Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

CorsiBot: Interfaz humano-computadora para la evaluación de memoria de trabajo

Un aporte computacional a las ciencias cognitivas

Tesis de Licenciatura en Ciencias de la Computación

Maximiliano Urso

Director: Matías Lopez-Rosenfeld
Codirector: Federico Giovannetti
Buenos Aires, Abril 2021

RESUMEN

La prueba de Bloques de Corsi es utilizada en Psicología y ciencias cognitivas para evaluar la memoria de trabajo viso espacial. Consiste en la presentación de una serie de estímulos visuales en un determinado orden, que deben ser recordados y repetidos en el mismo orden por las personas evaluadas. La versión original era administrada por un/a investigador/a manualmente, mostrando las secuencias, anotando errores, tiempos y aciertos. Esto implicó reiteradas dificultades y la imposibilidad de reproducir la experiencia de manera precisa.

Se han desarrollado diversas versiones digitales utilizando computadoras o tabletas. Si bien ha significado un gran avance para reducir la cantidad de procesos manuales, existe un debate respecto a cómo impacta en la memoria de trabajo el uso de versiones digitales mediante dispositivos con pantalla, respecto de las mismas pruebas en versiones físicas.

En esta tesis se presenta el desarrollo del *CorsiBot*: Una versión física y automatizada que permite estudiar el desempeño y las estrategias de resolución de la prueba *Bloques de Corsi*. Se presenta una versión física, similar a la original, pero con una administración y registro automatizados.

Se ha probado satisfactoriamente el funcionamiento del dispositivo, realizando la prueba a 30 personas a las que se las evaluó con esta versión y con una versión digital sobre una computadora. Se han verificado grandes similitudes en el desempeño y algunas diferencias en los tipos de errores cometidos por las personas evaluadas con ambas versiones de la prueba.

Palabras claves: Automatización y captura con embebidos. Interfaz Humano-Computadora. Experiencia de Usuario. Bloques de Corsi. Memoria de trabajo.

ABSTRACT

The Corsi block-tapping test is widely used in psychology and cognitive sciences to assess visuospatial working memory. It consists of a series of visual stimuli, given in a specific order, that the participant needs to remember and mimic. Originally, the stimuli sequence was manually set by a researcher, while also noting errors, matches, and time. Naturally, this entailed a constraint to run the test accurately.

Today, there are many digital versions available for computers or tablets. And although this meant a significant cut on manual steps, there are discussions on the impact these digital versions have on working memory as opposed to traditional ones.

This thesis focuses on the development of the CorsiBot: a new physical and automated version that enables researchers to assess performance and resolution strategies for the Corsi block-tapping test while combining the benefits of both tangible and digital versions.

The device was satisfactorily tested on 30 individuals. The same individuals also took the digital version of the test on a computer. There were great similarities in performance and some differences in the types of errors made by the people tested.

Keywords: Automation and input capture in an embedded system. Human-computer Interface. User Experience. Corsi block-tapping test. Working Memory.

AGRADECIMIENTOS

Agradezco en primer lugar a Mati, mi director y compañero, quien vino en el año 2018 a proponerme el sueño de desarrollar este desafiante proyecto, por toda su confianza y acompañamiento. Sin él, este desarrollo no hubiese existido.

A mi compañera Sofía e hijo Simón, por lo feliz que me hacen todos los días y por todas las horas de aguante sin las cuales este trabajo no hubiera sido posible.

A mis viejos especialmente, que dieron siempre hasta lo que no tenían para lograr que tengamos las mejores condiciones posibles para nuestro desarrollo personal y profesional.

A Fede, co-director de lujo de esta tesis, con quien aprendí muchísimo y espero seguir haciéndolo.

A mis compañeros docentes de Taller de la Escuela Técnica 36: Pablo, Sebastián y Guillermo, quienes colaboraron también con la parte electrónica y mecánica de este trabajo.

A los compañeros del Laboratorio de Robótica y Sistemas Embebidos con quienes compartí varios años de lucha y aprendizajes.

A todas las personas que colaboraron ofreciéndose a realizar el experimento de manera desinteresada. A Monchi que me ayudó con la traducción del resumen.

A la Educación y Universidad Pública de este país, por garantizarme de la mejor manera posible recorrer un camino de aprendizajes y desarrollo de pensamiento crítico que no debe abandonarse nunca. A todes les docentes y estudiantes que en estos ámbitos resisten y construyen una educación comprometida y de calidad, todos los días.

A mis amigues y compañeres de militancia con quien reafirmo día a día la necesidad de ser solidaries y luchar por un mundo más justo.

A la vida, que me ha dado tanto.

ENLACES

La presentación y todo el material del presente trabajo se encuentra disponible en: https://sites.google.com/view/defensa-maxi-urso/.

Índice general

1. INTRODUCCIÓN

El mundo está viviendo uno de los cambios tecnológicos más importantes de la historia: las Tecnologías de la información y la comunicación (TICs) están haciéndose lugar en la vida cotidiana. Este fenómeno abarca desde el dispositivo que llevamos en el bolsillo llamado vulgarmente *teléfono inteligente* hasta el sistema de climatización de una casa que se activa cuando los moradores están retornando. Eso se debe a que, en las últimas décadas y entre otros factores, las computadoras y los sistemas embebidos redujeron drásticamente sus dimensiones y su costo [1, 2].

En este contexto, diversas tareas manuales empezaron a ser asistidas (e incluso realizadas completamente) por computadoras. Desde las calculadoras electrónicas hasta el pago de peajes con tecnologías RFID [3], desde un lector de código de barras hasta las aspiradoras autónomas. La lista se agranda día a día, con nuevos desarrollos que permiten llegar un poco más allá. Además de dispositivos que actúan sobre el mundo, en los últimos años apareció una gran variedad de desarrollos que registran eventos de la vida cotidiana y se utilizan para el estudio de comportamiento. En muchos casos se estudia con fines comerciales y en otros casos con fines académicos.

El campo de las investigaciones científicas no es una excepción. Simulaciones de eventos complejos, manipulaciones de gran precisión, registro pormenorizado de eventos son algunos ejemplos. En particular, las investigaciones sobre el funcionamiento de la mente humana han podido capturar eventos antes imposibles de registrar [4, 5]. Esto permitió por ejemplo que en una evaluación, además de poder observar el resultado obtenido, también se pueda estudiar la trayectoria utilizada para su resolución [6]. Estudiando dichas resoluciones es posible observar reflejos de algoritmos de cómputo humano utilizados por las personas para enfrentar un desafío. [7–9].

La prueba de Bloques de Corsi es utilizada en Psicología y ciencias cognitivas para evaluar la memoria de trabajo viso espacial [10]. Consiste en la presentación de una serie de estímulos visuales en un determinado orden, que deben ser recordados y repetidos en el mismo orden por las personas evaluadas. La versión original del test estaba construida con nueve cubos de madera visualmente iguales y era un/a investigador/a quien señalaba con una varilla los bloques, siguiendo un protocolo previamente definido. Cada ensayo estaba definido con dos partes: 1) el/la investigador/a señalaba, en un orden preestablecido, una secuencia de bloques y al terminar daba paso a que, 2) el/la participante intentara repetir la misma secuencia que le fue mostrada. Mientras esto sucedía, el/la investigador/a registraba manualmente el desempeño, anotando errores, tiempos y aciertos.

La prueba estaba compuesta por una sucesión de secuencias. Al terminar cada secuencia, dependiendo del desempeño, se pasaba a la siguiente secuencia del protocolo o se terminaba la tarea si se alcanzaba un criterio de corte que podría estar dado, por ejemplo, por una cantidad consecutiva de errores. Las secuencias se ordenan por longitud. Al principio son simples, comenzado con secuencias de longitud de dos bloques, y a medida que el/la participante logra responder correctamente, la dificultad se va incrementando con el aumento de la cantidad y combinación de bloques en las secuencias.

La dificultad en la prueba de Bloques de Corsi está dada por una combinación de factores tales como la cantidad de bloques a señalar [11], el recorrido de la secuencia a reportar [12] y los tiempos entre estímulos [11]. La realización de estas pruebas implica

procesos para los cuales la computación nos brinda herramientas para su estudio.

Fig. 1.1. Corsi versión original de madera. La/el investigador/a señalaba manualmente los cubos de cada secuencia y, al finalizar, la persona participante debía recordar y señalarlos también. El registro de los aciertos y errores los realizaba manualmente el/la investigador/a introduciéndose errores de registro.

Desarrollos posteriores del Bloques de Corsi incluyeron luces con pulsadores mediante los cuales el/la investigador/a iluminaba los cubos en lugar de tener que indicarlos con una varilla o con la mano. Sin embargo, la tarea de registro de las respuestas continuó siendo manual. Estos procesos de administración de la tarea y registro manual del desempeño por parte de los/las investigadores/as implicó reiteradas dificultades (errores de anotaciones, confusión de protocolos, etc.) y la imposibilidad de reproducir la experiencia de manera precisa (distintos tiempos de exposición de estímulos, errores en la administración, etc.) [13].

Para tratar de sortear esas dificultades, se han desarrollado diversas versiones digitales de la prueba de Bloques de Corsi administradas mediante Computadoras o Tabletas [13, 14]. De esta manera, es el software el que se encarga de reproducir las secuencias, registrar las respuestas de las personas evaluadas y determinar el avance en la prueba o la finalización de la misma mediante su configuración previa.

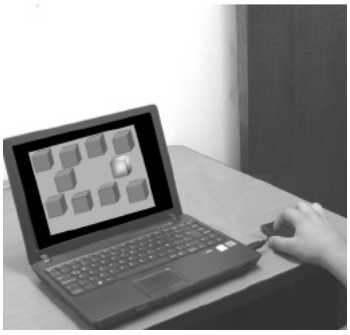

Fig. 1.2. Versión de Bloques de Corsi sobre Computadoras. La automatización de la reproducción de secuencias y registro de respuestas de las personas evaluadas, redujeron los errores manuales de registro.

El uso de estas versiones, si bien ha significado un gran avance para reducir la cantidad de procesos manuales por parte de los/as investigadores/as, presenta un debate respecto a cómo impacta en la memoria de trabajo el uso de versiones digitales mediante dispositivos con pantalla, respecto de las mismas pruebas en versiones físicas [15, 16].

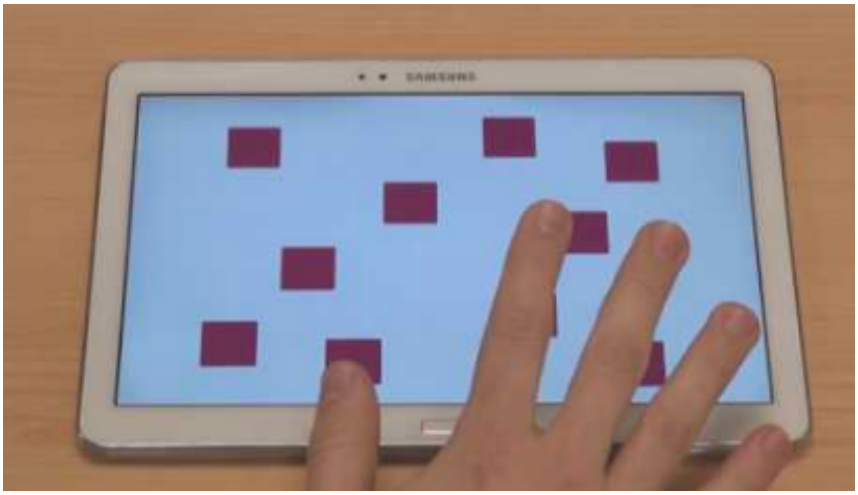

Fig. 1.3. Versión con pantalla táctil. Al igual que la versión de computadoras, automatiza la reproducción de secuencias y el registro de las respuestas. Por otra parte, evita dispositivos intermedios para responder como por ejemplo el mouse. Sin embargo, el uso de pantallas podría alterar el desempeño de las personas evaluadas, respecto de la versión física original.

En esta tesis nos propusimos construir un dispositivo físico y automatizado, para la administración y registro de las pruebas, que permita estudiar el desempeño y las estrategias de resolución sin pantalla del test *Bloques de Corsi*. Esto es posible mediante la incorporación de una computadora embebida para la automatización de los procesos relacionados con el tablero de luces y pulsadores.

Con este nuevo dispositivo de Corsi, esperamos recuperar el aspecto motriz-espacial de la tarea original pero con los beneficios de reproducir las secuencias de manera idéntica y de tener un registro con gran precisión de los eventos observables, reduciendo el sesgo y los errores humanos en la administración.

En este trabajo se detallará la construcción del dispositivo, las decisiones efectuadas, las dificultades encontradas y finalmente se incluirá una toma de datos donde se compara el uso del dispositivo con una versión digital del Bloques de Corsi.

El desarrollo del dispositivo está íntegramente publicado y es de acceso libre para su construcción y uso. Se espera de esta manera, dotar de un dispositivo confiable a grupos de investigación en las áreas de las Ciencias Cognitivas y las Neurociencias para sus investigaciones relacionadas con la memoria de trabajo.

1.1. Antecedentes

1.1.1. Debates entre lo analógico y lo digital

Existen debates en torno a las equivalencias o diferencias entre los diferentes test realizados de una forma tradicional (por ejemplo con lápiz y papel u otros objetos físicos) y las mismas pruebas realizadas mediante computadoras o tabletas. En la literatura científica del área se define el *Mode Effect* como el efecto por el cual idénticas pruebas, evaluadas sobre diferentes dispositivos, producen distintos resultados [17], Clariana y Wallace (2002). Leeson (2006) [16] realiza una exhaustiva revisión respecto de este efecto concluyendo que hay algunas pruebas y contextos donde la herramienta para realizar el test tiene impacto en los resultados.

Por ejemplo Wastlund y colegas (2005) [18], realizaron experimentos para comparar la lectura y comprensión de textos en papel y pantallas. Las pruebas consistían en leer un texto (desde una pantalla o desde un material impreso en papel) y responder preguntas para verificar la comprensión del texto. En su estudio concluyen que con los tests presentados a través de una pantalla se obtuvo un menor rendimiento respecto a las mismas versiones del test en papel. A su vez, las primeras generaron un mayor estrés y cansancio en las personas participantes.

De manera similar, Mayes y colegas (2001) [19] realizaron experimentos para comparar la lectura de textos y encontraron casos en que los tiempos de lectura eran menores dependiendo de los modos de presentación, aunque con desempeños más bajos en la comprensión de los textos en el caso de los experimentos en versiones de pantalla.

Por otra parte, Clariana y Wallace (2002) [17] introducen a la familiaridad previa que las personas tengan con estos dispositivos como un factor fundamental para el desempeño en los tests realizados en computadora. Otros estudios abordaron el tema, Taylor y colegas (1999) [20], Yu (2010) [21], aunque no encontraron evidencias suficientes para afirmar una correlación directa entre ambos factores.

Chua y colegas (1999) [22] afirman que la ansiedad en el uso de las computadoras podría estar inversamente relacionada con la experiencia previa en el uso de las mismas. McDonald, (2002) [23] hace una revisión de diversos estudios que relacionan estos factores y concluye que la evidencia disponible sugiere que la ansiedad en el uso de las computadoras puede causar un menor rendimiento en las pruebas cuando son evaluadas mediante estos dispositivos.

En cuanto a la carga de trabajo cognitiva, Noyes y colegas (2004) [24] publicaron los resultados de su experimento por el cual demostraron que el uso de pantalla demandó más esfuerzo para las personas evaluadas que la lectura sobre papel. El estudio consistió en que las personas debían leer un texto y luego responder un cuestionario *multiple choice*, seguido de un cuestionario tipo NASA-TLX como herramienta utilizada para medir la carga de trabajo cognitiva [25].

En cuanto a la memoria de trabajo viso espacial Carpenter y Alloway (2019) [15] concluyeron que para los tests que evaluaron la carga en la memoria de trabajo viso espacial, las versiones evaluadas con un soporte físico tuvieron mejor rendimiento que las proporcionadas mediante computadoras. Indican como razones posibles para esta diferencia el incremento en la carga cognitiva que representa el uso de las pantallas, los efectos de la tecnología y hasta incluso el estado socio económico o las diferencias en el desarrollo de acuerdo a las edades de las personas evaluadas.

Los estudios revisados, dejan en evidencia la necesidad de seguir evaluando similitudes y diferencias de los distintos tests realizados sobre diferentes plataformas para analizar el impacto del fenómeno conocido como *Mode Effect*.

1.1.2. La tarea de Bloques de Corsi

La memoria de trabajo es definida como un sistema de capacidad limitada, dedicado al mantenimiento y procesamiento de la información para el sostenimiento de la cognición y la acción (Baddeley et al., 2020) [26]. El modelo multicomponente es el modelo de uso más extendido en la literatura actual y estaría integrado por al menos cuatro componentes principales. Por un lado, se encontraría un componente ejecutivo central, encargado de guiar la atención y los recursos cognitivos hacia la información relevante y el uso de estrategias para la codificación, almacenamiento y recuerdo de la información. Vinculado a este componente, se encontraría el búfer episódico, que permitiría la asociación de información de distintas fuentes con la memoria de largo plazo. Finalmente, el modelo incluye un componente involucrado en el procesamiento verbal (el bucle fonológico), y otro involucrado en el procesamiento viso-espacial (la agenda visoespacial)(Baddeley et al., 2020) [26].

La tarea de Bloques de Corsi (Pickering, 2001) [27], es de uso extendido en el ámbito clínico y de investigación. La misma consiste en la presentación de una serie de estímulos visuales en un determinado orden, que deben ser recordados y repetidos por las personas participantes en el mismo orden. La tarea requiere la utilización de procesos de memoria de trabajo espacial mediante el compromiso de procesos ejecutivos y atencionales para la representación y el mantenimiento de la información (Patt et al., 2014) [28].

Los parámetros de dificultad de la tarea están dados por la combinación de distintos factores tales como la cantidad de bloques a recordar y las características del recorrido espacial realizado por la secuencia (Busch et al., 2005 [12]; De Lillo et al., 2016 [29]; A Orsini et al., 2001 [11]; Arturo Orsini et al., 2004 [30]). La cantidad de bloques, por un lado, representa el parámetro de dificultad más estudiado dentro de la literatura, asociado a la capacidad de almacenamiento de la memoria de trabajo. Sin embargo, diversos estudios han resaltado la importancia de estudiar el impacto de la estructura espacial de la secuencia a recordar. De Lillo y colegas (2016) [29], retomando la literatura del tema, definieron los parámetros de dificultad asociados con la estructura espacial: 1) La estructura, que refiere a la ubicación de los sucesivos bloques, indica que una secuencia es estructurada cuando contiene ítems consecutivos ubicados en la misma fila, columna o diagonal del espacio; 2) la longitud de la secuencia, definida como la superficie recorrida por la línea imaginaria que cruza a los bloques de la secuencia (medida en centímetros o milímetros); y 3) la cantidad de intersecciones generadas por el camino imaginario trazado por la secuencia espacial a recordar.

En términos generales, las personas participantes de diversos estudios mostraron mayores desempeños en secuencias estructuradas, de menor longitud y con menos intersecciones. Mientras que las secuencias menos estructuradas, con mayor longitud y más intersecciones

mostraron más errores. Además, los estudios muestran que cada uno de estos parámetros afecta de forma independiente a la dificultad de la secuencia (Busch et al., 2005 [12]; De Lillo et al., 2016 [29]; A Orsini et al., 2001 [11]; Arturo Orsini et al., 2004 [11]).

1.1.3. Versiones de Bloques de Corsi manual y digital

Un conjunto importante de estudios se dedicaron a comparar la implementación de diferentes adaptaciones computarizadas de la tarea Bloques de Corsi, con la versión original de madera.

Por ejemplo, Robinson y Brewer (2016) [13] comparan una versión de pantalla táctil sobre tabletas con la versión de madera original, concluyendo que no han encontrado mayores diferencias en el desempeño. Sin embargo, plantean la limitación de haber realizado el estudio con personas familiarizadas con tecnologías táctiles y sugieren que próximas investigaciones deberían incorporar muestras más heterogéneas y representativas.

Brunetti y colegas (2014) [14] también analizan una versión de pantalla táctil del Corsi mediante tableta y comparan sus resultados con resultados provenientes de estudios previos con la versión original [31,32]. Allí indican la ausencia de diferencias entre los datos provenientes de distintos dispositivos y concluyen que la versión computarizada *eCorsi* podría ser utilizada como reemplazo de la versión original. En sentido similar, Siddi y colegas (2020) [33] comparan la versión original del Bloques de Corsi con la versión de pantalla táctil en dos grupos diferentes de personas llegando a las mismas conclusiones.

Roser y colegas (2016) [34] comparan una versión de pantalla táctil del Bloques de Corsi con la versión *Walking* (una versión del Corsi a gran escala en la cual las secuencias se realizan caminando sobre un piso que contiene una representación 2D de los bloques, Fig. 1.4) encontrando diferencias entre las combinaciones de ambas modalidades.

Fig. 1.4. Versión *Walking*. Las personas evaluadas observan las secuencias mediante luces que iluminan los bloques y responden caminando sobre los mismos. Se hallaron diferencias en la comparación de rendimiento respecto de la versión sobre pantallas táctiles.

Como caso excepcional, Nelson y colegas (2000) [35] presentan una versión física y automatizada y la comparan con la versión original del Corsi. El equipo concluye no haber encontrado diferencias relevantes en los resultados entre ambas versiones. Un estudio posterior de dicho grupo de investigación [36] utiliza el mismo dispositivo, aunque no profundiza sobre la herramienta desarrollada. Este desarrollo es el único donde se planteó

utilizar una versión física y automatizada del Corsi, pero no se hallaron estudios posteriores en los cuales ese desarrollo sea retomado ni por el mismo equipo ni por otros grupos de investigación.

En el presente desarrollo, nos proponemos: 1) construir una versión física y automatizada del test Bloques de Corsi; 2) evaluar con un grupo de personas adultas dicha versión y una versión computarizada; y 3) comparar el desempeño en dichas tareas para analizar sus similitudes y diferencias así como también las ventajas y desventajas de una versión sobre la otra.

2. DESARROLLO

2.1. Objetivos

El problema que aborda la presente tesis articula las Ciencias de la Computación y las Ciencias Cognitivas. Desde esta perspectiva, se busca contribuir a la construcción de un dispositivo que permita el estudio de la memoria de trabajo espacial de una manera automatizada, emulando el dispositivo original. A partir de este dispositivo, esperamos hacer un aporte para la reducción de los errores en las tomas de datos, junto con la posibilidad de ofrecer una experiencia de evaluación para las personas participantes que no esté mediada por una pantalla.

En este marco, el objetivo general es desarrollar una versión física y automatizada del test Bloques de Corsi y evaluar su funcionamiento en comparación con una versión digital de características similares.

A partir de ello, se plantean los siguientes objetivos específicos:

- Desarrollar una versión física y automatizada del test Bloques de Corsi plausible de ser administrada a partir de dispositivos digitales portátiles (teléfonos inteligentes, tabletas y computadoras).
- Comparar el desempeño en la tarea de Bloques de Corsi de la versión construida con una versión digital de características similares.
- Comparar la experiencia de los sujetos experimentales en ambas versiones de la tarea de Bloques de Corsi en diversas dimensiones: usabilidad, experiencia de usuario, desempeño, errores.

2.2. Metodología

Previo a comenzar con la descripción de los requerimientos y proceso de construcción del Dispositivo, se definirán algunos conceptos con los que se trabajará fuertemente:

- Identificaremos como ***CorsiBot*** (o Dispositivo Físico) a la versión del test de Bloques de Corsi desarrollada en el presente trabajo, mientras que cuando se explicite el ***SoftCorsi*** (o Dispositivo Digital) se estará haciendo referencia a una versión del Test administrada mediante una computadora. La versión digital utilizada está disponible en el repositorio. [37].

  Cuando se mencionen los ***Dispositivos*** se estará haciendo referencia a estas dos versiones: *CorsiBot* y *SoftCorsi*.

- Los ***Estímulos*** son los cubos que se iluminan y permiten a la vez presionarlos en el momento de responder, tanto en el CorsiBot (mediante luces Led y Pulsadores) como en el SoftCorsi a través de la pantalla de la computadora donde se esté ejecutando el software y presionándolos mediante el uso de un mouse. El Test cuenta con nueve bloques que, más allá de su ubicación en el tablero, son indistinguibles para las personas participantes. Internamente, para la conformación de las secuencias y el

análisis de las respuestas identificaremos a los Estímulos con letras del conjunto [A..I].

- Una ***Secuencia*** es una lista de Estímulos. Las Secuencias podrán tener de dos a nueve Estímulos y éstos, a lo sumo, aparecerán sólo una vez en cada Secuencia. Identificaremos como *Longitud de la Secuencia* a la cantidad de Estímulos que contienen las mismas.
- Identificaremos como ***Protocolo*** al conjunto de Secuencias, agrupadas con un criterio de dificultad. Trabajaremos con dos Protocolos y los mismos tendrán 14 Secuencias con longitud creciente, desde dos hasta ocho Estímulos, incrementándose en un Estímulo cada dos Secuencias.
- Llamaremos ***Toma de Datos*** a la instancia en que se evalúa a personas con cada uno de los Dispositivos y Protocolos, en una fecha determinada. A su vez, llamaremos **evaluación** a la instancia individual en que una persona realiza el Test con un Dispositivo y Protocolo asignados.
- La ***persona que opera el sistema*** u ***operador/a*** será la que se encargue de administrar las evaluaciones, mientras que la ***persona evaluada*** será quien tenga el rol de responder luego de observar el encendido de cada una de las luces de las Secuencias. La entidad con la que se identificará a la persona evaluada en el sistema se denomina **Usuario**.

2.2.1. Construcción del dispositivo

Durante el proceso de entrevistas, se relevaron y documentaron los siguientes requerimientos para la construcción del CorsiBot:

Requerimientos

Aspecto y funcionalidades principales

REQ.1 El dispositivo deberá contar con nueve bloques visualmente iguales. Los mismos serán a la vez luces y pulsadores para que la persona que está siendo evaluada pueda ver el encendido y luego poder pulsar sobre los mismos. La distribución de los bloques estará dada por el plano de la Fig. 2.1.

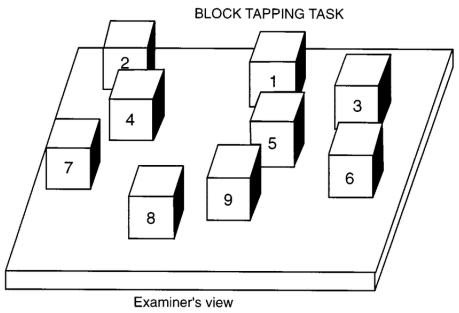

Fig. 2.1. Esquema de Bloques de Corsi original. El Dispositivo a construir deberá tener la distribución de Estímulos de la versión original [10] para que coincida a su vez con la versión SoftCorsi.

REQ.2 El Dispositivo deberá poder reproducir de forma automática las Secuencias de cada Protocolo encendiendo las luces que la componen y al finalizar informar a la persona que está siendo evaluada que puede comenzar a responder y presionar los botones.

REQ.3 El Dispositivo deberá poder registrar cada botón presionado por la persona que está siendo evaluada, con sellos temporales para indicar el intervalo de tiempo en el que se mantuvo presionado.

REQ.4 El Dispositivo deberá permitir la inclusión de Secuencias de práctica en los Protocolos cuya resolución por parte de las personas evaluadas no deberán ser verificadas.

REQ.5 El Dispositivo deberá incorporar un contador de tiempo disponible, configurable en el Protocolo, para la respuesta de las Secuencias del mismo por parte de la persona que está siendo evaluada. Una vez vencido ese plazo, el Dispositivo deberá computar como respuesta los botones presionados hasta ese momento para la Secuencia actual.

REQ.6 La persona que está siendo evaluada deberá poder informar al Dispositivo que ha concluido su respuesta para cada Secuencia.

REQ.7 El Dispositivo deberá poder indicar a la persona que está siendo evaluada si su respuesta ha sido correcta o incorrecta. Esta opción deberá poder habilitarse o deshabilitarse según se defina en el Protocolo.

REQ.8 El Sistema deberá tener la opción para definir un Criterio de Corte para una evaluación, con una cantidad de errores consecutivos definida previamente en el Protocolo. Superada esa cantidad, la evaluación deberá finalizar.

REQ.9 El Dispositivo deberá tener la opción configurable en el Protocolo para que la persona que está siendo evaluada pueda continuar con la próxima Secuencia de manera auto administrada.

Preparación de las evaluaciones

REQ.10 El sistema deberá permitir la carga previa de Protocolos con sus Secuencias y variables de configuración asociadas, de manera de no tener que configurarlas cada vez que se realiza una Toma de Datos.

REQ.11 El sistema no deberá registrar datos que identifiquen a las personas que serán evaluadas por lo cual deberá identificar a cada participante con un código de Usuario definido por la persona que opera el sistema.

REQ.12 El sistema deberá permitir la carga de una Toma de Datos con fecha y Usuarios a evaluar para la toma, y asociarla a un único Protocolo previamente cargado. Adicionalmente el sistema deberá incluir una opción para agregar un Usuario no contemplado inicialmente para la Toma de Datos.

Operación del Dispositivo

REQ.13 El Dispositivo no deberá incluir una pantalla por lo cual la persona que opera la evaluación deberá poder ver el estado y administrarla desde otro dispositivo pudiendo ser una computadora, tableta o teléfono inteligente.

REQ.14 El Dispositivo deberá permitir a la persona que opera el sistema seleccionar un Protocolo, Toma de Datos y Usuario previamente cargados para comenzar la evaluación.

REQ.15 Durante la ejecución de cada evaluación, tanto al reproducir la Secuencia como al estar registrando los botones presionados, el sistema deberá ofrecer las opciones de finalización de la presente Secuencia o de toda la evaluación para ese Usuario.

REQ.16 El sistema deberá registrar el resultado de cada respuesta, comparar si es correcta de acuerdo a la Secuencia presentada e informar el resultado a la persona que está operando el sistema.

REQ.17 Al finalizar la evaluación el sistema deberá presentar a la persona que opera el sistema un resumen con el resultado de cada una de las Secuencias que ha respondido la persona que realizó la evaluación.

Visualización de resultados y exportación de datos

REQ.18 La persona que opera el sistema deberá poder acceder a la revisión de los resultados y la navegación por los distintos Protocolos, Tomas de Datos y evaluaciones realizadas por cada Usuario.

REQ.19 El sistema deberá contener opciones para la exportación de los resultados de cada Usuario de manera individual, de una Toma de Datos o de todas las Tomas de Datos para un Protocolo, mediante archivos en formato CSV.

REQ.20 Los archivos en formato CSV deberán contar con la información correspondiente a las evaluaciones realizadas por los Usuarios, sus respuestas y se deberá poder visualizar la forma en que las respuestas han comenzado y finalizado (tanto por parte de la persona que opera el sistema como de la persona evaluada), los botones presionados con sus tiempos de presión y si la respuesta ha sido correcta o incorrecta respecto de la Secuencia original.

Diseño

Arquitectura del CorsiBot

Siguiendo los requerimientos relevados, se definió una arquitectura general que deberá contener los siguientes componentes:

- Una interfaz de entrada/salida para el encendido automático de las luces de las Secuencias y la captura de los botones presionados por las personas que realizan la evaluación. (REQ.1, REQ.2, REQ.9, REQ.6).
- Un enlace inalámbrico para la conexión entre el CorsiBot y la computadora, tableta o teléfono inteligente de la persona que opera el Sistema. (REQ.13).
- Una interfaz Web para administración de las pruebas por parte de la persona que opera el Sistema. (REQ.10, REQ.12, REQ.14, REQ.15, REQ.16, REQ.17, REQ.18, REQ.19).

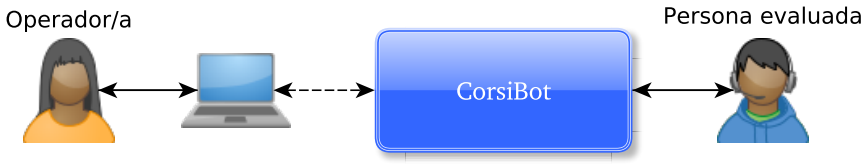

Fig. 2.2. Arquitectura general. La persona que opere el CorsiBot deberá conectarse vía wifi desde una computadora, tableta o teléfono inteligente para administrar las Tomas de Datos a través de una interfaz web. La persona que será evaluada observará las Secuencias y responderá presionando los botones del tablero del CorsiBot.

Diseño del Gabinete

Respetando la distribución del tablero original, se diseñó un gabinete para el CorsiBot, con las siguientes características para satisfacer el requerimiento REQ.1:

- Espacio para nueve bloques con la capacidad de ser luces y pulsadores a la vez.
- Barras verticales con la característica de ser de un material que pueda iluminarse de diferentes colores, mediante leds RGB [38]. Esto permitirá establecer distintos códigos de colores, tanto para las respuestas correctas o incorrectas como para códigos lumínicos que le indiquen a la persona que está siendo evaluada que comenzará una Secuencia, que puede comenzar a responder, etc.
- Espacio para dos pulsadores, uno en cada extremo, con la funcionalidad de confirmación de secuencia y avance de secuencia por parte de la persona que está siendo evaluada.

El diseño del gabinete puede observarse en la Fig. 2.3.

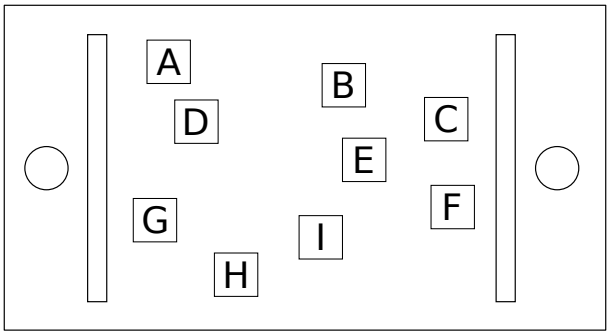

Fig. 2.3. Diseño del Gabinete. Cuenta con espacio para las 9 luces/botones de Secuencias, dos botones laterales para confirmación y selección de siguiente Secuencia y dos barras verticales con un material traslúcido para iluminarse mediante Leds RGB. Las dimensiones del mismo son: 390 mm de ancho, 217 mm de alto.

Hardware

Para el desarrollo del CorsiBot se definió utilizar una placa embebida *Raspberry Pi*, modelo *3B+* [39], la última versión disponible en el momento del desarrollo. La misma cuenta con las siguientes características:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- Memoria RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- 40 puertos de entrada/salida de propósito general
- Puerto HDMI
- 4 puertos USB 2.0
- Entre otras características.

Fig. 2.4. Raspberry Pi 3B+. Se seleccionó este sistema embebido puesto que provee las funcionalidades para controlar los puertos de entrada/salida en la cantidad necesaria y cuenta con hardware para conexión wifi. Además permite la ejecución de un sistema operativo lo que brinda muchas posibilidades de instalación de paquetes tanto para el desarrollo como para su uso.

Las características de este sistema embebido y su relativo bajo costo (U$D 35), se ajustan perfectamente a los requerimientos anteriormente mencionados.

Por una parte, sus interfaces de vídeo y USB permiten una rápida puesta en marcha y configuración inicial. Sus interfaces de red permiten, luego de configuraciones básicas, su administración remota y la funcionalidad de conexión inalámbrica que provee es uno de los requerimientos principales. (REQ.13)

La posibilidad de instalar un sistema operativo, ofrece gran flexibilidad para el desarrollo, permitiendo la instalación de paquetes necesarios, tanto para su uso final como para facilitar el desarrollo.

Los 40 puertos de entrada/salida que provee esta placa permiten el control independiente de las luces de Secuencia, los LEDs RGB (REQ.2, REQ.7) y los botones de siguiente Secuencia y confirmación (REQ.9). A su vez, se incluyó internamente un buzzer (parlante) como otra posibilidad que le permite al Dispositivo establecer feedback ante determinados eventos (por ejemplo distintos tonos para respuestas correctas o incorrectas REQ.7).

Para más información sobre la construcción y funcionamiento del Dispositivo, ver en el Apéndice Manual Técnico.

Software

Se definió la utilización del lenguaje Python como lenguaje principal, por su versatilidad, amplia cantidad de bibliotecas, extendida documentación y comunidad activa.

Sistema Operativo

La *Raspberry Pi* permite instalar y ejecutar un sistema operativo sobre el cual instalar diversos módulos y configurar sus periféricos. A su vez, esto permite la administración de manera remota, una vez configurados los accesos y las interfaces de red. Se eligió el sistema operativo *Raspbian Stretch versión 9.4*, un sistema operativo basado en *Linux Debian*.

Scripts de Entrada/Salida

Para el control de los pulsadores, luces de los bloques, buzzer y Leds RGB, se utilizaron bibliotecas en Python que permiten una sencilla configuración y control de los puertos de entrada/salida de la Raspberry Pi.

Framework Web

Para la interfaz web se seleccionó el Framework Web Django [40] para unificar el uso del lenguaje Python, lo que facilita la interacción con los scripts de control de las entradas/salidas de la Raspberry Pi.

Django incluye un servidor web liviano que permite la visualización de las páginas que permitirán a las personas que operan el sistema interactuar con el Dispositivo. Se utilizaron tecnologías responsive que permitieron adaptar la visualización de las páginas a los navegadores web de computadoras, tabletas y teléfonos inteligentes de manera automática.

A su vez el Framework Django provee una interfaz de operación simplificada con la base de datos, necesaria para el registro de los Protocolos, Secuencias, Botones presionados, entre otros.

Base de Datos

El diseño de la base de datos desarrollada, se representa mediante el Diagrama Entidad Relación de la figura 2.5.

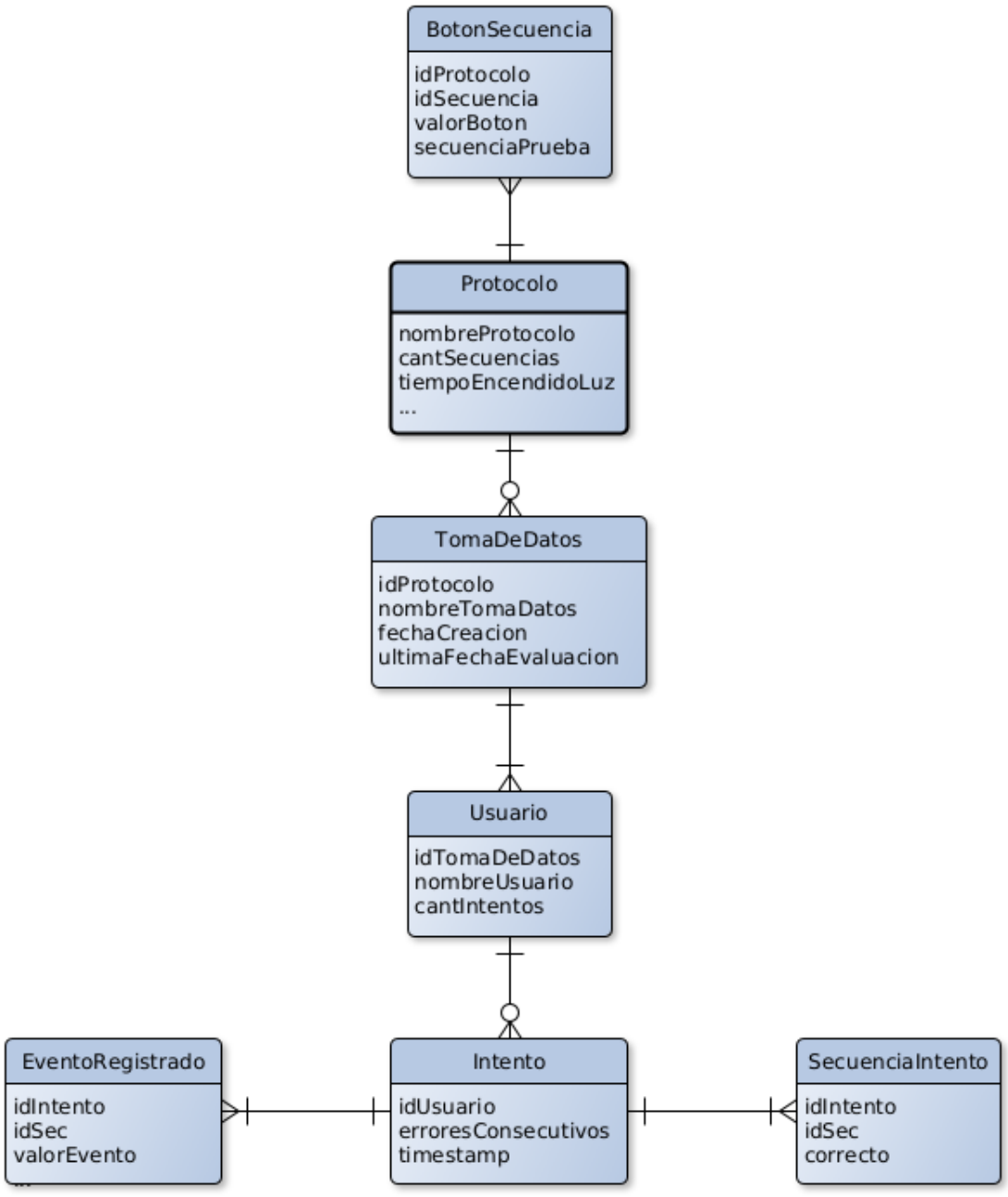

Fig. 2.5. Diagrama Entidad Relación. Entidades que modelan la información que debe ser registrada para el funcionamiento del Dispositivo.

Descripción general

La entidad ***BotonSecuencia*** representará a cada uno de los Estímulos que serán

parte de cada una de las Secuencias de un Protocolo. Podrá adquirir el valor de una letra que va desde el **A** hasta el **I**, puesto que el tablero cuenta con nueve luces/botones. Un BotonSecuencia podrá ser de práctica o no, dependiendo de la Secuencia y Protocolo al que se encuentre asociado.

La entidad ***Protocolo*** permitirá modelar y definir las características generales de los Protocolos como los tiempos de encendido de cada Led, demora entre encendido, todo tipo de feedback posible en cuanto a los Leds RGB, sonido, etc.

La entidad ***TomaDeDatos*** modelará la instancia de evaluación de un Protocolo para una cantidad de Usuarios.

La entidad ***Usuario*** es la entidad que modelará a cada persona que será evaluada con el CosiBot para una Toma de Datos particular.

La entidad ***Intento*** es la que permitirá modelar una participación de un Usuario en una TomaDeDatos.

La entidad ***EventoRegistrado*** modelará todo suceso que se produzca cuando el Usuario esté siendo evaluado. Representará los botones que ha presionado el mismo (incluyendo los Botones Rojos de inicio y confirmación) o eventos de control que indican cómo se ha iniciado o finalizado cada una de las Secuencias del experimento.

La entidad ***SecuenciaIntento*** modelará el resultado de la respuesta de un Usuario a una Secuencia.

Arquitectura Final

La arquitectura general del CorsiBot se representa con el siguiente diagrama:

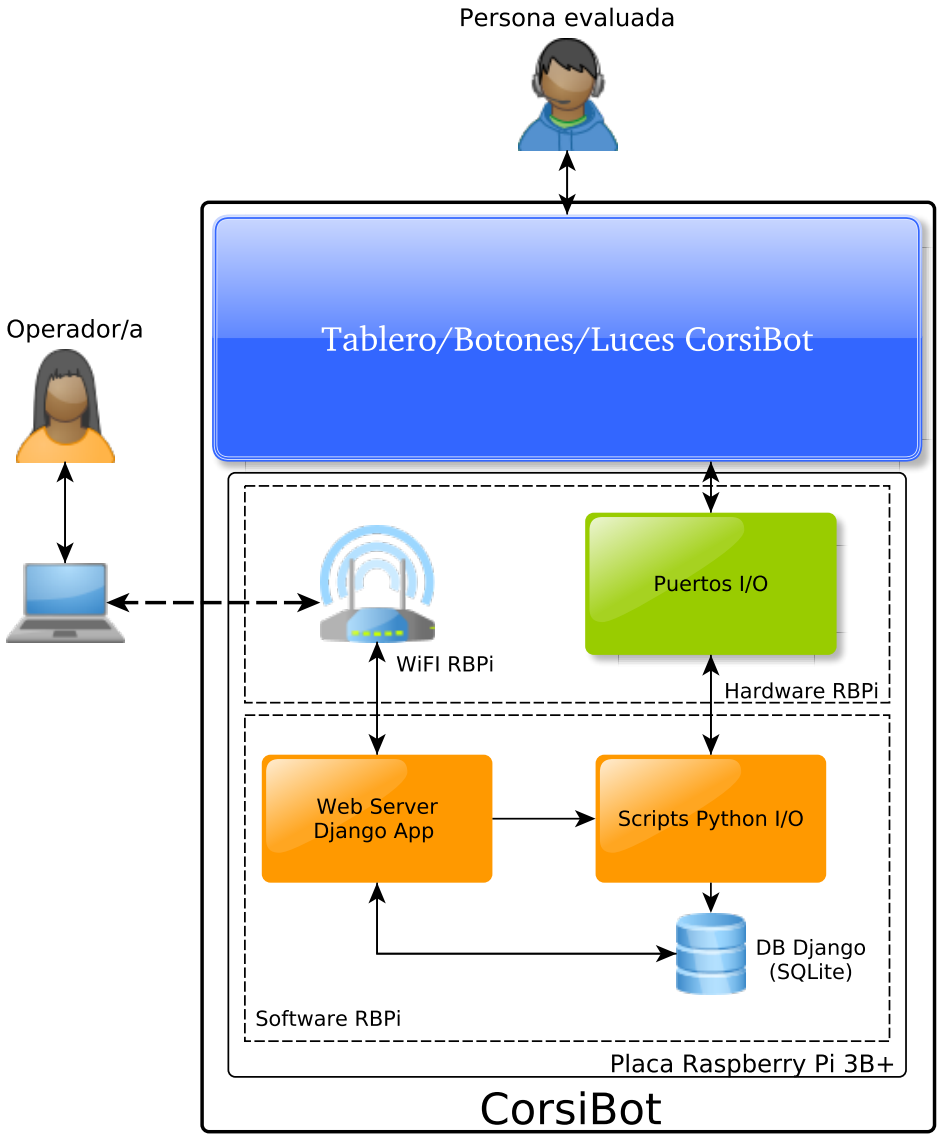

Fig. 2.6. Arquitectura Final. La persona que opera el sistema se conectará con su computadora, tableta o teléfono inteligente vía wifi al CorsiBot, e ingresará mediante un navegador web a la aplicación que será presentada por el Framework Django. Desde allí podrá cargar los Protocolos, Tomas de Datos y Usuarios que se registrarán en la base de datos. Podrá a su vez comenzar una instancia de evaluación, la cual se comunicará con los distintos Scripts de entrada/salida para reproducir las Secuencias y capturar los botones presionados por la persona que está siendo evaluada.

Para más información sobre la construcción y funcionamiento del Dispositivo, ver en el apéndice Manual Técnico.

Imágenes del Dispositivo desarrollado

Se presentan a continuación algunas imágenes del CorsiBot y de las pantallas de la interfaz web para la persona que opere el Dispositivo.

Fig. 2.7. CorsiBot. Vista frontal desde la posición de la persona que realizará la evaluación. Pueden observarse los 9 botones cuadrados con luces con la misma distribución que la versión original, los dos botones rojos laterales de confirmación y siguiente secuencia y las dos barras verticales traslúcidas que se iluminan mediante leds RGB.

Fig. 2.8. Vista lateral del CorsiBot. Se observa el conector de alimentación a 220 Volts.

Fig. 2.9. Pantalla principal de la interfaz web - Menú de opciones. Vista desde una PC. Con una resolución de pantalla menor, como ser una tableta o teléfono inteligente, se presentará un botón que mostrará un menú desplegable.

Analisis de la Evaluación

Protocolo: Juego
Nombre Toma Datos: Juego
Usuario: Juego
Fecha: Dec. 29, 2019

IdSec	Botones Secuencia	Botones Presionados	Correcto / Incorrecto
1	FBC	FBC	✓
2	BACD	BACD	✓
3	DCGB	DCGB	✓
4	CEAID	CEAID	✓
5	CHAFE	CHAFE	✓
6	IFGEBA	IFGBA	✗
7	DBEHCA	DBIHCA	✗
8	BCDEIFH	BCDEIFH	✓
9	DIGHBAC	DIGHAEC	✗
10	FEDIGACB	FEDIGCBA	✗
11	GDHAEFIB	DHGACFBE	✗

Volver Exportar CSV

Fig. 2.10. Vista de la interfaz web. Al finalizar una evaluación la persona que opera el sistema podrá visualizar un resumen con los resultados obtenidos, pudiendo exportar toda la información de la misma en formato CSV.

2.2.2. Diseño del experimento

Finalizada la construcción del CorsiBot, se procedió al diseño del experimento con el objetivo de comparar el Dispositivo desarrollado con una versión Digital del test de Bloques de Corsi. Dado que no existen Protocolos estandarizados para el Test, se confeccionaron dos Protocolos diferentes.

Diseño de los Protocolos

Para poder realizar el experimento y comparar las similitudes y diferencias entre el CorsiBot y la versión SoftCorsi, se definió confeccionar dos Protocolos con dos grados de dificultad distintos. Esto permitiría comparar el rendimiento entre ambos escenarios alternando la dificultad del experimento y el dispositivo con el cual se realizarán las evaluaciones.

Se definió que cada Protocolo debería tener un total de 14 Secuencias evaluables, incrementando la cantidad de Estímulos de manera creciente, sumando uno cada dos Secuencias, con un mínimo de 2 Estímulos y un máximo de 8 Estímulos. Definiremos la *longitud* de cada Secuencia como la cantidad de Estímulos que contiene.

Se trabajó con la restricción de que un mismo Estímulo no debía aparecer dos veces en la misma Secuencia. Adicionalmente y según el requerimiento REQ.4 se incorporarán al inicio de cada uno de los Protocolos cuatro Secuencias de práctica (no evaluables) para que las personas puedan comprender primeramente la dinámica de evaluación con los turnos de presentación de Estímulos de cada Secuencia y el momento para responder.

El objetivo buscado fue el de generar de la forma más automática posible un Protocolo más sencillo y un Protocolo con mayor grado de dificultad. Para ello se consideraron las siguientes variables para comparar el grado de dificultad:

Distancia total

La distancia total de una Secuencia en un Protocolo se calcula como la suma total de las distancias físicas entre cada par de botones consecutivos. Estas distancias se corresponden con la ubicación de los botones en el CorsiBot (expresadas en centímetros). En la versión SoftCorsi la ubicación y distancia será proporcional según la resolución de la pantalla con la que se esté evaluando. Se contempló como una de las variables de dificultad la distancia total puesto que era esperable que, en general, una Secuencia con mayor recorrido tendría mayor dificultad que una con menor recorrido. Por ello se trabajó con la idea de generar un **Protocolo Corto** y un **Protocolo Largo**, con una menor y mayor distancia total, respectivamente. Para comparar esta variable se implementó la suma de la norma vectorial entre cada par de botones consecutivos dentro de una secuencia (Ecuación 2.1) implementada mediante el Código 5, disponible en el apéndice Construcción de Protocolos.

$$\sum_{i=1}^{n-1} \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \tag{2.1}$$

2.1: Distancia total de una Secuencia. Se calcula como la sumatoria de la norma vectorial entre cada par de botones consecutivos de una Secuencia con n botones. Siendo (X_j, Y_j) la posición del botón j con $j \in [1, n]$.

Protocolos libres de pares repetidos

Un primer problema que surgió al automatizar la construcción de Protocolos fue la aparición reiterada de los mismos pares de Estímulos. Esto se produjo puesto que el algoritmo calculaba en cada paso la combinación óptima de Estímulos, tanto al minimizar como al maximizar las distancias. Es así que, por ejemplo si para la distancia mínima de dos Estímulos el par que minimizaba la distancia era la secuencia *A-B*, para secuencias de tres Estímulos resultaba ser la secuencia *A-B-C*.

Se trabajó para generar Protocolos "libres de pares repetidos", restringiendo la aparición de pares de Estímulos iguales en un mismo Protocolo con el fin de evitar que haya un "aprendizaje de pares frecuentes". Es decir, si la secuencia *A-B-C* ya era parte del Protocolo, se trataría de evitar que lo fuera la secuencia *A-B-C-D*, pero también la secuencia *A-B-E* o *B-C-F*.

Esta restricción promovió formas alternativas de construir los Protocolos, teniendo en cuenta que se intentaba maximizar o minimizar las distancias de las Secuencias en un Protocolo. Por un lado podrían construirse de mayor a menor, es decir maximizando o minimizando primero las distancias de las Secuencias con ocho Estímulos para luego ir construyendo las de menor cantidad contemplando que sean "libres de pares repetidos"; o construir primero las Secuencias de longitud dos minimizando o maximizando la distancia para luego seguir construyendo las de mayor cantidad de Estímulos. Para más detalle sobre la construcción de Protocolos, ver en el apéndice Construcción de Protocolos.

Cantidad de apariciones de Estímulos en cada Protocolo

Para tratar de evitar que los Protocolos se conformasen tomando una parte o algunas partes específicas del tablero (el sector superior, inferior, derecho, izquierdo o centro) se trabajó para que las Secuencias dentro de un Protocolo fuesen construidas con una distribución de Estímulos uniforme. Para trabajar con esta variable, por cada Protocolo generado se calculó la cantidad de apariciones de cada Estímulo y se tomó como métrica la **varianza** de cada una de esas listas, buscando minimizar dicho parámetro. La implementación de este cálculo puede observarse en el Código 6 del apéndice Construcción de Protocolos.

Intersecciones

Otra variable importante que se contempló en el momento de construir los Protocolos para el experimento, fue la cantidad de intersecciones en la trayectoria a recorrer para resolver las Secuencias de los mismos. Se considera una intersección cuando el recorrido realizado para completar una Secuencia implica atravesar un recorrido previamente realizado. En base a estudios previos [11, 12] se conoce que una secuencia con una o más intersecciones, tendrá una dificultad mayor que una que no contenga intersecciones. Ver Fig. 2.11.

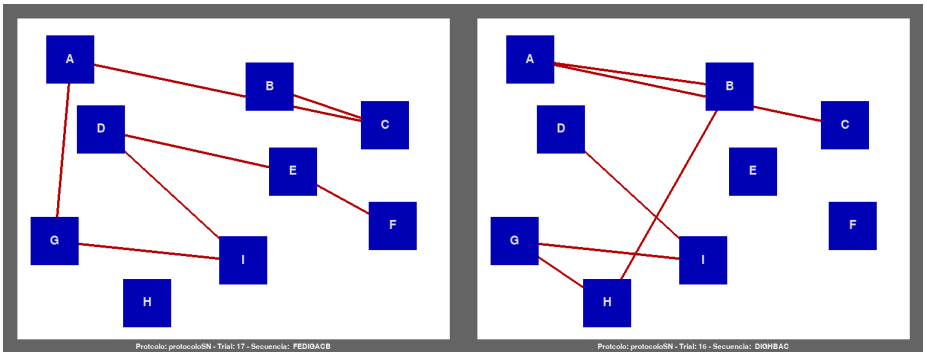

Fig. 2.11. Vista del tablero de la persona evaluada. A la izquierda, Secuencia de 8 Estímulos FEDIGACB sin intersecciones. A la derecha, Secuencia de 7 Estímulos DIGHBAC con intersecciones. Según estudios preliminares, la presencia de intersecciones en una Secuencia aumenta la dificultad de la misma.

Criterios finales para la construcción de los Protocolos

En la confección de los Protocolos, el criterio de que sean "libres de pares repetidos", resultó ser demasiado restrictivo restando muchísimas combinaciones posibles. Para flexibilizar este criterio pero cuidando el objetivo de no generar un "aprendizaje de pares frecuentes", se definió aceptar la presencia de los mismos pero con una distancia no menor a cinco Secuencias en un mismo Protocolo. Es decir, si el par *A-B* aparecía en la Secuencia número 5 del Protocolo, no podría volver a aparecer hasta la Secuencia número 10.

Por otra parte, el algoritmo que se definió utilizar para la construcción de los Protocolos fue el de conformar las secuencias de menor a mayor. Esto se debió a que en el caso contrario se evidenciaba un patrón común para las secuencias de mayor longitud, producto de la optimización en la construcción: Para el Protocolo Corto se producía un efecto "circular" en el recorrido dado que era el resultado de encontrar las mínimas distancias entre Estímulos en cada elección; mientras que para el Protocolo Largo se producía un efecto de cruces permanentes, puesto que era el resultado de encontrar siempre el Estímulo más lejano. Al comenzar la construcción con las Secuencias de menor cantidad de Estímulos y debido al criterio de que no haya pares repetidos cercanos, estos dos fenómenos se inhibieron en el momento en que el algoritmo construía las secuencias con mayor cantidad de Estímulos.

Se agrega que para la confección de las Secuencias candidatas por cada longitud dentro de un mismo Protocolo, y para no restringirse únicamente a la óptima en cada caso, finalmente se definió seleccionar con un criterio aleatorio dentro de las mejores, según se estuviera maximizando o minimizando las distancias.

Una última consideración que se tuvo en cuenta a la hora de definir los Protocolos que se utilizarían en cada experimento, fue la consideración de que en cada par de Secuencias de una misma longitud dentro de un mismo Protocolo, siempre se debería presentar la más sencilla en primer lugar. Puesto que, según las variables contempladas, se presentaban varias secuencias candidatas con igual o similares características, la elección final fue manual entre las posibilidades presentadas por los algoritmos utilizados.

Protocolos Finales

Finalmente se conformaron dos Protocolos para los experimentos: un protocolo de menor dificultad, denominado **"Protocolo Corto"** (Fig. 2.12) y otro con mayor dificultad, denominado **"Protocolo Largo"** (Fig. 2.13).

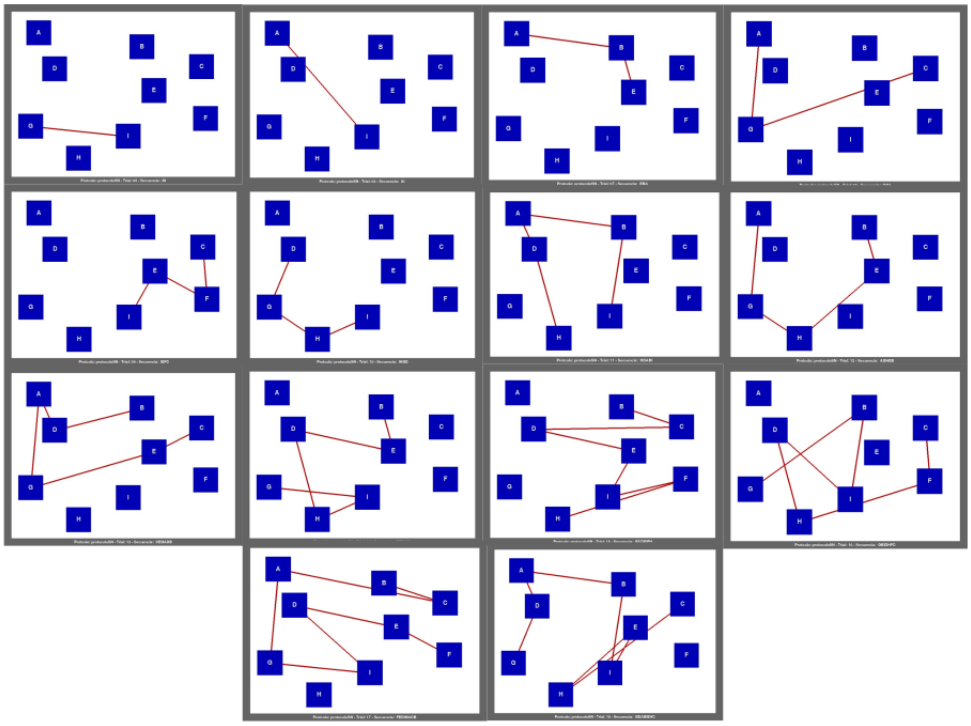

Fig. 2.12. Secuencias del Protocolo Corto ordenadas por cantidad de Estímulos de izquierda a derecha y de arriba a abajo. Contiene 14 Secuencias de longitud 2 a 8, dos por cada longitud. La distancia total del Protocolo Corto es de 122.85 cm. Sólo tres Secuencias tienen intersecciones.

Las distancias totales para ambos Protocolos, es decir, cuánto se debe desplazar la mano para resolver correctamente todas las Secuencias con el CorsiBot (o el proporcional según resolución de pantalla con el SoftCorsi) es de **122.85 cm** para el **Protocolo Corto** mientras que alcanza un total de **150.8 cm** para el **Protocolo Largo**.

Puede observarse una menor cantidad de intersecciones en el **Protocolo Corto** que en el **Protocolo Largo** (3 y 6 respectivamente) alterando este factor su dificultad. A su vez puede observarse en las figuras de las Secuencias de los Protocolos 2.12 y 2.13 que en general se cumple el objetivo buscado de que por cada par de Secuencias de igual longitud, la primera tiene menor dificultad que la segunda, sobre todo a partir de las secuencias con 5 y 6 Estímulos. Para más detalles sobre la construcción de los Protocolos, ver en apéndice Construcción de Protocolos.

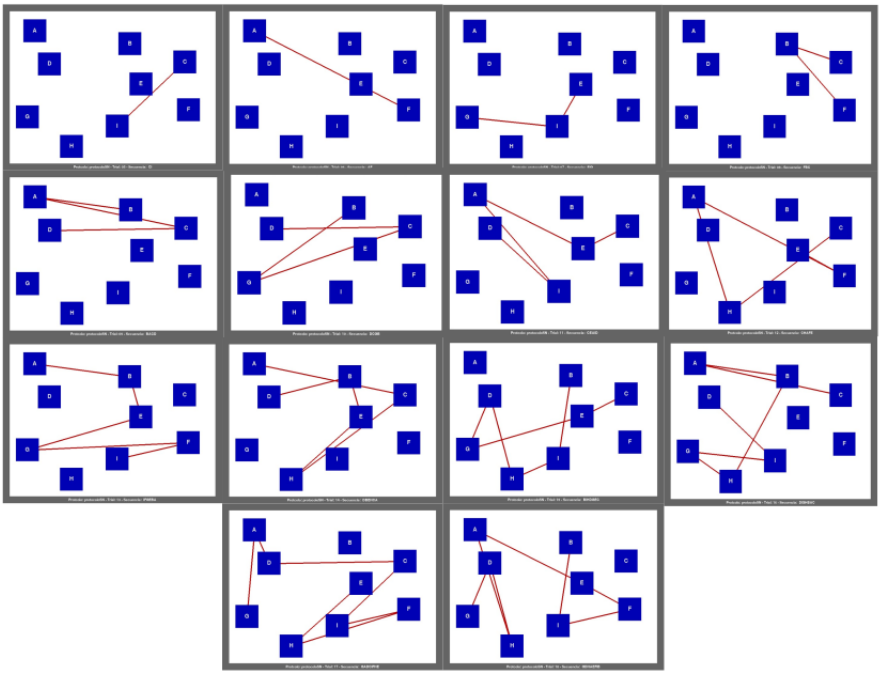

Fig. 2.13. Secuencias del Protocolo Largo ordenadas por cantidad de Estímulos de izquierda a derecha y de arriba a abajo. Contiene 14 Secuencias de longitud 2 a 8, dos por cada longitud. La distancia total del Protocolo Largo es de 150.8 cm. Son seis las Secuencias que tienen intersecciones.

Metodología de experimentación

Se diseñó un experimento para comparar dos versiones del Bloques de Corsi: la versión SoftCorsi ya existente y la nueva versión desarrollada. El estudio se estructuró de la siguiente manera:

- Cada persona evaluada realizará ambos protocolos: el **ProtocoloCorto** y el **ProtocoloLargo**, cada uno con un dispositivo diferente. Se identificarán como Toma 1 y Toma 2, según el orden cronológico en el que realice cada prueba.
- Se utilizará un ordenamiento contrabalanceado de las dos versiones de Dispositivos: **CorsiBot** y **SoftCorsi** y Protocolos: **Corto** y **Largo**, detallado en la tabla 2.1. Cada cuatro personas evaluadas se repetirá el orden de presentación de Dispositivo y Protocolo.
- Al finalizar cada una de las dos evaluaciones, la persona evaluada deberá realizar el cuestionario basado en el método NASA-TLX [25], incluido en el apéndice Cuestionario NASA-TLX.
- Las personas evaluadas deberán ser ciegas a las hipótesis del experimento.
- Como dispositivo Digital, se utilizará una computadora Notebook marca Samsung, modelo NP270E5E la cual tiene una pantalla de 14 pulgadas (de dimensiones muy similares al tablero del CorsiBot) y mediante el uso de un mouse para presionar los botones del SoftCorsi.

ID Usuario	Toma 1	Toma 2
Usr01	Corto - CorsiBot	Largo - SoftCorsi
Usr02	Largo - CorsiBot	Corto - SoftCorsi
Usr03	Corto - SoftCorsi	Largo - CorsiBot
Usr04	Largo - SoftCorsi	Corto - CorsiBot

Tab. 2.1: Cada persona evaluada realizará los dos Protocolos, cada uno con un Dispositivo diferente, según el orden en el que se tome la evaluación. El orden de asignación de Dispositivos y Protocolos se realizó según la tabla para balancear las condiciones. A la quinta persona se la evaluó con el ordenamiento de la primera y así sucesivamente.

El cuestionario tipo NASA-TLX que cada persona deberá completar al finalizar cada una de las Tomas de Datos consiste en una valoración numérica que deben calificar con una escala [1,20] para las siguientes variables subjetivas:

1. Exigencia Mental
2. Exigencia Física
3. Exigencia Temporal
4. Esfuerzo
5. Rendimiento
6. Nivel de Frustración
7. Nivel de Disfrute

Participantes

La experimentación con ambos Dispositivos (CorsiBot y SoftCorsi) se realizó en los meses de enero y febrero del año 2020. Las características de personas que participaron son:

- 30 personas, (16 mujeres).
- 26 de las personas manifestaron ser diestras mientras que sólo 4 manifestaron ser zurdas.
- La edad promedio fue de 33 años, con un máximo de 59 y un mínimo de 22 años.
- El nivel de estudios alcanzado por las personas evaluadas puede observarse en la Fig. 2.14
- El criterio para la selección de las personas fue el de *Muestreo por conveniencia.*
- Todas las personas manifestaron utilizar computadoras cotidianamente.

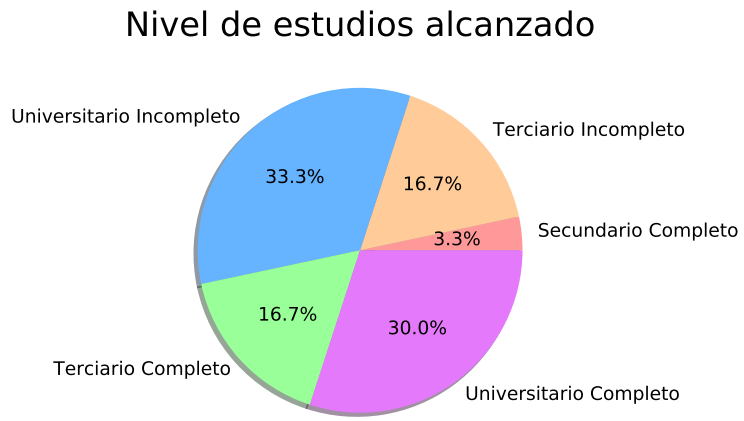

Fig. 2.14. Nivel de estudios alcanzado de las personas participantes

2.2.3. Plan de análisis

Con el propósito de responder a los objetivos de la tesis, se realizarán una serie de análisis estadísticos. En primer lugar, se calcularán los valores de media, desvío estándar, mediana y valores mínimos y máximos para la cantidad de respuestas correctas y máxima longitud de secuencia correcta alcanzada, agrupadas por Toma de Datos, Protocolo y Dispositivo.

Posteriormente, se analizará la homogeneidad entre la Toma de Datos 1 y la Toma de Datos 2 para las condiciones de Protocolo y Dispositivo mediante la prueba de U de Mann-Whitney. Esto permitirá establecer la similitud estadística entre las Tomas 1 y 2 para verificar que es posible sumarlas y poder generar análisis estadísticos más robustos en lo subsiguiente.

Luego de esto, se procederá a sumar ambas Tomas de Datos y comparar los valores de las variables de interés entre Protocolos y Dispositivos. Para esto, se realizarán comparaciones de mediana mediante el test de U de Mann-Whitney. Además, en todas las comparaciones se calcularán los tamaños del efecto mediante el estadístico r.

Con el objetivo de profundizar el análisis para determinar si aparecen fenómenos del tipo *Mode effect* [16], se procederá a analizar las respuestas erróneas con igual longitud que la Secuencia original, es decir, aquellas respuestas para las cuales las personas evaluadas presionaron la misma cantidad de botones, aunque no los esperados.

Para el análisis de errores y comparación entre Dispositivos y Protocolos se utilizarán las siguientes métricas:

Distancia de Levenshtein

Con el objetivo de analizar los errores respecto de las secuencias originales se realizará el análisis con la función conocida como Distancia de Levenshtein [41]. La misma permite comparar dos cadenas de texto y asignar un valor que será mayor en cuanto haya más diferencia entre ambas secuencias. Esto nos permitirá obtener una medida de la diferencia entre los errores de las respuestas con respecto a las Secuencias originales, para comparar luego entre Dispositivos y Protocolos. Por ejemplo, mientras la distancia de Levenshtein

entre las secuencias ABC y ABC es 0 (pues ambas son iguales), entre las secuencias ABC y ABD es 1, entre las secuencias ABC y AEF es 2 y así sucesivamente.

Será necesario introducir algunas modificaciones a la función de Levenshtein original, puesto que nuestro dominio implica ciertas restricciones, de manera que la nueva función pueda penalizar los distintos errores de manera diferente a la que lo realiza la función original. Las modificaciones a introducir deben contemplar:

- Penalizar más severamente cuando la secuencia de respuesta tiene un Estímulo que no estaba en la secuencia original.
- Penalizar más severamente cuando en la Secuencia de respuesta aparece un Estímulo duplicado.

El detalle de estas modificaciones puede encontrarse en el apéndice Distancia de Levenshtein.

Error de Camino

Un Error de Camino se define como aquel que tiene en la respuesta los mismos Estímulos que en la Secuencia original, aunque en un orden diferente. El resultado de esta métrica de error es binario: se respondieron utilizando los mismos Estímulos que en la secuencia original o se incluyeron Estímulos que no pertenecían a la misma.

Efectos de primacía y recencia

Los efectos de primacía y recencia [42] refieren a la capacidad de recordar determinados elementos de una secuencia a partir del orden de presentación de los mismos. El efecto de primacía hace referencia al recuerdo eficiente de los primeros elementos presentados, mientras que el efecto de recencia hace referencia al recuerdo eficiente de los últimos elementos presentados.

Para analizar el efecto de primacía se construyó como métrica el Índice del Primer Error (IPE) que indica 0 si el error se produjo en el primer botón presionado, 1 en el caso de que el error haya estado en el segundo botón presionado y así sucesivamente. Un IPE mayor implicará que hubo un mayor efecto de primacía puesto que el error se produjo "más tarde" en cuanto a la cantidad de Estímulos de la secuencia.

Para analizar el efecto de recencia se construyó como métrica el Índice del Último Error (IUE), que indica la última posición en que la respuesta y la Secuencia original son diferentes. De esta manera, si la persona que fue evaluada cometió un error en el último botón de la secuencia el IUE vale 0, si se equivocó en la penúltima vale 1 y así sucesivamente. Un mayor IUE indicará que hubo un mayor efecto de recencia puesto que la persona recordó mejor los últimos Estímulos observados.

Se analizarán las respuestas con errores de igual cantidad de Estímulos que la Secuencia original, para intentar determinar el momento en que las personas cometen dichos errores y analizar si hay mayor efecto de primacía o recencia en las respuestas.

Tiempo medio por botón presionado

Otra de las métricas que se analizarán en cuanto a las respuestas de las personas evaluadas es el tiempo en el que presionen cada uno de los botones. En principio, algo que se presupone es que las respuestas correctas tomarán menor tiempo que las incorrectas, puesto que en las segundas se prevé que haya un "efecto duda" que tomará más tiempo que el "efecto certeza" de las respuestas correctas. Para analizar esto se calculará el **tiempo**

medio por botón presionado, que consistirá en tomar la media del tiempo neto de respuesta (tiempo desde el inicio en que se presionó el último botón menos el tiempo desde el inicio en que se presionó el primer botón) sobre la cantidad de botones presionados (menos uno). Para este análisis también se tomarán las respuestas con igual longitud que la Secuencia original. Los tiempos de respuesta serán analizados sólo a nivel descriptivo dado que para realizar un análisis de significación estadística para cada condición y longitud de Secuencia se necesitaría una muestra de mayor tamaño.

Intersecciones y Secuencias con mayor cantidad de errores

Se analizarán las secuencias que contengan mayor cantidad de errores para comparar entre Dispositivos y se contemplará como variable la presencia de intersecciones en las Secuencias para analizar qué similitudes y diferencias se encuentran en las respuestas incorrectas producidas con ambos Dispositivos.

En resumen, se analizará el desempeño de las personas evaluadas en cuanto a cantidad de respuestas correctas y máxima longitud de respuesta correcta y para el análisis de errores y comparación entre Dispositivos y Protocolos se utilizarán las siguientes métricas:

- Distancia de Levenshtein entre las Secuencias originales y las respuestas incorrectas
- Errores de Camino
- Efectos de Primacía y Recencia
- Tiempo medio de respuesta por Botón
- Secuencias con mayor cantidad de errores
- Intersecciones

3. RESULTADOS

En esta sección se presentarán los resultados de la Toma de Datos definida en la sección Metodología de experimentación con la muestra detallada en la sección Participantes. Por cada participante se evaluaron dos Protocolos (**Protocolo Corto** y **Protocolo Largo**). Para ello, se utilizaron ambos dispositivos (**CorsiBot** y **SoftCorsi**) y por cada persona se evaluaron dos Tomas de Datos (**Toma 1** y **Toma 2**) según el orden de administración combinando cada uno de los dos Protocolos y Dispositivos.

3.1. Desempeño

Tomaremos como desempeño de las personas evaluadas a las siguientes dos métricas: Por un lado, la cantidad de Secuencias que logró responder correctamente la persona evaluada. Por otra parte, se analizará la máxima longitud de Secuencia correcta, es decir, la cantidad de Estímulos de la Secuencia con mayor longitud para la cual la persona evaluada ha respondido de manera correcta.

Análisis de homogeneidad entre las Tomas de Datos

Tal como se definió en la sección Plan de análisis, se procedió a realizar los análisis de homogeneidad entre las Tomas de Datos 1 y 2 entre los distintos Dispositivos y Protocolos con el fin de poder sumar los resultados de ambas Tomas de Datos y robustecer el análisis estadístico. Los análisis mostraron que tanto la proporción de respuestas correctas, como la máxima longitud de secuencia correcta fueron homogéneas para las Tomas 1 y 2 tanto en los dos Protocolos utilizados (Corto y Largo) como en los dos Dispositivos (CorsiBot y SoftCorsi). Ver Tabla 3.1.

Tab. 3.1: Homogeneidad en el desempeño entre las Tomas de Datos 1 y 2 para ambos Dispositivos y Protocolos.

	SoftCorsi				CorsiBot			
	Corto		Largo		Corto		Largo	
	U	p	U	p	U	p	U	p
Cantidad respuestas correctas	20	.36	25	.81	25.5	.80	29	.95
Máxima longitud de secuencia	25.5	.80	24.5	.70	34.5	.46	40	.16

Nota. U = Estadístico del test U-Mann – Whitney.

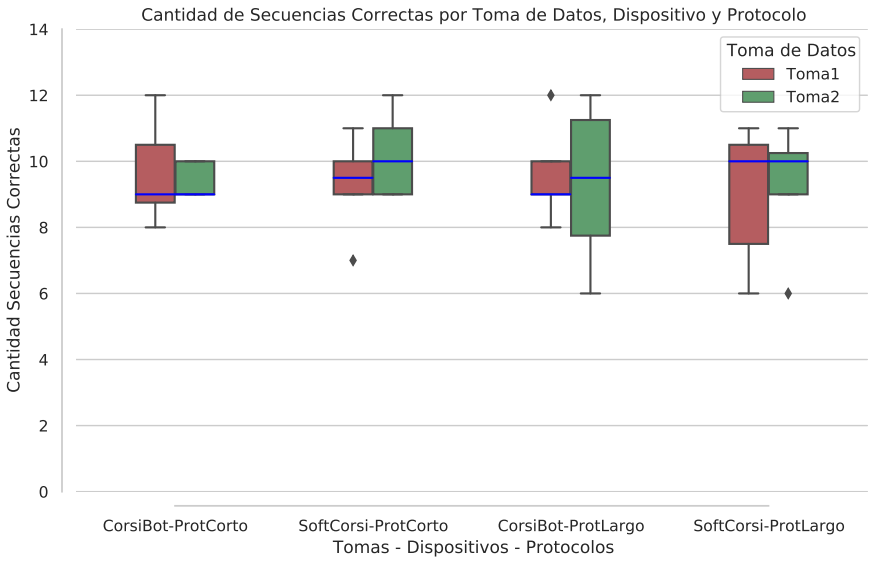

Fig. 3.1. Cantidad de respuestas correctas por Toma de Datos, Dispositivo y Protocolo.

En el análisis sobre la cantidad de respuestas correctas de las personas evaluadas, todas las medianas se mantienen entre los valores 9 y 10 y en ningún caso la diferencia es superior a 0.5, tal como puede observarse en la Fig. 3.1. Se puede afirmar que el orden de la Toma de Datos no impactó en el desempeño de las personas evaluadas, en cuanto a la cantidad de secuencias correctas que respondieron.

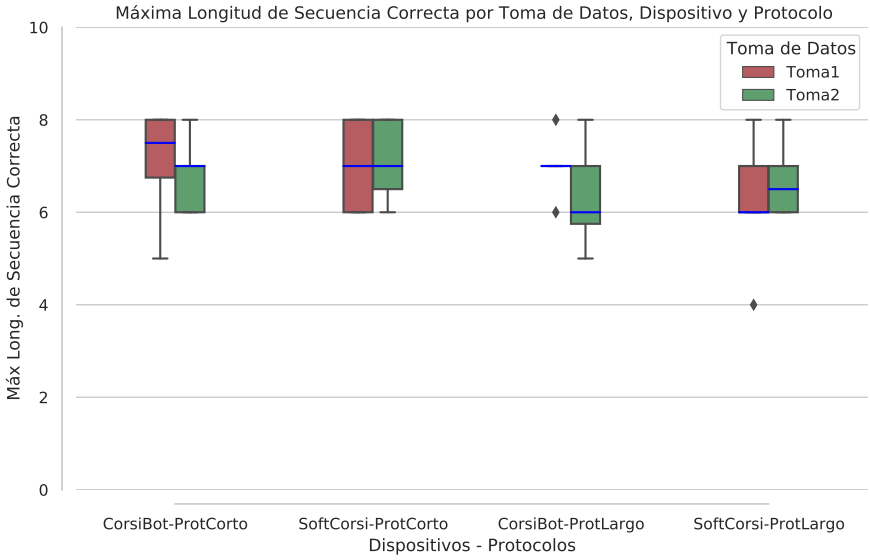

Fig. 3.2. Máxima Longitud de Secuencia Correcta alcanzada por Toma de Datos, Dispositivo y Protocolo

En cuanto a la máxima longitud de Secuencia para la cual las personas evaluadas

respondieron correctamente, al realizar la comparación entre Tomas de Datos, no se evidencian diferencias significativas, tal como puede observarse la Fig. 3.2. El único caso donde la distancia de las medianas alcanza una diferencia de uno, es con el Protocolo Largo realizado en el CorsiBot. Allí, para la Toma de Datos 1 se alcanza un valor de longitud de Secuencia de 7 Estímulos, mientras que para las personas que realizaron dicho Protocolo con el CorsiBot en la Toma de Datos 2, la mediana del máximo valor de Secuencia correcta es de longitud 6 Estímulos.

A partir de estos resultados, es posible afirmar que el orden en el cual las personas realizaron la evaluación de cada Protocolo con cada Dispositivo, no tuvo un impacto significativo en su desempeño. A partir de dichos resultados, se definió sumar los datos de ambas Tomas de Datos para los siguientes análisis.

3.2. Análisis de desempeño entre Dispositivos

Los análisis del desempeño mostraron que las personas evaluadas no mostraron diferencias estadísticamente significativas en el desarrollo de la tarea con ambos Dispositivos. Al comparar el desempeño en cuanto a la cantidad de Secuencias correctas por Dispositivo se observan valores similares para cada condición, tal como puede observarse en la Fig. 3.3.

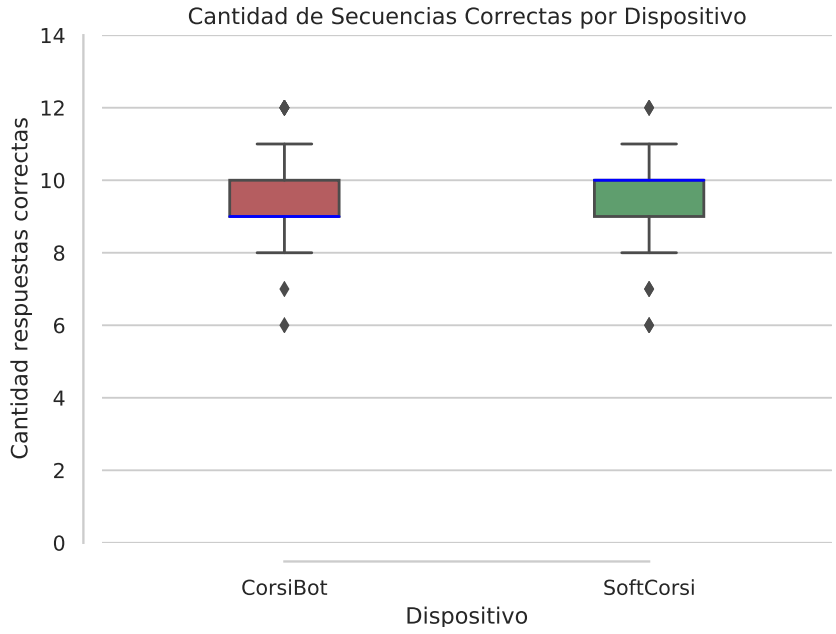

Fig. 3.3. Cantidad de Secuencias Correctas por Dispositivo: Sin distinguir por Tomas de Datos, se observa el desempeño de las personas evaluadas en cuanto a la cantidad de Secuencias que han respondido de manera correcta. Con el CorsiBot (Mdn: 9.0, SD: 1.5). Con el SoftCorsi (Mdn: 10.0, SD: 1.53).

Por otra parte, la máxima longitud de Secuencia correcta alcanzada se encuentra alrededor de 7 Estímulos con ambos Dispositivos, como puede observarse en la Fig. 3.4.

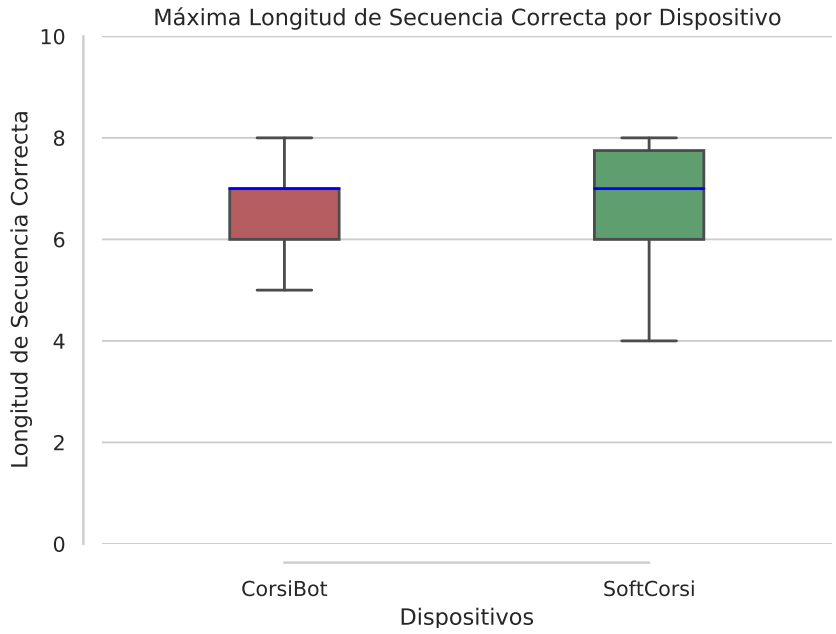

Fig. 3.4. Máxima Longitud de Secuencia Correcta: Sin distinguir por Tomas de Datos, se observa que el desempeño alcanzado en cuanto a la Máxima Longitud de Secuencia en la que las personas evaluadas respondieron de forma correcta es estadísticamente similar. CorsiBot (Mdn: 7.0, SD: 0.94) y SoftCorsi (Mdn: 7.0, SD: 0.97).

3.3. Análisis de desempeño entre Dispositivos y Protocolos

Al unificar las Tomas de Datos, por lo antes expuesto, podemos analizar si hubo diferencias en cuanto al desempeño de las 30 personas evaluadas, comparando los resultados entre Dispositivos y Protocolos. Los análisis no mostraron diferencias significativas en los desempeños en la tarea para los distintos Dispositivos y Protocolos administrados (tabla 3.2).

Tab. 3.2: Comparación del desempeño entre Dispositivos para los Protocolos.

	Ambos Protocolos			Protocolo Corto			Protocolo Largo		
	U	p	r	U	p	r	U	p	r
Cantidad de respuestas correctas	486.5	.58	.07	131	.43	.14	111	.96	.01
Máxima longitud de secuencia	459.5	.88	.01	125	.59	.09	106	.79	.05

Nota. U = Estadístico del test U-Mann – Whitney, r = Tamaño del efecto.

Cantidad de Secuencias Correctas por Dispositivo y Protocolo

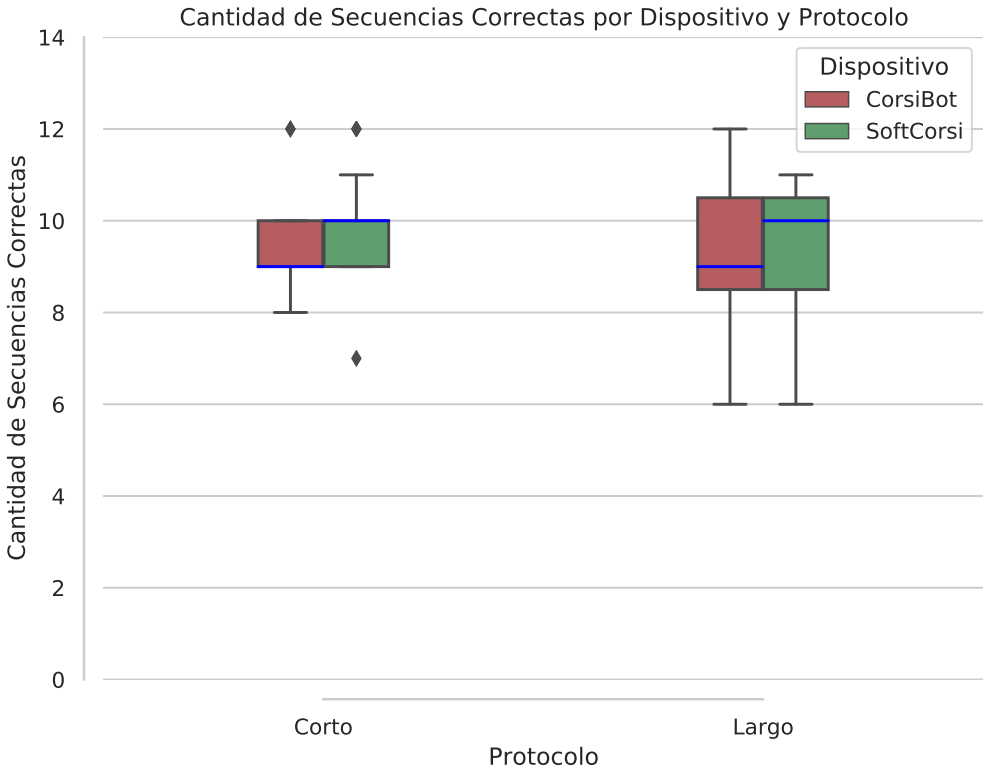

Fig. 3.5. Cantidad de Secuencias Correctas por Dispositivo y Protocolo: Unificando las Tomas de Datos, si se compara el desempeño de las personas evaluadas en cuanto a la cantidad de respuestas correctas, discriminando por Dispositivo y Protocolo, se observa que se mantuvo lo observado en la figura 3.3 con un desempeño bastante similar. ProtCorto CorsiBot (Mdn: 9.0, SD: 1.19), ProtCorto SoftCorsi (Mdn: 10.0, SD: 1.28), ProtLargo CorsiBot (Mdn: 9.0, SD: 1.8) y ProtLargo SoftCorsi (Mdn: 10.0, SD: 1.75).

Sin discriminar por Tomas de Datos, la cantidad de Secuencias que las personas respondieron de manera correcta discriminando por Dispositivos y Protocolos (Fig. 3.5), se mantuvieron en valores similares a los observados en el análisis de la comparación sólo por Dispositivos (Fig. 3.3). Es posible afirmar que no hubo diferencias significativas en el desempeño entre Dispositivos en ninguno de los Protocolos evaluados.

Máxima Longitud de Secuencia correcta por Dispositivo y Protocolo

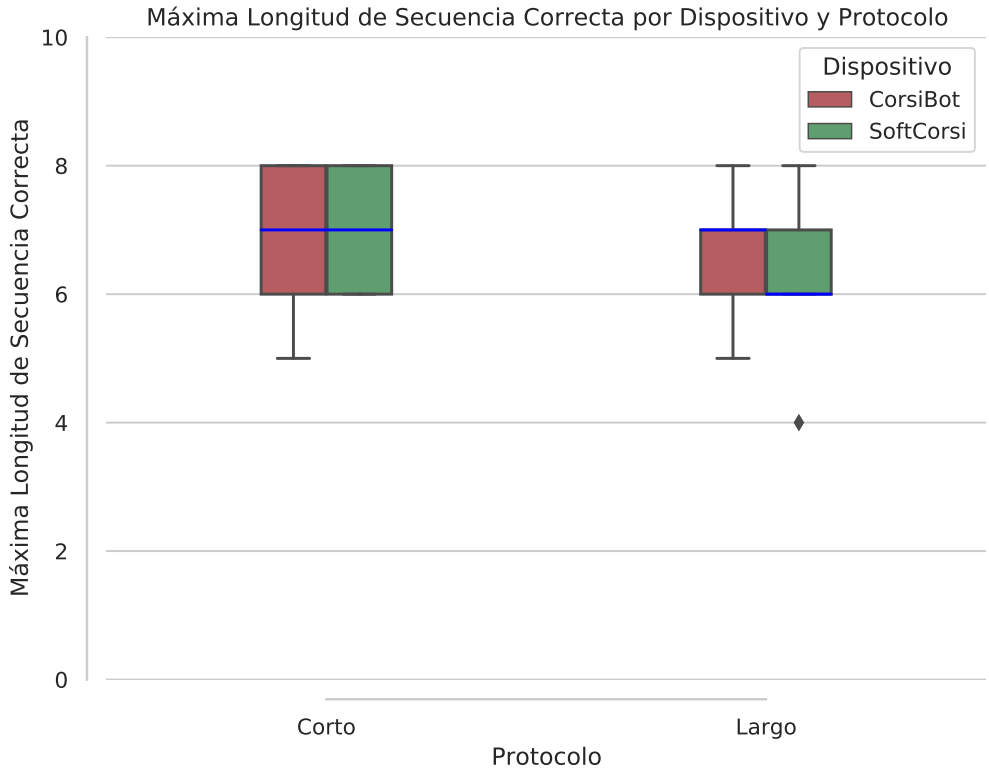

Fig. 3.6. Máxima Longitud de Secuencia correcta por Dispositivo y Protocolo: Unificando las Tomas de Datos, si se compara el desempeño de las personas evaluadas en cuanto a la máxima longitud alcanzada, discriminando por Dispositivos y Protocolos, se observa una leve diferencia con respecto a la comparación sólo por Dispositivos (Fig. 3.4). Mientras que para el Protocolo Corto el desempeño fue similar, en CorsiBot (Mdn: 7.0, SD: 0.96) y en SoftCorsi (Mdn: 7.0, SD: 0.88); para el Protocolo Largo se observa una leve diferencia entre Dispositivos: CorsiBot (Mdn: 7.0, SD: 0.91) y SoftCorsi (Mdn: 6.0, SD: 0.99).

De la misma forma, unificando las tomas de datos y al analizar la Máxima Longitud de Secuencia Correcta discriminando entre Dispositivos y Protocolos (Fig 3.6) no se observan diferencias significativas respecto de la comparación sólo por Dispositivos (Fig. 3.4). Sólo se observa una mínima diferencia en la mediana de la máxima longitud alcanzada para el Protocolo Largo realizado con el SofCorsi (6 Estímulos), en comparación con el CorsiBot (7 Estímulos). Para el Protocolo Corto no se evidencian diferencias entre Dispositivos, en cuanto a la máxima longitud de Secuencia correcta.

3.4. Análisis de errores

Cantidad de errores

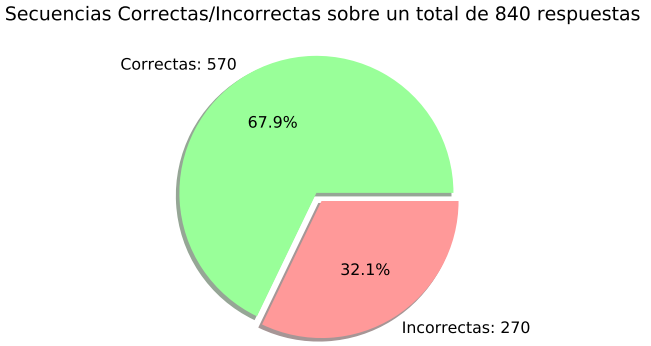

Fig. 3.7. Secuencias Correctas vs Incorrectas: Sobre un total de 840 respuestas, de las cuales las 30 personas evaluadas respondieron 28 cada una, se obtuvo un total de 570 respuestas correctas (67,9 %) mientras que 270 veces lo hicieron de manera incorrecta (32,1 %).

En la Fig. 3.7 puede observarse la distribución porcentual entre las Secuencias que las personas evaluadas respondieron de manera correcta o incorrecta. A cada persona participante se le presentaron en total 28 secuencias, 14 por cada Protocolo y Dispositivo. Del total de las 840 secuencias, fueron respondidas correctamente 570 (67,9 %) mientras que 270 respuestas fueron incorrectas (32,1 %).

Respuestas con Igual y Distinta Longitud que la Secuencia original

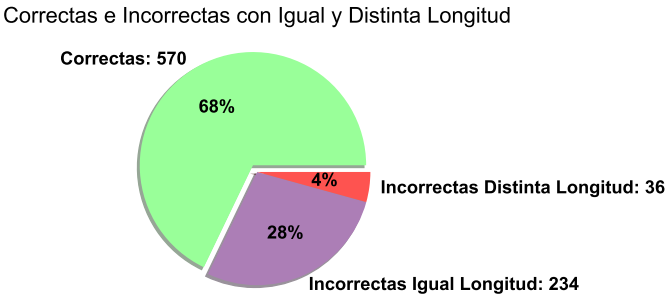

Fig. 3.8. Respuestas Correctas e Incorrectas con Igual y Distinta Longitud que la Secuencia original: Sobre un total de 840 respuestas, de las cuales las 30 personas evaluadas respondieron 28 cada una, se obtuvo un total de 570 respuestas correctas (67,9 %) mientras que 234 veces (27,9 %) lo hicieron de manera incorrecta pero presionando la misma cantidad de botones que la Secuencia presentada. En 36 oportunidades (4.3 %) respondieron con una cantidad de botones diferente a la longitud de la Secuencia presentada.

En la Fig. 3.8 se observa que de las 840 respuestas, 234 veces (27,9 %) las personas evaluadas respondieron erróneamente pero presionando la misma cantidad de botones que

la secuencia presentada, mientras que en 36 oportunidades lo hicieron con una cantidad de botones diferente a la longitud de la secuencia presentada, lo que representó un 4,3 % del total de respuestas.

Es importante destacar que la longitud de la Secuencia presentada en cada oportunidad era conocida por la persona que iba a responder. Por un lado, la persona que operaba el sistema se lo informó oralmente cuando se estuvo evaluando con el CorsiBot. Por otro lado, la longitud de la secuencia aparecía en la pantalla antes de reproducirse en el SoftCorsi.

Comparación entre Dispositivos

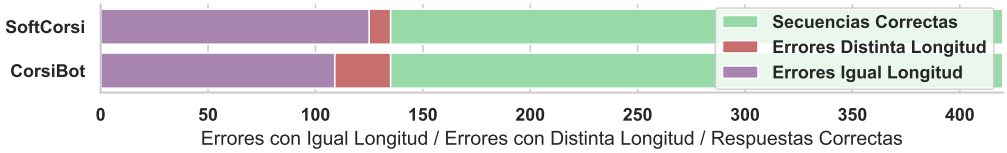

Fig. 3.9. Respuestas Correctas/Incorrectas por Dispositivo y errores con igual y distinta longitud: En cada Dispositivo las 30 personas evaluadas respondieron 420 Secuencias, 14 respuestas por cada persona. En ambos se obtuvo la misma cantidad de respuestas incorrectas (135, 32,1 %), aunque se observa una diferencia entre los errores que tienen igual y distinta longitud que la Secuencia presentada. Mientras que en el SoftCorsi las cantidades de respuestas incorrectas con igual y distinta longitud que la Secuencia original fue de 125 y 10, respectivamente; con el CorsiBot la diferencia fue mayor, con 109 respuestas incorrectas con igual longitud que la secuencia presentada y en 26 casos la longitud fue diferente.

En la Fig. 3.9 podemos observar la distribución de respuestas correctas e incorrectas con cada Dispositivo. A su vez, dentro de las incorrectas se puede diferenciar entre las que obtuvieron igual o distinta longitud que la Secuencia presentada, discriminando por Dispositivos. En primer lugar se puede observar que con cada Dispositivo se obtuvo la misma cantidad de respuestas correctas e incorrectas. De las 420 respuestas totales, en ambos casos las personas evaluadas respondieron 285 veces de manera correcta, mientras que en 135 oportunidades lo hicieron de manera incorrecta. Sin embargo, se observa una diferencia en la longitud de las respuestas incorrectas: mientras que con el SoftCorsi las personas respondieron 125 veces respetando la longitud de la Secuencia y sólo en 10 oportunidades no la respetaron; con el CorsiBot esta diferencia fue mayor, con 109 respuestas de igual longitud a la secuencia original y 26 respuestas con una longitud diferente.

Las posibles causas de estas diferencias entre errores fueron analizadas de forma cualitativa entre las 10 del SofCorsi y las 26 del CorsiBot. En primer lugar, se observó que algunas respuestas incorrectas con el CorsiBot podrían ser atribuidas a problemas en la interfaz Persona-Dispositivo. En algunos casos, se detectaron registros de botones duplicados (debido a que la persona consideró que no había presionado el botón correctamente y lo volvió a presionar); mientras que en otros casos se observaron Estímulos faltantes, atribuibles a que no se presionó correctamente el botón. Esto último podría llegar a deducirse en algunos casos por la frecuencia de respuesta, cuando presenta un incremento de tiempo mayor al promedio en el momento donde se omitió o no registró el botón presuntamente presionado; aunque también ese incremento de tiempo podría ser producido por la duda de la persona evaluada. Dado que las características de estos errores no pueden ser afirmadas con total seguridad y por la poca cantidad de casos de este tipo de respuestas (4.3 % del

total), no se considerarán para el análisis de errores. De esta manera, se evitó introducir al análisis respuestas erróneas que pudieran ser causadas por problemas con la interfaz del o de los Dispositivos.

Centraremos el análisis de errores sobre las 234 respuestas incorrectas que mantuvieron la misma longitud que la Secuencia presentada, que representaron el 86.7 % de los errores cometidos por las personas evaluadas.

Distancia de Levenshtein entre las Secuencias originales y las respuestas

Tal como se mencionó en la sección Plan de análisis, utilizaremos esta métrica para comparar las diferencias entre las Secuencias presentadas y las respuestas incorrectas.

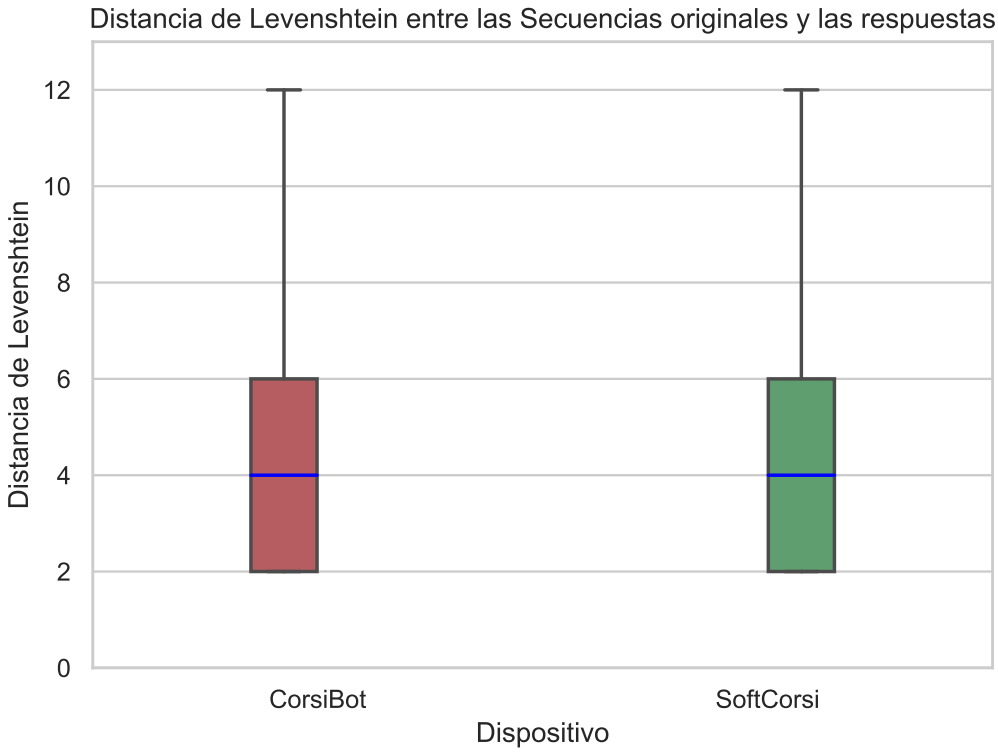

Fig. 3.10. Método extendido de Distancia de Levenshtein entre respuestas y secuencias presentadas: En la comparación entre Dispositivos con el método extendido de Distancia de Levenshtein entre las Secuencias presentadas y las respuestas de las personas evaluadas, no se observan diferencias significativas. Valores de distancia para el CorsiBot (Mdn: 4.0, SD: 2.48) y para el SoftCorsi (Mdn: 4.0, SD: 2.55).

Comparando por Dispositivos, para esta métrica, no se obtuvieron diferencias sustanciales entre los errores, tal como puede observarse en la Fig. 3.10.

Distancia de Levenshtein por Dispositivo y Protocolo

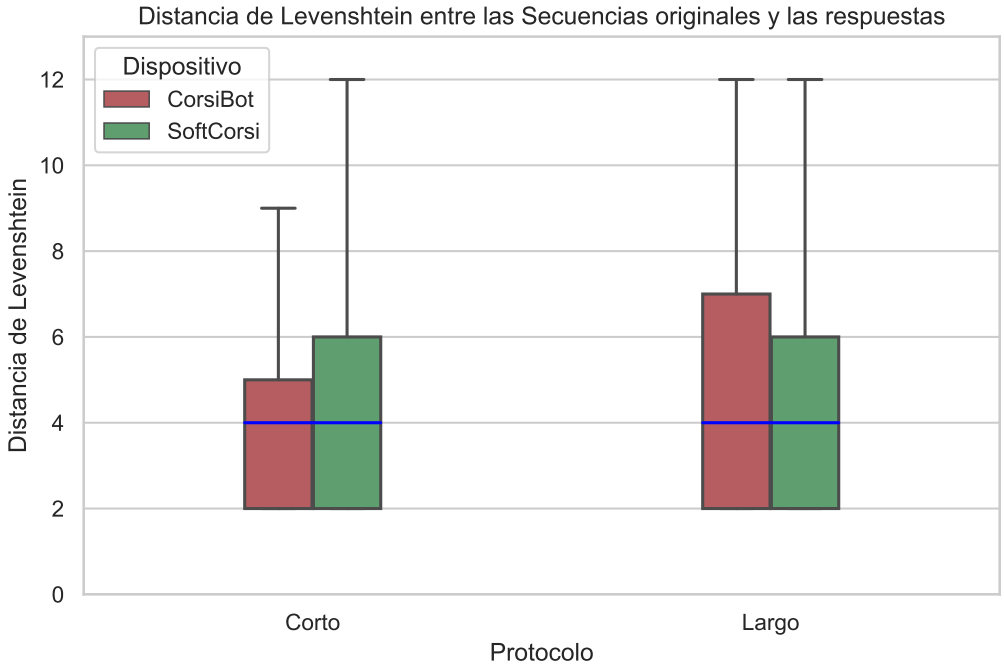

Fig. 3.11. Método extendido de Distancia de Levenshtein entre respuestas incorrectas y Secuencias presentadas comparando por Dispositivo y Protocolo: En la comparación entre Dispositivos y Protocolos tampoco se observan diferencias significativas. El valor medio de la distancia entre respuestas erróneas y Secuencias presentadas con este método es de 4.0 en todos los casos.

Como puede observarse en la Fig. 3.11, al comparar con el método extendido de Distancia de Levenshtein entre las Secuencias esperadas y las obtenidas, se evidencia que no hubo diferencias significativas en cuanto a los errores, comparando por Dispositivos y Protocolos. Es así que los errores, según esta métrica, presentan características similares independientemente de los Dispositivos y Protocolos. Ver tabla 3.5.

Errores de Camino

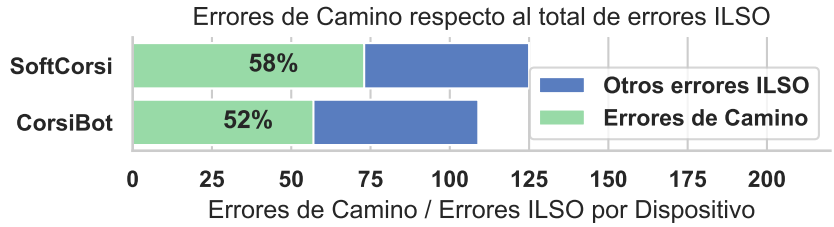

Fig. 3.12. Errores de camino respecto de otros errores por Dispositivo: Sobre las respuestas con errores ILSO (igual longitud que la secuencia original), se observa que con el SoftCorsi cometieron una cantidad mayor de Errores de Camino (73, 58.4 %) mientras que con el CorsiBot la cantidad de Errores de Camino fue muy similar a la del resto de los errores (57, 52.3 %).

Los análisis de la cantidad de Errores de Camino, no mostraron una diferencia estadísticamente significativa en estos tipos errores comparando ambos Dispositivos. Sin embargo, en la figura 3.12, puede observarse que mientras con el CorsiBot los Errores de Camino (52 %) tuvieron una cantidad prácticamente igual a otro tipo de errores, con el SoftCorsi se observaron una mayor proporción de éstos (58 %). Ver Tab. 3.3.

Dispositivo	Errores ILSO	Errores de Camino	Porcentaje
SoftCorsi	125	73	58.4 %
CorsiBot	109	57	52.3 %

Tab. 3.3: Errores de Camino: Dentro de los Errores ILSO (con Igual Longitud que la Secuencia Original), los Errores de Camino son aquellos en los que las personas evaluadas presionaron los botones correctos pero en un orden diferente al presentado.

Errores de Camino por Protocolo y Dispositivo

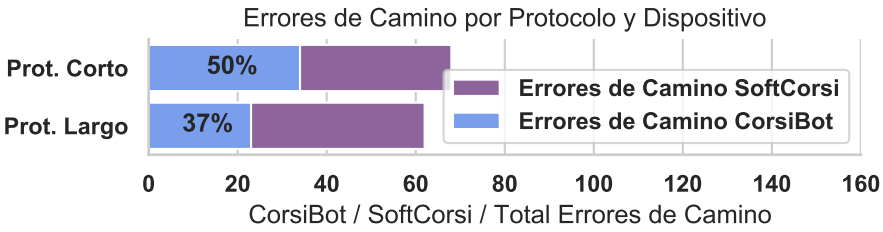

Fig. 3.13. Errores de Camino por Protocolo y Dispositivo: Sobre el total de Errores de Camino discriminando por Protocolo y Dispositivo, se observa que mientras que para el Protocolo Corto las personas evaluadas cometieron la misma cantidad de errores con ambos Dispositivos (34); para el Protocolo Largo hubo una cantidad mayor con el SoftCorsi (39; 63 %) mientras que con el CorsiBot fue significativamente menor. (23; 37 %).

Al discriminar los Errores de Camino por Protocolo y Dispositivo, se encontraron diferencias estadísticamente significativas con el Protocolo Largo (ver tabla 3.4). Mientras que con el CorsiBot los Errores de Camino fueron sólo 23/62 errores (37 %), con el SoftCorsi se cometieron una cantidad considerablemente mayor: 39/62 errores (63 %). Este fenómeno no se produjo con el Protocolo Corto, donde las personas evaluadas cometieron exactamente la misma cantidad de Errores de Camino con ambos Dispositivos: 34/68 errores. (Tab. 3.4).

Se puede afirmar que para el Protocolo de mayor dificultad, las personas evaluadas cometieron más Errores de Camino con el SoftCorsi que con el CorsiBot, tal como puede observarse en la Fig. 3.13.

Protocolo	Errores de Camino	CorsiBot	SoftCorsi
Corto	68	34	34
Largo	62	23	39

Tab. 3.4: Errores de Camino por Protocolo y Dispositivo.

Efecto de Primacía

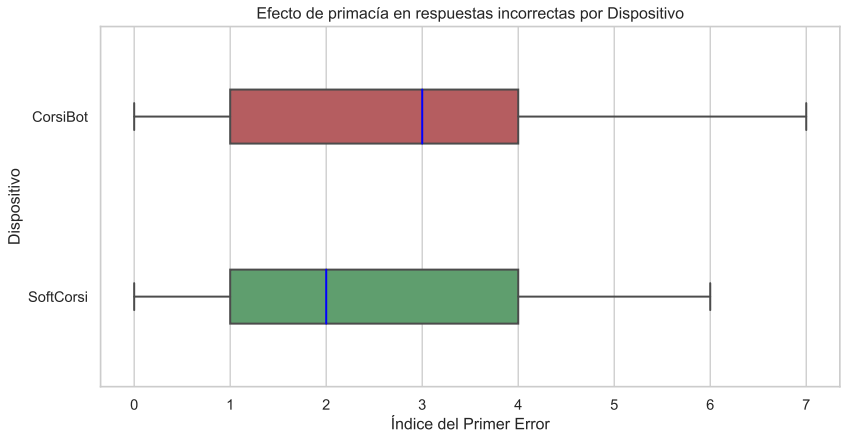

Fig. 3.14. Efecto de Primacía por Dispositivo: Para los errores con igual longitud que la secuencia original, con el CorsiBot se obtuvo un efecto de primacía con un Índice del Primer Error (Mdn: 3.0, SD: 1.74), mientras que en el SoftCorsi el Índice del Primer Error fue de (Mdn: 2.0, SD: 1.71).

Tal como se mencionó en el Plan de análisis el efecto de primacía refiere a la capacidad que tuvieron las personas para recordar de forma eficiente los primeros elementos presentados. El efecto de primacía no mostró diferencias estadísticamente significativas entre Dispositivos. Sin embargo, tal como puede observarse en la Fig. 3.14, a nivel descriptivo se observa una leve diferencia: con el CorsiBot los valores del Índice del Primer Error fueron de (Mdn: 3.0, SD: 1.74) mientras que resultaron de (Mdn: 2.0, SD: 1.71) con el SoftCorsi.

Efecto de Primacía por Protocolo y Dispositivo

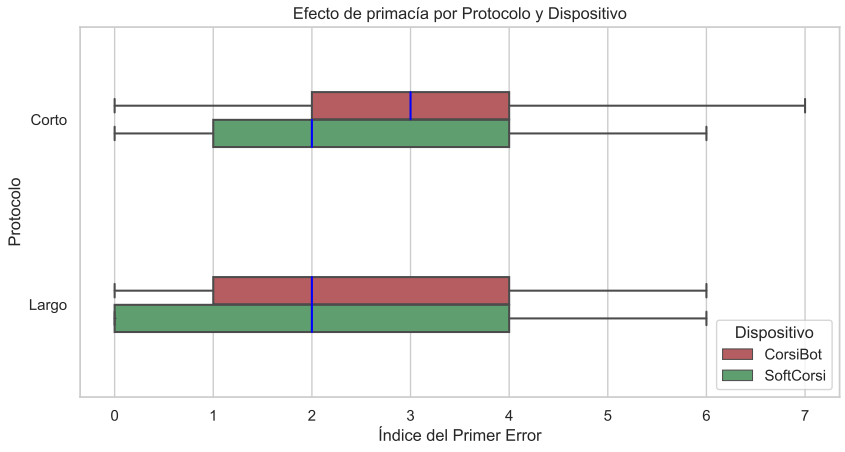

Fig. 3.15. Efecto de Primacía por Protocolo y Dispositivo: Al discriminar también por Protocolo, se observa que para las respuestas incorrectas con longitud esperada, mientras que para el Protocolo Largo el efecto de primacía fue muy similar entre el CorsiBot con un IPE (Mdn: 2.0, SD: 1.76) y el SoftCorsi (Mdn: 2.0, SD: 1.72); para el Protocolo Corto hubo una leve diferencia, con un mayor efecto de Primacía con el CorsiBot con un IPE (Mdn: 3.0, SD: 2.86) que con el SoftCorsi con un IPE (Mdn: 2.0, SD: 1.71).

En la figura 3.15 se muestra la comparación del Índice del Primer Error (IPE) de las respuestas incorrectas que mantuvieron igual longitud que la secuencia presentada, entre los Dispositivos SoftCorsi y CorsiBot para ambos Protocolos. Los análisis estadísticos no mostraron diferencias estadísticamente significativas entre Protocolos y Dispositivos para esta variable. Sin embargo, los valores descriptivos mostraron la tendencia a un mayor efecto de primacía con el Protocolo Corto en el CorsiBot respecto del SoftCorsi. Es decir, para el Protocolo corto se observa que las personas evaluadas tuvieron una leve tendencia a recordar de mejor manera los primeros Estímulos con el CorsiBot que con el SoftCorsi.

Efecto de Recencia

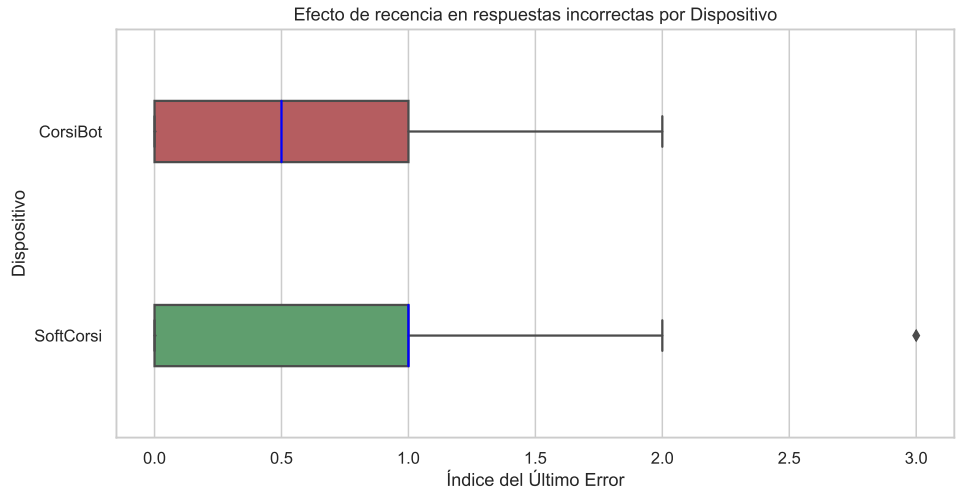

Fig. 3.16. Efecto de Recencia por Dispositivo: Para los errores con igual longitud que la Secuencia presentada, se observa que con el SoftCorsi tuvieron un mayor efecto de recencia, con un Índice de Último Error (Mdn: 1.0, SD: 0.75) con el SoftCorsi y de (Mdn: 0.5 , SD: 0.72) con el CorsiBot.

El efecto de recencia no mostró diferencias estadísticamente significativas entre dispositivos. Sin embargo, como puede observarse en la Fig. 3.16 el efecto de recencia habría mostrado valores levemente mayores a nivel descriptivo con el SoftCorsi que con el CorsiBot. Los valores del Índice del Último Error fueron de (Mdn: 0.5, SD: 0.72) con el CorsiBot y de (Mdn: 1.0, SD: 0.75) con el SoftCorsi.

Efecto de Recencia por Protocolo y Dispositivo

Los análisis mostraron que no hay diferencias estadísticamente significativas en el índice del último error en la comparación entre Dispositivos y Protocolos, tal como puede observarse en la Fig. 3.17.

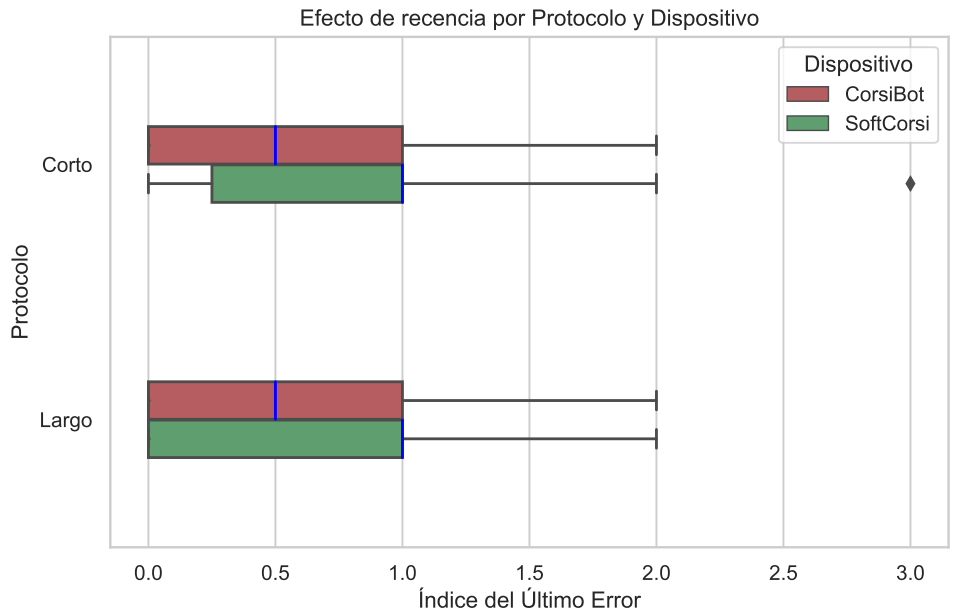

Fig. 3.17. Efecto de Recencia por Dispositivo y Protocolo: Al discriminar también por Protocolo, para las respuestas incorrectas con igual longitud que la secuencia presentada, se observan valores similares para el efecto de recencia. Para el Protocolo Corto se obtuvo un Índice de Último error (Mdn: 0.5, SD: 0.75) con el CorsiBot y (Mdn: 1.0, SD: 0.86) con el SoftCorsi, mientras que para el Protocolo Largo se obtuvo un Índice de Último Error de (Mdn: 0.5, SD: 0.70) con el CorsiBot y de (Mdn: 1.0, SD: 0.86) con el SoftCorsi.

Tab. 3.5: Comparación entre Dispositivos de las variables de error para los Protocolos

	Ambos Protocolos			Protocolo Corto			Protocolo Largo		
	U	p	r	U	p	r	U	p	r
Dist. de Levenshtein(*)	419.5	.65	.05	96.5	.50	-.12	110.5	.95	.01
Error de Camino	528	.23	.15	104	.73	-.66	**160.5**	**.04**	**.37**
Índice Primer Error	330.5	.07	-.023	78	.15	-.26	77.5	.14	-.26
Índice Último Error	570	.05	.24	125	.60	.09	132	.40	.15

Nota. U = Estadístico del test U-Mann – Whitney, r = Tamaño del efecto. (*) Método extendido.

Tiempo de respuesta por botón presionado

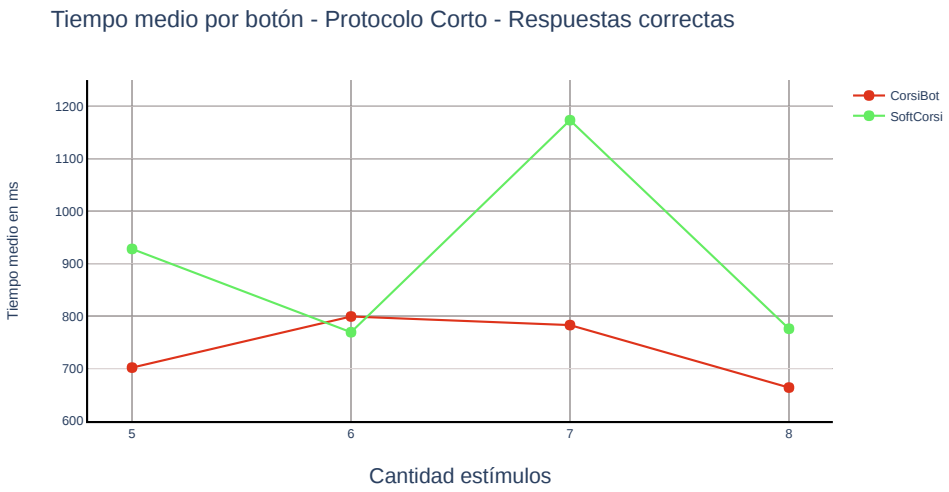

Fig. 3.18. Tiempo medio de respuesta por botón para las Secuencias correctas del Protocolo Corto. Se evidencia en general un menor tiempo con el CorsiBot, a excepción de las secuencias de 6 Estímulos donde el tiempo medio es levente menor con el SoftCorsi. Luego, se observa un pico de tiempo medio con el SoftCorsi para las respuestas con 7 Estímulos.

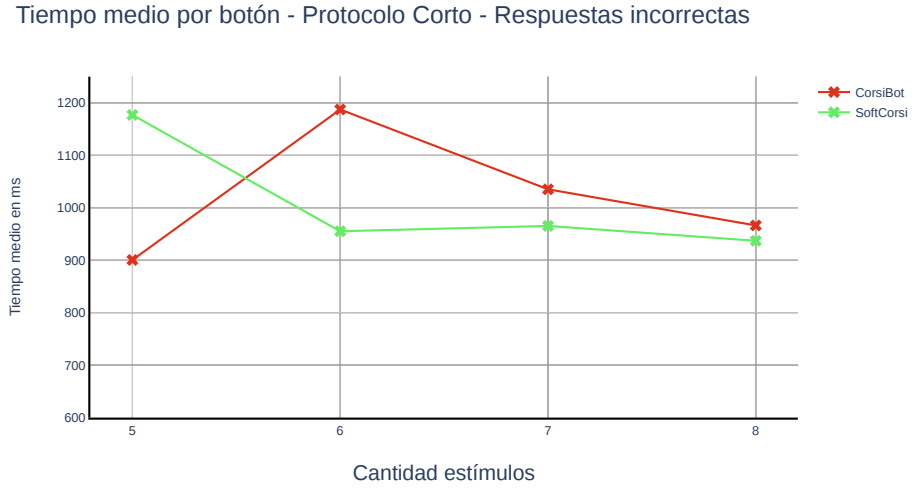

Fig. 3.19. Tiempo medio de respuesta por botón para las secuencias incorrectas del Protocolo Corto. Se observa en general un mayor tiempo de respuesta con el CorsiBot, a excepción de las respuestas con 5 Estímulos donde el tiempo medio de respuesta por botón fue bastante mayor con el SoftCorsi. También se observa que con el CorsiBot hubo un pico de tiempo medio por botón para las respuestas incorrectas de las secuencias con 6 Estímulos, similar a las respuestas de 5 Estímulos realizadas con el SoftCorsi.

Se compararon los tiempos de las respuestas correctas e incorrectas para ambos Dispositivos y Protocolos con las secuencias de longitud de 5 a 8 (con secuencias de longitud

menores a 5 prácticamente no se produjeron errores), es decir secuencias de 5 a 8 estímulos. Para esta evaluación también se trabajó, en las respuestas incorrectas, con las que tuvieron igual longitud que la secuencia original, es decir, la misma cantidad de botones presionados en las respuestas incorrectas que los estímulos presentados en las secuencias originales.

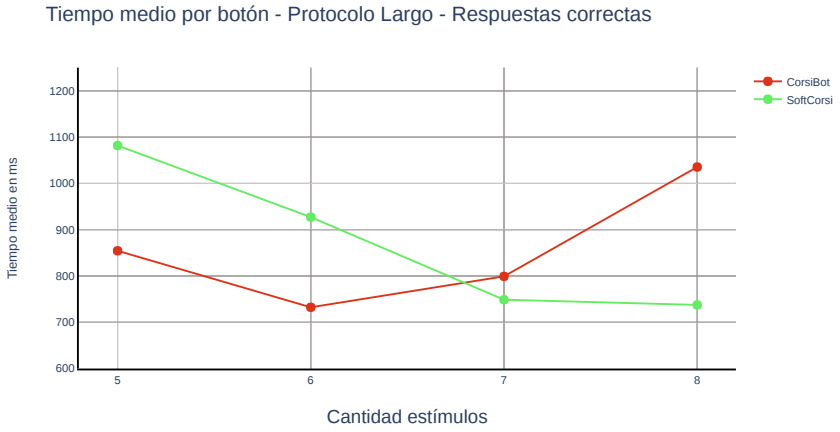

Fig. 3.20. Tiempo medio de respuesta por botón para las secuencias correctas del Protocolo Largo. Mientras que con el SoftCorsi las respuestas correctas tuvieron un menor tiempo medio por botón a medida que se incrementó la cantidad de Estímulos de las Secuencias, con el CorsiBot sucedió casi lo opuesto: A partir de las Secuencias con 6 Estímulos el tiempo medio de respuesta por botón se incrementó a medida que se incrementó la longitud de la Secuencia, en cuanto a la cantidad de Estímulos.

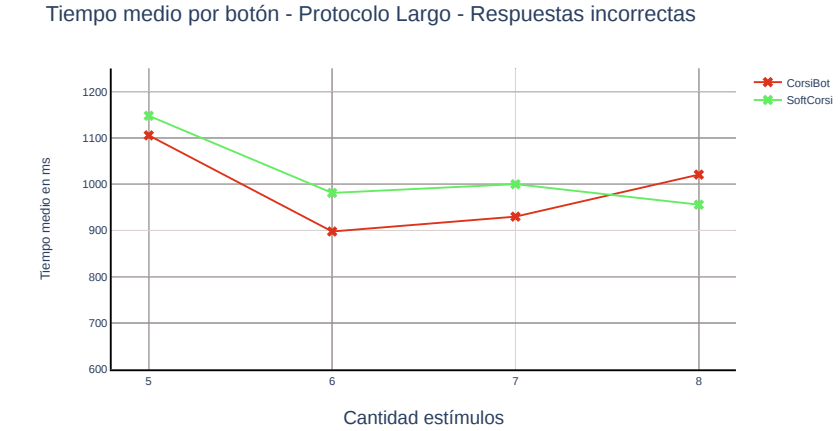

Fig. 3.21. Tiempo medio de respuesta por botón para las secuencias incorrectas del Protocolo Largo. En este escenario se evidencia que hubo un tiempo medio de respuesta por botón bastante similar para las secuencias de 5 a 7 Estímulos, con un leve menor tiempo en el CorsiBot. Sin embargo, para las respuestas a las secuencias de longitud 8, el tiempo fue menor con el SoftCorsi.

En general, los datos descriptivos (Fig. 3.18, Fig. 3.19, Fig. 3.20 y Fig. 3.21) muestran que la media de los tiempos de respuesta por botón en las Secuencias de longitud 5 a 8 respondieron a los siguientes patrones:

- Las respuestas correctas tomaron en general menor tiempo medio por botón que las incorrectas. Además fue menor el tiempo medio de respuesta con el CorsiBot que con el SoftCorsi, para ambos Protocolos. Particularmente, el SoftCorsi mostró mayores tiempos de respuesta en las Secuencias con 5 y 7 Estímulos.
- En las respuestas incorrectas, se observa en general un menor tiempo medio por botón con el SoftCorsi para el Protocolo Corto, mientras que para el Protocolo Largo no se encontraron diferencias significativas.

Secuencias con mayor cantidad de errores

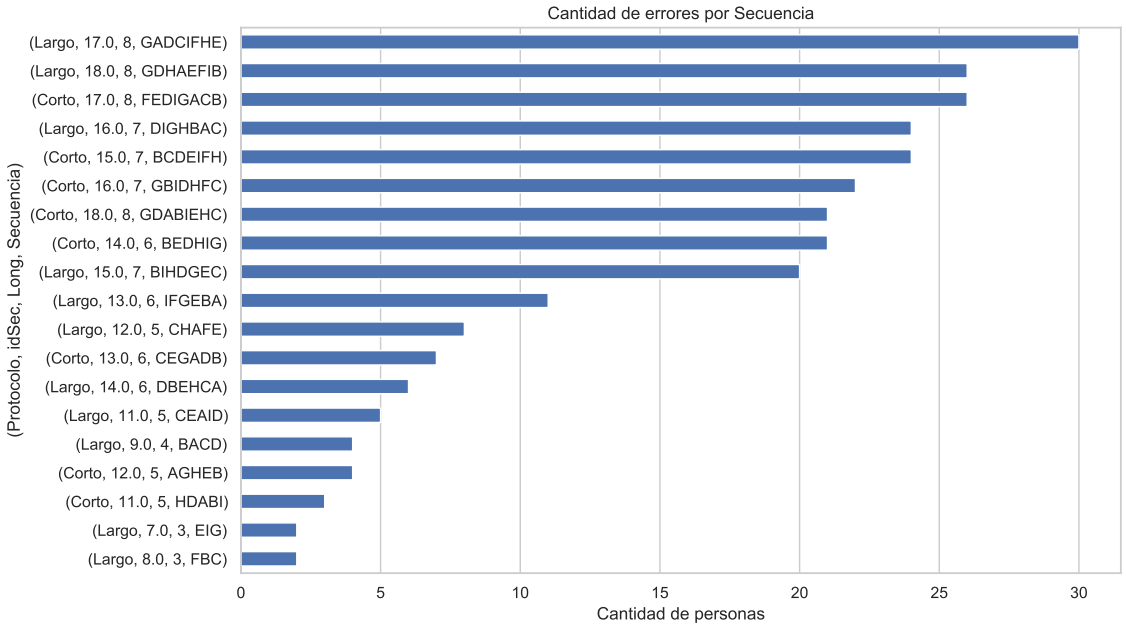

Fig. 3.22. Secuencias con su cantidad de errores. Sin discriminar por Dispositivo, se muestran ordenadas las Secuencias con la cantidad de errores cometidos por las personas evaluadas. Se destaca la Secuencia *GADCIFHE* del Protocolo Largo para la cual ninguna de las 30 personas evaluadas pudo responder correctamente.

Con el objetivo de analizar si existieron algunas Secuencias que tuvieron una mayor cantidad de errores con un Dispositivo específico, se ordenó a las mismas por cantidad de errores que cometieron las personas evaluadas, tal como puede observarse en la Fig. 3.22. Es así que la Secuencia GADCIFHE del Protocolo Largo cuya longitud es de 8 Estímulos, fue la Secuencia con más cantidad de errores siendo que ninguna de las 30 personas evaluadas ha podido responderla correctamente, con ninguno de los dos Dispositivos.

Secuencias con mayor cantidad de errores por Dispositivo

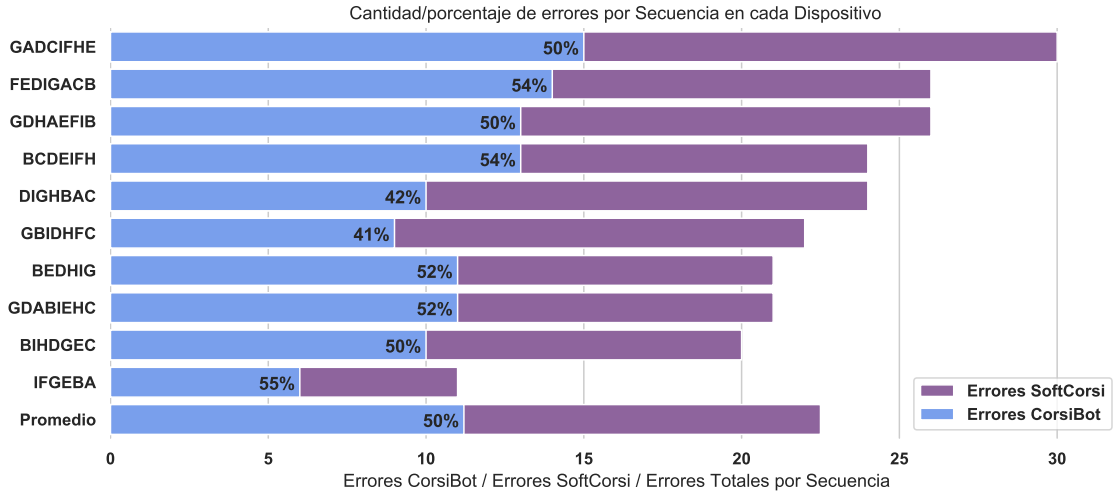

Fig. 3.23. Secuencias con mayor cantidad de errores por Dispositivo: Tomando las secuencias para las cuales 10 o más personas evaluadas respondieron de forma incorrecta, se puede observar la distribución que tuvieron los errores con cada Dispositivo, con una distribución de errores promedio igual para ambos Dispositivos. Se destacan dos secuencias en particular, la Secuencia *DIGHBAC* y la Secuencia *GBIDHFC* para las cuales el porcentaje de errores cometidos con el CorsiBot fue sensiblemente menor que las cometidas con el SoftCorsi, con porcentajes de 42 % y 41 % respectivamente.

Una vez identificadas las Secuencias con mayor cantidad de errores se conservó para la comparación entre Dispositivos aquellas para las cuales al menos 10 personas las respondieron de manera incorrecta. Tal como puede observarse en la Fig. 3.23, en líneas generales, la distribución de los errores en los Dispositivos es similar, aunque se destacan la Secuencia DIGHBAC del Protocolo Largo (con longitud 7) y la Secuencia GBIDHFC del Protocolo Corto (con longitud 7) con una diferencia mayor en la cantidad de errores, cometiéndose un 16 % y 18 % más de errores con el SoftCorsi que con el CorsiBot.

Intersecciones

Tal como se mencionó en la sección Diseño de los Protocolos, según estudios preliminares [11, 12], uno de los factores que influyen en la dificultad de una Secuencia es la presencia de intersecciones en la misma.

Secuencia	**#Intersec.**	**Total Errores**	**Errores CorsiBot**	**Errores SoftCorsi**	**Diferencia**
GADCIFHE	0	30	15	15	0
FEDIGACB	0	26	14	12	2
GDHAEFIB	2	26	13	13	0
BCDEIFH	0	24	13	11	2
DIGHBAC	**3**	24	10	14	**4**
GBIDHFC	**2**	22	9	13	**4**
BEDHIG	1	21	11	10	1
GDABIEHC	3	21	11	10	1
BIHDGEC	2	20	10	10	0
IFGEBA	0	11	6	5	1

Tab. 3.6: Secuencias con cantidad de intersecciones y errores cometidos con CorsiBot y SoftCorsi: Se observa que las secuencias destacadas en el apartado anterior (DIGHBAC, GBIDHFC) por ser las que tuvieron mayor diferencia de errores cometidos con los Dispositivos (4 menos con el CorsiBot) cuentan con 3 y 2 intersecciones, respectivamente. Otras secuencias con misma cantidad de intersecciones, no tuvieron diferencias significativas en los errores cometidos con los Dispositivos.

Si se compara la cantidad de errores cometidos por las personas evaluadas con la cantidad de intersecciones de las Secuencias, tal como se muestra en la tabla 3.6, puede observarse que las dos Secuencias distinguidas en el apartado anterior, DIGHBAC y GBIDHFC, tienen 3 y 2 intersecciones respectivamente y son las que mayor diferencia en la cantidad de errores tuvieron, con mayor cantidad de errores en el SoftCorsi que en el CorsiBot. En el resto de los casos de Secuencias que presentan un número similar de intersecciones no se observan diferencias sustanciales entre la cantidad de errores cometidos con los distintos Dispositivos. Es pertinente destacar que las dos Secuencias con mayor cantidad de errores no tienen intersecciones.

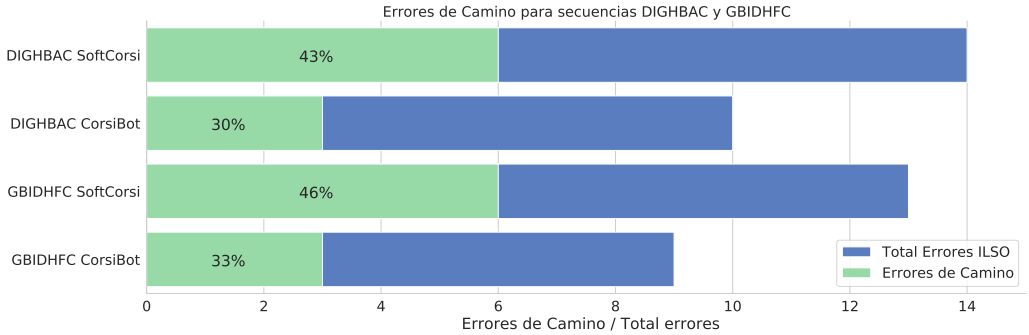

Fig. 3.24. Errores de Camino y errores totales de las Secuencias que presentaron mayor diferencia de errores entre Dispositivos. Para las Secuencias con mayor diferencia en la cantidad de errores (DIGHBAC y GBIDHFC) se evidencia una menor proporción de Errores de Camino con el CorsiBot: 30 % y 33 % respectivamente; mientras que con el SoftCorsi la proporción de Errores de camino fue mayor: 43 % y 46 % respectivamente.

Por último, se verificaron cuántos de los errores cometidos en estas dos Secuencias particulares fueron Errores de Camino para comprobar si la diferencia en la cantidad de errores con ambos Dispositivos mencionada anteriormente fue producida con ese tipo de error. Como puede observarse en la Fig. 3.24 con el CorsiBot hubo una menor proporción de Errores de Camino que con el SoftCorsi, para ambas Secuencias.

3.5. Encuestas a personas evaluadas

Tal como se detalló en la sección Metodología de experimentación a cada persona evaluada se le solicitó que, al finalizar cada Toma de Datos, completara un cuestionario del tipo NASA-TLX [25].

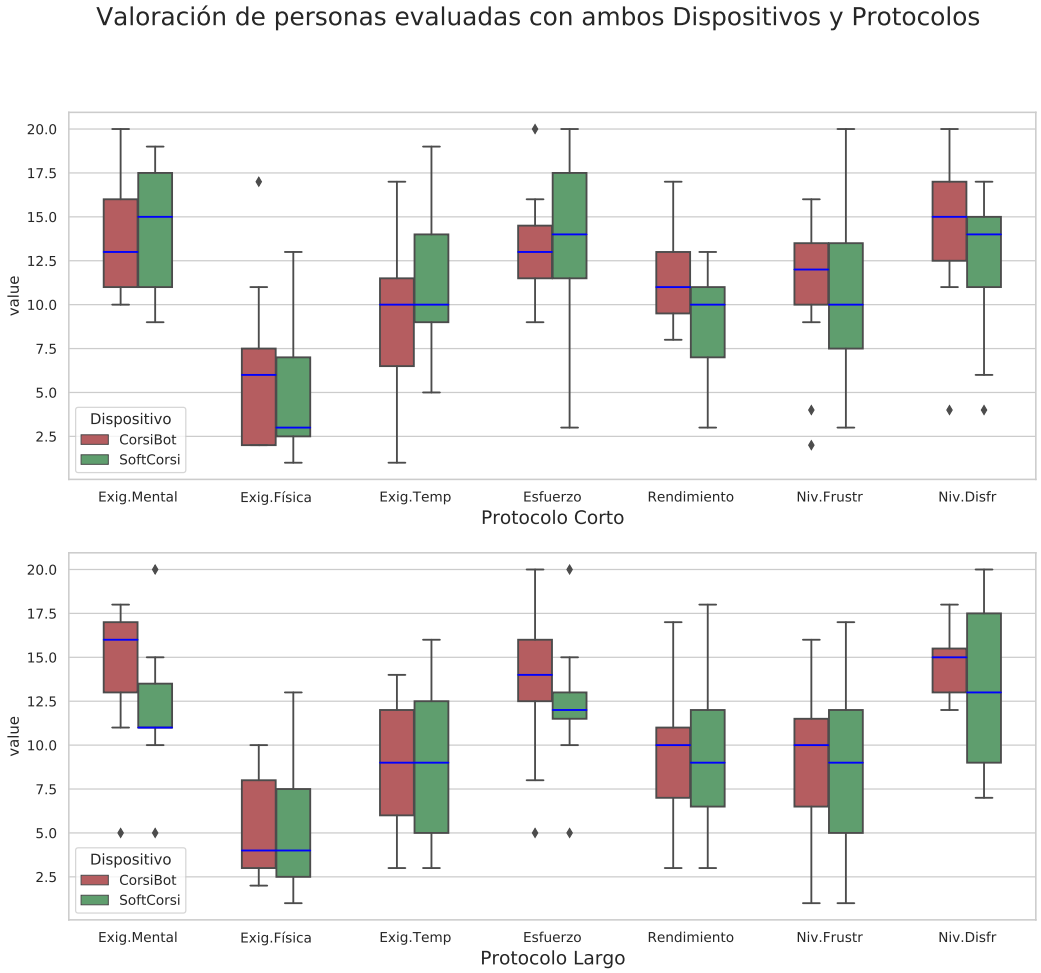

Fig. 3.25. Valoración de las personas evaluadas respecto de la experiencia con cada Dispositivo y Protocolo. En general los valores son similares. Las únicas que se diferencian son, para el Protocolo Largo, las de **Exigencia Mental** con (Mdn: 16.0, SD: 3.48) para el CorsiBot y (Mdn: 11.0, SD: 3.22) para el SoftCorsi y la de **Esfuerzo** con (Mdn: 14.0, SD: 3.78) para el CorsiBot y (Mdn: 12, SD: 3.12) para el SoftCorsi.

Los análisis de las respuestas de las personas evaluadas con el cuestionario NASA-TLX mostraron que se realizó una valoración similar de los Dispositivos en casi la totalidad de las variables. Sin embargo, se hallaron diferencias estadísticamente significativas en la valoración de la exigencia mental requerida por el Protocolo Largo. En particular, se reportó que el CorsiBot implicó una mayor exigencia mental que el SoftCorsi (Tab. 3.7).

Por otro lado, los niveles de esfuerzo mostraron la misma tendencia a nivel descriptivo, aunque la diferencia no fue estadísticamente significativa (Fig. 3.25).

Tab. 3.7: Comparación de la valoración de la tarea que realizaron las personas con cada Dispositivo

	Ambos Protocolos			Protocolo Corto			Protocolo Largo		
	U	p	r	U	p	r	U	p	r
Exigencia Mental	372.5	.24	-.15	129.5	.48	.13	**55.5**	**.01**	**-.43**
Exigencia física	403.5	.49	-.08	102	.67	-.08	105.5	.78	-.05
Exigencia temporal	500.5	.45	.09	135.5	.34	.17	111.5	.98	-.007
Esfuerzo	433	.80	-.03	137	.31	.18	75.5	.12	-.28
Rendimiento	373	.25	-.14	69.5	.07	-.32	113.5	.98	.007
Nivel de frustración	413.5	.59	-.06	95.5	.49	-.12	109	.90	-.02
Nivel de disfrute	355	.15	-.18	89	.33	-.17	89	.33	-.17

Nota. U = Estadístico del test U-Mann – Whitney, r = Tamaño del efecto.

4. DISCUSIÓN Y CONCLUSIONES

Conclusiones generales

El desarrollo de versiones computarizadas del Bloques de Corsi, si bien ha significado un gran avance al automatizar los procesos de presentación de secuencias y registro de las respuestas, presenta un debate acerca de cómo los dispositivos con pantalla impactan sobre la memoria de trabajo de las personas cuando realizan la tarea, respecto de las versiones físicas [15, 16].

Se ha desarrollado el CorsiBot: una versión física y automatizada del Bloques de Corsi que cumple con los requerimientos solicitados y detallados en la sección Requerimientos. El mismo ha funcionado de manera satisfactoria para la experimentación con personas. Los planos y códigos fuente del dispositivo, se encuentran abiertos para su uso a través del siguiente enlace:

https://github.com/maxiwebs/CorsiBot.

El Dispositivo ha permitido realizar evaluaciones y registrar de forma eficiente las respuestas de distintas personas evaluadas, permitiendo luego la visualización de los resultados junto con la exportación de los datos para su posterior análisis.

En el experimento realizado, se buscó comparar el desempeño de una muestra de personas adultas entre una versión computarizada y el dispositivo desarrollado para la evaluación de procesos de memoria de trabajo. El análisis mostró valores de desempeño similares en ambos dispositivos para la mayor parte de las variables de interés. Específicamente, no se encontraron diferencias significativas entre los dispositivos ni para la cantidad de respuestas correctas ni para la máxima longitud de secuencia respondida correctamente, tal como se observa en la sección Desempeño.

Sin embargo, la comparación de los errores cometidos por las personas evaluadas evidenció algunas diferencias de acuerdo al dispositivo utilizado. En forma específica, los tipos de errores cometidos por los participantes difirieron de acuerdo al dispositivo utilizado.

En primer lugar, la cantidad de errores debido a que la persona evaluada presionó una cantidad de botones diferente al esperado (mayor o menor cantidad, dependiendo el caso), fue mayor con el CorsiBot que con el SoftCorsi. Esta diferencia podría deberse a errores de registro causados porque las personas evaluadas no presionaron de forma correcta los botones del dispositivo o sí lo hicieron pero consideraron que no lo habían hecho y lo presionaron nuevamente. Estos errores fueron observados en el momento de tomar la evaluación con el CorsiBot y pueden deducirse, en algunos casos, en los registros de los experimentos.

En segundo lugar, otros tipos de errores sucedieron cuando los ensayos fueron respondidos incorrectamente pero respetando la longitud de la secuencia presentada. En este caso, se encontraron diferencias significativas en la cantidad de Errores de Camino cometidos para cada dispositivo. Este tipo de errores implicó que las personas evaluadas presionaran los botones correctos de la secuencia a recordar, pero en un orden diferente al de presentación. Los resultados mostraron que, cuando los participantes utilizaban el SoftCorsi, cometían más Errores de Camino que con el CorsiBot.

Los errores de camino podrían ser explicados por procesos asociados a la codificación de la secuencia. El estudio de De Lillo (2016) [29] apoya la hipótesis de que las personas construyen imágenes mentales del camino realizado por la secuencia a recordar. Esta estrategia

funcionaría mejor con caminos estructurados y con menos intersecciones, formando figuras geométricas familiares. Los errores de camino reportados en nuestro estudio podrían estar asociados a una codificación de las secuencias en un orden diferente al presentado, pero consistentes con una simplificación de la figura trazada por la secuencia. Esto concuerda con lo reportado en estudios de Orsini y colegas (2001, 2004) [11, 30] donde se refiere a la simplificación como el proceso por el cual las personas evaluadas recuerdan la secuencia presentada en un orden diferente al otorgado, evitando las intersecciones presentes en la secuencia original. Sin embargo, el tamaño de la muestra utilizada, no permitió profundizar en análisis que puedan determinar el origen de los errores de camino.

Por otro lado, estos tipos de errores fueron menores en el caso de las personas evaluadas que utilizaron el CorsiBot. De acuerdo a nuestro conocimiento, entre los estudios que compararon el desempeño en modalidad manual y digital del Corsi, este es el primer estudio que encontró diferencias en los tipos de errores cometidos por las personas evaluadas. Si bien, al igual que en la literatura del tema, las personas evaluadas mostraron una proporción similar de ensayos correctos, presentaron errores cualitativamente diferentes. Estos resultados podrían ser consecuencia del *Mode Effect* mencionado en los Antecedentes, resaltando la necesidad de continuar el estudio de las consecuencias de las distintas modalidades de intervención utilizadas en la práctica clínica.

El trabajo de Perrochon y colegas (2014) [43], constituye un posible ejemplo del impacto del *Mode Effect* en los tipos de errores en una tarea con demanda de memoria de trabajo espacial. Allí, Perrochon y colegas (2014) compararon el desempeño de una muestra de personas adultas mayores con Deterioro Cognitivo Leve (DCL) en una versión SoftCorsi y en una versión del *Walking Corsi*. Sus resultados mostraron que las personas con DCL evaluadas mostraron la misma capacidad de memoria que las personas sin DCL evaluadas, en términos de ensayos correctos. Sin embargo, ambos grupos de participantes mostraron distintos tipos de errores, mostrando que quienes no tenian DCL tendían a presentar errores de un bloque, mientras que quienes tenían DLC erraban en más de dos. Próximos trabajos podrían profundizar en estos resultados con una mayor cantidad de casos para determinar con mayor precisión el tipo de errores que son afectados por el *Mode Effect* y las implicancias que ello podría tener para la práctica clínica.

Más allá de las diferencias significativas en los Errores de Camino, se han utilizado otras métricas para comparar los errores de las personas evaluadas, algunas desarrolladas especialmente para este trabajo.

Para la métrica del método extendido de Distancia de Levenshtein, evaluando sólo las respuestas incorrectas con igual longitud que la secuencia original, no se hallaron diferencias significativas entre los Dispositivos.

Otras de las métricas de error utilizadas fueron las variables de Índice del Primer Error e Índice del Último Error, destinadas a medir efectos de primacía y recencia, respectivamente. Si bien se hallaron algunas diferencias a nivel descriptivo, los análisis no mostraron diferencias estadísticamente significativas para estas variables, mostrando que los efectos de primacía y recencia no se verían modulados por el dispositivo utilizado.

Luego de analizar los errores cometidos por las personas evaluadas, se procedió a analizar los tiempos de respuesta tanto para los ensayos correctos como los incorrectos. Debido al tamaño de muestra obtenido no se realizaron análisis de significación estadística para la comparación de las variables de tiempo. Por lo tanto el análisis realizado tuvo un carácter descriptivo limitando la posibilidad de generalizar las conclusiones. En general se obtuvo un menor tiempo de respuesta para las secuencias correctas que para las incorrectas. Estos

tiempos más lentos en las respuestas incorrectas podrían indicar un mayor requerimiento de recursos cognitivos por parte de las personas evaluadas dado tanto por la dificultad de la secuencia presentada, como por los requerimientos del dispositivo utilizado.

Se verificó en general menor tiempo medio por botón para las respuestas correctas con el CorsiBot para ambos protocolos. En cuanto a las respuestas incorrectas, se verificó un menor tiempo medio de respuesta por botón con el SoftCorsi para el Protocolo Corto, mientras que para el Protocolo Largo el tiempo medio de respuesta por botón fue menor con el CorsiBot.

Con la variable de tiempos, se encontró una limitación en el registro de tiempos del CorsiBot, al inicio del registro de los botones de la evaluación, lo que produjo la imposibilidad de analizar un factor de interés como lo era el Tiempo de Preparación para la comparación entre Dispositivos. Quedará para el futuro corregir este error y profundizar el análisis de tiempos de respuesta.

Respecto de las valoraciones que realizaron las personas evaluadas de su experiencia con los Dispositivos se puede decir que en general son bastante similares, a excepción de la valoración sobre la Exigencia Mental y el Esfuerzo de las personas que realizaron el Protocolo Largo con el CorsiBot con valoraciones superiores respecto de las que fueron evaluadas con el SoftCorsi. Si bien esta valoración no está correlacionada con el desempeño que tuvieron en general, (de hecho su consideración sobre rendimiento fue similar para ambos dispositivos) se podría profundizar el análisis tomando un mayor cantidad de casos y despejando la variable de haber realizado dos Tomas de Datos, lo cual podría haber influido en esta valoración superior cuando realizaron el Protocolo Largo con el CorsiBot luego de haber realizado el Protocolo Corto con el SoftCorsi.

Si bien la muestra de 30 personas evaluadas fue algo acotada, del análisis de los experimentos realizados surgieron algunas variables en la comparación con la versión SoftCorsi que presentan nuevos interrogantes y abren la posibilidad de seguir experimentando para evaluar el impacto de las distintas formas de suministrar el test de Bloques de Corsi en relación a lo denominado como el *Mode Effect* [16], mencionados en la sección Antecedentes.

El CorsiBot quedará a disposición de los equipos de investigación que necesiten este dispositivo para realizar sus experimentos, asistiéndolos en su configuración y uso. Con el compromiso de seguir evaluando el funcionamiento y realizando mejoras que surjan de su uso.

Como trabajo adicional a la construcción del CorsiBot, se han desarrollado algunos algoritmos que podrán aportar en las sucesivas investigaciones que utilizan el Test de Bloques de Corsi. Por una parte, los algoritmos y métricas que permitieron la construcción de los dos protocolos utilizados, tal como se detalla en la sección Construcción de Protocolos. Por otra parte, las métrica utilizada para la comparación de los errores, para la cual se ha realizado una modificación al algoritmo de la Distancia de Levenshtein, tal como se detalla en el apéndice Distancia de Levenshtein.

Para concluir, en esta Tesis construimos un dispositivo automatizado para la evaluación de la prueba Bloques de Corsi. Este esfuerzo va en consonancia con el proceso que está viviendo el mundo, en el cual la tecnología y los dispositivos electrónicos se encuentran cada vez más presentes, muchas veces facilitando y agilizando la vida, y que nos exigen poder evaluar su impacto de manera crítica. De esta manera, se materializó un esfuerzo interdisciplinario que permite indagar sobre procesos de resolución de la tarea, antes ocultos, gracias a la factibilidad de realizar registros de gran precisión. Este aporte

se encuentra disponible bajo licencia libre para su uso, reproducción y mejora para toda la comunidad científica con el espíritu que la Ciencia y la Tecnología deben ponerse al servicio de las necesidades de los pueblos.

4.1. Trabajos Futuros

Como trabajos de mejoras hacia el Dispositivo desarrollado en la presente Tesis se proponen algunas prioritarias y otras más deseables.

En primer lugar sería importante corregir el registro de tiempo de inicio de respuesta en el CorsiBot, puesto que sólo se están registrando los valores relativos y esto imposibilita medir de forma certera el tiempo de preparación previo a presionar el primer botón de cada respuesta. A su vez, sería importante realizar una prueba más exhaustiva de los botones del Dispositivo para evaluar, si fuera necesario, cambiar los pulsadores por otros con mayor sensibilidad o agregar sensores que permitan evitar los posibles errores que las personas cometerían por no presionar correctamente los botones. Si bien este problema se detectó en muy pocas ocasiones (a lo sumo en 15 sobre 420 respuestas lo que representaría el 3.6 % de los casos) no habría que descartar este tipo de chequeos, sobre todo si va a utilizarse con niños/as.

Otra de las mejoras que se podrían realizar, es la de independizar la conexión de entrada que captura los botones rojos de confirmación (en la presente implementación no se discrimina) para que pueda registrarse si han confirmado la secuencia con el botón izquierdo o derecho y, por ejemplo, relacionarlos con cuál fue el último botón presionado, si era cercano o no, si la persona que respondió la evaluación era zurda o diestra, entre otros.

Un detalle a mejorar en el gabinete del CorsiBot es el de incluir en el diseño una manera de identificar de forma clara la ubicación de la persona que opera el sistema y de la persona que va a realizar la evaluación.

En cuanto a mejoras en el funcionamiento del dispositivo podría contemplarse el uso de un protocolo para Internet de las Cosas como por ejemplo del Protocolo MQTT [44]. Esto requeriría grandes cambios en el desarrollo actual pero mejoraría la sincronización entre los eventos de respuesta y detección de cambios, sin la necesidad de tener que consultar de forma periódica a la base de datos.

En cuanto a mejoras en la interfaz web, podrían adoptarse tecnologías que permiten una mayor versatilidad para la visualización de los eventos, de forma tal que permita en tiempo real a la persona que está administrando el experimento visualizar las respuestas de la persona que está siendo evaluada, pudiendo detectar si han cometido algún error, sin tener que esperar a la confirmación final por parte del participante. Ejemplos de estas tecnologías son los Frameworks Angular.js, Node.js o, React.js. A su vez, podrían mejorarse con estas herramientas la visualización de algunas resoluciones de pantallas para las cuales el framework Django y su biblioteca responsive han presentado problemas. También podría incluirse un módulo de estadísticas para un mejor procesamiento y visualización de los experimentos realizados.

En cuanto a la administración de los experimentos, podrían incluirse variantes que detecten el desempeño de la persona que está siendo evaluada, para modificar de forma dinámica la siguiente secuencia a presentar, adaptándose a los distintos rendimientos que pudieran tener las diferentes personas.

Bibliografía

[1] Michael Barr. Michael barr's embedded systems glossary. *Netrino. June*, 2, 2003.

[2] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[3] Jerry Landt and Barbara Catlin. Shrouds of time: The history of rfid. *AIM inc*, 2001.

[4] Pedro Pinheiro-Chagas, Dror Dotan, Manuela Piazza, and Stanislas Dehaene. Finger tracking reveals the covert stages of mental arithmetic. *Open Mind*, 1(1):30–41, 2017.

[5] Dror Dotan, Pedro Pinheiro-Chagas, Fosca Al Roumi, and Stanislas Dehaene. Track it to crack it: Dissecting processing stages with finger tracking. *Trends in cognitive sciences*, 23(12):1058–1070, 2019.

[6] Luis Von Ahn. Games With A Purpose (GWAP). Verbosity. `http://www.gwap.com/gwap/gamesPreview/verbosity/`.

[7] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, 2004.

[8] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[9] Luis Von Ahn. Games with a purpose. *Computer*, 39(6):92–94, 2006.

[10] Philip M Corsi. *Human memory and the medial temporal region of the brain.* PhD thesis, ProQuest Information & Learning, 1973.

[11] Arturo Orsini, M Pasquadibisceglie, L Picone, and R Tortora. Factors which influence the difficulty of the spatial path in corsi's block-tapping test. *Perceptual and Motor skills*, 92(3):732–738, 2001.

[12] Robyn M. Busch, Kathleen Farrell, Krista Lisdahl-Medina, and Robert Krikorian. Corsi block-tapping task performance as a function of path configuration. *Journal of Clinical and Experimental Neuropsychology*, 27(1):127–134, 2005.

[13] Sarita J Robinson and Gayle Brewer. Performance on the traditional and the touch screen, tablet versions of the corsi block and the tower of hanoi tasks. *Computers in Human Behavior*, 60:29–34, 2016.

[14] Riccardo Brunetti, Claudia Del Gatto, and Franco Delogu. ecorsi: implementation and testing of the corsi block-tapping task for digital tablets. *Frontiers in psychology*, 5:939, 2014.

[15] Rachel Carpenter and Tracy Alloway. Computer versus paper-based testing: Are they equivalent when it comes to working memory? *Journal of psychoeducational assessment*, 37(3):382–394, 2019.

[16] Heidi V Leeson. The mode effect: A literature review of human and technological issues in computerized testing. *International Journal of Testing*, 6(1):1–24, 2006.

[17] Roy Clariana and Patricia Wallace. Paper–based versus computer–based assessment: key factors associated with the test mode effect. *British Journal of Educational Technology*, 33(5):593–602, 2002.

[18] Erik Wästlund, Henrik Reinikka, Torsten Norlander, and Trevor Archer. Effects of vdt and paper presentation on consumption and production of information: Psychological and physiological factors. *Computers in human behavior*, 21(2):377–394, 2005.

[19] Daniel K Mayes, Valerie K Sims, and Jefferson M Koonce. Comprehension and workload differences for vdt and paper-based reading. *International Journal of Industrial Ergonomics*, 28(6):367–378, 2001.

[20] Carol Taylor, Irwin Kirsch, Joan Jamieson, and Daniel Eignor. Examining the relationship between computer familiarity and performance on computer-based language tasks. *Language Learning*, 49(2):219–274, 1999.

[21] Guoxing Yu. Effects of presentation mode and computer familiarity on summarization of extended texts. *Language Assessment Quarterly*, 7(2):119–136, 2010.

[22] Siew Lian Chua, Der-Thanq Chen, and Angela FL Wong. Computer anxiety and its correlates: a meta-analysis. *Computers in human behavior*, 15(5):609–623, 1999.

[23] Angus S McDonald. The impact of individual differences on the equivalence of computer-based and paper-and-pencil educational assessments. *Computers & Education*, 39(3):299–312, 2002.

[24] Jan Noyes, Kate Garland, and Liz Robbins. Paper-based versus computer-based assessment: is workload another test mode effect? *British Journal of Educational Technology*, 35(1):111–113, 2004.

[25] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[26] AD Baddeley, GJ Hitch, and RJ Allen. *A multicomponent model of working memory.* Oxford: Oxford University Press, 2020.

[27] Susan J Pickering. The development of visuo-spatial working memory. *Memory*, 9(4-6):423–432, 2001.

[28] Virginie M Patt, Michael L Thomas, Arpi Minassian, Mark A Geyer, Gregory G Brown, and William Perry. Disentangling working memory processes during spatial span assessment: A modeling analysis of preferred eye movement strategies. *Journal of clinical and experimental neuropsychology*, 36(2):186–204, 2014.

[29] Carlo De Lillo, Melissa Kirby, and Daniel Poole. Spatio-temporal structure, path characteristics, and perceptual grouping in immediate serial spatial recall. *Frontiers in psychology*, 7:1686, 2016.

[30] Arturo Orsini, S Simonetta, and MS Marmorato. Corsi's block-tapping test: some characteristics of the spatial path which influence memory. *Perceptual and motor skills*, 98(2):382–388, 2004.

[31] Roy PC Kessels, Martine JE Van Zandvoort, Albert Postma, L Jaap Kappelle, and Edward HF De Haan. The corsi block-tapping task: standardization and normative data. *Applied neuropsychology*, 7(4):252–258, 2000.

[32] André Vandierendonck, Eva Kemps, Maria Chiara Fastame, and Arnaud Szmalec. Working memory components of the corsi blocks task. *British journal of psychology*, 95(1):57–79, 2004.

[33] Sara Siddi, Antonio Preti, Elvira Lara, Gildas Brébion, Regina Vila, Maria Iglesias, Jorge Cuevas-Esteban, Raquel López-Carrilero, Anna Butjosa, and Josep Maria Haro. Comparison of the touch-screen and traditional versions of the corsi block-tapping test in patients with psychosis and healthy controls. *BMC psychiatry*, 20(1):1–10, 2020.

[34] Andrea Röser, Gregor Hardiess, and Hanspeter A Mallot. Modality dependence and intermodal transfer in the corsi spatial sequence task: Screen vs. floor. *Experimental Brain Research*, 234(7):1849–1862, 2016.

[35] Roy E Nelson, Andrew L Dickson, and James H Baños. An automated administration of corsi's block-tapping test. *Perceptual and motor skills*, 91(2):578–580, 2000.

[36] Scott M Crawford, Andrew L Dickson, and James H Baños. A revised spatial serial learning and memory procedure using corsi's block-tapping apparatus. *Perceptual and motor skills*, 91(2):669–674, 2000.

[37] Matías Lopez-Rosenfeld. Repositorio versión softcorsi. `https://github.com/matiaslopez/pygame-corsi`.

[38] Leds rgb. `https://es.wikipedia.org/wiki/Led#RGB`.

[39] Raspberry Pi Foundation. Sistema embebido raspberry pi, modelo 3b+. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[40] Django Software Foundation. Django project. `http://www.https://www.djangoproject.com/`.

[41] V Levenshtein. Levenshtein distance, 1965.

[42] Bennet B Murdock Jr. The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482, 1962.

[43] Anaïck Perrochon, Gilles Kemoun, Benoit Dugué, and Alain Berthoz. Cognitive impairment assessment through visuospatial memory can be performed with a modified walking corsi test using the 'magic carpet'. *Dementia and geriatric cognitive disorders extra*, 4(1):1–13, 2014.

[44] Dr Andy Stanford-Clark and Arlen Nipper. Mqtt is an oasis standard for iot connectivity. `https://mqtt.org///`.

[45] Maximiliano Urso. Repositorio dispositivo corsibot. https://github.com/maxiwebs/CorsiBot.

[46] NumPy community. Librería numpy. https://numpy.org//.

[47] RaspberryConnect.com. Raspberry pi - hotspot/access point dhcpcd method. https://www.raspberryconnect.com/projects/65-raspberrypi-hotspot-accesspoints/168-raspberry-pi-hotspot-access-point-dhcpcd-method.

[48] Ben Croston. Librería rpi.gpio. https://pypi.org/project/RPi.GPIO/.

[49] joan@abyz.me.uk. Librería pigpio. https://pypi.org/project/pigpio/.

[50] Django Software Foundation. Django mvt. https://www.javatpoint.com/django-mvt.

[51] Mark Lavin. Librería django-responsive. https://django-responsive.readthedocs.io/en/latest/.

[52] SQLite Consortium. Sqlite 3. https://www.sqlite.org/index.html.

Apéndice

A. CUESTIONARIO NASA-TLX

A fin de evaluar la percepción de cada persona evaluada, tal como se detalló en la sección Metodología de experimentación, se le solicitó al finalizar cada Toma de Datos que completara un cuestionario del tipo NASA-TLX [25]. El mismo consiste en una valoración numérica con una escala [1,20] sobre las siguientes preguntas:

1. **Exigencia Mental (M)**: *¿Cuánto esfuerzo mental fue necesario? (Pensar, decidir, recordar). ¿Se trata de una tarea fácil o difícil?*
2. **Exigencia Física (F)**: *¿Cuánto esfuerzo físico fue necesario? (Pulsar, mover, arrastrar). ¿Se trata de una tarea fácil ó difícil, lenta o rápida, relajada o cansadora?*
3. **Exigencia Temporal (T)**: *¿Cuánta presión de tiempo sintió, debido al ritmo al cual se sucedían las tareas o los elementos de la tareas?¿Era un ritmo lento y pausado o rápido y frenético?*
4. **Esfuerzo (E)**: *¿En qué medida ha tenido que trabajar (física o mentalmente) para alcanzar su nivel de resultados?*
5. **Rendimiento (Performance) (R)**: *¿Hasta qué punto cree que ha tenido éxito en los objetivos establecidos por el investigador (o por ud. mismo/a)? ¿Cuál es su grado de satisfacción con su nivel de ejecución?*
6. **Nivel de Frustración (Fr)**: *Durante la tarea, ¿En qué medida se ha sentido inseguro/a, desalentado/a, irritado/a, tenso/a) o preocupado/a o, por el contrario, se ha sentido seguro/a, contento/a, relajado/a y satisfecho/a?*
7. **Nivel de Disfrute (D)**: *¿Cuán agradable le resultó la tarea? ¿Le pareció divertida o aburrida?*

El cuestionario se completó de manera digital mediante la plantilla que se muestra en la Fig. A.1

Cuestionario Bloques de Corsi

Nombre: **Edad:**
Zurd@/Diestr@: IZ DE Ambas **Género:**
Nivel de estudios alzanzado: SC SI TC TI UC UI

1. EXIGENCIA MENTAL (M) ¿Cuánto esfuerzo mental fue necesario? (Pensar, decidir, recordar).
¿Se trata de una tarea fácil o difícil?

Baja Alta

2. EXIGENCIA FÍSICA (F) ¿Cuánto esfuerzo físico fue necesario? (Pulsar, mover, arrastrar)
¿Se trata de una tarea fácil ó difícil, lenta o rápida, relajada o cansadora?

Baja Alta

3. EXIGENCIA TEMPORAL (T) ¿Cuánta presión de tiempo sintió, debido al ritmo al cual se sucedían las tareas o los elementos de la tareas?
¿Era un ritmo lento y pausado ó rápido y frenético?

Baja Alta

4. ESFUERZO (E) ¿En qué medida ha tenido que trabajar (física o mentalmente) para alcanzar su nivel de resultados?

Bajo Alto

5. RENDIMIENTO («Performance») (R) ¿Hasta qué punto cree que ha tenido éxito en los objetivos establecidos por el investigador (o por ud. mism@)?
¿Cuál es su grado de sastisfaccion con su nivel de ejecución?

Bueno Malo

6. NIVEL DE FRUSTRACIÓN (Fr) Durante la tarea, ¿En qué medida se ha sentido insegur@, desalentad@, irritad@,
tens@) o preocupad@ o por el contrario, se ha sentido segur@, content@, relajad@ y satisfech@ ?

Bajo Alto

7. NIVEL DE DISFRUTE (D) ¿Cuán agradable te resultó la tarea? ¿Le pareció divertida o aburrida?

Bajo Alto

Protocolo: **Dispositivo:**
MU: IZ DE Ambas

Fig. A.1. Cuestionario tipo NASA-TLX que las personas evaluadas completaron al finalizar cada toma de datos.

B. DISTANCIA DE LEVENSHTEIN

La distancia de Levenshtein [41], es una métrica que permite entre dos palabras establecer cuál es el número mínimo de operaciones requeridas para transformar una en la otra. Las operaciones para una carácter pueden ser: la inserción, la eliminación o la sustitución. Por ejemplo, mientras la distancia de Levenshtein entre las secuencias ABC y ABC es 0 (pues ambas son iguales), entre las secuencias ABC y ABD es 1, entre las secuencias ABC y AEF es 2 y así sucesivamente.

Como las secuencias de estímulos y las respuestas pueden ser almacenadas como palabras, esta métrica fue incluida para el análisis de las respuestas con errores.

La implementación en *Python* puede observarse en el Código 1. Fue necesario introducir algunas modificaciones a la función de Levenshtein original, puesto que nuestro dominio implicaba ciertas restricciones, de manera que la función pueda penalizar los distintos errores de manera diferente a la que lo realizaba la función original. Las modificaciones introducidas consistieron en:

- Penalizar más severamente cuando la secuencia de respuesta tiene una letra que no estaba en la secuencia original. De esta forma, el algoritmo deberá otorgar más distancia para un error proveniente de incluir una letra que no se encontraba en la secuencia original. Luego de reiteradas evaluaciones se decidió otorgar una ponderación extra de valor 3 por cada nueva letra no presente en la secuencia original.

- Penalizar más severamente cuando en la Secuencia de respuesta aparece una letra duplicada puesto que era una condición conocida que las secuencias no podían contener dos veces el mismo estímulo. De esta forma, se penalizará más severamente cuando en la respuesta se encuentren letras repetidas, ya sea de forma consecutiva como en forma alternada. Luego de reiteradas evaluaciones se decidió otorgar una ponderación extra de valor 2 por cada letra duplicada.

Puede observarse el algoritmo con las modificaciones en el Código 2. Se encuentra disponible en el script mayoresDiferencias.py del repositorio [45].

```
def levenshtein(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = numpy.zeros ((size_x, size_y))
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1,y] + 1,
                    matrix[x-1,y-1] + 1,
                    matrix[x,y-1] + 1
                )
    return (matrix[size_x - 1, size_y - 1])
```

Código 1: Función de Levenshtein original. Brinda una métrica que permite determinar cuánta distancia hay entre dos secuencias. Fue necesario extenderla para que se ajuste mejor al dominio del problema.

```
#Indica cuántas letras de seq2 no tenían que estar.
def letrasDeMas(seq1,seq2):
    letras = list(str(seq2))

    letrasSobrantes = 0

    for letra in letras:
        if not(letra in list(seq1)):
            letrasSobrantes += 1

    return letrasSobrantes

#Indica cuántas letras de seq1 están repetidas en seq2.
def cantRepetidos(seq1,seq2):
    letras = list(seq1)
    repeticiones = 0

    for letra in letras:
        apariciones = (str(seq2)).count(letra)
        if apariciones > 1:
            repeticiones+=apariciones

    return repeticiones

#Complementa la función original sumando penalizaciones
#por letras repetidas o sobrantes.
def levenshteinExtendido(seq1,seq2):
    letrasSobrantes = 0
    repeticiones = 0

    #Los factores de multiplicación se determinaron luego
    #de experimentar con muchos casos.
    repeticiones = cantRepetidos(seq1,seq2)*2
    letrasSobrantes = letrasDeMas(seq1,seq2)*3

    #Retorna la distancia de Lev+penalización por repeticiones
    #y letras que no están en la seq1.
    return levenshtein(str(seq1),str(seq2))+repeticiones+letrasSobrantes
```

Código 2: Método extendido de Distancia de Levenshtein. Brinda una métrica de la diferencia entre dos secuencias. Se utilizó como método para el análisis de los errores, comparando las respuestas de las personas evaluadas, respecto de las secuencias originales. A la función de Levenshtein original se le sumaron penalizaciones por estímulos duplicados en las respuestas (cuestión restringida en el dominio del test de Bloques de Corsi) y por la inclusión de estímulos que no pertenecían a la secuencia original.

C. CONSTRUCCIÓN DE PROTOCOLOS

Se comparten en este Anexo como complemento a la sección Diseño de los Protocolos el algoritmo desarrollado para la construcción de los dos protocolos que se utilizaron para los experimentos. Como métrica principal en cuanto a la dificultad se utilizó la distancia total de cada una de las secuencias de un protocolo y se tomó como restricción la ausencia de repeticiones de pares cercanos en las secuencias un mismo protocolo.

El script que permite la construcción de protocolos candidatos es el theory.py, disponible en el repositorio [45]. El mismo permite configurar con diferentes parámetros el protocolo a construir: longitud mínima y máxima de las Secuencias; maximizar o minimizar la distancia total; cantidad de secuencias por cada longitud; y el método de construcción del protocolo: de mayor a menor o de menor a mayor.

La distancia total de una Secuencia en un Protocolo se calcula como la suma total de las distancias físicas entre cada par de botones consecutivos. Estas distancias se corresponden con la ubicación de los botones en el CorsiBot (expresadas en centímetros). Se contempló como una de las variables de dificultad la distancia total puesto que era esperable que, en general, una Secuencia con mayor recorrido tendría mayor dificultad que una con menor recorrido. Por ello se trabajó con la idea de generar un **Protocolo Corto** y un **Protocolo Largo**, con una menor y mayor distancia total, respectivamente. Para comparar esta variable se implementó la suma de la norma vectorial entre cada par de botones consecutivos dentro de una secuencia (Ecuación C.1) implementada mediante el Código 5.

$$\sum_{i=1}^{n-1} \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \tag{C.1}$$

C.1: Distancia total de una Secuencia. Se calcula como la sumatoria de la norma vectorial entre cada par de botones consecutivos de una Secuencia con n botones. Siendo (X_j, Y_j) la posición del botón j con $j \in [1, n]$

Para tratar de evitar que los Protocolos se conformasen tomando una parte o algunas partes específicas del tablero (el sector superior, inferior, derecho, izquierdo o centro) se trabajó para que las Secuencias dentro de un Protocolo fuesen construidas con una distribución de Estímulos uniforme. Para trabajar con esta variable, por cada Protocolo generado se calculó la cantidad de apariciones de cada Estímulo y se tomó como métrica la **varianza** de cada una de esas listas, buscando minimizar dicho parámetro. La implementación de este cálculo puede observarse en el apéndice, Código 6.

Se muestran en los Códigos 3 y 4 la lista de secuencias de los dos Protocolos finalmente utilizados, introduciendo al comienzo cuatro secuencias de longitud 1 y 2, que serán de práctica para la persona evaluada, definidas con el parámetro booleano junto a cada secuencia tal como se mencionó en la sección Base de Datos.

```
{
    "trials": {
            "1": ["A", "True"],
            "2": ["BC", "True"],
            "3": ["D", "True"],
            "4": ["EF", "True"],
            "5": ["IG", "False"],
            "6": ["AI", "False"],
            "7": ["EBA", "False"],
            "8": ["CGA", "False"],
            "9": ["IEFC", "False"],
            "10": ["IHGD", "False"],
            "11": ["HDABI", "False"],
            "12": ["AGHEB", "False"],
            "13": ["CEGADB", "False"],
            "14": ["BEDHIG", "False"],
            "15": ["BCDEIFH", "False"],
            "16": ["GBIDHFC", "False"],
            "17": ["FEDIGACB", "False"],
            "18": ["GDABIEHC", "False"]
        }
}
```

Código 3: Secuencias Protocolo Corto. Se agregaron para los experimentos las primeras cuatro secuencias, definidas como de práctica a través del parámetro booleano configurado en "True"(no computan error en caso de que la persona responda de manera incorrecta). La distancia total del Protocolo Corto (sin contemplar las secuencias de práctica) es de: **122.85 cm**.

```
{
    "trials": {
            "1": ["A", "True"],
            "2": ["BC", "True"],
            "3": ["D", "True"],
            "4": ["EF", "True"],
            "5": ["CI", "False"],
            "6": ["AF", "False"],
            "7": ["EIG", "False"],
            "8": ["FBC", "False"],
            "9": ["BACD", "False"],
            "10": ["DCGB", "False"],
            "11": ["CEAID", "False"],
            "12": ["CHAFE", "False"],
            "13": ["IFGEBA", "False"],
            "14": ["DBEHCA", "False"],
            "15": ["BIHDGEC", "False"],
            "16": ["DIGHBAC", "False"],
            "17": ["GADCIFHE", "False"],
            "18": ["GDHAEFIB", "False"]
        }
}
```

Código 4: Secuencias Protocolo Largo. Se agregaron para los experimentos las primeras cuatro secuencias, definidas como de práctica a través del parámetro booleano configurado en "True"(no computan error en caso de que la persona responda de manera incorrecta). La distancia total del Protocolo Largo (sin contemplar las secuencias de práctica) es de: **150.8 cm**.

```
def distances_ret_total(sequence):
    d = []
    for i in range(len(sequence)-1):
        a = box_positions[sequence[i]]
        b = box_positions[sequence[i+1]]
        d.append(numpy.linalg.norm(numpy.array(a)-numpy.array(b)))

    return sum(d)
```

Código 5: Método que retorna la distancia total de una secuencia de estímulos. Para ello utiliza la norma vectorial mediante la función *linalg.norm* de la biblioteca numpy [46], calculando la distancia para secuencias con dos o más botones, con la ubicación de los mismos definida previamente en el diccionario *box_positions*. El método construye una lista de distancias por cada par de botones, por lo que para calcular la distancia total basta con sumar todos los elementos de la lista. La distancia total de las Secuencias fue la variable de dificultad principal para el armado de los Protocolos, definiéndolos finalmente como Protocolo Corto y Protocolo Largo.

```
def cantApariciones_ret_varianza(letras, secuencias_seleccionadas):
        array_apariciones = []
        for letra in letras:
                apariciones = (str(secuencias_seleccionadas)).count(letra)
                array_apariciones.append(apariciones)

        return statistics.variance(array_apariciones)
```

Código 6: Método que retorna la varianza en base a la aparición de los botones en un Protocolo. Se tomó como métrica en el momento de elegir las Secuencias candidatas para cada Protocolo, con el objetivo de lograr una distribución lo más uniforme posible para los Estímulos.

D. MANUAL DE USUARIO

El CorsiBot es un dispositivo que permite la creación, administración y registro de la prueba bloques de Corsi con una interfaz completamente computarizada. El presente Manual está pensado para el uso de las personas que operen el CorsiBot. Contiene instrucciones para operar el dispositivo al iniciarlo, cargar protocolos y tomas de datos, realizar las evaluaciones y visualizar los resultados.

Inicio del Corsibot

Para iniciar el CorsiBot sólo debe conectarse a una fuente eléctrica convencional de 220 Volts y esperar unos segundos a que el dispositivo inicie. Para iniciar el uso del CorsiBot, la persona que lo opere deberá conectarse a través de un dispositivo con conexión inalámbrica (notebook, tablet o smartphone).

Para hacerlo deberá conectarse con la red cuyo nombre (SSID) es *CorsiWifi* y la contraseña por defecto es *Corsi2019*. Una vez establecida la conexión deberá ingresar mediante un navegador web (Google Chrome, Firefox, Opera, etc) a la url: **http://corsi.local/** o a la dirección: **http://192.168.50.10/**. Fig. D.1.

Fig. D.1. Pantalla principal - Menú de opciones visto desde una computadora. Si la resolución de pantalla es menor (como ser una tablet o smarthpone), se presentará como menú vertical desplegable.

Desde la pantalla principal se presentarán las opciones para:

- Cargar en la base de datos un protocolo y una toma de datos con los usuarios asociados a través de la sección *Protocolos*.
- Explorar las tomas de datos realizadas, sus resultados y exportarlos a formato csv, también en la sección *Protocolos*.
- Acceder a la documentación sobre el uso del dispositivo, sección *Documentación*.
- Seleccionar un protocolo, toma de datos y usuario y comenzar un experimento, opción *Evaluar*.

Luego de que la persona que opere el CorsiBot haya preparado el dispositivo, se podrá invitar a la persona a evaluar que se ubique frente al dispositivo para comenzar la evaluación. Desde la posición que se sitúe deberá poder observar el encendido de las luces de las Secuencias y presionar los botones para responder. El dispositivo consta de 9 botones/luces que se usarán para presentar los estímulos y para tomar las respuestas. Dos barras lumínicas para dar *feedback*, y dos botones para indicar la finalización de una etapa (ver Fig. D.2) o el paso a la siguiente secuencia.

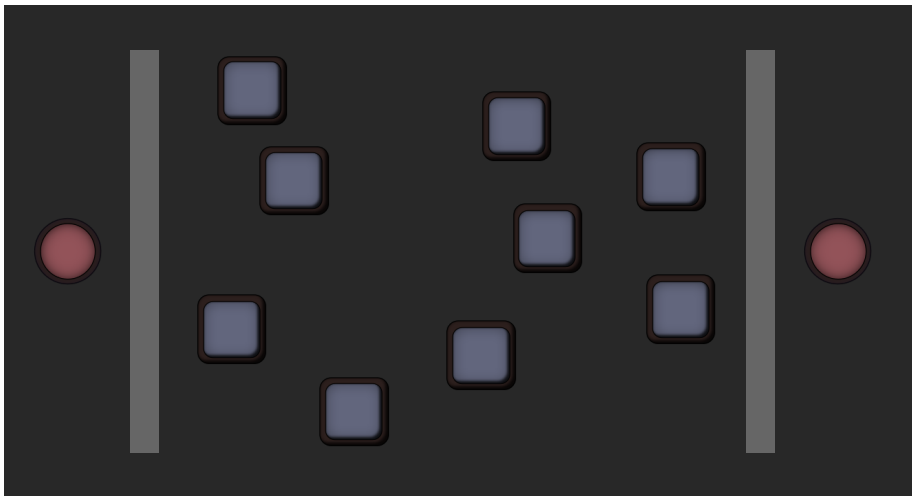

Fig. D.2. Tablero del Dispositivo. Se muestran los 9 botones/luces que componen las Secuencias, las barras lumínicas laterales y los botones rojos de confirmación y siguiente secuencia, cuyo funcionamiento dependerá de estar habilitados por la configuración del Protocolo.

D.1. Carga de Protocolos, Tomas de Datos y Usuarios

Previo a comenzar con los experimentos, la persona que opere el dispositivo deberá cargar en la base de datos al menos un Protocolo y una Toma de Datos con sus Usuarios asociados.

Para ello se deberán confeccionar archivos en formato *.json* y luego subirlos a la base de datos mediante la interfaz web, sección *Protocolos*. Es importante destacar que los Protocolos y Tomas de Datos subidos quedarán almacenados en la base de datos, independientemente de si se interrumpe la alimentación eléctrica o se apaga el dispositivo.

Ejemplo de definición de Protocolo mediante archivo json

En el Código 7 puede observarse el formato del archivo json para la definición de las variables del Protocolo. El archivo debe continuar con la definición de las secuencias, tal como se muestra en el Código 8. Pueden encontrarse ejemplos para descargar y modificar dos archivos de Protocolo en la sección *Documentación* del menú principal del Dispositivo.

```
{
    "protocolo": {
        "nombreProtocolo": "Protocolo X"
    },
    "properties": {
                "tiempoEncendidoLuz": 500,
                "tiempoEntreLuces": 500,
                "tiempoComienzaIntento": 0,
                "timeoutRespuesta": 60000,
                "cantErroresCorte": 0,
                "reverso": "False",
                "botonRojoIniciaTrial":"True",
                "luzComienzaSecuencia":"True",
                "colorLuzComienzaSecuencia":"blue",
                "tiempoLuzComienzaSecuencia":"500",
                "repeticionLuzComienzaSecuencia":"3",
                "intervaloRepeticionLuzComienzaSecuencia":"100",
                "luzEsperaBotones":"True",
                "colorLuzEsperaBotones":"blue",
                "luzAnalisisIntento":"True",
                "colorLuzAnalisisIntento":"yellow",
                "feedbackLuces": "True",
                "colorSecuenciaCorrecta":"green",
                "colorSecuenciaIncorrecta":"red",
                "tiempoLuzFeedback":"500",
                "repeticionLuzFeedback":"3",
                "intervaloRepeticionLuzFeedback":"100",
                "luzEsperaProxSecuencia": "False",
                "colorLuzEsperaProxSecuencia": "blue",
                "feedbackSonido":"True",
                "luzTerminaTomaDatos":"True"
    },
(...)
}
```

Código 7: Definición de Parámetros de un Protocolo mediante archivo json con sus valores por defecto. (continúa)

En el mismo archivo json de Protocolo, se deberán definir las secuencias, tal como se indica en el Código 8.

```
{
(...)
    "trials": {
                "1": ["A", "True"],
                "2": ["BC", "True"],
                "3": ["D", "True"],
                "4": ["EF", "True"],
                "5": ["IG", "False"],
                "6": ["AI", "False"],
                "7": ["EBA", "False"],
                "8": ["CGA", "False"],
                "9": ["IEFC", "False"],
                "10": ["IHGD", "False"],
                "11": ["HDABI", "False"],
                "12": ["AGHEB", "False"],
                "13": ["CEGADB", "False"],
                "14": ["BEDHIG", "False"],
                "15": ["BCDEIFH", "False"],
                "16": ["GBIDHFC", "False"],
                "17": ["FEDIGACB", "False"],
                "18": ["GDABIEHC", "False"]
        }
}
```

Código 8: Definición de Secuencias de un Protocolo mediante archivo json del Protocolo

Las secuencias pueden ser de práctica, es decir que no computan error, o de evaluación, que sí computan error. El parámetro booleano junto a cada secuencia indica si es de práctica (True) o no (False).

Cada atributo de un Protocolo tiene un valor por defecto. Si no se redefine al cargarlo tomará dicho valor. Es posible una definición mínima de un Protocolo con la configuración de *protocolo* y *trials* (Secuencias), tal como se observa en el Código 9. Los valores por defecto pueden consultarse en la interfáz web y son los mismos que se definen en el Código 7.

```
{"protocolo": {
        "nombreProtocolo": "Protocolo Minimo"
},
"properties": {
},
"trials": {
        "1": ["A","true"],
        "2": ["B","false"]
}}
```

Código 9: Definición mínima de archivo json de Protocolo, aceptada por el Dispositivo. Los parámetros no definidos explícitamente en el archivo json, tomarán los valores por defecto, indicados en el Código 7.

Descripción de los atributos

- **nombreProtocolo**: Cadena de caracteres con la que se identificará un Protocolo.
- **tiempoEncendidoLuz**: Tiempo que estará encendido cada led en las secuencias del Protocolo. Por defecto 500 ms.
- **tiempoEntreLuces**: Intervalo entre el encendido de dos leds en una secuencia del Protocolo. Por defecto 500 ms.
- **tiempoComienzaIntento**: Tiempo de demora entre que se da la orden de reproducir una secuencia y el encendido del primer led, para todas las secuencias de un protocolo. Por defecto 0 ms.
- **timeoutRespuesta**: Tiempo de espera para la respuesta del usuario, una vez que se ha terminado de reproducir la secuencia. Por defecto: 60 segundos.
- **cantErroresCorte**: Criterio de errores consecutivos para dar por concluido un experimento. Por defecto 0. Si se define en 0, no habrá criterio de corte.
- **reverso**: Por defecto en *False*. Si se define como *True*, los usuarios deberán responder en el orden inverso al que fue presentada la secuencia para que el sistema considere correcta la respuesta.
- **botonRojoIniciaTrial**: Permite habilitar o deshabilitar la posibilidad de que el usuario inicie el siguiente Trial mediante el Botón Rojo. Por defecto en *True*.
- **luzComienzaSecuencia**: Por defecto en *True*. Habilita a que se enciendan las barras RGB laterales al iniciar cada una de las secuencias, como forma de alertar al usuario. Los siguientes cuatro atributos están relacionados.
- **colorLuzComienzaSecuencia**: Color en el que se encenderá las barras laterales RGB al inicio de la reproducción de cada secuencia, en caso de que *luzComienzaSecuencia* esté definido como *True*. Por defecto en *blue*.
- **tiempoLuzComienzaSecuencia**: Tiempo de encendido de las barras laterales RGB al inicio de la reproducción de cada secuencia, en caso de que *luzComienzaSecuencia* esté definido como *True*. Por defecto 500 ms.

- **repeticionLuzComienzaSecuencia**: Cantidad de veces de encendido de las barras laterales RGB al inicio de la reproducción de cada secuencia, en caso de que *luzComienzaSecuencia* esté definido como *True*. Por defecto 3 veces.
- **intervaloRepeticionLuzComienzaSecuencia**: Tiempo de demora entre el encendido de las barras laterales RGB al inicio de la reproducción de cada secuencia, en caso de que *luzComienzaSecuencia* esté definido como *True*. Por defecto 500 ms.
- **luzEsperaBotones**: Por defecto en *True*. Mantiene encendida las barras RGB laterales mientras el usuario responde la secuencia. El siguiente atributo está relacionado.
- **colorLuzEsperaBotones**: Por defecto en *blue*. Es el color con el cual se mantendrán encendidas las barras RGB laterales mientras el usuario responde la secuencia.
- **luzAnalisisIntento**: Por defecto en *True*. Mantiene encendida las barras RGB laterales mientras se evalúa la respuesta del usuario. Funciona como feedback de confirmación ante la presión del Botón Rojo por parte del usuario o la Validación del intento por parte del operador mediante el navegador web. El siguiente atributo está relacionado.
- **colorLuzAnalisisIntento**: Por defecto en *yellow*. Es el color con el que se mantendrán encendidas las barras RGB laterales mientras se evalúa cada respuesta del Usuario a una secuencia.
- **feedbackLuces**: Por defecto en *True*. Habilita o deshabilita la respuesta lumínica con las tiras de Leds RGB laterales a la respuesta del Usuario ante la secuencia presentada. Los siguientes cinco atributos están relacionados.
- **colorSecuenciaCorrecta**: Por defecto en *green*. Color con el que se encienden las barras laterales RGB en caso de que el usuario haya respondido correctamente.
- **colorSecuenciaIncorrecta**: Por defecto en *red*. Color con el que se encienden las barras laterales RGB en caso de que el usuario haya respondido incorrectamente.
- **tiempoLuzFeedback**: Tiempo de encendido de las barras laterales RGB ante la respuesta del Usuario, en caso de que *feedbackLuces* esté definido como *True*. Por defecto 200 ms.
- **repeticionLuzFeedback**: Cantidad de veces de encendido de las barras laterales RGB ante la respuesta del Usuario, en caso de que *feedbackLuces* esté definido como *True*. Por defecto 3 veces.
- **intervaloRepeticionLuzFeedback**: Tiempo de demora entre el encendido de las barras laterales RGB ante la respuesta del Usuario, en caso de que *feedbackLuces* esté definido como *True*. Por defecto 100 ms.
- **luzEsperaProxSecuencia**: Por defecto en *True*. Habilita o deshabilita el encendido de las barras laterales RGB mientras se espera que se seleccione una nueva secuencia a reproducir. El siguiente atributo está relacionado.
- **colorLuzEsperaProxSecuencia**: Por defecto en *blue*. Color con el que se encienden las barras laterales RGB mientras se espera que se seleccione una nueva secuencia a reproducir, siempre que *luzEsperaProxSecuencia* esté definido en True.
- **feedbackSonido**: Por defecto en *True*. Habilita o deshabilita un feedback sonoro ante una respuesta correcta o incorrecta del Usuario.

- **luzTerminaTomaDatos**: Por defecto en *True*. Habilita o deshabilita una secuencia fija de colores en las barras laterales RGB para indicarle al usuario que ha finalizado la Toma de Datos.

La Fig. D.3 muestra cómo los parámetros de un Protocolo, anteriormente descriptos, son utilizados al realizar una iteración de una secuencia en una evaluación. Los rectángulos verticales representan las barras laterales RGB y los cuadrados representan los leds/botones. El color gris indica el momento en que se encuentran apagados y cualquier otro color, encendidos. El círculo rojo indica el Botón Rojo de confirmación de secuencia. Los valores entre corchetes indican los valores por defecto de los parámetros.

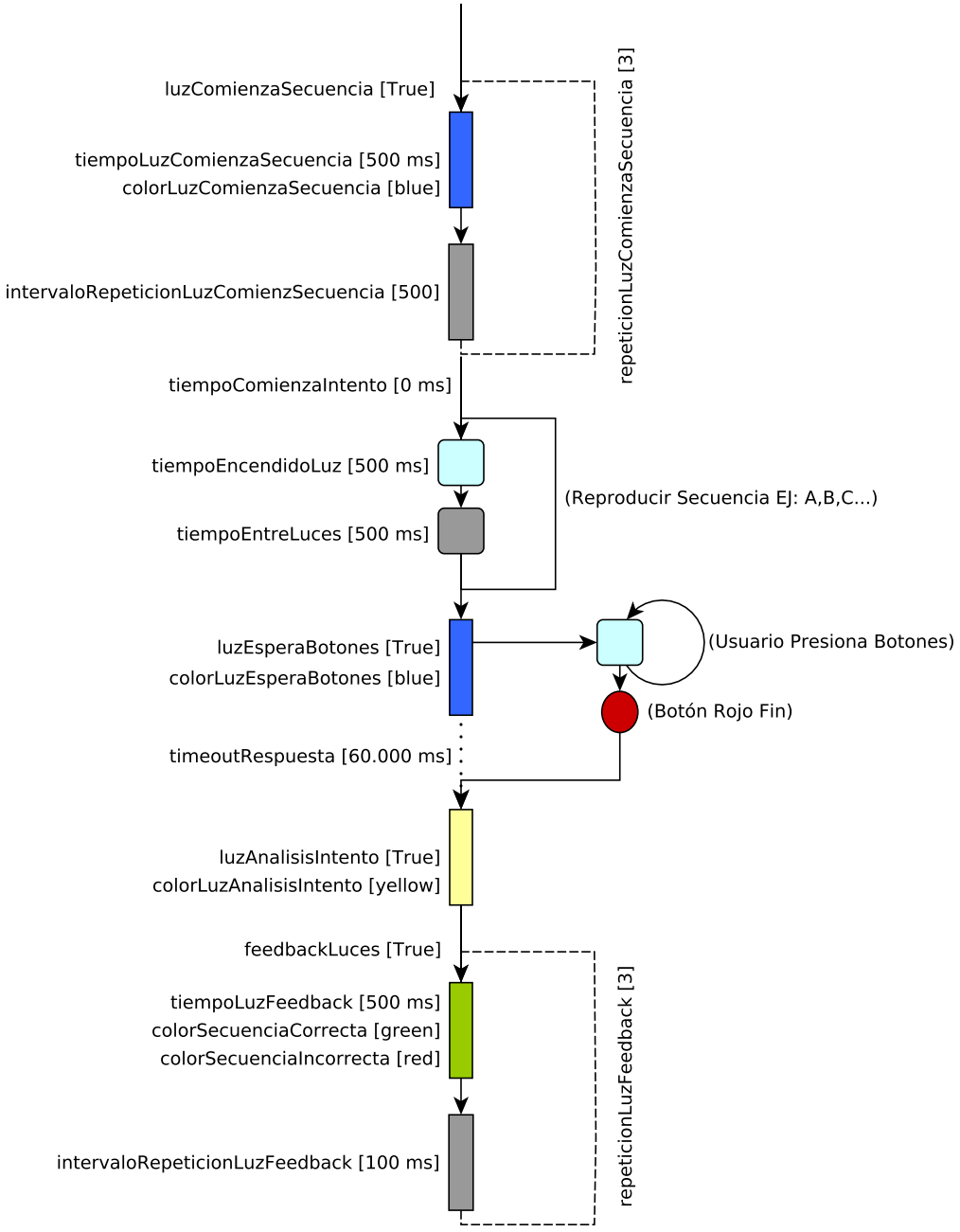

Fig. D.3. Parámetros de un Protocolo. Los rectángulos verticales representan las barras laterales RGB y los cuadrados representan los Leds/botones. El color gris indica el momento en que se encuentran apagados y cualquier otro color, encendidos. El círculo rojo indica el Botón Rojo de confirmación de secuencia. Los valores entre corchetes indican los valores por defecto de los parámetros.

Para subir un nuevo Protocolo, el operador deberá ingresar a la sección ***Protocolos*** y luego seleccionar la opción "Subir Protocolo", tal como indica la Fig. D.4. En la siguiente pantalla se presentará un campo para seleccionar y subir el archivo con formato json,

desde la computadora de la persona que esté operando el sistema. En esta misma sección, podrán visualizarse los Protocolos cargados.

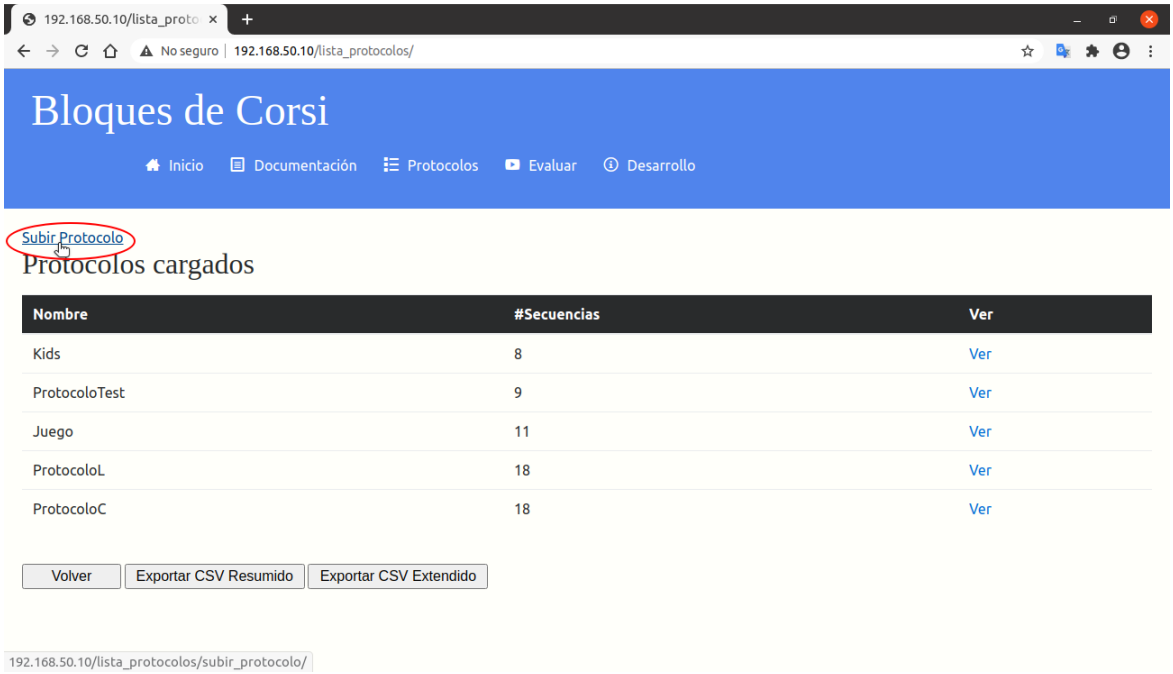

Fig. D.4. Pantalla de carga de Protocolo.

Para cargar una nueva Toma de Datos el operador deberá seleccionar el Protocolo al cual desee ingresar una nueva Toma de Datos, y seleccionar la opción "Ver". En esa sección aparecerá la opción "Subir Toma de Datos".

Ejemplo de definición de Toma De Datos mediante archivo json

Para la Toma de Datos, el procedimiento será de similar manera. Se definirá un archivo con formato *json* como el definido en el Código 10.

```
{
    "tomaDeDatos": {
        "nombreTomaDeDatos": "TomaDatosEjemplo"
    },
    "usuarios": {
        "nombres": ["User1","User2","User3"]
    }
}
```

Código 10: Ejemplo de definición de archivo json para la Toma de Datos.

Como se evidencia, la definición de este archivo es bastante más sencilla. Habrá que definir un nombre con el cual se identifique a la Toma de Datos y una lista con los Usuarios

que van a realizar el experimento. Vale la pena aclarar que luego de subida una Toma de Datos podrán agregarse manualmente Usuarios de forma individual pero no podrán eliminarse los previamente cargados.

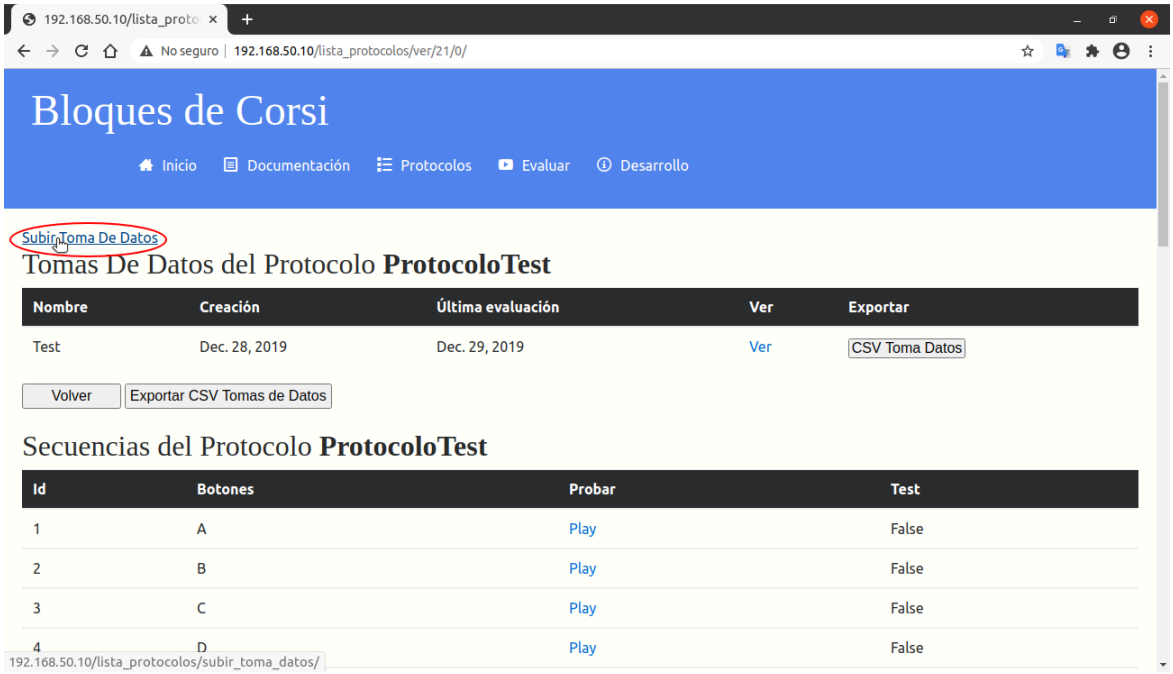

Fig. D.5. Pantalla de visualización de un Protocolo, la cual permite la carga de una nueva Toma de Datos.

Como se indicó previamente, en la sección ***Protocolos*** del menú principal, se muestran todos los Protocolos cargados en el Dispositivo. Para cargar una toma de datos, se deberá seleccionar un Protocolo ya cargado, hacer click en la opción "Ver" y en la siguiente pantalla ir a la opción ***Subir Toma de Datos***, como muestra la Fig. D.5. La siguiente pantalla mostrará un campo que permitirá seleccionar un archivo json con el formato de Toma de Datos previamente definido.

D.2. Visualización de resultados y exportación de datos

Fig. D.6. Al finalizar una evaluación la persona que opera el sistema visualizará los resultados mediante esta vista.

La la Fig. D.6 permite analizar el resultado de una evaluación. Como se puede deducir, en la columna de Correcto/Incorrecto, los círculos verdes con tilde blanca indicarán que la respuesta del usuario ha sido correcta, mientras que los círculos rojos con cruz blanca indicarán que la respuesta ha sido incorrecta. La persona que opera el sistema en esta instancia podrá seleccionar la opción de "Volver" para regresar al inicio o "Exportar CSV" para descargar los datos de la evaluación que ha realizado recientemente la persona evaluada.

En el caso de que la persona que opera el sistema decida exportar los resultados para su posterior análisis, deberá presionar el botón "Exportar CSV". El mismo procesará los datos almacenados en la base de datos y permitirá descargar un archivo con formato csv similar al de la Fig. D.7.

Usuario	idEvaluacion	numRegistroToma	idSec	timeStamp	Tipo Evento	Valor Evento	Tiempo Presionado	Tiempo Desde Inicio
Usr30	110	75	12	2019-12-29 01:45:09.115466	Control	OI	100	0
Usr30	110	76	12	2019-12-29 01:45:18.601680	Control	ITR	100	0
Usr30	110	77	12	2019-12-29 01:45:19.892524	Boton	C	300	907
Usr30	110	78	12	2019-12-29 01:45:20.859303	Boton	H	300	1874
Usr30	110	79	12	2019-12-29 01:45:21.733074	Boton	A	300	2748
Usr30	110	80	12	2019-12-29 01:45:22.702037	Boton	F	300	3717
Usr30	110	81	12	2019-12-29 01:45:23.709259	Boton	E	501	4724
Usr30	110	82	12	2019-12-29 01:45:24.623906	Control	RF	200	5639
Usr30	110	83	12	Correcto	True	['C', 'H', 'A', 'F', 'E']	['C', 'H', 'A', 'F', 'E']	5639
Usr30	110	84	13	2019-12-29 01:45:30.586901	Control	OI	100	0
Usr30	110	85	13	2019-12-29 01:45:41.817449	Control	ITR	100	0
Usr30	110	86	13	2019-12-29 01:45:44.976664	Boton	F	500	3017
Usr30	110	87	13	2019-12-29 01:45:46.297990	Boton	G	501	4338
Usr30	110	88	13	2019-12-29 01:45:47.462548	Boton	E	300	5503
Usr30	110	89	13	2019-12-29 01:45:48.789568	Boton	B	300	6830
Usr30	110	90	13	2019-12-29 01:45:50.767700	Boton	A	701	8808
Usr30	110	91	13	2019-12-29 01:45:56.064971	Boton	D	501	14105
Usr30	110	92	13	2019-12-29 01:45:57.803976	Control	RF	200	15844
Usr30	110	93	13	Correcto	False	['I', 'F', 'G', 'E', 'B', 'A']	['F', 'G', 'E', 'B', 'A', 'D']	15844
Usr30	110	94	14	2019-12-29 01:46:03.583108	Control	OI	100	0
Usr30	110	95	14	2019-12-29 01:46:14.586931	Control	ITR	100	0
Usr30	110	96	14	2019-12-29 01:46:17.038045	Boton	D	300	2305
Usr30	110	97	14	2019-12-29 01:46:18.135469	Boton	B	300	3403
Usr30	110	98	14	2019-12-29 01:46:20.300421	Boton	E	300	5568
Usr30	110	99	14	2019-12-29 01:46:21.786972	Boton	H	300	7054
Usr30	110	100	14	2019-12-29 01:46:22.624532	Boton	A	300	7892
Usr30	110	101	14	2019-12-29 01:46:23.652409	Boton	C	300	8920
Usr30	110	102	14	2019-12-29 01:46:24.663208	Control	RF	200	9931
Usr30	110	103	14	Correcto	False	['D', 'B', 'E', 'H', 'C', 'A']	['D', 'B', 'E', 'H', 'A', 'C']	9931

Fig. D.7. Vista en formato CSV de los resultados de la evaluación realizada por un usuario. En la misma se pueden observar las secuencias originales, las respuestas del usuario con los tiempos en los cuales presionó cada botón y eventos de control que permiten reconstruir cómo ha sido el comienzo y final de cada secuencia, con códigos de eventos de control, según la tabla D.1.

En la Fig. D.7 puede observarse las respuestas del usuario *Usr30* ante tres secuencias de longitud 5 y 6, resultando sus respuestas incorrectas en dos oportunidades y correcta en una de ellas. Se observa en las columnas "Tipo Evento", "Valor Evento" y "Tiempo Presionado", información sobre los momentos en que son presionados cada uno de los botones por el usuario, eventos de control sobre cómo se iniciaron o finalizaron las respuestas, y el resultado sobre si es correcta la respuesta o no (True o False), junto a los botones esperados y los presionados por el usuario.

Código Evento	Significado
RI	El Usuario ha presionado el botón Rojo para dar Inicio a la secuencia.
RF	El Usuario ha presionado el botón Rojo para dar Fin y confirmar su respuesta.
OI	El/la Operador/a ha Iniciado manualmente, mediante el navegador, la siguiente secuencia.
OF	El/la Operador/a ha Finalizado y validado manualmente, mediante el navegador, la respuesta del Usuario.
OAS	El/la Operador/a ha Anulado la Secuencia manualmente mediante el navegador.
TO	Ha vencido el TimeOut para responder, definido en el Protocolo.
OTI	El/la operador/a ha finalizado manualmente el Intento/evaluación, mediante el navegador.
ITR	Inicio de Tiempo Respuesta. Momento cero desde el cual se cuenta el tiempo en el que se comienza a registrar los botones que presiona la persona evaluada.

Tab. D.1: Códigos de Eventos de Control. Permiten visualizar en los archivos CSV una traza de los eventos de control en las respuestas a las secuencias, tal como puede observarse en la Fig. D.7.

E. MANUAL TÉCNICO

Los planos y código fuente del dispositivo, se encuentran abiertos para su descarga a través del siguiente enlace:

https://github.com/maxiwebs/CorsiBot/.

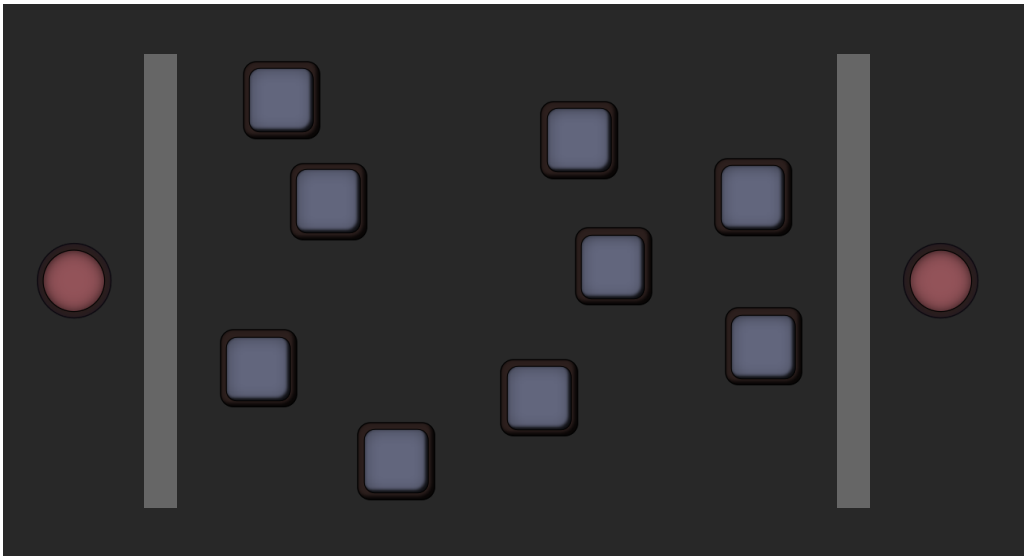

Fig. E.1. Tablero del Dispositivo. Contiene las nueve luces/botones de secuencias, dos barras verticales traslúcidas con Leds RGB y dos botones rojos de confirmación y siguiente secuencia.

En cuanto al tablero del dispositivo, Fig. E.1, los nueve pulsadores con sus correspondientes luces pueden ser controlados cada uno con un puerto de entrada/salida independiente de la Placa Raspberry Pi 3B+ [39]. Es decir, por cada luz/botón tendrá asociado un puerto de la placa Raspberry configurado como entrada (para capturar el botón) y otro como salida (para encender el Led). El tipo de botón instalado es el modelo Arcade-A0127 pero se les reemplazó la lámpara original de 12 Volts por leds blancos de alta luminosidad para que puedan ser alimentados directamente desde los 5 Volts de la Placa Raspberry y así evitar la necesidad de un módulo para el manejo de ambos de voltajes.

A su vez, los dos pulsadores externos de confirmación están conectados a una misma entrada y las tiras de LEDs RGB tienen una salida por cada uno de los tres colores. Por último, se cuenta con un buzzer instalado a una salida adicional. Fueron utilizados 23 de los 40 puertos de entrada/salida disponibles en la placa Raspberry Pi, sin contar los pines de puesta a tierra. Puede observarse en la Fig. E.3 el detalle del circuito interno. En la Fig. E.2 se muestra el esquema de entradas/salidas de la Placa Raspberry Pi 3B+. El mapeo de Puertos puede observarse en la tabla E.1.

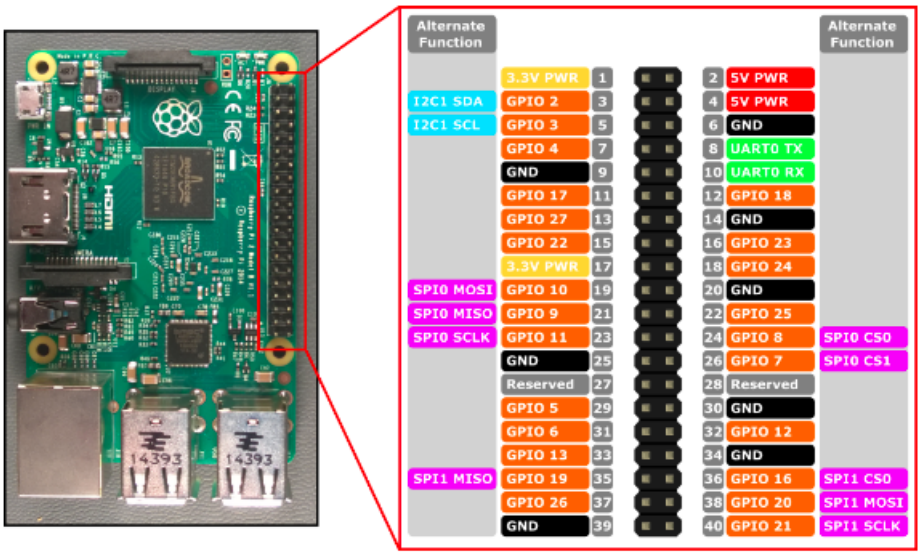

Fig. E.2. Pinout Raspberry Pi 3B+. Identificadores de puertos de entrada/salida.

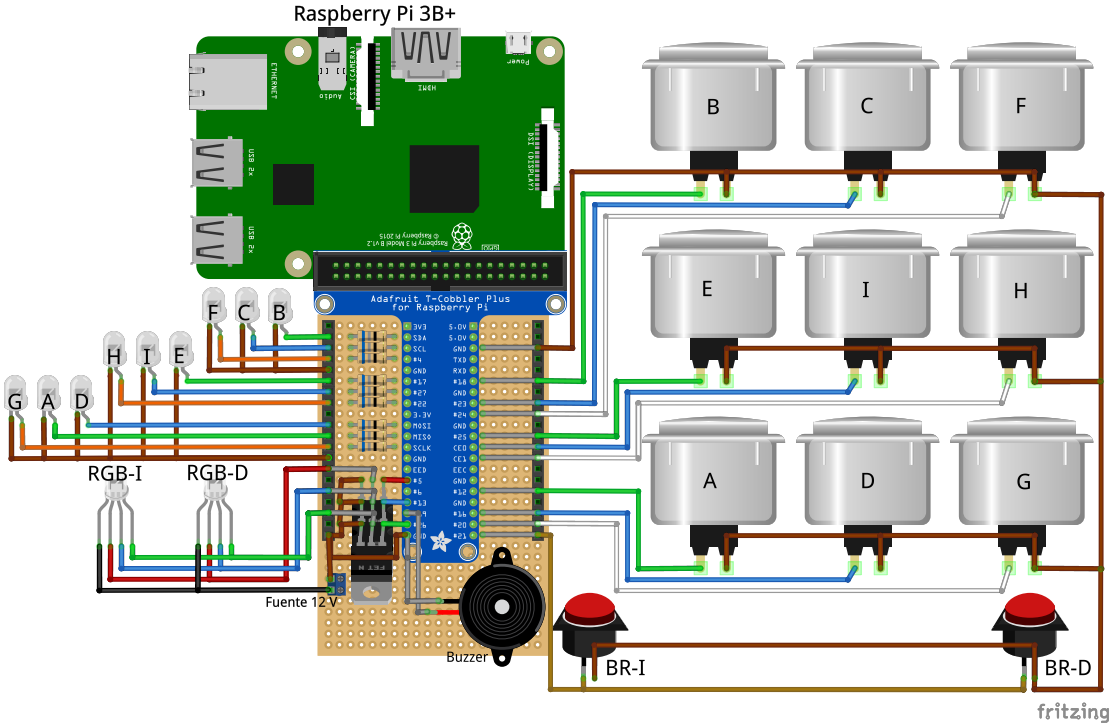

Fig. E.3. Circuito interno del Dispositivo. Los Leds Blancos están integrados a los pulsadores (según la letra correspondiente). Se muestran separados para mejorar la comprensión del circuito. Tanto las dos tiras de Leds RGB como los botones rojos de confirmación (Izquierdo y Derecho) están conectados a las mismas entradas/salidas con lo cual es el mismo control para ambas tiras de Leds RGB y para ambos botones rojos.

Id Luz/Salida	Puerto RBPi3B+	Id Botón/Entrada	Puerto RBPi3B+
A	21/GPIO 9	A	32/GPIO 12
B	3/GPIO 2	B	12/GPIO 18
C	5/GPIO 3	C	16/GPIO 23
D	19/GPIO 10	D	36/GPIO 16
E	11/GPIO 17	E	22/GPIO 25
F	7/GPIO 4	F	18/GPIO 24
G	23/GPIO 11	G	38/GPIO 20
H	15/GPIO 22	H	26/GPIO 7
I	13/GPIO 27	I	24/GPIO 8
RGB Rojo	29/GPIO 5	—	—
RGB Verde	33/GPIO 13	—	—
RGB Azul	37/GPIO 26	—	—
Buzzer (Z)	35/GPIO 19	Botón Rojo	40/GPIO 21

Tab. E.1: Mapeo de Leds, Buzzer y Botones del Dispositivo con los puertos de Entrada/Salida de la Placa Raspberry Pi 3B+, según la Fig. E.2.

Alimentación

El dispositivo contiene en su interior, dos fuentes de alimentación de corriente continua. Por una parte, la fuente de alimentación propia de la placa Raspberry Pi (5V, 2500 mA) y una fuente de alimentación extra para el control de las tiras de Leds RGB cuyo voltaje de operación es de 12 Voltios. Para esto se utilizó una fuente switching de 12 Volts y 3 Ampers, Fig. E.4.

Fig. E.4. Fuente Switching interna para control de Leds RGB.

La interfaz de alimentación externa está compuesta por un conector macho tipo C14 al cual se conecta un cable interlock directo a un toma corriente de 220 Volts, tal como se muestra en la Fig. E.5.

Fig. E.5. Vista lateral del CorsiBot. Se observa el conector de alimentación a 220 Volts.

Configuración de Raspberry como punto de acceso Wifi

Para habilitar la placa Wifi de la Raspberry Pi como punto de acceso y que permita la conexión de dispositivos inalámbricos como notebooks, tablets o smartphones, se utilizaron las bibliotecas **hostapd** y **dnsmasq**. Las configuraciones que permiten su funcionamiento están definidas en los siguientes archivos y disponibles en el repositorio [45].

- /etc/hostapd/hostapd.conf
- /etc/dhcpcd.conf
- /etc/network/interfaces

Para dichas configuraciones, se trabajó con un artículo del sitio *RaspberryConnect* [47].

Scripts de Entrada/Salida

Para el control de los pulsadores y luces de los bloques, se utilizó la biblioteca **RPi.GPIO** [48] que permite una sencilla configuración y control de los puertos de entrada/salida de la Raspberry Pi. Se desarrollaron los scripts de *scripts/tests/testLeds.py* y *scripts/tests/testBotones.py* que hacen uso de esta biblioteca y permiten comprobar el funcionamiento de los leds y pulsadores.

Por otra parte, para el control de los leds RGB se hace uso de la biblioteca **pigpio** [49] que mediante un seteo para los tres valores RGB permite controlar las luces de las barras verticales. Se desarrolló un script de prueba llamado *scripts/tests/testRGB.py* para la verificación de su funcionamiento.

Framework Web

Para la interfaz web se seleccionó el Framework Web Django [40]. El mismo utiliza el patrón de diseño MVT (Model-View-Template) [50] lo que facilita la modularización del código. Se seleccionó este Framework para unificar el uso del lenguaje Python, lo que

facilita la interacción con los scripts de control de las entradas/salidas de la Raspberry Pi. La versión de Django instalada actualmente en el dispositivo es la **1.11.14**.

Django incluye un servidor web liviano que permite la visualización de los *templates*, las páginas que finalmente verán las personas que operen el sistema. Se utilizó además la biblioteca *django-responsive* [51] para adaptar la visualización de las páginas a los navegadores web de PCs, tabletas y smartphones de manera automática.

Django también contiene un módulo de base de datos (como motor por defecto usa SQLite 3 [52]) para las cuales se define la estructura en el archivo ***Models.py*** y provee una API que facilita el manejo de las mismas, pudiendo abstraerse por completo de la sintaxis de SQL.

El Framewrok Django tiene básicamente tres secciones que se relacionan para implementar el modelo MVT:

- **urls.py**: Aquí se definen los puntos de entrada a los diferentes scripts y páginas que redirijen a los distintos métodos del views.py.
- **views.py**: Aquí se define toda la lógica, interacción con la base de datos e invocación a las vistas que finalmente se muestran en los navegadores.
- **Templates**: Son las vistas que se muestran en los navegadores. Son básicamente archivos html, aunque permiten una determinada cantidad de sentencias de programación en un lenguaje propio.

La comunicación entre estos tres módulos principales puede representarse con el esquema de la Fig. E.6.

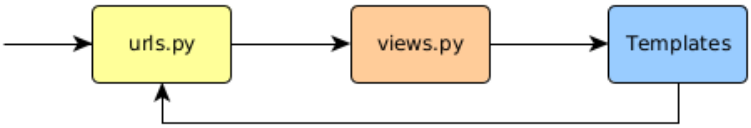

Fig. E.6. Esquema Módulos Django.

Para el presente desarrollo, además se utilizaron scripts en python que permiten la interacción con la placa Raspberry Pi y sus puertos de entrada/salida, donde están conectados los leds y botones.

Base de Datos - Descripción detallada

El diseño de la base de datos desarrollada, se representa mediante el Diagrama Entidad Relación de la Fig. E.7.

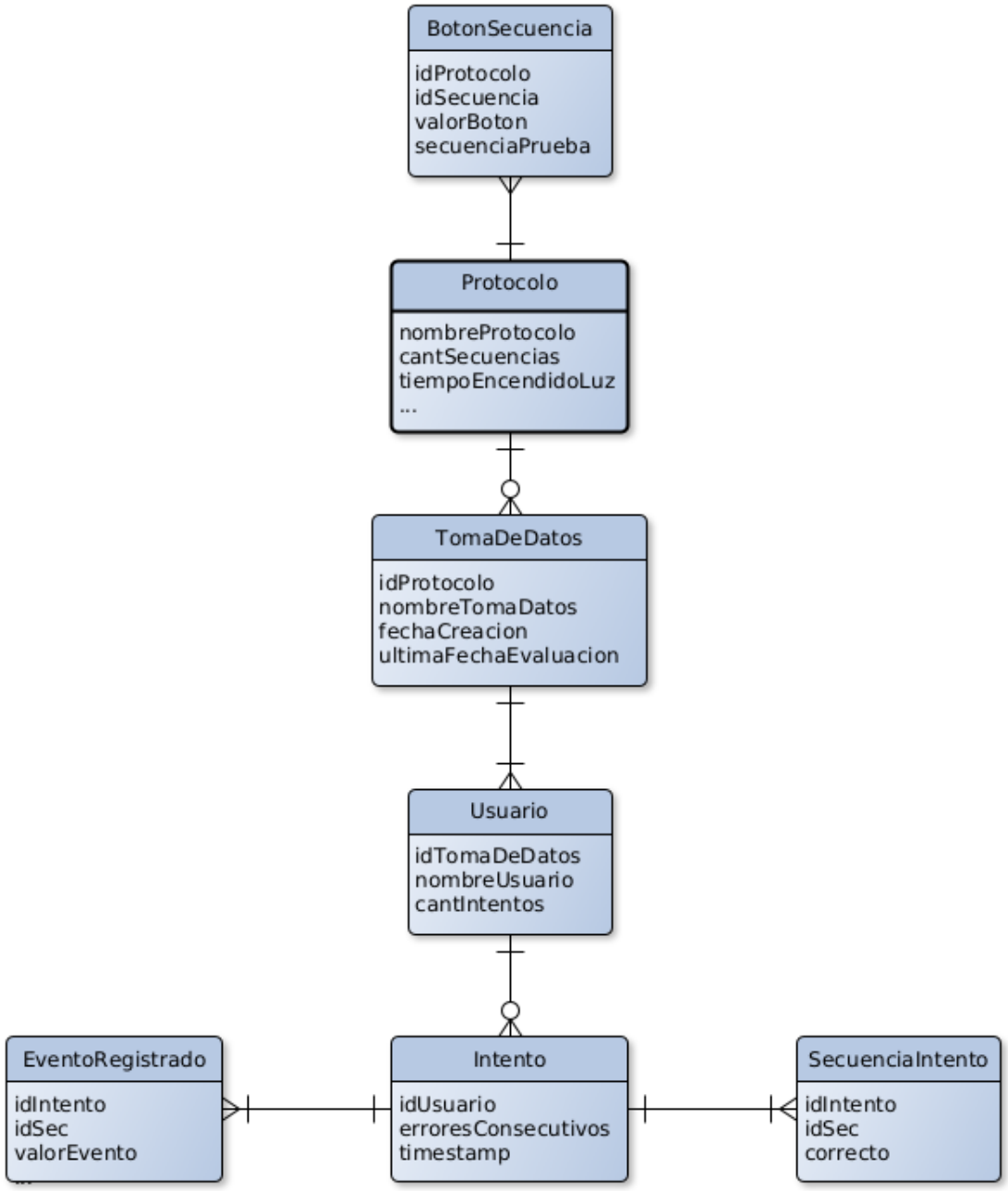

Fig. E.7. Diagrama Entidad Relación.

Entidad BotonesSecuencias

Cada una de las secuencias de un protocolo está conformada por una serie de luces/botones, con una cantidad mínima de uno y una cantidad máxima de nueve luces/botones. Cada luz/botón tendrá asociada una letra según el orden del tablero definido en la Fig. E.8. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class BotonesSecuencias(models.Model):
    idProtocolo = models.ForeignKey(Protocolo, on_delete=models.CASCADE, default=0)
    idSec = models.IntegerField(default=0)
    valorBoton = models.CharField(max_length=1)
    secuenciaPrueba = models.BooleanField(default = False)
```

Código 11: Entidad BotonesSecuencias.

Descripción de los atributos

- **idProtocolo**: Clave foránea, que referencia al protocolo asociado.
- **idSec**: Dentro de un protocolo, indica a qué número de secuencia pertenece.
- **valorBoton**: Identificador del botón/luz. Valores posibles: [A-I].
- **secuenciaPrueba**: Indica si la secuencia a la que pertenece el botón es de prueba. En ese caso no se computan errores.

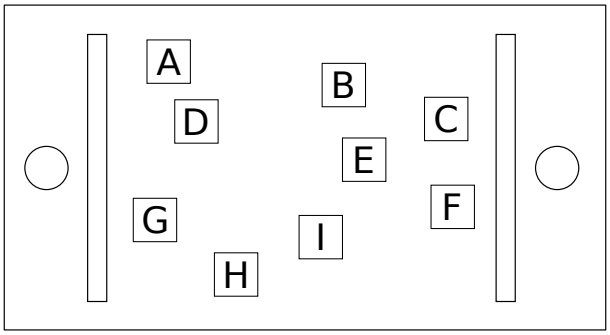

Fig. E.8. Ubicación de los botones/luces en el tablero.

Entidad Protocolo

Esta es la entidad principal ya que define todos los atributos de un protocolo, con múltiples posibilidades que permitirán ajustar su funcionamiento. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class Protocolo(models.Model):
        nombreProtocolo = models.CharField(max_length=100, default = 0)
        cantSecs = models.IntegerField(default=0)
        tiempoEncendidoLuz = models.IntegerField(default=500)
        tiempoEntreLuces = models.IntegerField(default=500)
        tiempoComienzaIntento = models.IntegerField(default=0)
        timeoutRespuesta = models.IntegerField(default=60000)
        cantErroresCorte = models.IntegerField(default=0)
        reverso = models.BooleanField(default = False)
        botonRojoIniciaTrial = models.BooleanField(default = True)
        luzComienzaSecuencia = models.BooleanField(default = True)
        colorLuzComienzaSecuencia = models.CharField(max_length=20, default = "blue")
        tiempoLuzComienzaSecuencia = models.IntegerField(default=500)
        repeticionLuzComienzaSecuencia = models.IntegerField(default=3)
        intervaloRepeticionLuzComienzaSecuencia = models.IntegerField(default=500)
        luzEsperaBotones = models.BooleanField(default = True)
        colorLuzEsperaBotones = models.CharField(max_length=20, default = "blue")
        luzAnalisisIntento = models.BooleanField(default = True)
        colorLuzAnalisisIntento = models.CharField(max_length=20, default = "yellow")
        feedbackLuces = models.BooleanField(default = True)
        colorSecuenciaCorrecta = models.CharField(max_length=20, default = "green")
        colorSecuenciaIncorrecta = models.CharField(max_length=20, default = "red")
        tiempoLuzFeedback = models.IntegerField(default=500)
        repeticionLuzFeedback = models.IntegerField(default=3)
        intervaloRepeticionLuzFeedback = models.IntegerField(default=100)
        luzEsperaProxSecuencia = models.BooleanField(default = True)
        colorLuzEsperaProxSecuencia = models.CharField(max_length=20, default = "blue")
        feedbackSonido = models.BooleanField(default = True)
        luzTerminaTomaDatos = models.BooleanField(default = True)
```

Código 12: Entidad Protocolo con sus valores por defecto.

La descripción de los atributos puede observarse en el apéndice D.1.

Entidad TomaDeDatos

Esta es la entidad que permitirá que los Usuarios puedan experimentar con un Protocolo. En general una Toma de Datos se realizará en una fecha determinada y con un grupo de usuarios previamente definidos. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class TomaDeDatos(models.Model):
        idProtocolo = models.ForeignKey(Protocolo, on_delete=models.CASCADE, default=0)
        nombreTomaDeDatos = models.TextField(max_length=50, default = "nombre")
        fechaCreacion = models.DateField(auto_now_add=True)
        ultimaFechaEvaluacion = models.DateField(null = True, blank=True)
```

Código 13: Entidad TomaDeDatos.

Descripción de los atributos

- **idProtocolo**: Clave foránea, que referencia al protocolo asociado.
- **nombreTomaDeDatos**: Cadena de caracteres con la que se identificará la instancia de Toma de Datos.
- **fechaCreacion**: Campo de fecha, con el momento en que es generada la Toma de Datos.
- **ultimaFechaEvaluacion**: Campo de fecha, en que la Toma de Datos es realizada por última vez. Esto permitirá realizar un ordenamiento de fechas en la vista para colocar a la última que fue realizada en el primer lugar.

Entidad Usuario

Es la entidad que modela a cada Usuario que realizará uno o más experimentos para una misma Toma de Datos. Por cuestiones de preservar la confidencialidad de las personas que realizan los experimentos, esta entidad no tiene campos de identificación personal. Si se quisiera que una misma persona realice más de un experimento con distintos Protocolos, se deberá cargar un Usuario específico para cada Toma de Datos. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class Usuario(models.Model):
        idTomaDeDatos = models.ForeignKey(TomaDeDatos, on_delete=models.CASCADE, default=0)
        nombre = models.CharField(max_length=20)
        cantIntentos = models.IntegerField(default=0)
```

Código 14: Entidad Usuario.

Descripción de los atributos

- **idTomaDeDatos**: Clave foránea, que referencia a la TomaDeDatos asociada.

- **nombre**: Cadena de caracteres con la que se identificará al usuario dentro de la TomaDeDatos.
- **cantIntentos**: Cantidad de veces que el Usuario ha realizado un experimento dentro de la misma TomaDeDatos y Protocolo.

Entidad Intento

Esta entidad modela cada una de las veces que un Usuario realiza una evaluación sobre un mismo Protocolo y TomaDeDatos. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class Intento(models.Model):
        idUsr = models.ForeignKey(Usuario, on_delete=models.CASCADE, default=0)
        erroresConsecutivos = models.IntegerField(default=0)
        timeStamp=models.DateTimeField(default=datetime.now)
```

Código 15: Entidad Intento.

Descripción de los atributos

- **idUsr**: Clave foránea, que referencia al Usuario asociado.
- **erroresConsecutivos**: Se registra los errores consecutivos que ha cometido el Usuario en un Intento para una TomaDeDatos y Protocolo. Este atributo se contemplará en caso de estar habilitado el criterio de corte en el Protocolo.
- **timeStamp**: Fecha y hora en que comienza el Intento.

Entidad EventoRegistrado

Esta entidad modela cada uno de los botones presionados por el usuario como respuesta de una secuencia. También incluye los eventos de control de inicio y finalización de cada intento. De esta forma, el sistema permite tener trazabilidad sobre las respuestas del usuario y las intervenciones del operador para cada experimento. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class EventoRegistrado(models.Model):
        idIntento = models.ForeignKey(Intento, on_delete=models.CASCADE, default=0)
        idSec = models.IntegerField(default=0)
        valorEvento = models.CharField(max_length=3)
        tipoEvento = models.CharField(max_length=7) #Boton/Control
        tiempoPresionadoMS = models.IntegerField(default = 100)
        timeStamp=models.DateTimeField(default=datetime.now)
        tiempoDesdeInicio = models.IntegerField(default = 0)
```

Código 16: Entidad EventoRegistrado.

Descripción de los atributos

- **idIntento**: Clave foránea, que referencia al Intento asociado.
- **idSec**: Dentro de un protocolo, indica a qué número de secuencia pertenece.
- **valorEvento**: Si es un botón tomará valores en [A-I]. Si es un evento de control podrá tomar los valores descriptos a continuación.
- **tipoEvento**: Puede tomar los valores "Boton" o "Control".
- **tiempoPresionadoMS**: En el caso de los botones, indica cuánto tiempo han sido presionados, medidos en milisegundos.
- **timeStamp**: Indica la marca de tiempo con el momento exacto en que se ha registrado el evento.
- **tiempoDesdeInicio**: Para facilitar el análisis de datos, se calcula y registra el tiempo que ha transcurrido desde que el sistema permite registrar las respuestas de una secuencia y el momento en que ha registrado este evento.

Valores de los Eventos de Control

- **RI**: Indica que el Usuario ha presionado el botón Rojo para dar Inicio;
- **RF**: Indica que el Usuario ha presionado el botón Rojo para dar Fin y confirmar su respuesta;
- **OI**: Indica que el Operador ha Iniciado manualmente, mediante el navegador, la siguiente secuencia;
- **OF**: Indica que el Operador ha Finalizado y validado manualmente la respuesta del Usuario mediante el navegador;
- **OAS**: Indica que el Operador ha Anulado la Secuencia manualmente mediante el navegador;
- **TO**: Indica que ha vencido el TimeOut definido en el Protocolo;
- **OTI**: Indica que el operador ha finalizado manualmente el Intento, mediante el navegador.
- **ITR**: Indica el Inicio de Tiempo Respuesta, que el es momento cero desde el cual se cuenta el tiempo en el que se comienza a registrar los botones que presiona la persona evaluada.

Entidad SecuenciaIntento

Esta entidad permite registrar si la respuesta del Usuario en un Intento y para una secuencia determinada dentro de una TomaDeDatos y Protocolo, ha sido correcta o incorrecta. A continuación se muestra la definición de la entidad en el archivo *models.py* y la descripción de cada atributo.

```
class SecuenciaIntento(models.Model):
        idIntento = models.ForeignKey(Intento, on_delete=models.CASCADE, default=0)
        idSec = models.IntegerField(default=0)
        correcto = models.BooleanField(default = True)
```

Código 17: Entidad SecuenciaIntento.

Descripción de los atributos

- **idIntento**: Clave foránea, que referencia al Intento asociado.
- **idSec**: Dentro de un protocolo, indica a qué número de secuencia pertenece.
- **correcto**: Valor booleano. Indica si la respuesta ha sido correcta (True) o incorrecta. (False).

E.1. Funcionamiento del Dispositivo

La persona que vaya a operar el experimento se conectará al Dispositivo desarrollado a través de un dispositivo con conexión inalámbrica (Notebook, tablet o smarthpone). Podrá conectarse con la red cuyo SSID es *CorsiWifi* y el password por defecto es *Corsi2019*. Una vez establecida la conexión deberá ingresar mediante un navegador web (Google Chrome, Firefox, Opera, etc.) a la url: **http://corsi.local/** o a la dirección IP: **http://192.168.50.10/**

La solicitud será respondida por el Framework Django que presentará una página inicial con un menú de opciones entre los que podrá seleccionar:

- Cargar en la base de datos un protocolo y una toma de datos con los usuarios asociados a través de la sección *Protocolos*.
- Explorar las tomas de datos realizadas, sus resultados y exportarlos a formato csv, también en la sección *Protocolos*.
- Acceder a la documentación sobre el uso del dispositivo, sección *Documentación*.
- Seleccionar un protocolo, toma de datos y usuario y comenzar un experimento, opción *Evaluar*.

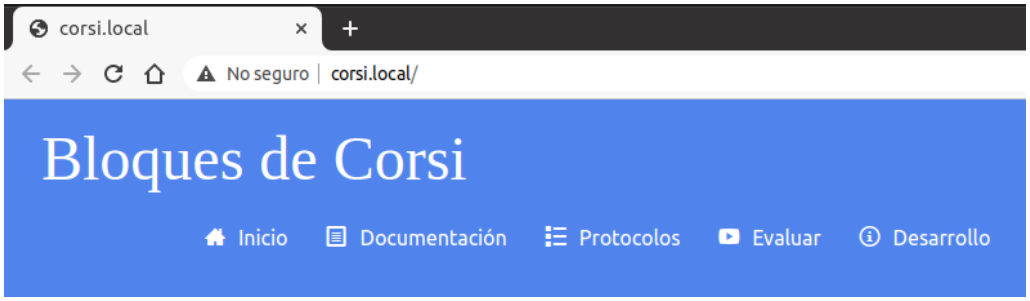

Fig. E.9. Pantalla principal - Menú de opciones. Vista desde una PC. Con una resolución de pantalla menor, como ser una tableta o teléfono inteligente, se presentará un botón que mostrará un menú desplegable.

Al comenzar una evaluación la aplicación Django se vinculará con los scripts de entrada/salida que permitirán el encendido de los leds con las secuencias del experimento así como las barras lumínicas laterales o el buzzer emisor de sonido según los parámetros de configuración del Protocolo. La aplicación quedará a la espera de la respuesta y confirmación por parte del usuario a través de los botones rojos laterales (chequeando la aplicación web mediante una rutina en javascript periódicamente la inserción de un registro en la base de datos) o a la confirmación o cancelación por parte de la persona que opera el sistema mediante los botones en el navegador, para pasar al template con los resultados de la secuencia realizada, pudiendo continuar con el siguiente o finalizando el experimento.

La persona que esté siendo evaluada se ubicará frente al dispositivo desarrollado, podrá observar el encendido de las secuencias y una vez que sea el turno de responder presionará

los botones de secuencia y confirmación. Un script de entrada/salida se encargará de registrar la pulsación de los botones y el registro en la base de datos de los mismos. A su vez, podrá continuar con el siguiente ensayo mediante los botones rojos de confirmación en caso de estar habilitados por la configuración del Protocolo.

Scripts de carga de Protocolos

Al seleccionarse la opción "Subir Protocolo", se invoca la URL */lista_protocolos/subir_protocolo/* que será capturada por la definición del **urls.py**, llamará al método *subir_protocolo()* del **views.py** y mostrará el template *protocolo_nuevo.html*. Este template, en primera instancia mostrará un campo de "Examinar" para seleccionar el archivo json del Protocolo.

Una vez que se haya seleccionado el archivo json y clickeado la opción "Subir" se seguirá el camino inicial pero esta vez el método *subir_protocolo()* del **views.py** invocará el script *scripts/levantoProtocoloJson.py*, quien será el encargado de chequear que el formato del json sea correcto y de la definición por defecto de los parámetros que no hubieran sido predefinidos, para luego subir el Protocolo a la base de datos.

En caso de que todo esté definido correctamente, el método *subir_protocolo()* del **views.py** hallará en la base de datos el protocolo cargado e invocará al template *secuencias/listado_secuencias.html* con los datos del Protocolo recientemente cargado. En caso de haber algún error en el json o en la carga, se mostrará nuevamente el template *protocolo_nuevo.html* informando el error correspondiente.

Scripts de carga de Toma de Datos

Al seleccionar la opción "Subir Toma De Datos", se invoca la URL */lista_protocolos/subir_toma_datos/* que será capturada por la definición del **urls.py**, llamará al método *subir_toma_datos()* del **views.py** y mostrará el template *toma_datos_nueva.html*. Este template, en primera instancia mostrará un campo de "Examinar" para seleccionar el archivo json de la Toma de Datos.

Una vez que se haya seleccionado el archivo json y clickeado la opción "Subir" se seguirá el camino inicial pero esta vez el método *subir_toma_datos()* del **views.py** invocará el script *scripts/levantoTomaDatosJson.py*, quien será el encargado de chequear que el formato del json sea correcto y subir la Toma de Datos a la base de datos.

En caso de que todo esté definido correctamente, el método *subir_toma_datos()* del **views.py** hallará en la base de datos la Toma De Datos cargada e invocará al template *usuarios/toma_datos_usuarios.html* con los datos de la Toma de Datos recientemente cargada. En caso de haber algún error en el json o en la carga, se mostrará nuevamente el template *toma_datos_nueva.html* informando el error correspondiente.

E.2. Ejecución de una evaluación - Toma de Datos

En el diagrama de la Fig. E.10 se muestra cómo interactúan los distintos métodos y módulos de Django con scripts de entrada/salida y la base de datos, en una ejecución de una evaluación donde se asumen los datos de Protocolo, Toma de Datos y Usuarios cargados. Para mayor claridad, agregamos las interacciones con el sistema que el/la operador/a mantiene mediante el navegador y la persona evaluada mediante el tablero del dispositivo.

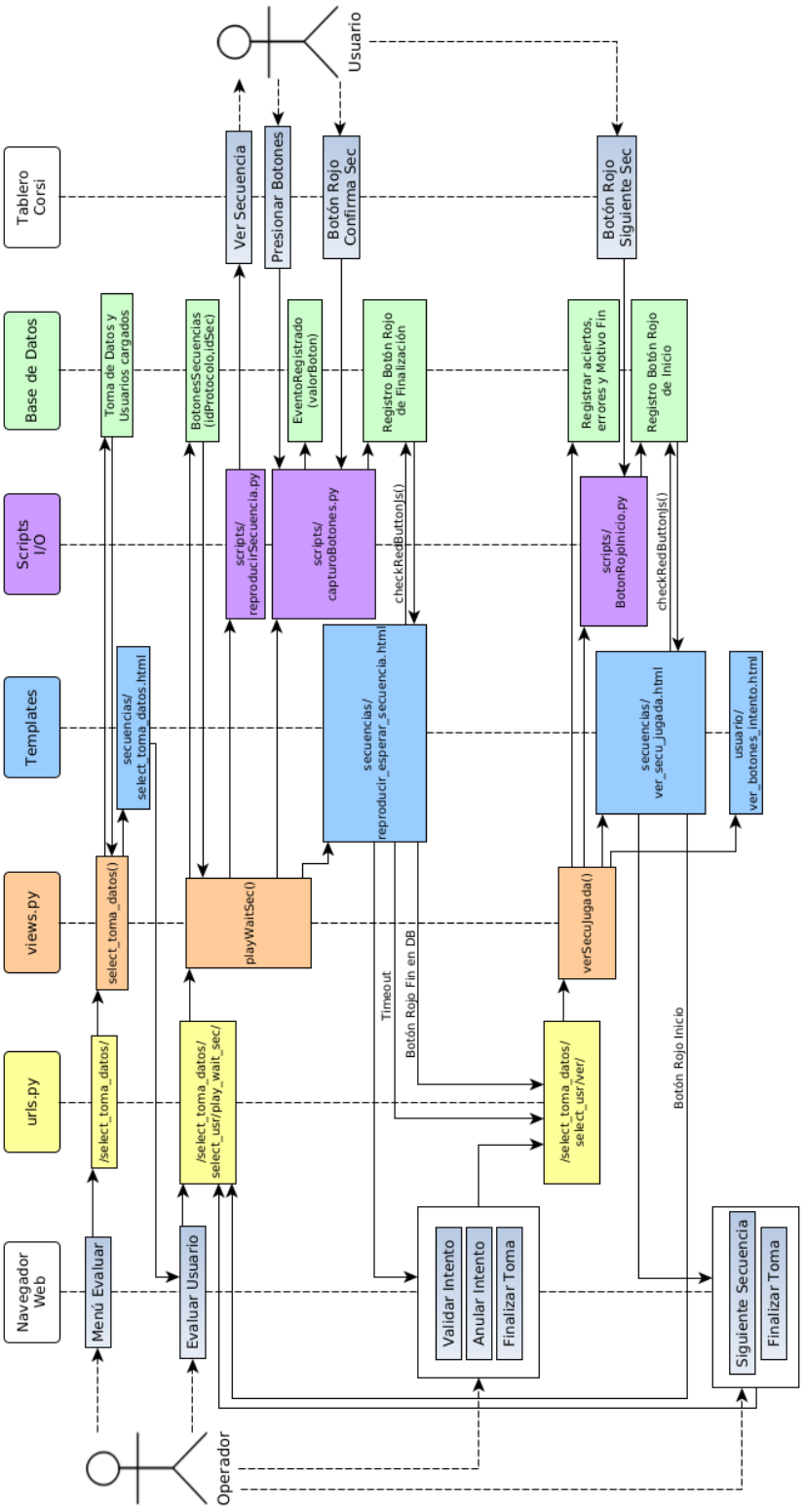

Fig. E.10. Sincronización de procesos durante una evaluación. Se puede observar cómo interactúa el/la operador/a con la aplicación web, y éstos con los distintos módulos de vistas y templates, así cómo con la base de datos. También la persona evaluada con el tablero del dispositivo y éste los scripts de entrada/salida.

La persona que opera el sistema podrá comenzar el experimento mediante la opción ***Evaluar*** del menú de la página principal.

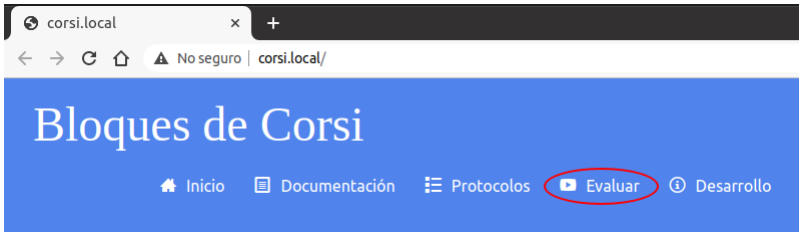

Fig. E.11. Pantalla principal - Opción "Evaluar". Permite seleccionar una Toma de Datos y Usuario para comenzar una evaluación.

Al hacer click sobre esta opción se llamará a la url */select_toma_datos/* que será capturada por el módulo **urls.py** e invocará la función *select_toma_datos()* del módulo **views.py**. Este procedimiento se comunicará con la base de datos para solicitar todas las tomas de datos cargadas (y los usuarios asociados). Con esta información se llamará al template */select_toma_datos.html* visualizando en la pantalla la última toma de datos cargada, con sus datos de usuarios y protocolo asociados, más un menú que permitirá seleccionar otras tomas de datos previamente cargadas.

Fig. E.12. Seleccionar Usuario y Comenzar Experimento.

Una vez hallado el Usuario precargado, el operador presionará el botón *"Evaluar"*, que redireccionará a la entrada */select_toma_datos/select_urs/play_wait_sec/* del urls.py, dando comienzo al ciclo principal de los experimentos, mediante la invocación al procedimiento ***playWaitSec()*** del views.py.

Este es uno de los procedimientos principales y tendrá varias funciones:

- Comunicarse con la base de datos para solicitar la secuencia a reproducir (leds a encender, tiempos, etc).
- Lanzar el script de I/O llamado ***reproducirSecuencia.py*** que será el encargado de manejar la interfaz con los Leds para mostrar la secuencia al usuario en el orden y con los tiempos indicados. Éste a su vez se encargará de encender las secuencias de leds RGBs con el código de colores predefinidos en el Protocolo, si las variables del Protocolo así lo habilitan.
- Lanzar el script de I/O llamado ***capturoBotones.py*** que será el encargado de registrar la respuesta del usuario, tanto de los botones de secuencia como de alguno de los botones rojos laterales que confirman la respuesta. Éste a su vez registrará los botones presionados en la base de datos.
- Por último, llamar al template *reproducir_esperar_secuencia.html* que será la vista del navegador para la persona que esté operando el sistema.

Fig. E.13. Vista del operador mientras se reproduce la secuencia y la persona evaluada responde.

Este template tendrá información de la secuencia que se está reproduciendo y presentará varias maneras de pasar a la próxima sección, según las opciones disponibles:

- Un script en Javascript que se encargará periódicamente de consultar en la base de datos si ha sido registrado un evento de "Botón Rojo Fin (RF)"que indica que el

usuario ha terminado de contestar la secuencia y confirmado la misma. En caso de encontrar el registro invocará a la url */select_toma_datos/select_urs/ver/*.

- Un script en Javascript que decrementará cada un segundo un contador predefinido en el Protocolo. Al llegar dicho contador al valor cero, se dará por terminado el tiempo disponible de respuesta y se invocará a la url */select_toma_datos/select_urs/ver/*.
- Un botón de "Validar Intento" que podrá presionar la persona que opera el sistema en caso de que la persona que está siendo evaluada omita confirmar con el botón rojo la secuencia de respuesta, evitando de esta forma tener que esperar a que se venza el Timeout. Esta acción también invocará a la url */select_toma_datos/select_urs/ver/*.
- Un botón de "Anular Intento" que podrá presionar la persona que opera el sistema en caso de determinar que la persona que está siendo evaluada no pudo completar la secuencia por algún motivo que no se desea registrar. Vale la pena aclarar que al presionar este botón no se evaluará la respuesta por ende no se computará como error en caso de que no sea correcta. Esta acción también invocará a la url */select_toma_datos/select_urs/ver/*.
- Un botón de "Finalizar Toma" que podrá presionar la persona que opera el sistema en caso de determinar que el experimento debe finalizar por algún motivo. Esta acción también invocará a la url */select_toma_datos/select_urs/ver/*.

Como se evidencia, todas las opciones del template *reproducir_esperar_secuencia.html* invocan a la url */select_toma_datos/select_urs/ver/* del urls.py, aunque con distintos parámetros según la opción mediante la cual haya sido invocado, que será registrada luego en la base de datos. Para ello, el urls.py invocará el método ***verSecuJugada()*** del views.py. Este es el segundo método principal en la ejecución de los experimentos y tiene la siguiente funcionalidad:

- Registrar en la Base de Datos el resultado de la respuesta del Usuario, comparando los botones presionados con la secuencia original y la opción por la cual se ha concluido la secuencia previa, en cualquier caso que no sea el que el usuario haya presionado el botón rojo de finalización.
- En el caso de que la secuencia realizada previamente sea la última de un experimento o en el caso de que el operador haya presionado el botón de "Finalizar Toma", el template a mostrar será el *ver_botones_intento.html* que mostrará un resumen de todo el experimento con las secuencias originales, las respuestas del usuario y el resultado (Correcto o Incorrecto). Ver Fig. E.15.
- En el caso de que haya una próxima secuencia para continuar el experimento, se lanzará el script de I/O *BotonRojoInicio.py* que permitirá registrar en la Base de Datos si el usuario presiona el Botón Rojo para continuar con una próxima secuencia.
- En este caso el template invocado será el *ver_secu_jugada.html*, que permitirá visualizar información sobre la secuencia recientemente realizada e información parcial sobre el estado del experimento (Secuencias restantes, errores acumulados, etc) y mediante la ejecución de la función en Javascript llamada *checkRedButtonJs()* podrá enviar la orden al navegador de dar llamada a la url

/select_toma_datos/select_urs/play_wait_sec/ (en caso de que el usuario haya presionado el botón rojo de siguiente secuencia) que comenzará nuevamente con la reproducción de la próxima secuencia. También podrá invocarse esta misma url si el operador presiona mediante el navegador el botón de "Siguiente Secuencia".

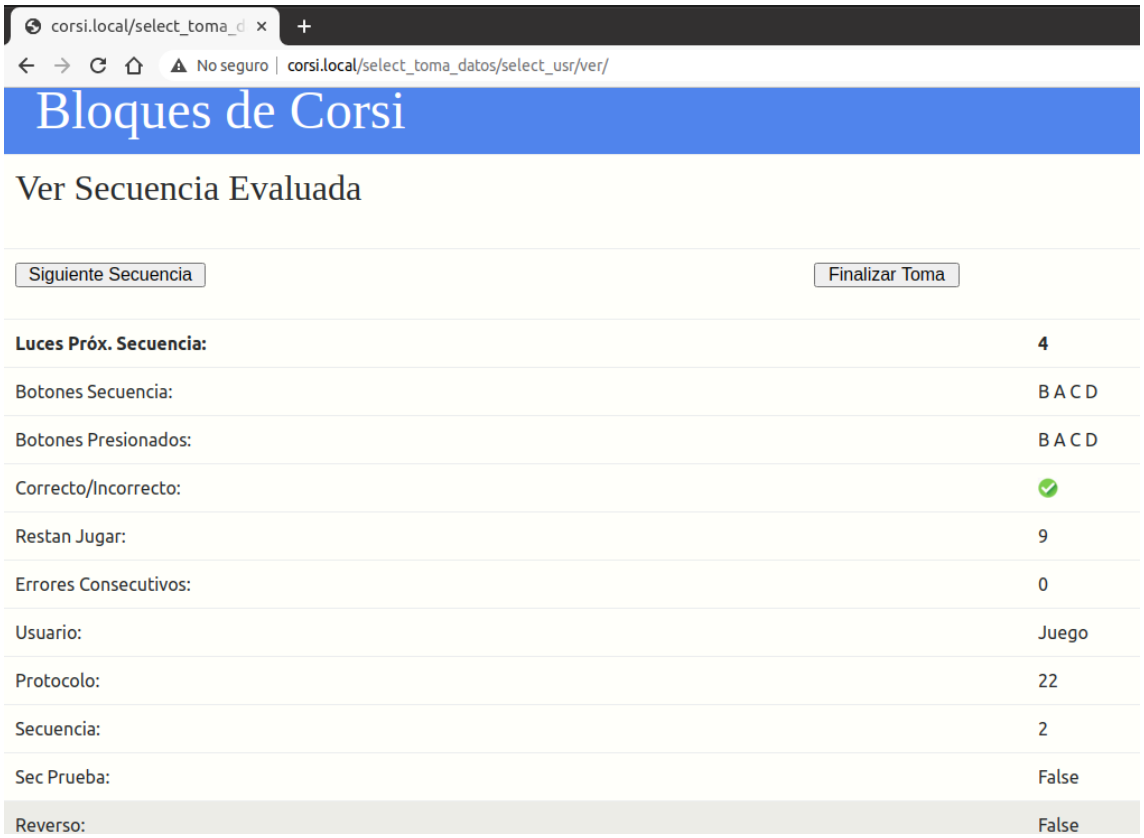

Fig. E.14. Vista de la persona que opera el dispositivo luego de que la persona evaluada ha finalizado la respuesta a una secuencia.

E.3. Visualización de resultados y exportación de datos

La Fig. E.15 permite analizar el resultado de una evaluación.

Fig. E.15. Al finalizar una evaluación se muestra un resumen mediante esta vista.

Como se puede deducir, en la columna de Correcto/Incorrecto, los círculos verdes con tilde blanca indicarán que la respuesta del usuario ha sido correcta, mientras que los círculos rojos con cruz blanca indicarán que la respuesta ha sido incorrecta. La persona que opera el sistema en esta instancia podrá seleccionar la opción de "Volver" para regresar al inicio o "Exportar CSV" para descargar los datos de la evaluación que ha realizado el Usuario.

En el caso de que la persona que opera el sistema decida exportar los resultados para su posterior análisis, deberá presionar el botón "Exportar CSV". Este invocará la url **/export_csv/** que será capturado por el *urls.py* invocando el método **export_csv()** del *views.py*. El mismo procesará los datos almacenados en la base de datos y permitirá descargar un archivo con formato csv con el formato de la Fig. E.16.

Usuario	idEvaluacion	numRegistroToma	idSec	timeStamp	Tipo Evento	Valor Evento	Tiempo Presionado	Tiempo Desde Inicio
Usr30	110	75	12	2019-12-29 01:45:09.115466	Control	OI	100	0
Usr30	110	76	12	2019-12-29 01:45:18.601680	Control	ITR	100	0
Usr30	110	77	12	2019-12-29 01:45:19.892524	Boton	C	300	907
Usr30	110	78	12	2019-12-29 01:45:20.859303	Boton	H	300	1874
Usr30	110	79	12	2019-12-29 01:45:21.733074	Boton	A	300	2748
Usr30	110	80	12	2019-12-29 01:45:22.702037	Boton	F	300	3717
Usr30	110	81	12	2019-12-29 01:45:23.709259	Boton	E	501	4724
Usr30	110	82	12	2019-12-29 01:45:24.623906	Control	RF	200	5639
Usr30	110	83	12	Correcto	True	['C', 'H', 'A', 'F', 'E']	['C', 'H', 'A', 'F', 'E']	5639
Usr30	110	84	13	2019-12-29 01:45:30.586901	Control	OI	100	0
Usr30	110	85	13	2019-12-29 01:45:41.817449	Control	ITR	100	0
Usr30	110	86	13	2019-12-29 01:45:44.976664	Boton	F	500	3017
Usr30	110	87	13	2019-12-29 01:45:46.297990	Boton	G	501	4338
Usr30	110	88	13	2019-12-29 01:45:47.462548	Boton	E	300	5503
Usr30	110	89	13	2019-12-29 01:45:48.789568	Boton	B	300	6830
Usr30	110	90	13	2019-12-29 01:45:50.767700	Boton	A	701	8808
Usr30	110	91	13	2019-12-29 01:45:56.064971	Boton	D	501	14105
Usr30	110	92	13	2019-12-29 01:45:57.803976	Control	RF	200	15844
Usr30	110	93	13	Correcto	False	['I', 'F', 'G', 'E', 'B', 'A']	['F', 'G', 'E', 'B', 'A', 'D']	15844
Usr30	110	94	14	2019-12-29 01:46:03.583108	Control	OI	100	0
Usr30	110	95	14	2019-12-29 01:46:14.586931	Control	ITR	100	0
Usr30	110	96	14	2019-12-29 01:46:17.038045	Boton	D	300	2305
Usr30	110	97	14	2019-12-29 01:46:18.135469	Boton	B	300	3403
Usr30	110	98	14	2019-12-29 01:46:20.300421	Boton	E	300	5568
Usr30	110	99	14	2019-12-29 01:46:21.786972	Boton	H	300	7054
Usr30	110	100	14	2019-12-29 01:46:22.624532	Boton	A	300	7892
Usr30	110	101	14	2019-12-29 01:46:23.652409	Boton	C	300	8920
Usr30	110	102	14	2019-12-29 01:46:24.663208	Control	RF	200	9931
Usr30	110	103	14	Correcto	False	['D', 'B', 'E', 'H', 'C', 'A']	['D', 'B', 'E', 'H', 'A', 'C']	9931

Fig. E.16. Vista la evaluación de un usuario, en formato CSV.