 baudios) para realizar intercambio de datos con el lente de manera directa. Tienen en cuenta que este modo de operación se activa únicamente cuando se pierde conexión con el teléfono, en este modo no es posible configurar la distancia de medición del bastón y del lente pues este es un plan de contingencia frente a fallos de conexión con el teléfono que pueden derivar de muchas causas.

En esta sección se presentan los módulos citados anteriormente.

```
void inicializar() {
  if (inicio == 0) {
    delay(2000);
    digitalWrite(enBT , HIGH);
    delay(1000);
    digitalWrite (powerBT, LOW);
    delay(1000);
    digitalWrite (powerBT, HIGH);
    delay(1000);
    Serial.println("Entre");
    inicio = 1;
    BT.println("AT");
    resultado1 = obtenerRespuesta();
    Serial.println(resultado1);
    Serial.println("Es maestro, cambiando a esclavo");
    BT.println("AT+ROLE=0");
    //delay(2000);
    resultado1 = obtenerRespuesta();
    BT.println("AT+CMODE=1");
    //delay(2000);
    resultado1 = obtenerRespuesta();
    Serial.println(resultado1);
    if (resultado1.equals("OK\r")) {
      Serial.println("Es esclavo");
    }
    BT.println("AT+ROLE");
    resultado1 = obtenerRespuesta();
    resultado2 = obtenerRespuesta();
    Serial.println(resultado1);
    Serial.println(resultado2);
    Serial.println("Channel de comunicación al 38400");
    digitalWrite(enBT , LOW);
    delay(2000);
    digitalWrite(powerBT , LOW);
    delay(3000);
    configurarPuertoSerieEsclavo();
    digitalWrite(powerBT , HIGH);
    Serial.println("Testeando channel 38400");
    desconectado = false;
    flag = false;
    BT.setTimeout (60);
  }
}
```

Figura 29: Función Iniciar.

```
}
int checkSum(String chars) {
  int check = 0;
  int aux = 0;
  for (int c = 0; c < chars.length(); c++) {
    aux = chars.charAt(c);
    check = check + aux;
  }
  return check;
}
```

Figura 30: Función CheckSum.

```
void ejecutarSentencia() {
  switch (orden_actual) {                                          //segun la orden del bluetooth sera la funcion que se ejecuta
    case '-':
      //con el mensaje "0" apagamos el vibrador.
      apagar();                                                    //ejecutamos la funcion para "apagar()".
      break;
    case '+':                                                      //con el mensaje "1" encendemos el vibrador.
      calcularDistancia();                                         //toma una medición de distancia con el Ultrasonido.
      if (distancia > 1 && distancia < cotaSuperior) {             //si hay una distancia menor a 50 centimetros vibra fuerte.
        intensidad = map(round((distancia * 255) / cotaSuperior), 1 , 255, 255, 1);

      }
      else  {                                                      //si no apaga el vibrador.
        intensidad = 0;
      }
      vibrar(intensidad);
      break;
    case '/':                                                      //con el mensaje "1" encendemos el vibrador.
      orden_actual = '+';
      vibrarLente(command.substring(1, command.length() - 1).toInt());
      digitalWrite(led, LOW);      //apagamos led del pin 13 como testigo.
      break;
  } //FIN SWITCH
}
```

Figura 31: Función EjecutarSentencia.

```
void cambiarAMaestro() {
  Serial.println("Entrando configuracion Maestro");
  digitalWrite (powerBT, LOW);
  delay(1000);
  digitalWrite (enBT, LOW);
  delay(1000);
  configurarPuertoSerieMaestro();
  digitalWrite (powerBT, HIGH);
  delay(1000);
  digitalWrite(enBT , HIGH);
  delay(1000);
  digitalWrite (powerBT, LOW);
  delay(1000);
  digitalWrite (powerBT, HIGH);
  delay(1000);
  BT.println("AT");
  resultado1 = obtenerRespuesta();
  BT.println("AT+ROLE");
  delay(1000);
  resultado1 = obtenerRespuesta();
  Serial.println(resultado1);
  if (resultado1.equals("+ROLE:0\r")) {
    Serial.println("Es esclavo, cambiando a maestro");
    BT.println("AT+ROLE=1");
    //delay(2000);
    resultado1 = obtenerRespuesta();
    Serial.println(resultado1);
    if (resultado1.equals("OK\r")) {
      Serial.println("Es maestro");
      BT.println("AT+CMODE=0");
      //delay(2000);
      resultado1 = obtenerRespuesta();
      if (resultado1.equals("OK\r")) {
        Serial.println("Se conectará a un mac específica");
       // BT.println("AT+BIND=13,EF,7C4E");
        BT.println("AT+BIND=98D3,31,F73F70");
        //delay(2000);
        resultado1 = obtenerRespuesta();
        if (resultado1.equals("OK\r")) {
          Serial.println("Se conectará a la mac 98D3:31:F73F70");
          //Serial.println("Se conectará a la mac 13:EF:7C4E");
          BT.println("AT+BIND");
          //delay(2000);



          //delay(2000);
          resultado1 = obtenerRespuesta();
          resultado2 = obtenerRespuesta();
          Serial.println(resultado1);
          Serial.println(resultado2);
          Serial.println("Comando AT+UART");
          BT.println("AT+UART");
          resultado1 = obtenerRespuesta();
          Serial.println(resultado1);
          resultado2 = obtenerRespuesta();
          Serial.println(resultado2);
          Serial.println("Cambiando channel de comunicación al 9600");
          digitalWrite(enBT, LOW);
          delay(1000);
          digitalWrite(powerBT , LOW);
          delay(2000);
          configurarPuertoSerieEsclavo();
          //BT.setTimeout (1000);
          delay(2000);
          digitalWrite(powerBT , HIGH);
          Serial.println("Puerto Serial 38400");
          flag = true;
          delay(5000);
        }
      }
    }
  }
}
```

Figura 32: Función CambiarAMaestro.

En esta versión final incluimos la conexión automática de los dispositivos bluetooth al ingresar a la aplicación, en la sección de “Camino Seguro”. Para que los dispositivos se conecten automáticamente replicamos el circuito que nos permite ingresar en configuración AT y cambiar los nombres de los dispositivos denominándose “LENTE” y “BASTÓN”. Estos mismos aparecen cuando se encienden los dispositivos al ser suministrados de corriente. La actividad fue necesaria ya que el teléfono necesita utilizar los nombre para conectarse y realizar la asociación de los módulos con las funcionalidades de la aplicación sin tener que seleccionar manualmente qué dispositivo sirve para qué propósito. En las versiones anteriores las MACs de los dos dispositivos estaban hardcodeados en el código y el teléfono establecía la conexión mediante un botón presionado por el usuario.

Además, en esta versión se tuvo en cuenta lo conversado con los chicos no videntes y se cambiaron los botones para aumentar o disminuir la distancia de los sensores tanto del bastón como del lente, los botones fueron reemplazados por barras seekbar que son manejadas desde el botón de volumen del teléfono.

La vista resultante de esta nueva versión se muestra en la siguiente imagen:

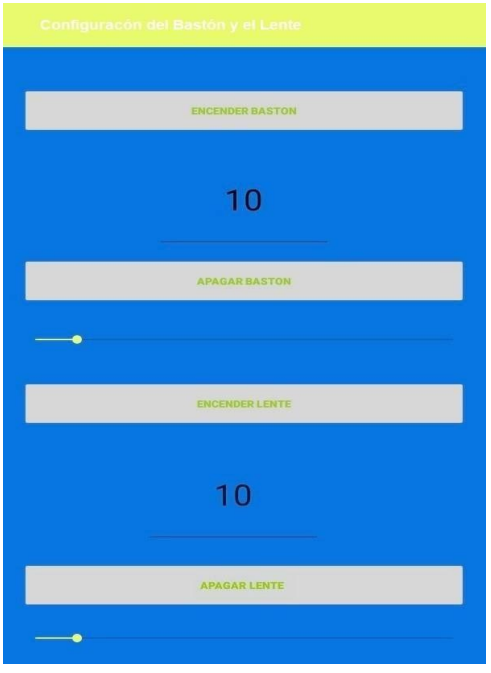

Figura 33: Prototipo Final conexión Bastón y Lente.

A simple vista se aprecian sólo cambios estéticos. Los botones de encender bastón y lente ya no tienen asociado un MAC fija, sino que estos se recuperan de la lista de dispositivos vinculados y se asocian a los que estén denominados como BASTÓN y LENTE. Además esta versión da soporte a la **Tercera Versión de Hardware** del bastón incorporando las funciones de checksum del lado del teléfono.

```
private Integer calculateCheckSum(String commando){
    int nameLenght = commando.length();
    Integer  result=0;
    for(int i = 0; i < nameLenght ; i++){   // while counting characters if less than the length add one
        char character = commando.charAt(i); // start on the first character
        result = result + (int) character; //convert the first character

    }
    return result;
}
```

Figura 34: Función CalculateCheckSum.

5.2 Diseño y desarrollo del lente

5.2.1 Ensamblando el lente - Primera prototipo Hardware

El desarrollo del lente, así como el desarrollo del bastón pasó por varias etapas. Primeramente se realizó el relevamiento de información al igual que con el bastón y determinamos la funcionalidad a implementar teniendo en cuenta la comodidad en el uso para el individuo. Replicamos la idea original del bastón como prototipo para el lente.

Se procedió al ensamblaje del primer prototipo con los siguientes componentes:

- Placa de desarrollo Arduino Nano 3.0 Atmel Atmega328p.

- Módulo Bluetooth Hc05 Maestro Esclavo.
- Sensor Ultrasonido Hc-sr04
- Protoboard
- Luz led verde
- Cables macho para protoboard

A continuación se expone la interconexión entre los dispositivos citados.

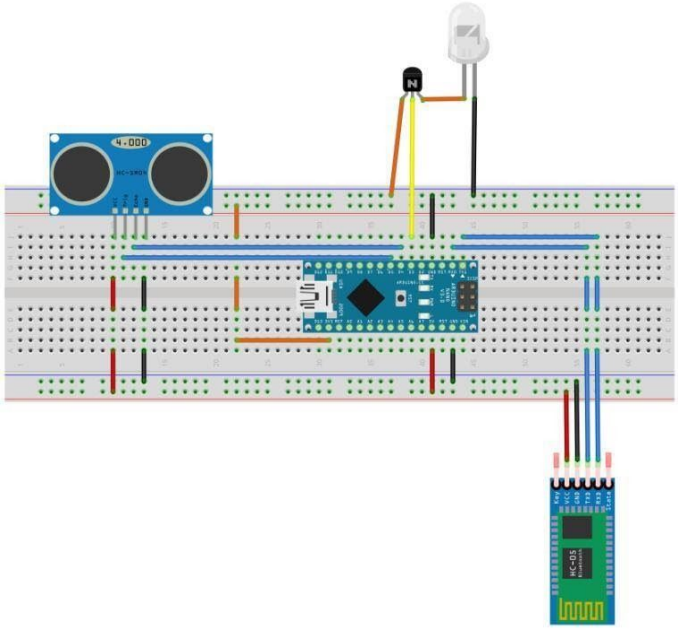

Figura 35: Diagrama que expone los elementos citados.

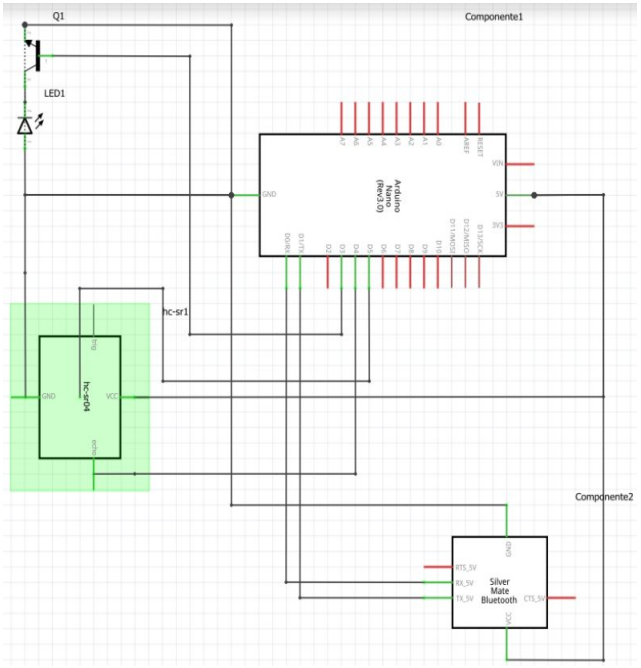

Figura 36: Diagrama que muestra el circuito electrónico.

Se descargó el mismo firmware en el módulo de lente, si bien el dispositivo funcionaba correctamente el modo de operación era cuestionable. Replicamos el mismo comportamiento en el prototipo del lente y se realizaron pruebas con las siguientes conclusiones preliminares. El prototipo del lente necesitaba de un tratamiento distinto al bastón, como el lente se encuentra en una zona de mayor sensibilidad, las componentes de ensamblaje debían ser más compactas, livianas y de mayor estética. Estas características fueron de gran importancia a la hora de armar el producto final.

En la imagen de la Figura 37 se puede observar que el lente tiene las mismas prestaciones que la versión inicial del hardware del bastón salvo el Arduino UNO Mega que en la codificación no presenta cambio alguno con respecto al Arduino Nano, es así que la imagen del circuito aplica correctamente a esta descripción.

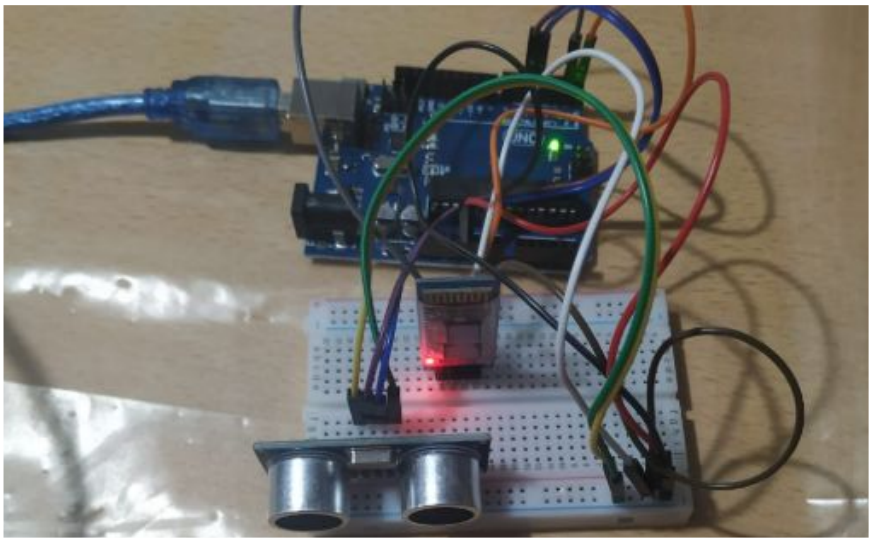

Figura 37: Primer prototipo de lente.

Éste carece de motor vibratorio, una funcionalidad cuestionada por nuestros encuestados a la hora de realizar el relevamiento.

5.2.2 Desarrollando la estética del lente - Segundo prototipo Hardware

Para el segundo prototipo utilizar los siguientes componentes:

- Placa de desarrollo Arduino Nano 3.0 Atmel Atmega328p.
- Módulo Bluetooth Hc05 Maestro Esclavo.
- Sensor Ultrasonido Hc-sr04
- Batería portátil Power Bank Externo 2000mah
- Lentes de sol negros.

La ingeniería de este prototipo minimalista fue desarrollada por el Ingeniero Emiliano Albarracín, el cual realizó un trabajo de reingeniería de las componentes mencionadas para presentarlas de una manera más estética y cómoda. El segundo prototipo no requirió agregado alguno sobre el código Arduino.

En las siguientes imágenes se puede observar el prototipo integrado.

Figura 38: Prototipo del lente con todos los componentes mencionados, vista de atrás.

Figura 39: Prototipo del lente con todos los componentes mencionados, vista frontal.

5.2.3 Adaptando software del bastón al lente - Primer Prototipo Software

Se basa en la modificación del código Arduino. Como el hardware del lente carece de un mecanismo de alerta perceptible para las personas con discapacidad visual (el led de notificación que utilizamos para realizar el sensado es inviable para el uso real), se requirió su modificación para enviar las señales de sensado al bastón y éste pueda notificar por medio de vibraciones. Para realizar este algoritmo estudiamos diferentes propuestas

de conectividad. La más rápida de abordar fue realizar una conexión múltiple entre el lente, el bastón y el teléfono.

Lo que se requería es que tanto el bastón como el lente puedan notificar al usuario sobre algún obstáculo en su trayecto. El bastón tiene incorporado un motor vibratorio para realizar esta funcionalidad, en el módulo del lente fue omitido porque los encuestados determinaron que el motor vibratorio en la cabeza les resultaría molesto.

El software del lente en esta versión es muy parecido a la versión inicial del bastón con la salvedad de que en vez de vibrar con la intensidad acorde a la medición realizada, envía dicha orden de vibración al puerto serie haciendo que el bastón la reciba y vibre de una manera determinada.

A continuación se expone el fragmento de código que realiza dicha actividad en el lente:

```
#include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
SoftwareSerial BT(6,7);    // Definimos los pines RX y TX del Arduino conectados al Bluetooth


int led = 13;        //Conexión del led testigo (viene incorporado en la placa arduino).
int eco = 4;         //Conexión del pin receptor del Ultrasonido.
int trig  = 5;        //Conexión del pin emisor del Ultrasonido.
char orden;          //lo usamos para almacenar el dato del bluetooth.
int distancia;       //lo usamos para almacenar el valor de la medición tomada.
char aux="";

//--------------------------------------------------------------------

void setup() {
  BT.begin(9600);              // inicializamos el puerto serie BT que hemos creado
  Serial.begin(9600);          // se utiliza para hacer debugg.
  pinMode (led, OUTPUT);       //activamos el pin 13 como salida, vamos a usar el led que ya trae el arduino como testigo.
  pinMode(trig, OUTPUT);       //configuramos el pin del Trig como salida.
  pinMode(eco, INPUT);         //configuramos el pin del Edo como entrada.
  digitalWrite(led, LOW);      //configuramos que el led del pin 13 inicie apagado.

}

//--------------------------------------------------------------------

void apagar(){
  digitalWrite(led, LOW);      //apagamos led del pin 13.
}

//--------------------------------------------------------------------

void vibrar( int aux){
  if (aux>0){
    digitalWrite(led, HIGH);      //encendemos led del pin 13 como testigo.
    String result=String("/")+String(aux)+".";
    BT.print(result);
    delay(200);                   //dormidmos 200 milisegundos para reenviar una posible señal de detección
  }else{
    digitalWrite(led, LOW);       //apagamos led del pin 13 como testigo.
  }

}
```

Figura 40: Fragmento de código que ejecuta la vibración del bastón.

```
}
switch (orden_actual) {                                    //segun la orden del bluetooth sera la funcion que se ejecuta
  case '-':                                                //con el mensaje "0" apagamos el vibrador.
     apagar();                                             //ejecutamos la funcion para "apagar()".
     while (Serial.available()){
       aux = Serial.read();
     }
     break;
  case '+':                                          //con el mensaje "1" encendemos el vibrador.
    calcularDistancia();                             //toma una medición de distancia con el Ultrasonido.
    if (distancia > 1 && distancia < cotaSuperior) {          //si hay una distancia menor a 50 centimetros vibra fuerte.
      intensidad = map(round((distancia * 255) / cotaSuperior), 1 , 255, 255, 1);
    }
    else  {                                          //si no apaga el vibrador.
      intensidad = 0;
    }
    vibrar(intensidad);
    break;
  case '/':                                          //con el mensaje "1" encendemos el vibrador.
    orden_actual = '+';
    vibrarLente(atol(numStr));
    digitalWrite(led, LOW);      //apagamos led del pin 13 como testigo.
    break;
} //FIN SWITCH

}    //FIN LOOP
```

Figura 41: Fragmento de código que ejecuta las condiciones del lente.

```
void vibrar( int aux) {                //escribimos el comportamiento de vibración como una función aparte para futuras modificaciones.
  analogWrite(vibrador, aux);     //el vibrador está conectado a 5v, con lo que 0PWM es equivalente a 0v y 255PWM es equivalente a 5v (vibra mas suave o mas fuerte).
  if (aux > 0) {
    digitalWrite(led, HIGH);      //encendemos led del pin 13 como testigo.
  } else {
    digitalWrite(led, LOW);      //encendemos led del pin 13 como testigo.
  }

}

void vibrarLente( int aux) {
  int starttime = millis();
  int endtime = starttime;
  int loopcount = 0;
  while ((endtime - starttime) <= 200) // do this loop for up to 1000mS
  {
    loopcount = loopcount + 1;
    endtime = millis();
    analogWrite(vibrador, aux);
    if (aux > 0) {
      digitalWrite(led, HIGH);      //encendemos led del pin 13 como testigo.
    } else {
      digitalWrite(led, LOW);      //encendemos led del pin 13 como testigo.
    }
  }
}

//--------------------------------------------------------------------
```

Figura 42: Función vibrarLente..

5.2.4 Permitiendo conexión directa bastón y lente - Segundo Prototipo Software

Se estudió la posibilidad de utilizar múltiples dispositivos bluetooth para realizar una conexión directa. Esto lo abordamos técnicamente en la tercer versión del hardware del bastón del cual se rescata la incorporación del transistor NPN para la configuración automática del bluetooth por comandos AT.

El algoritmo presenta una variante con respecto al utilizado en el bastón más que nada en la función de vibración, en vez de realizar la señal

analógica al pin 3 escribe en el puerto serial un código “3”, dicho código es captado por el teléfono y retransmitido al módulo del bastón con una orden inmediata de vibración haciendo que éste active el motor vibratorio a su máxima potencia.

En la siguiente imágen se presenta la función de vibrado:

```
//--------------------------------------------------------------------

void vibrar( int aux){
  if (aux>0){
    digitalWrite(led, HIGH);      //encendemos led del pin 13 como testigo.
    Serial.print("3");
    delay(200);                   //dormidmos 200 milisegundos para reenviar una posible señal de detección
  }else{
    digitalWrite(led, LOW);       //apagamos led del pin 13 como testigo.
  }

}

//--------------------------------------------------------------------
```

Figura 43: Función vibrar.

Desarrollamos una aplicación Android adicional para realizar pruebas del prototipo de conexión múltiple y verificar la conectividad. La aplicación se gestiona mediante botones y permite activar y desactivar el módulo que se desee, posee un botón que gestiona la configuración de los dos parámetros de medición presentes en cada prototipo.

A continuación se presenta el diseño del menú de configuración del bastón desarrollado en Android Studio:

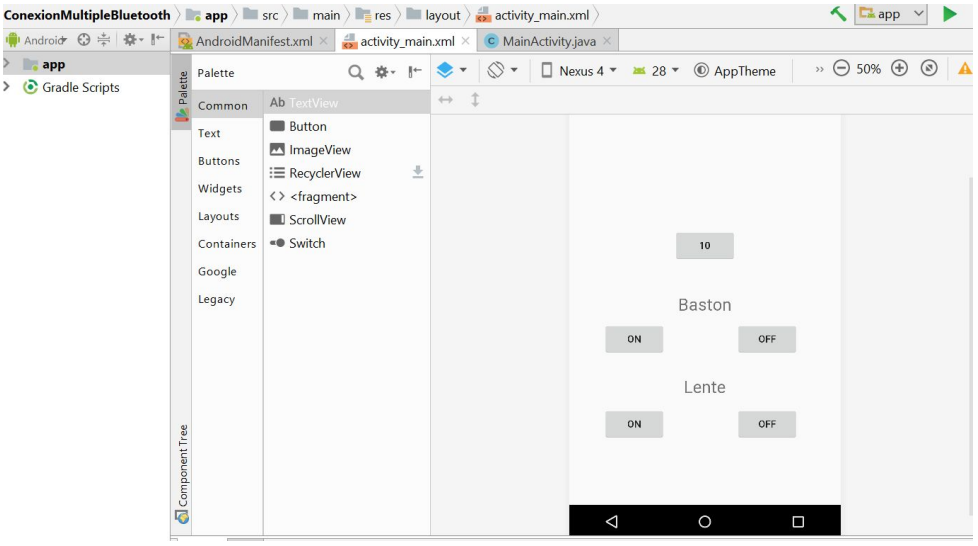

Figura 44: Segundo prototipo Software.

5.2.5 Chequeo de conectividad para manejar fallos - Tercer Prototipo Software

En esta versión se dió soporte a la **Tercera versión de Hardware** del bastón junto al software del teléfono. La única diferencia con el anterior código es la incorporación de la función checkSum utilizada para validar los datos de entrada provenientes del teléfono o del bastón más no así los datos de salida. Como los datos enviados por este módulo se deben interpretar en tiempo real, la retransmisión en directo (*streaming)* haría que el bastón vibre cuando el obstáculo ya no se encuentre en dicho rango de medición si este obstáculo pasa por el rango sin detenerse.

5.3 Diseño y desarrollo del software de recorridos

En paralelo a la configuración y desarrollo del kit compuesto por el hardware y la configuración del mismo se realizó una funcionalidad con la capacidad de consultar y gestionar los recorridos de los ómnibus de la ciudad de La Plata. Este desarrollo constó de varias etapas.

Se realizaron análisis sobre las posibles fuentes de información y/o APIs para obtener y procesar dicha información. Por disposiciones legales, en algunas provincias de Argentina se pueden obtener los recorridos de los transportes públicos consultando en la aplicación de Google Maps. Dicha aplicación posee una API para consultar los recorridos de los transportes públicos, junto con otras que permiten realizar trazabilidad, gestión de mapas, funciones de GPS en tiempo real, entre otros. En la ciudad de La Plata, esta funcionalidad de recorridos de transportes no se encuentra disponible, pero la Municipalidad nos proveyó un servicio web para obtener dicha información. El servicio web se consume bajo el protocolo de transporte HTTP por medio del intercambio de mensajes SOAP. Este servicio posee un conjunto de operaciones que fueron estudiados en detalle para realizar la trazabilidad del recorrido de los transportes. En total posee 20 operaciones.

Para realizar pruebas y consultar la información utilizamos la herramienta SoapUI, que sirve como cliente para enviar peticiones al servicio web de la Municipalidad. En el proceso de pruebas y obtención de información se observa que la información no está completa y tampoco está ordenada. Se intentó realizar un tratamiento de información sobre los datos obtenidos para ajustarlos y poder utilizarlos al momento de la trazabilidad, puesto que la información expuesta por medio de operaciones dentro de los servicio están muy atomizadas se requiere una interacción constante entre los servicios.

Las operaciones utilizadas fueron las siguientes:

- RecuperarLineaPorCuandoLlega: Por medio de esta iteración pasándole un código de área permite recuperar los ómnibus que se encuentran registrados en este servicio.
- RecuperarLineaPorLocalidad: Al igual que la operación anterior, permite obtener los ómnibus pero con más unidades.
- RecuperarParadasPorLineaParaCuandoLlega: Esta operación permite consultar por todas las paradas de una línea en particular, retornando el nombre, la posición geográfica y las calles principal y secundaria entre las que se encuentra.
- RecuperarProximosArribos2: Por medio de esta operación al pasarle un código de parada se puede obtener el identificador de la unidad próxima a arribar, el tiempo estimado de arribada y la ubicación de la unidad en tiempo real.

Las operaciones deben ser utilizadas en conjunto para ubicar una unidad y determinar la dirección y sentido de la misma, las paradas no poseen un código y ordenamiento que permita determinar cuál es la anterior y cuál es la siguiente por lo que se debe realizar un trabajo adicional para relacionar la información obtenida. Utilizamos el Android Studio para trabajar sobre los datos, la información se encuentra muy dispersa y la

cantidad de requerimientos a realizar para unificarla demanda poder computacional .

Se obtuvo la API del sistema de transporte de La Plata a través de la municipalidad de dicha ciudad junto con la documentación y manual de uso. Se realizaron un conjunto de pruebas para determinar el funcionamiento de la API y los métodos necesarios para obtener la posición del ómnibus para una parada determinada.

Detectamos una limitación en la API provista por la municipalidad. Las paradas carecían de sentido (ida/vuelta) y los números representativos e unívocos de paradas no tenían un ordenamiento. Por ende, se procedió a diseñar un algoritmo cuya función era consultar en simultáneo todas las paradas para determinar el recorrido realizado por el ómnibus más cercano. Se registró un resultado de 5 minutos de espera entre la solicitud del usuario y la respuesta del algoritmo con la ubicación del ómnibus, en el mejor de los casos. En este proceso se descubrió que el dispositivo utilizado para la experiencia no soportaba los tiempos de espera necesarios para determinar el recorrido y proveer una respuesta. Por este motivo se acotó a dos recorridos, desde la Facultad de Informática (calles 50 y 120) hasta el Dardo Rocha (calles 7 y 50).

El algoritmo utilizado fue diseñado e implementado en la plataforma Android Studio dentro de una aplicación para los teléfonos con Android Lollipop 5.1 en adelante. Dicha aplicación, en un principio, permitía la consulta de los ómnibus para las paradas más cercanas desde la posición actual (obtenida del dispositivo GPS del teléfono) por medio de una interfaz “estímulo respuesta” que respondía a comandos de voz utilizando la API SOAP provista por la Municipalidad de La Plata.

Luego del ensamblado, se instaló el firmware a la placa de desarrollo con el fin de que éste pueda controlar la medición sobre el sensor ultrasónico, prendiendo una luz led del propio Arduino cuando un objeto se encuentra dentro del rango de medición. El mismo firmware permite al Arduino configurar el dispositivo bluetooth en modo esclavo, para recabar información por medio de este.

La funcionalidad que se pretende desarrollar para este apartado es la de consultar desde un punto determinado la parada más cercana junto con el ómnibus que se debe tomar para llegar a un lugar determinado. La aplicación pretende guiar a los usuarios a dicha parada, indicando los desvíos y rutas a tomar para llegar al destino, la distancia en metros desde la ubicación actual del usuario hasta la parada destino donde debe realizar el abordaje. También permitirá informar mediante comandos de voz el tiempo estimado de llegada de la unidad a la parada de abordaje.

La imagen de la Figura 45 muestra el área geográfica sobre la cual se pretende realizar el trabajo, en ella se observan las paradas que comprenden el recorrido de ida desde 1 y 50 hasta 7 y 50. Estas paradas representan a la línea de ómnibus 214 obtenidas por medio del servicio web de la Municipalidad de La Plata.

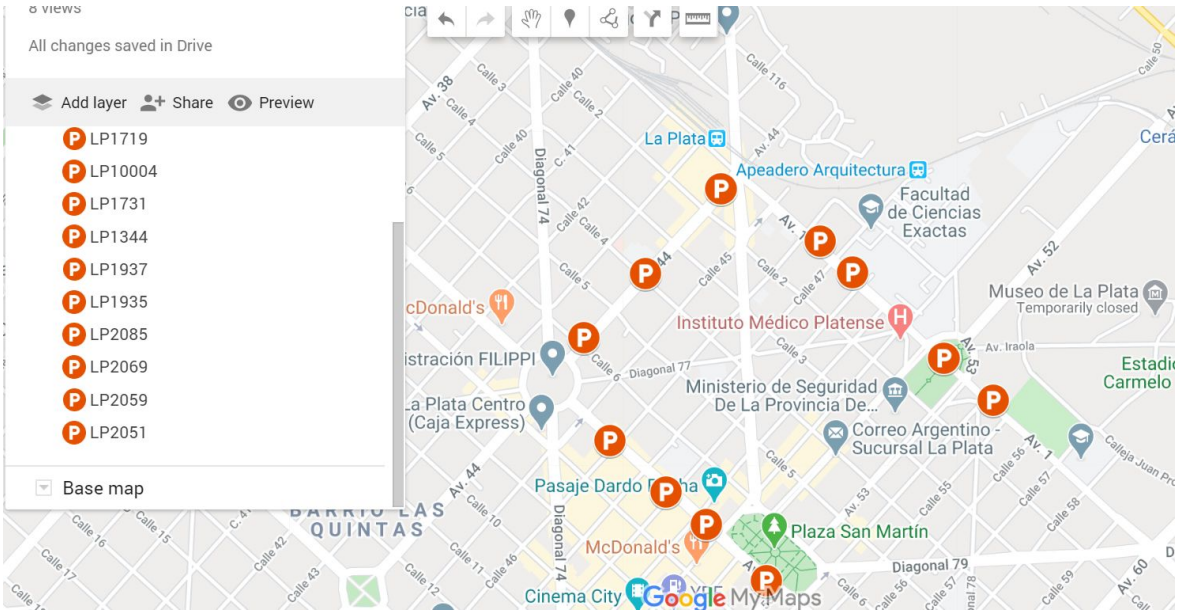

Figura 45: Área geográfica donde se realizó el prototipo, con las paradas correspondientes.

Consumiendo la operación RecuperarProximosArribos2 para una parada determinada podemos obtener la ubicación de la misma, los ómnibus próximos a arribar a dicha parada y los tiempos de arribo para cada ómnibus en la lista devuelta, además obtenemos la ubicación en tiempo real del ómnibus próximo a arribar. Con dicha información y

conociendo el orden de las paradas en que los ómnibus van arribando podemos ubicar todos los puntos en el mapa y realizar trazabilidad.

Las paradas obtenidas ubicando el próximo ómnibus en cada una de las demás paradas nos arrojó un resultado de posible ordenamiento y sentido de calles. Luego ordenando los datos de manera ascendente de acuerdo al tiempo de arribo obtenemos una lista de paradas posiblemente ordenadas, con el resultado de dicha operación ubicamos los puntos en GoogleMaps y se observa que para algunos tramos el algoritmo de ordenamiento aplicado era correcto, pero el mismo algoritmo aplicado a paradas más cercanas entre sí con recorridos circulares fallaba. Además de presentar inconsistencias en los tramos, el algoritmo requería más capacidad de cómputo y networking que un hilo de ejecución del sistema operativo Android pudiera ofrecer.

Por la falta de orden en las paradas no se puede realizar cálculos de distancia simples y determinar cuál es la parada más cercana al que el usuario debe ir para llegar al destino final. Los datos de las paradas sobre la cual trabajamos fueron almacenados localmente en una lista ordenada con sentido realizada a mano, así recorriendo la lista podemos obtener las paradas destino, su geolocalización y con ello realizar el cálculo de distancia entre puntos para determinar si el camino que se está procesando actualmente es el correcto.

5.3.1 Proveedor de servicios Offline

Por problemas con el proveedor de servicios de los ómnibus de La Plata, indisponibilidad del servicio web que utilizamos para obtener las coordenadas de los ómnibus y las paradas y, por la baja calidad y precisión de los datos obtenidos procedimos a replicar el servicio web con algunas de las operaciones provistas por el proveedor original. En éste punto se realizan algunas aclaraciones:

Para montar una réplica del servicio utilizamos una raspberry pi b3+. La raspberry es un ordenador de placa reducida, placa única o simple (SDB) de bajo costo desarrollado en el Reino Unido por Raspberry PI Foundation con el objetivo de estimular la enseñanza informática en las escuelas que luego fue extendiéndose a diferentes áreas, no incluye periféricos o carcasa [22]. Dicho ordenador puede alojar diferentes distribuciones de sistemas operativos de código abierto. Para este caso en particular nos inclinamos por la versión oficial derivada de Debian conocida como Raspbian.

Como tecnología de almacenamiento utilizamos MySQL, un sistema de gestión de base de datos relacional de código abierto bajo los términos de la Licencia Pública General GNU, también se distribuye bajo otras licencias patentadas. Este producto es un componente de la pila de software de aplicación web LAMP, acrónimo de Linux, Apache. MySQL, Perl/PHP/Python. Dicha pila de software fue instalada en la Raspberry.

Usamos PHP, un popular lenguaje de scripting de propósito general muy adecuado para el desarrollo web, generalmente procesado en un servidor web por un intérprete PHP implementado como módulo, DAEMON o como un ejecutable de CGI. El servidor de aplicaciones utilizado es Apache, que al igual que PHP forma parte del empaquetado de aplicaciones provisto por LAMP.

Se construyeron servicios de prueba con la librería NuSoap, son un conjunto de clases PHP que permiten crear y consumir servicios web basados en SOAP 1.1, WSDL 1.1 y HTTP 1.0 / 1.1. En las pruebas del desarrollo de los servicios, la librería respondía favorablemente a requerimientos simples, se debía construir el cliente y el servidor como funcionalidades diferentes dentro de sus respectivas clases, todo orientado a objetos y bastante ordenado, el inconveniente que tiene esta librería es que todo el contrato (firma WSDL) debe ser construido mediante clases y métodos, esto incluye las operaciones, los tipos de datos simples y compuestos, mensajes entre otros haciendo muy tedioso el trabajo. No respondía favorablemente en la construcción de las respuestas a los

servicios que tenían elementos compuestos como listas, o tipo de datos complejos (complexType) definidos en la estructura de respuesta no pudiendo representarlos correctamente, esto hace que la validación de los mensajes en la respuesta fallara, normalmente el testeo de los mensajes SOAP de entrada como de salida forman parte de las buenas prácticas al utilizar este protocolo de mensajería.

Realizando un poco más de investigaciones encontramos las clases SoapClient y SoapServer incluidas como librerías en las versiones de PHP 5 y 7. Con dichas clases podemos desarrollar servicios y consumirlos por medio del protocolo de mensajería SOAP. El agregado de estas clases con respecto al anterior es la utilización de un template como base, es decir los servicios se construyen alrededor de un contrato WSDL, con elementos, mensajes y operaciones los cuales representarán la interfaz de entrada/salida para el servicio. La clase que realizará el Servidor utiliza como base el documento WSDL para ejecutar las operaciones definidas en ella, recuperar los datos de entrada y responder, es aquí donde se pueden presentar inconvenientes, pues uno puede colocar en la respuesta algún elemento que no corresponda con el definido en el contrato haciendo que el testeo del mensaje de respuesta pueda fallar. Otra gran diferencia es la simplicidad en el manejo de los elementos compuestos a la hora de armar los mensajes de respuesta. Por estas características los servicios con las operaciones necesarias para simular la API de la Municipalidad fue desarrollada con estas librerías.

Una vez aclaradas todas las componentes necesarias para simular el servicio web procedemos con la explicación de la construcción del servicio.

Se agrega una política de seguridad al servicio, en la metodología del proveedor anterior utilizaban un elemento *custom* de seguridad en texto plano, para la versión desarrollada utilizamos el tipo de autenticación básico *WSS Username Token* (estándar). Además el servicio implementa un registro en base de datos donde almacena todos los orígenes de peticiones junto con la cantidad de accesos exitosos y con fallos por credenciales, una verificación de seguridad que bloquea las peticiones

después 3 intentos durante el transcurso de 2 minutos entre peticiones. En la siguiente imagen se muestra el esquema de base de datos utilizado.

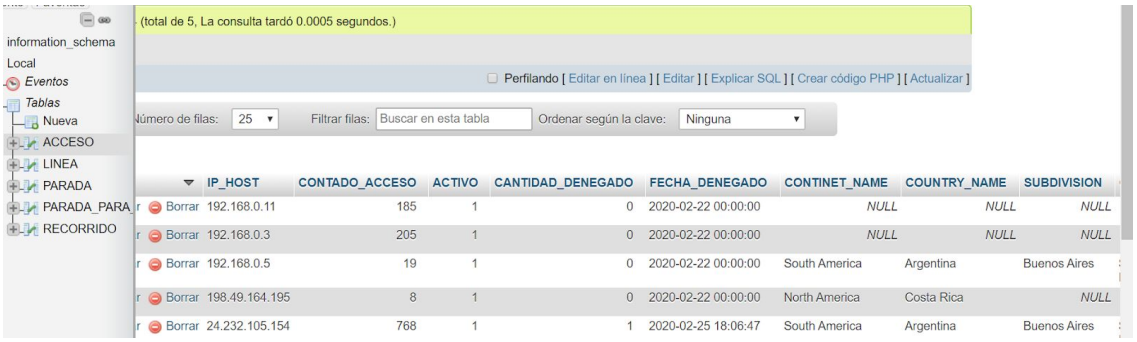

Figura 46: Base de datos MySQL utilizada para probar el prototipo.

A continuación se pueden observar las clases implementadas en PHP para conexión con el esquema de base de datos.

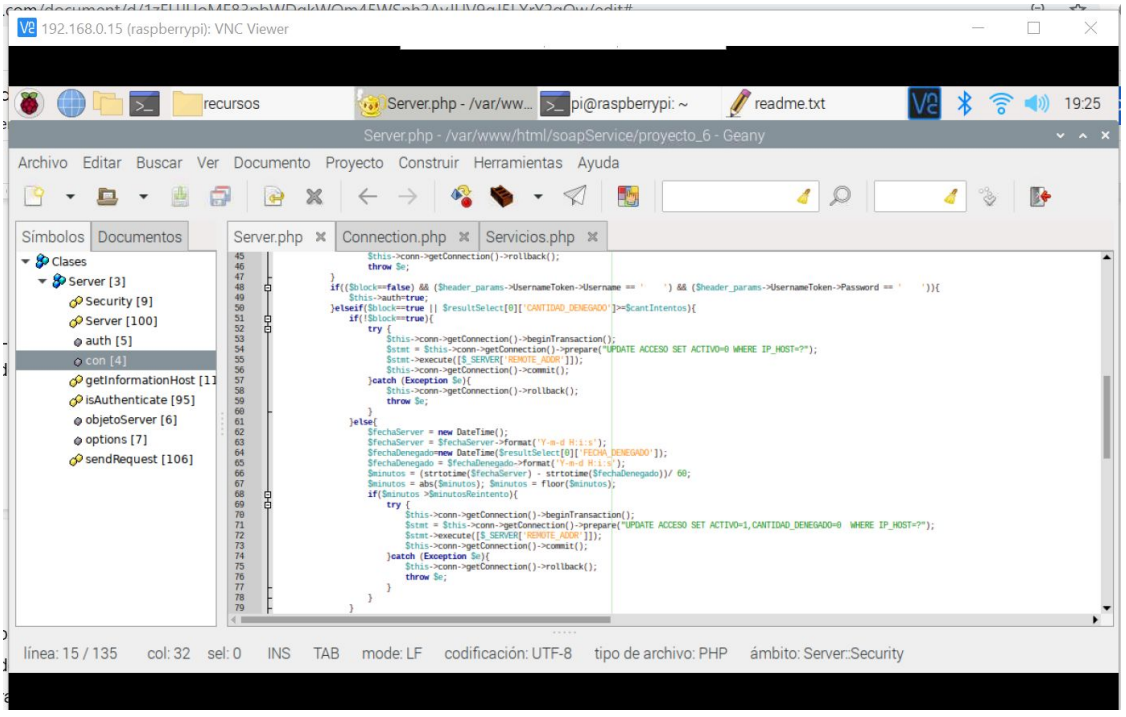

Figura 47: Conexión con la base de datos servidor Apache y BD MySQL.

La ubicación en tiempo real de los ómnibus es simulada por medio de un conjunto de puntos y tratamiento de los mismos por un proceso (*schedule*). Para ello se procede a la incorporación de las coordenadas de los recorridos en un tabla de base de datos con una columna que identifica la posición actual del ómnibus. En la imagen se puede observar la tabla de recorridos.

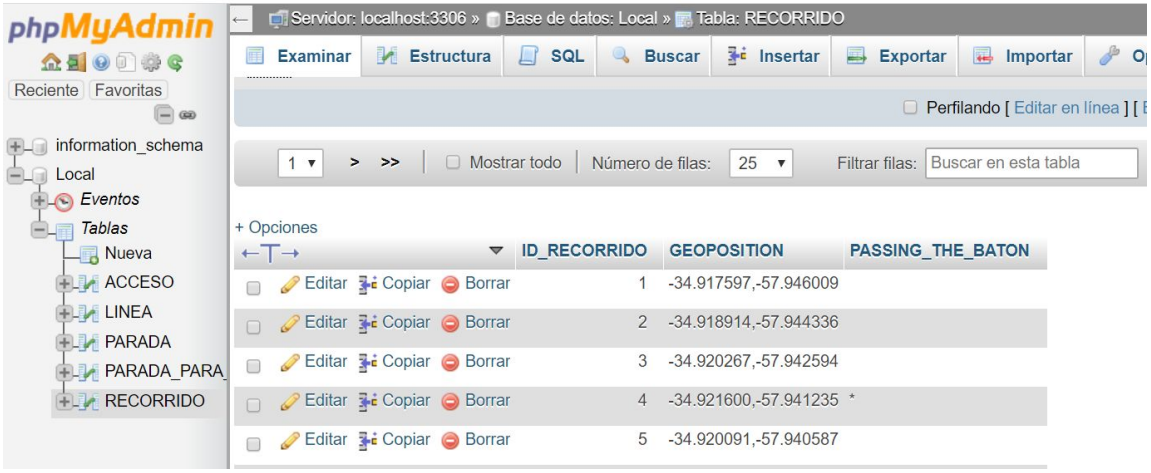

Figura 48: Columna que simula la geolocalización en tiempo real del ómnibus en MySQL.

Para simular el desplazamiento del ómnibus utilizamos la marca dentro de la columna PASSIG_THE_BATON junto con un evento programado sobre esta tabla que corre cada 1 minuto pasando la marca “*” a la fila siguiente. En la imagen se muestra el evento programado en la base de datos MySQL.

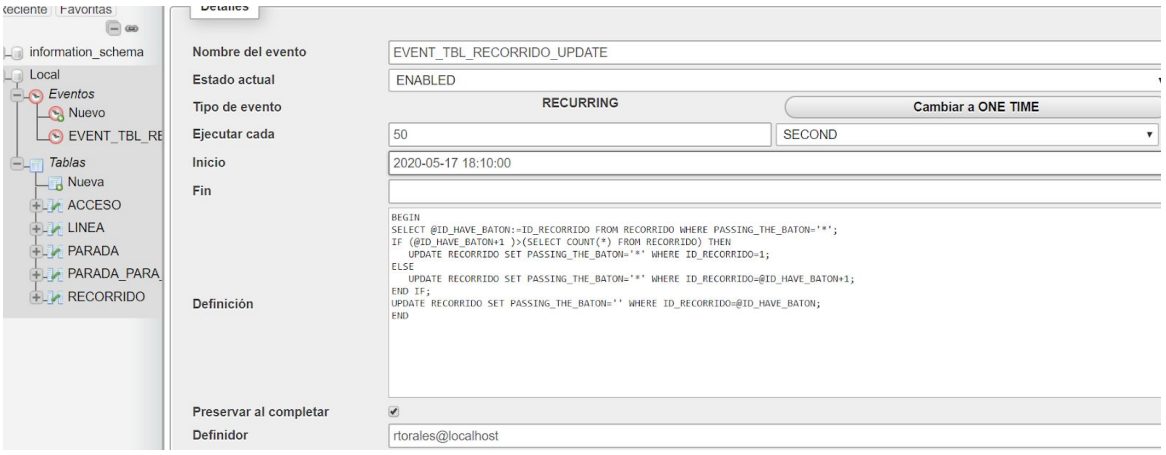

Figura 49: Evento que simula el desplazamiento del micro en MySQL

La obtención del mensaje de “arribando” en el servicio se obtiene utilizando la API REST de GoogleMaps para determinar la cercanía del ómnibus a la parada de abordaje. Se puede observar en la imágen la codificación de la operación RecuperarTiempoDeEspera.

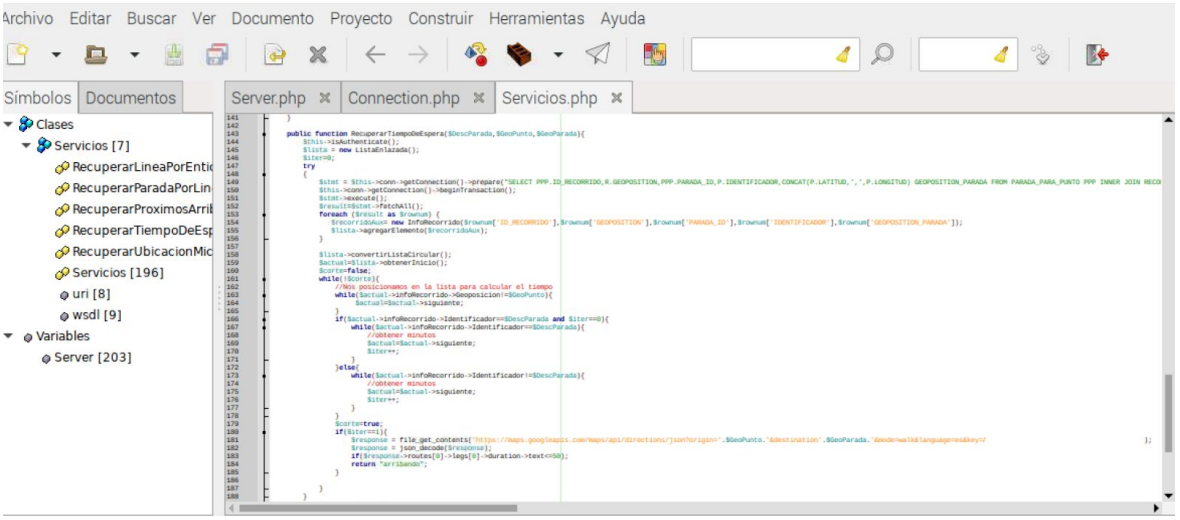

Figura 50: Obtención del mensaje “Arribando” a través de la API de Google Map servidor Apache.

Finalmente en la imagen se muestra una fracción del código en la clase Servicio que sirve a la operación RecuperarPróximoArribo2 donde se recupera la información de la parada junto con la posición actual del ómnibus.

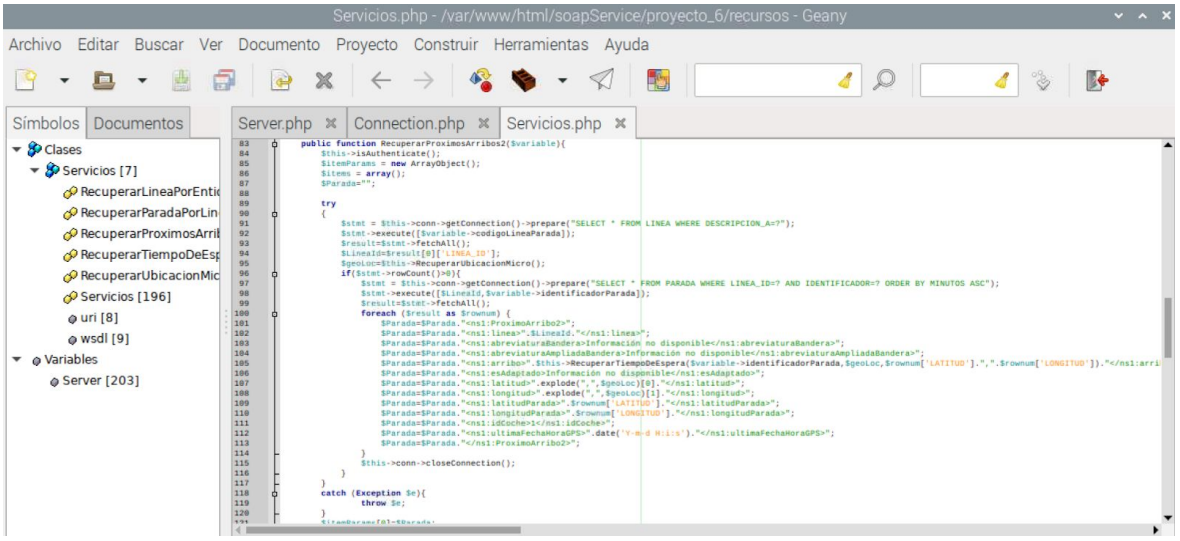

Figura 51: Función RecuperarProximoArribo2 en el servidor Apache.

Por último ejecutamos el servicio por SoapUi para obtener la información de los ómnibus como se muestra en la imagen.

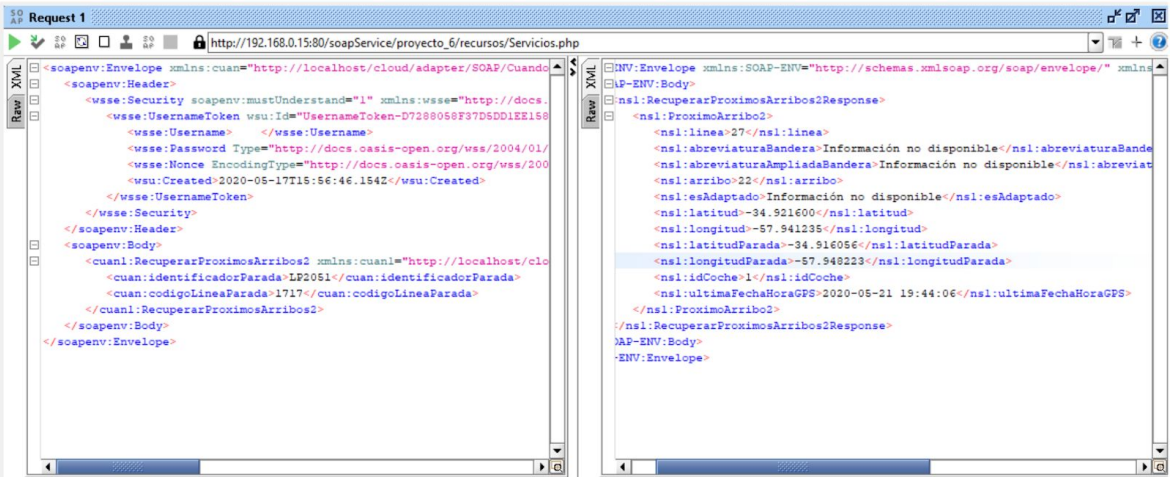

Figura 52: Invocación al WebServiceCuandoLlegaPI opración RecuperarProximoArribo2.

Luego de realizar todas las aclaraciones pertinentes, el producto final que fue pasando por estas etapas de desarrollo, consta de un kit formado por un bastón, lente y una aplicación para Android. Se puede apreciar que el bastón y el lente pueden ser configurados desde el teléfono por medio de una aplicación la cual permite además, consultar servicios web posicionando ómnibus y recorridos para la ciudad de La Plata. También se puede apreciar que las dos funcionalidades, recorrido y configuración del kit funcionan de manera aislada, pueden ejecutarse por separado en caso de ser necesario (ej: al presentarse algún tipo de falla de conectividad por el servicio offline).

5.4 Diseño final del software

Finalmente, luego de varias versiones del software, se llegó a la versión final. Teniendo en cuenta las distintas recomendaciones brindadas por los usuarios ciegos de la Facultad de Informática de la UNLP.

A continuación se pueden observar las distintas pantallas de la aplicación móvil:

Figura 53: Pantallas que forman parte de la aplicación mobile.

La primera imagen hace referencia a la aplicación, es la imagen del ícono del aplicativo que aparece al momento de realizarse la descarga al móvil.

La segunda imagen pertenece a las dos funcionalidades provistas por la aplicación cada una de ellas denominada como “BONDIS A CASA” y “CAMINO SEGURO”.

La tercera imagen corresponde a la configuración del bastón y del lente. Luego de elegir la opción “CAMINO SEGURO”. Por medio de botones en la pantalla se pueden encender y apagar tanto el bastón como el lente además de configurar la distancia de los sensores. La configuración de distancias de cada módulo puede realizarse teniendo el talkback activo apretando los botones de volumen (+) para aumentar la distancia de medición o volumen (-) para disminuirlas.

Y por último, la cuarta imagen corresponde al recorrido de los ómnibus. Luego de elegir la opción “BONDIS A CASA”. En esta pantalla se puede consultar los ómnibus más cercanos desde donde está el usuario (detectando su posicionamiento por medio del GPS del celular o por medio de internet) hacia dónde quiere ir. En nuestro caso y por motivos de complicaciones con el Servicio Web brindado por la Municipalidad de La

Plata, marcamos un recorrido fijo desde la Facultad de Informática (calles 50 y 120) hasta el Dardo Rocha (calles 7 y 50).

6

6. Prueba de campo

Para obtener una impresión del impacto que generaba el kit en usuarios ciegos y el nivel de accesibilidad que tenía la aplicación móvil, se hicieron algunas pruebas.

6.1 Pruebas del software

Durante el desarrollo de los prototipos, el software fue pasando por varios procesos de aceptación. Se realizaron pruebas de conectividad de las componentes internas de manera aislada para corroborar el funcionamiento, posterior a las pruebas de ensamblado (puesta en marcha de las componentes y sus interconexiones), empezamos a realizar pruebas unitarias de los módulos, bastón y lente sometiéndolos a tests de detección de obstáculos con distintos objetos, ángulos de visión respecto al sensor ultrasónico, velocidad de detección de obstáculos. Dichas pruebas arrojaron resultados que fueron retroalimentando las versiones posteriores. Al principio, los test del software en el bastón Arduino arrojaron interferencias en sensados de obstáculos, una limitante entre la ubicación del sensor ultrasónico con respecto el módulo bluetooth (conectado de una manera determinada la placa de desarrollo Arduino) haciendo que la luz inicial que representaba al objeto dentro del rango de medición parpadeara en ocasiones de manera intermitente sin haber obstáculos. También había diferencias en la capacidad de alimentación entre un módulo de desarrollo Arduino y otro más compacto que tuvo un resultado inesperado. Casi todo el proceso de construcción del kit se realizó sobre la placa de desarrollo Arduino UNO (la placa convencional) pues dicha placa es resistente y soporta mucho mejor los fallos de conectividad. También se presentaron

problemas asociados al hardware para configurar los modos de operación de los dispositivos bluetooth. Dichos problemas se presentaron al migrar el kit de la versión del prototipo a la versión final haciendo necesario un re-test unitario y de integración de toda la funcionalidad para detectar posibles inconvenientes.

Las pruebas realizadas relacionadas con el recorrido de los ómnibus en principio nos arrojaron retrasos con respecto a la ubicación real. En algunos casos el tiempo de procesamiento excede los límites de ejecución de actividades en el dispositivo haciendo que la aplicación se detuviese. Dicha información como se explicó en los apartado anteriores tuvo que ser reemplazada porque además del inconveniente mencionado bajaron el servicio haciendo inutilizable la aplicación de recorrido. En este apartado se tiene en cuenta que el recorrido de ómnibus es una simulación de la API real por lo que su testeo no presentó mayores inconvenientes fuera de la misma construcción y puesta en marcha.

Una vez finalizadas las pruebas funcionales se procedió con las pruebas sobre el software asociadas al kit, testeando la aplicación móvil desarrollada con la aplicación “Test de Accesibilidad” descargada desde la PlayStore. Este test nos arrojó algunas observaciones en las diferentes pantallas.

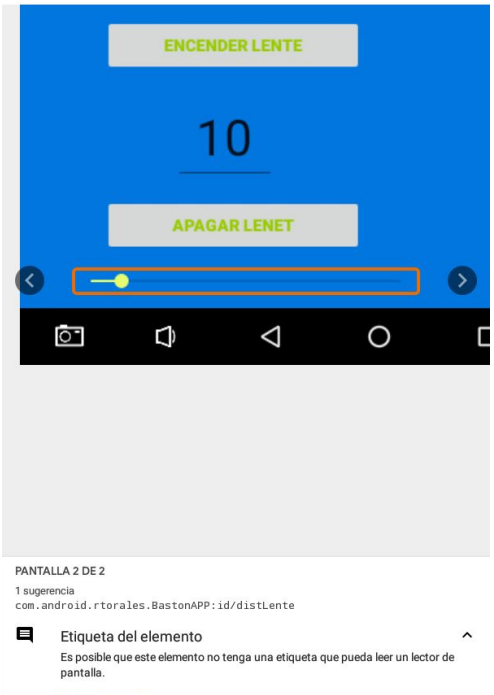

Figura 54: Test en la pantalla “Camino Seguro”

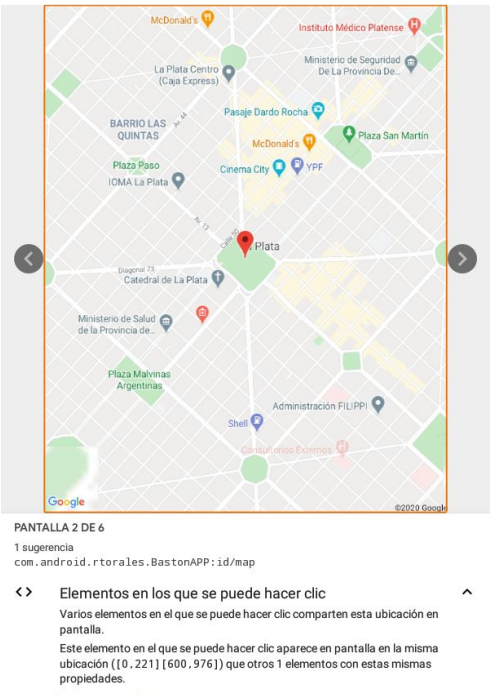

Figura 55: Test en la pantalla “Bondis a Casa”

Las imágenes anteriores nos muestran las sugerencias arrojadas por el test.

En la Figura 54 podemos observar que la seekbar no tiene una etiqueta, la cual ayudaría al lector de pantalla.

En la Figura 55 podemos observar que hay varios elementos en los que se puede hacer clic compartiendo ubicación en la pantalla. Esto puede resultar difícil al momento de que el usuario interactúe con la pantalla.

6.2 Pruebas del hardware

El desarrollo de los prototipos del hardware, al igual que el del software, pasó por varios procesos de evaluación.

Se analizaron con los estudiantes ciegos, cuestiones de ergonomía, aspectos de diseño, de peso, ubicación y formas de uso para evaluar su nivel de aceptación y satisfacción.

En el bastón se tuvo en cuenta la ubicación de los sensores ya que requiere cerrarse y abrirse. Además del diseño compacto de todos los componentes para que puedan ser puestos en un mango fácil de agarrar.

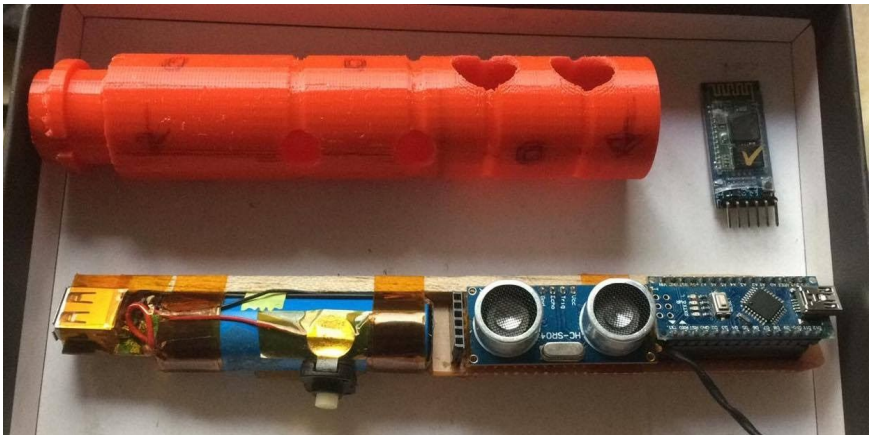

Figura 56: Prototipo del mango del bastón..

En cuanto al lente, se tuvo en cuenta que los sensores no queden de manera muy “vistosa” para que no llame la atención. Además que el peso de la batería sea el menor posible.

Figura 57: Prototipo del lente..

6.3 Prueba del kit

Durante la Expo de Ciencia y Tecnología que se realizó en la Facultad de Informática, pudimos probar el primer prototipo del kit con Tomás, uno de los usuarios ciegos que colaboró con nosotros.

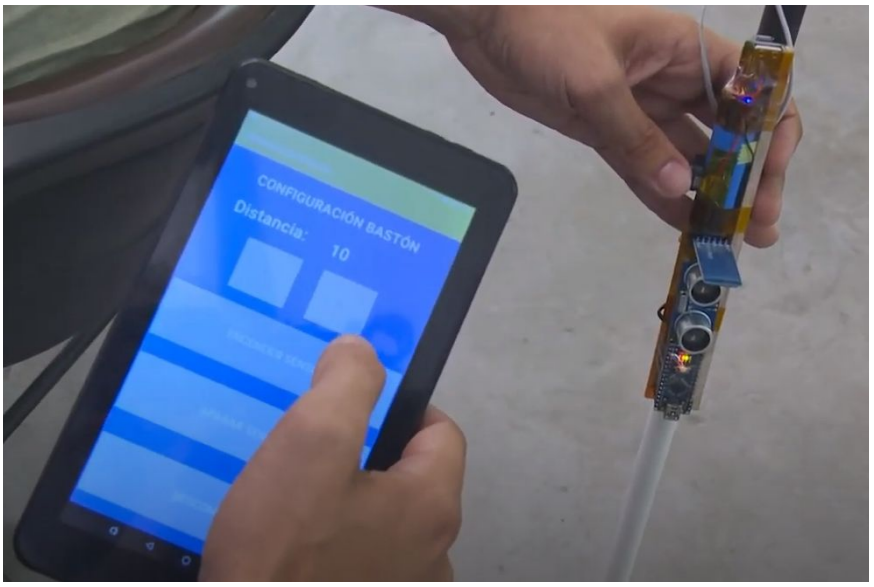

Figura 58: Primer prototipo del kit.

Figura 59: Tomás testeando el primer prototipo del kit.

En las imágenes anteriores, podemos observar el primer prototipo del kit y el momento en el que Tomás lo testeó. Pudimos ver un gran entusiasmo de su parte con respecto a la independencia que le puede dar este kit para su vida cotidiana.

Finalizado el testeo, Tomás nos comentó algunas mejoras que se podrían hacer para que el kit sea más fácil de utilizar.

En ese primer prototipo testeado, la distancia de los sensores se configuraba con botones, Tomás nos comentó que para ellos es más fácil si eso se controlaba con los botones del volumen. Estas sugerencias fueron tomadas y en la versión final de software se puede observar que que dicha característica fue implementada.

7

7. Conclusiones y Futuras líneas de investigación

7.1 Conclusiones

Este trabajo de tesina intenta encontrar una solución integral entre el hardware y el software que mejore la calidad de vida de las personas ciegas y disminuidas visuales para un tránsito autónomo e independiente. Fue desarrollado mediante la participación y opinión de personas ciegas como así también teniendo en cuenta otros antecedentes y desarrollos previos para lograr una propuesta mejoradora.

El desarrollo del kit fue construido en base a características existentes en otros prototipos desarrollados que fueron analizados. También combinamos funcionalidades adicionales de ubicación y gestión de movilidad encontradas en otras aplicaciones analizadas que no necesariamente fueron construidas para ser utilizadas por personas con discapacidad.

El análisis realizado al bastón desarrollado por los investigadores de la UMH (Universidad Miguel Hernández) fue de mucha utilidad. Las características de dicho bastón lo hacen muy fácil y práctico de usar. Sus principales características como la alerta por vibración y la detección de obstáculos suspendidos lo hacen interesante para el estudio, ya que en otros prototipos presentes, no se consideraban los obstáculos suspendidos, los cuales representan una problemática para las personas con disminución visual y ciegas. En contraposición al prototipo de UMH, las alertas

realizadas por los sensores ultrasónicos de otros prototipos estudiados, invaden el sentido de la audición, es decir, intentan subsanar la discapacidad de la vista pero al mismo tiempo limitan su sentido auditivo.

Sin dejar de lado el diseño, el bastón presentado no se veía muy diferente al bastón blanco tradicional, se mantiene pequeño y los dispositivos electrónicos se encuentran bien protegidos, el bastón es ergonómico en comparación a otras soluciones.

También se estudió una solución de lentes inteligentes para ciegos con el objetivo de detectar obstáculos suspendidos desarrollada en Nuestra Señora de las Mercedes de Bollullos, España, el dispositivo emite una alerta sonora al momento de detectar un obstáculo.

Estas funcionalidades fueron expuestas a los alumnos con discapacidad visual de la Facultad de Informática de la UNLP para determinar si eran adecuadas para el prototipo a desarrollar, con el agregado de una aplicación que sirva de nexo entre el bastón y los lentes. Producto de esta exposición se desarrolló un kit de movilidad capaz de detectar obstáculos de manera más precisa, pudiendo ser configurado desde el teléfono con una aplicación adaptada a las personas con discapacidad visual, dicha aplicación posibilita consultar las paradas de los ómnibus más cercanos para dirigirse a puntos comprendidos entre la Facultad de Informática y el Dardo Rocha.

Luego del ensamblaje y la integración de los tres dispositivos y funcionalidades que conforman el kit, se desarrollaron pruebas en la Facultad de Informática con los chicos no videntes las cuales resultaron ser satisfactorias. Las pruebas compendian la detección de obstáculos tanto debajo de la cintura como por encima de ella, recorriendo las instalaciones de la facultad. También se consultó por la disponibilidad de ómnibus para dirigirse al Dardo Rocha desde la Facultad de Informática mediante la asistencia que provee la aplicación (enrutamiento mediante GPS).

7.2 Futuras líneas de investigación

Si bien el kit de movilidad es bastante innovador e intentó mejorar/optimizar las características de prototipos desarrollados con anterioridad, se decidió acotar las funcionalidades para cerrar una idea y comenzar con el proceso de desarrollo. A lo largo del desarrollo de la tesina se encontraron inconvenientes funcionales y de infraestructura. Uno de estos inconvenientes consistía en la limitación presente al consumir el Servicio Web provisto por la Municipalidad de La Plata. La información obtenida no se encontraba ordenada, requería que los datos sean procesados previamente. Los celulares convencionales no soportan este tipo de procesamiento, les exige una mayor velocidad de cómputo que las que pueden ofrecer y un mayor tiempo de espera. Se requiere de una mayor velocidad de cómputo para armar los recorridos de las distintas líneas de transporte, almacenando los resultados obtenidos dentro de algún repositorio persistente y así poder consultarlo más rápido. Al ser un algoritmo bastante complejo y que requiere muchos recursos computacionales, la información debe poder ser actualizada cada cierto tiempo, de tal manera de mantener el sistema en línea el mayor tiempo posible, y tener actualizado los recorridos de los ómnibus, ya que éstos pueden ir variando con el tiempo. Esta actualización por ejemplo, puede ser desarrollada como un proceso programado en el tiempo, de tal manera que cada una semana (los domingos a la noche) el algoritmo puede ejecutarse nuevamente, y actualizar la información de los recorridos.

Teniendo en cuenta que el servicio de ómnibus proporcionado por la municipalidad de La Plata fue dado de baja, parte de la solución propuesta fue implementada para mostrar el funcionamiento del kit. Queda abierta la posibilidad de estudiar distintos proveedores de servicios y evaluar la sustitución del servicio implementado para la muestra.

Otra de las funcionalidades adicionales y que agregaría un valor importante, es la de incorporar algún módulo (cámaras en el bastón) con reconocimiento de imágenes, que detecte los tipos de objetos que se encuentran en el recorrido del usuario ciego. Este procesamiento al igual que el anterior, debe hacerse de manera centralizada, ya que las prestaciones de los celulares convencionales en la actualidad, no soportarían semejante carga y cálculo de información. Esta centralización requiere de un protocolo de comunicación adicional para manejar las solicitudes de los diferentes dispositivos conectados a la central, los cuales estarían enviando información constantemente para ser procesadas, y quedarían esperando una respuesta de la central para determinar si hay necesidad de levantar alertas dependiendo del objeto detectado.

Por último, se podría agregar algún tipo de aviso que alerte a los usuarios ciegos que las baterías tanto del bastón como del lente, están por acabarse.

8

8. Bibliografía

[1] *La OMS presenta el primer Informe mundial sobre la visión,* Ginebra, 8 de Octubre del 2019, https://www.who.int/es/news-room/detail/08-10-2019-who-launches-first-world-report-on-vision

[2] *Identificación de la población con discapacidad en la Argentina: aprendizajes y desafíos hacia la Ronda Censal 2020*, Buenos Aires, abril de 2019, https://www.indec.gob.ar/ftp/cuadros/publicaciones/discapacidad_ronda_censal_2020.pdf

[3] *Inclusión de personas con discapacidades, Centro para el Control y Prevención de Enfermedades, Departamento de Salud y Servicios Humanos*, GobiernoUSA.gov, 1 de Julio del 2020, https://www.cdc.gov/ncbddd/spanish/disabilityandhealth/disability-inclusion.html

[4] *El Bastón Blanco*, Escuela de Educación Especial Para Formación Laboral N°37 "Francisco Gatti", https://www.buenosaires.gob.ar/areas/educacion/escuelas/escuelas/especial/esp37de03/bastonblanco.htm

[5] [6] *Ley 26.858, Personas con discapacidad acompañadas por Perro Guía o de Asistencia.* Sancionada el 22 de Mayo de 2013, promulgada el 10 de Junio de 2013. Ministerio de Justicia y Derechos Humanos, Presidencia de la Nación Argentina

[7] *Recursos tecnológicos para personas con discapacidad*, Andalucía es Digital, 11 de Octubre del 2016, http://www.blog.andaluciaesdigital.es/recursos-tecnologicos-para-personas-con-discapacidad/#Ventajas_del_uso_de_los_recursos_tecnologicos_TIC_y_discapacidad

[8] *Google Maps*, Google Inc., 29 de Julio del 2020, https://google-maps.uptodown.com/android

[9] *Estado del Arte: Servicios Web*, Carlos Andrés Morales Machuca (Universidad Nacional de Colombia), 1 de Octubre del 2019, sites.google.com/site/camoralesma/articulo2.pdf

[10] *SoapUI: jugando con web services*, Iván García Puebla, 28 de Diciembre del 2019, https://www.adictosaltrabajo.com/2009/12/28/introduccion-soap-ui/

[11] *¿Qué es XML?,* Culturacion.com, https://culturacion.com/que-es-xml/

[12] *Application Programming Interface*, Wikipedia The Free Encyclopedia, 2 de Octubre del 2019, https://en.wikipedia.org/wiki/Application_programming_interface#cite_note-Braunstein2018-1

[13][15] *Discapacidad Visual*, Universidad de Palermo, 5 de Octubre del 2019. https://fido.palermo.edu/servicios_dyc/proyectograduacion/archivos/793.pdf

[14] *Sistema de Protección Integral de los Discapacitados Ley N. 24.314*, InfoLEG (Información Legislativa) Ministerio de Justicia y Derechos Humanos. Presidencia de la Nación, 8 de Abril de 1994, http://servicios.infoleg.gob.ar/infolegInternet/anexos/0-4999/713/norma.htm

[16] *Microcontrolador ATMega328P*, About Components 101, 4 de Abril del 2018, https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet

[17] *Arduino Nano*, Arduino.cl, 10 de Noviembre del 2019, https://arduino.cl/arduino-nano/

[18] *Comunicación serial USART AVR*, Dr. Rubén Estrada, 15 de Abril del 2014. https://hetpro-store.com/TUTORIALES/comunicacion-serial/

[19] *Comandos AT - 3Cu Electrónica*, 5 de Junio del 2020, https://sites.google.com/site/3cuelectronica/home/comandos-at-1#:~:text=Los%20comandos%20AT%20se%20denominan,poder%20comunicarse%20con%20sus%20terminales

[20] *Aprender Sobre La Electrónica*, 5 de Junio del 2020. http://www.learningaboutelectronics.com/Articulos/Diferencia-entre-transistores-NPN-y-PNP.php

[21] *Checksum*, Wikipedia The Free Encyclopedia, 16 de Junio del 2020, https://en.wikipedia.org/wiki/Checksum

[22] *Raspberry Pi*, 5 de Junio del 2020, https://es.wikipedia.org/wiki/Raspberry_Pi