



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE PSICOLOGÍA
DIVISIÓN DE ESTUDIOS PROFESIONALES

INTERFAZ DE CONTROL EXPERIMENTAL EN DISPOSITIVOS ANDROID
BASADA EN ARDUINO (RATUINO MOBILE) COMO HERRAMIENTA DIDÁCTICA
PARA LA ENSEÑANZA DEL ANÁLISIS DE LA CONDUCTA

T E S I S

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN PSICOLOGÍA
PRESENTA:
CARLOS ALEXIS PÉREZ HERRERA

DIRECTOR DE TESIS

DR. ROGELIO ESCOBAR HERNÁNDEZ

REVISOR DEL PROYECTO

DRA. ALICIA ROCA COGORDAN

SINODALES

DR. ÁLVARO FLORENCIO TORRES CHÁVEZ

DR. ÓSCAR VLADIMIR ORDUÑA TRUJILLO

DR. CHRISTIAN LÓPEZ GUTIÉRREZ



MÉXICO, D.F.

ENERO 2015

ESTA TESIS FUE REALIZADA GRACIAS AL APOYO DEL PROYECTO PAPIIT TA300213-2



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

*A mi abuelo Rafael Herrera por todas sus enseñanzas, apoyo y cariño
que me ha brindado a lo largo de mi vida.*

Agradecimientos

Agradezco a mi tutor el Dr. Rogelio Escobar, por todas las enseñanzas, oportunidades, consejos, correcciones, y lecciones no sólo académicas sino personales que me ha compartido durante estos años.

Agradezco a la Dra. Alicia Roca y el Dr. Christian López por la confianza que depositaron en mi ofreciendome la oportunidad para continuar mi desarrollo profesional así como las observaciones para mejorar mi trabajo día a día.

Agradezco al Dr. Álvaro Torres y el Dr. Vladimir Orduña por todas las observaciones y comentarios a través de mi vida académica, sus palabras de aliento y observaciones a mis diferentes proyectos me ayudaron a crecer como estudiante.

Agradezco a Nadia y Ale por las pláticas que tuvimos durante mi carrera, sus comentarios me inspiraron e impulsaron a trabajar en este proyecto.

A mis compañeros de laboratorio Nadia, Mayela, Katya, Rodrigo, Ale, Varsovia, Jorge, Karina y Luis por hacer del laboratorio más que un recinto de conocimiento, un hogar donde los compañeros se vuelven familia.

A Liz por estar a mi lado, siempre con un abrazo y una sonrisa para darme los animos a seguir adelante, por compartir sus días conmigo y hacerme crecer como persona día a día.

A las personas que he tenido el honor de llamar amigos y que me han acompañado a lo largo de mi carrera ya sea con un consejo, un regaño, un abrazo o un chismazo como el Harry, Manuel, Janeth, Nestor y Bety.

Agradezco a mi mamá Pachis por el apoyo que me brindó para lograr terminar una carrera y a mi familia por generar el ambiente que moldeó mi comportamiento.

Y por último agradezco a todos los alumnos que emplearon las interfaces experimentales, gracias por su ayuda y comentarios.

Tabla de contenido

Resumen	1
Ratuino-Mobile: Interfaz de control experimental en dispositivos Android basada en Arduino para la enseñanza del análisis de la conducta.....	2
Parte I: Diseño de la interfaz Arduino – Android: Ratuino Mobile	15
. Comunicación de la interfaz.....	16
. Ratuino-Mobile	21
Parte II: Aplicación de la interfaz Ratuino Mobile como herramienta auxiliar en la enseñanza del análisis de la conducta.....	24
. Pruebas del equipo	24
. Resultados	35
. Discusión.....	40
Referencias.....	47
Apéndice A.....	52
. Código del microcontrolador Atmega328.....	52
Apéndice B.....	53
. Código del programa Ratuino-Mobile.....	53
Apéndice C.....	102
. Manual del usuario.....	102
Apéndice D.....	127
. Ratuino Mobile App: errores conocidos	127

Resumen

El uso de tecnología ha acompañado el desarrollo del análisis de la conducta desde sus inicios y ha favorecido la generación de nuevas formas de estudiar el comportamiento de los individuos. El uso de tecnología también ha sido importante para la enseñanza de la psicología que en gran parte se puede facilitar o limitar por los recursos de personal, espacio y herramientas tecnológicas con las que se cuenta. Por ejemplo, la automatización de la presentación de eventos experimentales y el registro conductual en las cámaras de condicionamiento operante por medio de interfaces de control no solo ha sido central en el desarrollo de la investigación de los principios básicos del comportamiento sino también para la enseñanza de la profesión en cursos de laboratorio. Sin embargo, los altos costos de los equipos comerciales de control experimental han limitado la expansión de laboratorios y, por tanto, de su uso dentro del salón de clases. El presente trabajo describe el diseño de una interfaz de bajo costo que consiste en una tarjeta Arduino en combinación con el sistema operativo Android mediante la aplicación Ratuino Mobile para el control de experimentos en condicionamiento operante así como diversas pruebas de desempeño para evaluar su precisión y estabilidad para el registro conductual en tiempo real. Adicionalmente se describe el posible uso de dicha interfaz como una herramienta didáctica que permita acercar a los estudiantes a la manera en la cual se realiza la investigación de los fenómenos conductuales, así como para familiarizarlos con el quehacer profesional del análisis de la conducta.

Palabras clave: Arduino, Android, instrumentación, análisis de la conducta.

Ratuno-Mobile: Interfaz de control experimental en dispositivos Android basada en Arduino para la enseñanza del análisis de la conducta.

Las herramientas tecnológicas han permitido una mejor comprensión de los fenómenos psicológicos gracias a que conforme estas se vuelven más precisas, así lo hacen también las formas de estudiar el comportamiento de los individuos (Lattal, 2008). Lattal notó la importancia de la interacción entre la tecnología endógena creada dentro del análisis de la conducta y la tecnología exógena que es tomada de otras disciplinas y adaptada para mejorar nuestro entendimiento de la conducta. Un ejemplo notable es la interacción entre las cámaras de condicionamiento operante y el equipo que permite automatizar la presentación de eventos y el registro de la conducta.

La cámara de condicionamiento operante también conocida como “caja de Skinner” (Skinner, 1979) es de las herramientas icónicas en psicología desarrolladas dentro de la tradición del análisis de la conducta. Con dicha herramienta pudieron mostrarse relaciones ordenadas entre el ambiente y la conducta de diversos organismos (e.g., Skinner, 1938) que no hubiera sido posible observar con las herramientas usadas previamente en psicología experimental como laberintos, corredores y cajas de salto que estaban basadas en la lógica del estudio de la conducta en ensayos discretos. Skinner logró la automatización de la presentación de eventos experimentales y el registro conductual por medio de relevadores electromecánicos conectados a motores y “*timers*” y el uso de registradores acumulativos (véase Skinner, 1956).

La combinación de la cámara de condicionamiento operante con el equipo de control y registro automático de la conducta fue central para estudiar sistemáticamente los principios de la conducta como el reforzamiento, el castigo, la extinción, la discriminación, la

generalización y el reforzamiento condicionado (véase Iversen, 1992). Adicionalmente, permitió el desarrollo de los programas de reforzamiento que permitieron estudiar patrones regulares de conducta a partir de cambios en reglas de presentación de los reforzadores (e.g., Ferster & Skinner, 1957).

Un aspecto crucial en el establecimiento de los primeros programas de estudios doctorales en análisis de la conducta en las Universidades de Columbia e Indiana a finales de la década de 1940, fue el entrenamiento de estudiantes en el uso de cámaras de condicionamiento operante con el propósito didáctico de mostrar el control conductual que se obtiene con esta herramienta y de entrenar a los estudiantes en el uso del equipo experimental (e.g. Keller & Schoenfeld, 1949; Guttman & Estes, 1949). Keller y Schoenfeld afirmaron que un nuevo *curriculum* de psicología en la Universidad de Columbia debía enfocarse en tres preguntas: qué estudiar, por qué estudiarlo y cómo estudiarlo. Las dos primeras preguntas estaban relacionadas con la importancia del condicionamiento operante para la psicología en general. La tercera pregunta estaba relacionada con la forma de los cursos. Respecto a los cursos de laboratorio mencionaron:

Rápidamente hemos concluido que el trabajo de laboratorio debe ser una parte integral en la instrucción básica. El estudio de la conducta en esencia es una ciencia experimental, y sentimos que los cursos basados en lecturas, o lecturas y demostraciones únicamente no son la forma óptima de enseñanza. Hemos acordado que el trabajo de laboratorio puede lograr una mejor comprensión de la ciencia y sus métodos. Es un acierto pedagógico que permite una participación más activa de los estudiantes en el proceso educativo. Aterriza de manera concreta los materiales de lectura y da un sentimiento de seguridad y comprobabilidad a los conocimientos que se aprenden. Además, una disciplina experimental necesita preparaciones de

laboratorio si se quiere lograr un avance en el conocimiento. No cabe duda, según hemos visto, que nuestra disciplina se beneficiará siguiendo a nuestras ciencias hermanas en este criterio. (p. 166, traducido por el autor).

Esta descripción hace evidente la importancia de los cursos de laboratorio en una disciplina experimental. Sin embargo, para realizar cursos de laboratorio es importante contar con el equipo de control que los estudiantes puedan operar.

En otro artículo, Frick, Schoenfeld y Keller (1948) describieron el uso de cámaras de condicionamiento operante y de las interfaces de control que podían usarse como herramienta didáctica en los cursos en la Universidad de Columbia. Este equipo fue diseñado específicamente para los cursos de laboratorio en la Universidad de Columbia y permitió a los estudiantes la oportunidad de replicar los hallazgos en los cuales se basan los principios que rigen el comportamiento. El equipo incluía una caja en la que se introducía a la rata, una palanca, un kimógrafo para registrar la conducta, un aparato de registro temporal con el cual podían medir latencias de respuesta y un dispositivo para la entrega de comida. Este laboratorio de enseñanza fue importante en el establecimiento del programa de la Universidad de Columbia como uno de los más exitosos en la historia del análisis de la conducta (véase e.g., Dinsmoor, 1990).

Posteriormente la idea de usar cámaras experimentales como un apoyo en la enseñanza de la psicología se extendió por otras universidades en los Estados Unidos (e.g., Guttman & Estes, 1949). Unos años después, los avances tecnológicos en electrónica permitieron el uso de equipo de control basado en transistores, conocido como “de estado sólido”, y eventualmente del uso de computadoras conectadas a interfaces comerciales de control experimental y registro conductual (Lattal, 2008).

A partir de la influencia de Sidney Bijou y Fred Keller, el análisis de la conducta empezó a extenderse a Latinoamérica en países como México y Brasil (Colotla & Ribes, 1981; Todorov, 2006). Sin embargo, la expansión a Latinoamérica fue problemática debido a la dificultad para conseguir el equipo de control experimental y las cámaras de condicionamiento operante. Martínez (2006) describió que las primeras demostraciones de condicionamiento operante en México realizadas por Ribes, fueron en cámaras de condicionamiento operante construidas con madera. Todorov (2006) describió que en Brasil

El operandum fue una pieza de metal enrollado de tal forma que cuando la rata presionaba un extremo, el otro extremo hacía contacto con una placa metálica; entonces el experimentador presentaba manualmente un tubo mojado para que la rata pudiera lamer el agua que se encontraba en él (p.31, traducción del autor).

En relación con dichos procedimientos, Matos (1996) mencionó que Keller les enseñó en Brasil que un buen profesor siempre “encuentra la forma”. Sin embargo, como también notaría Matos (1998), la construcción de cámaras de condicionamiento operante y especialmente del equipo para el control experimental, en la mayoría de las ocasiones, requiere de conocimientos específicos en electrónica. Adquirir este conocimiento, que no forma parte de la formación de los psicólogos, podrían consumir gran parte del tiempo de los académicos.

Martínez (2006) realizó un recuento histórico sobre el desarrollo de la psicología en México y en este se puede notar como la disponibilidad de espacios y equipo de investigación han influido fuertemente en el desarrollo de la enseñanza de la psicología ya sea con la creación de posgrados o mejorando la calidad de las clases sobre análisis de la conducta (véase también Lopes Miranda & Días Cirino, 2010, para una descripción similar en Brasil). Un ejemplo de esto es el proyecto encabezado por Ribes en la ENEP Iztacala (hoy FES

Iztacala), Ribes, Fernández, Rueda, Talento y López (1980), en su libro *Enseñanza y ejercicio de la investigación de la psicología. Un modelo integral*, argumentaron sobre la necesidad de contar con prácticas de laboratorio y en escenarios naturales que se asemejen al quehacer profesional de un analista de la conducta. Ribes impulsó fuertemente la creación de un modelo donde no sólo se busque las habilidades verbales necesarias sino también las habilidades conductuales para realizar investigación en análisis de la conducta.

Actualmente existen diversas empresas (e.g., Med Associates, Coulbourn, Lafayette) que comercializan interfaces de control y cámaras de condicionamiento operante para realizar experimentos en análisis de la conducta. Sin embargo no todos los centros de investigación pueden adquirir estos equipos debido a su alto costo (Escobar & Lattal, 2010). Este problema dificulta la creación de nuevos laboratorios y por ende la expansión del análisis experimental de la conducta. El escaso acceso a las interfaces de control experimental limita también la enseñanza de la psicología, debido a que en numerosas instituciones los cursos teóricos sobre aprendizaje no pueden complementarse con la replicación de los fenómenos en cursos de laboratorio.

Los avances recientes en electrónica han producido tecnología de bajo costo y fácil de usar que supera las limitaciones de las herramientas de registro utilizadas con anterioridad y que pueden adecuarse para registrar diferentes dimensiones del comportamiento así como presentar una amplia variedad de estímulos. Una de estas tecnologías son las tarjetas de microcontroladores. Estas tarjetas son dispositivos compactos equipados con microcontroladores que pueden ser programados para activar *outputs* (e.g., luces o motores) y detectar cambios en el ambiente con una variedad de sensores, *inputs*. Existen diversas tarjetas con microcontroladores disponibles alrededor del mundo (e.g., Arduino®, BASIC

stamp®, Parallax Propeller®), cada una con su particular entorno de desarrollo integrado y lenguaje de programación.

Palya (1988) enumeró una serie de nueve propiedades fundamentales para el equipo de control experimental: (1) Bajo costo, (2) fácil de construir y de conectar, (3) versatilidad para adaptarse a una variedad de situaciones experimentales, (4) fácil de aprender y fácil de programar, (5) alta confiabilidad, (6) bajo costo de mantenimiento, (7) capacidad de resistir la obsolescencia, (8) un adecuado ambiente para el usuario y capaz de tener herramientas complementarias, (9) portabilidad e intercambiabilidad entre laboratorios. Los 9 criterios propuestos por Palya (1988) se pueden resumir en cuatro criterios básicos fundamentales para generar equipo de registro auxiliar en la conducción de experimentos en condicionamiento operante los cuales son precisión, simplicidad, versatilidad y bajo costo.

La precisión es esencial para que el registro conductual refleje adecuadamente la conducta bajo estudio. La simplicidad se refiere a la facilidad con la que se logran construir o implementar los nuevos diseños sin conocimientos previos en electrónica, esto se logra utilizando herramientas o programas que sean comunes dentro de la comunidad de psicólogos o que sean fáciles de aprender. Por otro lado, la versatilidad es necesaria en el diseño pues este debe poder abarcar una amplia gama de experimentos que se puedan realizar con él y a la vez poder proporcionar la opción de una pronta modificación para situaciones muy específicas. El bajo costo en las interfaces alternativas es una necesidad imperante ya que no todos los centros de investigación tienen los recursos necesarios para adquirir las interfaces comerciales, además de que el bajo costo del equipo alternativo al comercial podría permitir la creación de nuevos centros de investigación con poca inversión económica.

Se han propuesto numerosas alternativas de bajo costo al equipo comercial. Un ejemplo notable es la tarjeta de control de Walter y Palya (1984), dicha tarjeta estaba

equipada con un microprocesador programado a través de ECBASIC. A pesar de que la tarjeta fue usada en numerosos estudios, requería de componentes especializados y era difícil de ensamblar para investigadores que no estaban familiarizados con la electrónica.

Otras alternativas para el control experimental han hecho énfasis en el puerto paralelo de las computadoras (e.g., Escobar, Hernández-Ruíz, Santillán & Pérez-Herrera, 2012; Escobar & Lattal, 2010; Escobar & Pérez-Herrera, en prensa; Goodman & Maskell, 1985; Santillán & Escobar, 2013; Stewart, 2006). Goodman y Maskell (1985), describieron la construcción de una caja llamada *parallel box* para su uso como herramienta de registro conductual. Dicha herramienta constaba de un cableado a través circuitos integrados de gran escala (LSI por sus siglas en inglés) para conectar el puerto paralelo de la computadora a switches digitales arreglados en un prisma rectangular que interactúan con el sujeto experimental. Esta interfaz permitía cambiar el arreglo experimental mediante código sin alterar el arreglo físico de los switches, permitiendo su uso en una variedad de experimentos. Sin embargo pese a su versatilidad en situaciones experimentales, eran necesarios conocimientos profundos sobre electrónica para poder ensamblar la interfaz.

Stewart (2006) empleó un nuevo diseño de la herramienta *parallel box*, apoyándose en Linux, para aumentar su precisión de registro. Mediante programación en dicha plataforma su diseño mostró precisión de milisegundos a través del registro con puerto paralelo. En otro estudio Escobar y Lattal (2010) mostraron la eficacia del puerto paralelo para registrar eventos experimentales mediante Visual Basic, esta interfaz ha sido modificada para registrar los sucesos experimentales en una cámara de condicionamiento operante (Escobar et al, 2012), así como para proporcionar soporte a diversos mecanismos de registro como palancas retráctiles basadas en unidades de disco de 3.5 pulgadas (*floppy disk*) (Escobar & Pérez-Herrera, en prensa) o la interrupción de un haz de luz entre un LED infrarrojo y un

fototransistor (Santillán & Escobar, 2013). Sin embargo, pese a la versatilidad y precisión de registro que se puede lograr con el puerto paralelo, hoy en día este componente de las computadoras se encuentra discontinuado, por lo que el acceso a equipo de cómputo que tiene este puerto es limitado.

Además de estas soluciones Canto, Bufalari & D'Ausilio (2011) implementaron el puerto USB para realizar experimentos que requieren mayor precisión temporal. En dicho estudio los autores emplearon un dispositivo similar a las *parallel box* (Goodman & Maskell, 1985) que se comunicaba con el puerto USB y registraba los eventos experimentales mediante el programa E-Prime. La descripción de la construcción de este dispositivo muestra que se requiere un conocimiento amplio sobre electrónica y métodos de comunicación entre dispositivos electrónicos para su ensamblado. En las pruebas realizadas por Canto et al. (2011) se observó que se puede lograr una precisión de milisegundos y sugirieron que la precisión podría aumentar con las configuraciones adecuadas.

Otros autores (e.g., Hoffman, Song, & Tuttle, 2007; Varnon & Abramson, 2014) propusieron sobrepasar las desventajas de la interfaz con el equipo de cómputo implementando microcontroladores, que regulen y graben los sucesos en experimentos específicos dotando así de mayor portabilidad a los aparatos de registro experimental. Hoffman et al. diseñaron un dispositivo al que llamaron *electronic operant testing apparatus* (ELOPTA). Este dispositivo basado en el circuito integrado PIC16f877A, es capaz de registrar y almacenar todos los eventos experimentales en una determinada cámara de condicionamiento operante de manera independiente al equipo de cómputo. Sin embargo esta independencia se logra a partir del uso de un arreglo único en la construcción de la cámara de condicionamiento operante. Hoffman et al. describieron con precisión la manera en que se debe construir la cámara de condicionamiento operante de la cual se hará cargo ELOPTA.

Sin embargo la modificación del diseño requiere vasto conocimiento en electrónica puesto que es necesario reconstruir las cámaras operantes y a que el circuito integrado PIC16f877A no puede ser reprogramado sin programadores especializados, lo cual limita la versatilidad de esta herramienta.

Por otra parte Varnon y Abramson (2014), propusieron la implementación de la tarjeta Parallax Propeller® para su uso en diferentes experimentos en psicología comparativa principalmente con invertebrados. Cada diseño consta de componentes y programación específica de la tarjeta, por lo que es una opción muy versátil para la conducción de experimentos de manera independiente a una computadora. Sin embargo la programación de estas tarjetas requiere de conocimientos en el lenguaje Spin y el entorno de desarrollo integrado (IDE) de propeller para poder crear nuevos experimentos.

Pérez-Herrera y Escobar (2012) propusieron una interfaz de control experimental y registro conductual a través de la combinación del IDE de Visual Basic y la tarjeta de código abierto Arduino UNO para la conducción de experimentos en el análisis de la conducta que podría cumplir con los cuatro criterios básicos de precisión, simplicidad, versatilidad y bajo costo. El lenguaje de Visual Basic es un lenguaje de programación usado frecuentemente en psicología (Cabello, Barnes-Holmes, O'Hara, & Stewart, 2002) y se ha descrito extensamente su uso en el análisis de la conducta para generar diversos programas de reforzamiento (Dixon & McLin, 2003). Además existe una versión gratuita (Visual Basic 2010 Express Edition) que puede descargarse de la página oficial de Visual Studio en la siguiente dirección: <http://www.visualstudio.com/downloads/download-visual-studio-vs#d-2010-express> Asimismo existen diversos recursos en línea que describen cómo emplearlo e incluso se puede buscar ayuda en el *Microsoft Developer Network* (<http://msdn.microsoft.com/es-mx/default.aspx>).

La tarjeta de código abierto Arduino UNO, es una tarjeta de prototipos con la cual se puede programar el microcontrolador Atmega328 a través de una adaptación del lenguaje C. La tarjeta Arduino UNO cuenta con catorce pines (conectores) digitales que pueden usarse a manera de *inputs* u *outputs* según lo requiere el programador, dos de estos pines digitales (el pin 0 y el pin 1) se utilizan para establecer comunicación serial con otros dispositivos. La tarjeta Arduino UNO cuenta también con seis pines analógicos para registrar o emitir señales analógicas, así como también cuenta con un resonador de cristal de 16 MHz que permite el registro temporal con una resolución de 4 microsegundos. La Figura 1, muestra la tarjeta Arduino UNO así como varios de sus componentes señalados.

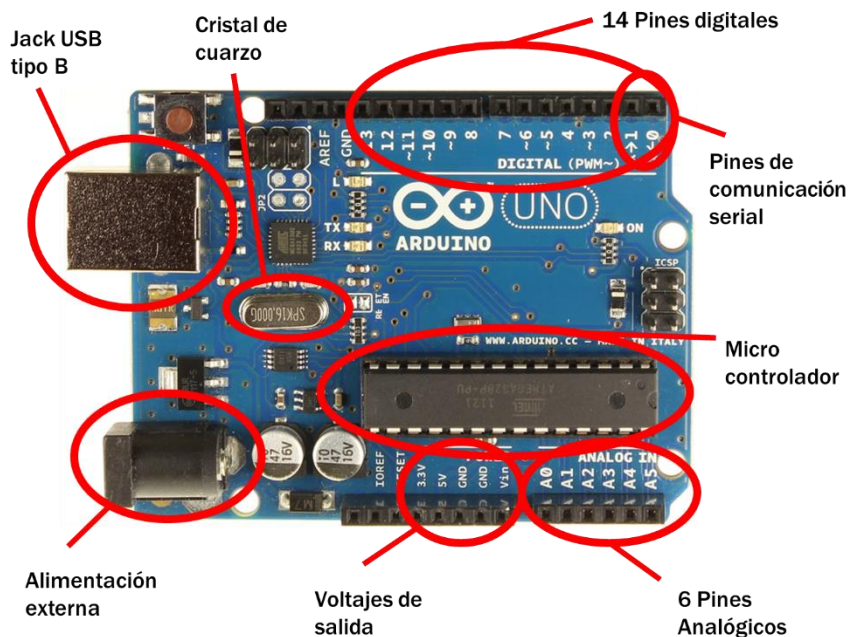


Figura 1. Esquema de la tarjeta Arduino UNO con los componentes principales señalados en rojo. Esta tarjeta de *hardware* abierto incluye un micro controlador Atmega328 y un resonador de cristal de 16 MHz.

La tarjeta Arduino UNO puede conectarse mediante un cable USB tipo-B (comúnmente usado en impresoras) a cualquier computadora ya sea para reprogramar el microcontrolador o para proporcionarle la energía necesaria para su funcionamiento. Esta tarjeta funciona con voltajes de 5 a 20 Vcd por lo que puede emplearse usando los 5 Vcd proporcionados por el puerto USB de las computadoras. La tarjeta Arduino UNO cuenta también con 3 pines de voltaje de salida para adecuarse a los diferentes componentes que pudieran utilizarse, por lo que puede enviar 3.3 Vcd, 5 Vcd o todo el voltaje proporcionado mediante el *jack* de alimentación integrado.

La tarjeta Arduino UNO, puede adquirirse en tiendas físicas o virtuales de electrónica especializada, por un costo menor a los \$400 pesos mexicanos y cuenta también con una extensa comunidad de usuarios que comparten proyectos y tutoriales ya sea en blogs, YouTube o en la página de *Arduino playground*. Pérez-Herrera y Escobar (2012)

describieron que es posible programar un código único para la tarjeta Arduino de tal forma que los usuarios únicamente tengan que modificar el programa en Visual Basic para realizar una amplia gama de experimentos sin necesidad de reprogramar la tarjeta Arduino.

La velocidad de comunicación entre la tarjeta y la computadora es un requisito importante para poder llevar a cabo experimentos que requieran una gran precisión temporal. Se ha mostrado que la tarjeta Arduino alcanza una comunicación eficiente y rápida (D'Ausilio, 2012). Pese a que la tarjeta Arduino puede controlar por sí misma la cámara de condicionamiento operante, Pérez-Herrera y Escobar (2012) describieron su uso en conjunto con una computadora para poder ofrecer una interfaz amigable con el experimentador, así como proporcionar la ventaja de permitir el funcionamiento de varias cámaras de condicionamiento operante con un solo programa. De esta manera la interfaz propuesta por Pérez-Herrera y Escobar (2012), cubre los requisitos de precisión, simplicidad, versatilidad y bajo costo antes planteados, por lo cual es una alternativa viable a las interfaces comerciales para conducir experimentos.

La interfaz descrita por Pérez-Herrera y Escobar (2012) incluso ha sido optimizada para generar diferentes programas de reforzamiento (Escobar & Pérez-Herrera, 2013a) y para su rápida implementación como herramienta de control experimental (Escobar & Pérez-Herrera, 2013b; Escobar & Pérez-Herrera, 2013c). Sin embargo, la programación en Visual Basic y el requisito de controlar la tarjeta Arduino UNO por medio de una computadora podrían limitar su uso en salones de clase en cursos de laboratorio debido a que los salones no siempre están equipados con computadoras ni con la instalación adecuada para conectar múltiples equipos al tomacorriente. Por tanto, una interfaz adecuada para emplearse en cursos de laboratorio debe funcionar sin la necesidad de una computadora y al mismo tiempo

presentar una interfaz amigable con el usuario para dar seguimiento a los eventos experimentales.

Una solución es reemplazar el uso de computadoras con dispositivos móviles como teléfonos y tablets. Gracias a los avances tecnológicos, estos dispositivos son cada vez más comunes y muchos de ellos tienen la capacidad de permitir el control de una cámara de condicionamiento operante y registrar eventos en tiempo real. Por lo tanto, el objetivo del presente trabajo es describir el diseño de una nueva interfaz para dispositivos móviles con sistema operativo Android basada en la tarjeta Arduino para realizar experimentos en condicionamiento operante, así como evaluar su precisión para registrar eventos experimentales dadas las condiciones de uso continuo que se producen en un salón de clase.

El presente trabajo se divide en dos partes, en la primera se describe el diseño de una interfaz de bajo costo para el control de experimentos en salones de clase para dispositivos Android en conjunto con cámaras de condicionamiento operante de bajo costo. Debido a que esta interfaz puede usarse con dispositivos conocidos como móviles o portátiles, no depende de la disponibilidad de computadoras por lo que es viable su uso en el salón de clase. En la segunda parte del trabajo se analiza la viabilidad de la interfaz como una herramienta de registro conductual capaz de mantener un funcionamiento adecuado pese a las horas continuas de trabajo y los menesteres propios de un salón de clase, permitiendo el desarrollo de habilidades conductuales tanto para la instrumentación de laboratorios, la conducción de experimentos en condicionamiento operante y el análisis de datos pertinente a las investigaciones realizadas.

Parte I: Diseño de la interfaz Arduino – Android: Ratuino Mobile

El sistema operativo Android se ha convertido en uno de los sistemas operativos para dispositivos móviles más populares que se encuentran actualmente en el mercado con más de 1,000 millones de dispositivos activados (<http://www.android.com>), su extensión se ha debido entre otras cosas al hecho de ser una plataforma de código abierto y a su amplio rango de precios que permite la adquisición de un equipo desde los \$1,299 pesos mexicanos.

Por otra parte la plataforma de código abierto Arduino, es una tarjeta prototípica que permite fácilmente programar un microcontrolador con un lenguaje de programación similar a C, esta tarjeta tiene integrados varios componentes útiles como lo son las entradas de voltaje, un Jack para USB y un cristal de cuarzo que permite registrar con precisión de microsegundos. Aunado a esto se pueden aumentar las capacidades de la tarjeta Arduino a través de la implementación de *shields*, los cuales son tarjetas que se acoplan directamente a la tarjeta Arduino y tienen funciones específicas no incluidas en la tarjeta Arduino.

Comunicación de la interfaz

Las interfaces de control que se emplearon en este estudio tienen varias similitudes con las interfaces descritas por Pérez-Herrera y Escobar (2012) por ejemplo la forma digital de transmisión de datos y la forma de comunicación y de control sobre la cámara de condicionamiento operante se mantiene constante, debido a que comparten la misma programación en el microcontrolador de la tarjeta Arduino.

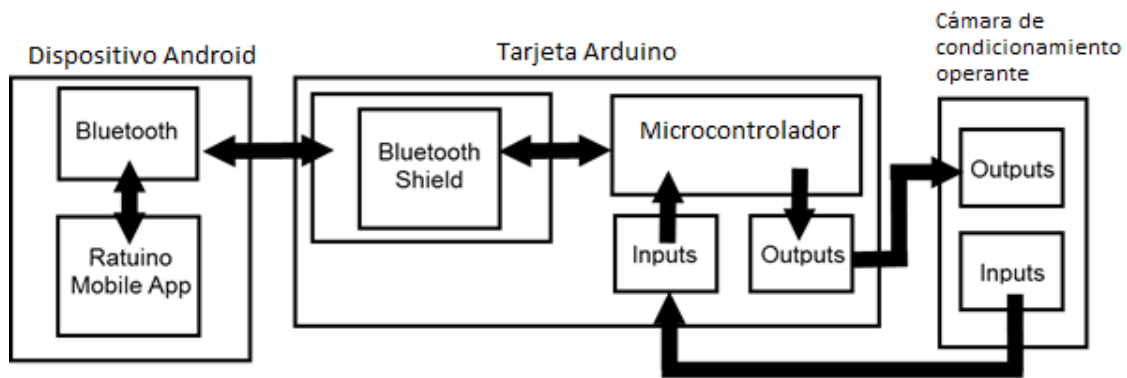


Figura 2. Esquema de la estructura de la interfaz.

La Figura 2 muestra un esquema de la estructura de la interfaz. Es importante notar que la aplicación Ratuino Mobile usa el protocolo Bluetooth para establecer comunicación serial de manera bidireccional con el microcontrolador en la placa Arduino por medio de un Shield Bluetooth (Itead Studio Modelo BT Shield 2.2 -Bluetooth to Serial Port Module Shield). El microcontrolador activa los *outputs* en la cámara de condicionamiento operante y detecta los cambios en los valores de los *inputs* digitales. A continuación se describen las características de la comunicación y las tareas específicas de cada componente de la interfaz.

La programación de la tarjeta Arduino se logra a través del IDE de Arduino a través de un lenguaje basado en C. Este código se carga previamente en el microcontrolador de la tarjeta Arduino y tiene el siguiente funcionamiento:

- i. Determina que pines se van a utilizar para el experimento y si estos pines permiten la entrada o salida de información, estos pines no pueden cambiar de función una vez que inicia el programa. Por lo tanto, la función de *input* u *output* se mantiene constante para todas las sesiones.
- ii. Revisa el valor lógico de cada pin de entrada, estos valores lógicos pueden ser únicamente 1 o 0 indicando que el circuito se encuentra cerrado (valor 1) o que se encuentra abierto (valor 0). Debido a que se emplea un micro interruptor conectado a una palanca estos valores representarían la ausencia (1) u ocurrencia (0) de una respuesta.
- iii. Por medio de comunicación serial, la información del valor lógico de los pines de entrada se envía al programa de interfaz con el usuario. Esta comunicación está programada de tal forma que permite el envío de caracteres (carácter “1” o carácter “0”). Esta programación facilita que el usuario pueda identificar la ausencia o presencia de una respuesta o alguna falla en la comunicación serial.
- iv. Busca información (instrucciones) en el canal de comunicación serial antes mencionado proveniente del programa de interfaz con el usuario. Esta comunicación utiliza caracteres alfanuméricos. En caso de existir una instrucción, esta produce una ejecución particular que puede ser: iniciar sesión, reforzar o terminar sesión. Cada una de estas ejecuciones consta de cambios en los pines de salida del microcontrolador. En caso de no existir una instrucción realiza el siguiente paso.

- v. Espera 50 milisegundos para asegurar que no exista un fallo en la comunicación y reinicia el ciclo.

El código que ejecuta los pasos i a v y que se utiliza en la tarjeta Arduino puede observarse en el Apéndice A.

Por su parte el dispositivo de control y registro (el dispositivo móvil con Android), contiene otro programa que, en el momento de comunicación con la tarjeta Arduino, realiza las siguientes acciones:

- i. Abre un espacio para la comunicación serial con la tarjeta Arduino.
- ii. Una vez abierto el espacio de comunicación comienza a cronometrar el tiempo que se mantiene abierto este canal de comunicación (tiempo total de la sesión).
- iii. Crea un archivo de extensión “.txt” en el cuál almacena los datos generales de la sesión, las acotaciones necesarias para su lectura y lo mantiene abierto para grabar los eventos experimentales.
- iv. Muestra un registro en pantalla del tiempo y respuestas desde el último reforzador (o el inicio de la sesión en caso de no haber un reforzador anterior).
- v. Muestra un registro en pantalla de las respuestas totales, el tiempo total y los reforzadores totales de la sesión.
- vi. Recibe los valores lógicos de tipo caracter de parte de la tarjeta Arduino, los convierte en variables numéricas y las compara con el valor lógico inmediatamente anterior para determinar si existe un cambio en el registro. Debido a la programación de la tarjeta Arduino, este cambio puede ser únicamente de 0 a 1 o de 1 a 0 lo que significa el inicio de una conducta o el final de ésta.

- vii. En caso de existir un cambio en el registro evalúa conforme a los criterios de reforzamiento si es necesario enviar un mensaje y de qué tipo al microcontrolador. Además escribe en el archivo de extensión “.txt” el tiempo en milisegundos desde el inicio de la sesión y un código que hace referencia al tipo de evento que sucedió en la sesión (el cual se podrá interpretar con las acotaciones del archivo).
- viii. Muestra los eventos que el experimentador puede realizar manualmente como forzar la culminación de la sesión o entregar un reforzador independiente de la conducta.
- ix. En caso de que se determine la entrega de un reforzador ya sea dependiente de la conducta del sujeto o de manera gratuita por el experimentador, reinicia los registros que toman en cuenta el último reforzador.
- x. Reinicia el ciclo a partir del punto vi.

Una vez que se han especificado las acciones que realizan los dos sistemas de registro digital con la tarjeta Arduino y con el programa en el dispositivo Android, se describirá brevemente el funcionamiento global de la interfaz. Debido a que ya se describieron las características propias de la comunicación, únicamente se hace referencia a los eventos en términos del experimento para facilitar su comprensión:

- i. La tarjeta Arduino determina la ocurrencia o ausencia de una conducta a través del sensor de entrada (la palanca).
- ii. La tarjeta Arduino envía la información recopilada al puerto de comunicación serial que a su vez se comunica con el dispositivo de control Android.

- iii. El dispositivo de control Android “evalúa” la conducta según los criterios de reforzamiento definidos por el experimentador, y le comunica a la tarjeta Arduino si debe o no reforzarse la conducta.
- iv. El dispositivo de control Android registra con una resolución de milisegundos los cambios en la conducta a la vez que la tarjeta Arduino activa los componentes para presentar los estímulos que defina el experimentador.
- v. El ciclo se repite hasta cubrir los criterios de tiempo o reforzadores definidos por el experimentador.

La comunicación serial se logra por medio de un dispositivo *Shield Bluetooth* (Itead Studio Modelo BT Shield 2.2 -Bluetooth to Serial Port Module Shield). En la siguiente dirección se encuentra una hoja de datos de este dispositivo: ftp://imall.iteadstudio.com/IM120417010_BT_Shield_v2.2/DS_IM120417010_BTShield.pdf. El *shield Bluetooth* se monta sobre la tarjeta Arduino alineando los conectores. Todos los *shields Bluetooth* están configurados de fábrica para usar el mismo nombre de dispositivo y la misma contraseña. Debido a esto, si se utiliza más de un Shield podría existir interferencia entre los dispositivos. Para evitar la interferencia en la comunicación fue necesario reprogramar los *shields* de tal forma que cada uno contara con un nombre y una contraseña específica. La reprogramación puede realizarse mediante el mismo IDE de Arduino con los comandos descritos en la hoja de datos del *Shield*.

• **Ratuino-Mobile**

La aplicación para dispositivos Android descrita en este trabajo puede descargarse de manera gratuita del sitio *Google play store* en la siguiente dirección electrónica:

<https://play.google.com/store/apps/details?id=com.Ratuino.ratuinomobile&hl=en>

La aplicación fue programada a través del IDE *Android Studio*.

Para el correcto funcionamiento de la aplicación el dispositivo móvil Android debe cubrir como mínimo con los siguientes requisitos de sistema:

- Sistema Operativo: Android Honeycomb 3.1 o Mayor
- Bluetooth 2.1 o mayor.
- Memoria flash interna o externa del dispositivo referenciada como memoria SD (por defecto todas las memorias internas de los dispositivos Android hacen referencia a esta como memoria SD).

Para realizar los experimentos utilizando esta interfaz Arduino-Bluetooth-Android el alumno debe, en primer lugar, enlazar su dispositivo móvil y el *shield Bluetooth* (la aplicación Ratuino Mobile permite trabajar y enlazar el dispositivo pero se recomienda realizar esta acción a través de los ajustes del dispositivo para poder tener un mejor desempeño). Posteriormente debe abrirse la aplicación, llenar el formulario de datos generales de la sesión, elegir el shield apropiado e iniciar la sesión.

Las aplicaciones de los dispositivos móviles a pesar de tener características similares a los programas de computadora poseen nombres diferentes. Para facilitar la lectura de este texto se utilizan los términos usados para describir programas de computadora pues es probable que más lectores estén familiarizados con estos términos. Sin embargo, también se mencionan los términos usados en aplicaciones Android enmarcados en paréntesis.

La aplicación Ratuino Mobile consta de tres pantallas sucesivas (una pantalla *Activity* con dos esquemas de pantalla asociados y un menú):

- i. La primer pantalla (primer esquema *Activity*) es un formulario para la recopilación de los datos generales de la sesión, tales como: nombre del sujeto, número de sesión, tiempo máximo de la sesión, número de reforzadores máximos de la sesión, programa de reforzamiento (elegible de entre los cuatro programas básicos: razón fija [RF], razón variable [RV], intervalo fijo [IF] e intervalo variable [IV]) y valor del programa. En el caso de los programas variables la aplicación también permite especificar el número de iteraciones necesarias para generar de manera automática valores conforme a la progresión de Fleshler y Hoffman (1962).
- ii. La segunda pantalla (el menú) es accesible en la parte superior del dispositivo. Se recomienda su acceso únicamente cuando se ha terminado de llenar el formulario (pues de lo contrario la aplicación podría presentar problemas de desempeño). Dentro del menú se muestra la lista de dispositivos Bluetooth disponibles o previamente enlazados con sus direcciones de comunicación. En esta pantalla se debe elegir el nombre del shield Bluetooth apropiado.
- iii. La última pantalla (el segundo esquema asociado al *Activity*) muestra el desarrollo de la sesión en cinco bloques:
 - a. El primer bloque muestra los datos de la sesión como criterio de reforzamiento, sujeto, número de sesión y la información que se está transmitiendo a través del puerto de comunicación. Esta línea permite observar el funcionamiento de la interfaz en tiempo real.
 - b. El botón de inicio de la sesión.

- c. El tercer bloque muestra las respuestas y el tiempo que ha transcurrido desde el último reforzador.
- d. El cuarto bloque muestra las respuestas totales, el tiempo total y el número de reforzadores obtenidos durante la sesión.
- e. El quinto y último bloque consta de dos botones. Uno permite al experimentador señalar y registrar la entrega de un reforzador independiente de la respuesta. El segundo botón permite cerrar la pantalla. Este botón sólo se habilita cuando la sesión concluye por los criterios establecidos por el experimentador.

Debido a los diferentes tamaños de pantalla que existen para los dispositivos móviles cada bloque está habilitado de manera independiente para recorrerse horizontalmente y de esta forma poder tener acceso a la información específica que se busque.

La interfaz inicia la sesión únicamente cuando el experimentador lo solicita, al presionar el botón de inicio de sesión (bloque b). Antes de iniciar la sesión debe verificarse que no existan problemas en la comunicación (observable en el primer bloque de la tercera pantalla). El botón de “inicio de sesión” se deshabilita una vez que se presiona. Para evitar la pérdida de datos, la interfaz no permite cerrar la sesión de manera rápida. En caso de suceder algún error se debe cerrar la aplicación por completo antes de volver a iniciarse. El código de la aplicación se muestra en el Apéndice B.

Parte II: Aplicación de la interfaz Ratuino Mobile como herramienta auxiliar en la enseñanza del análisis de la conducta

Para determinar la viabilidad de emplear la interfaz Ratuino-Mobile como una herramienta de registro en el aula para la enseñanza del análisis de la conducta, se realizaron una serie de pruebas en las cuales se utilizó dicha aplicación como parte de un curso de laboratorio. Durante el curso se determinó si la interfaz Arduino-Bluetooth-Android puede implementarse en escenarios educativos y puede soportar las continuas horas de uso dentro de un salón de clase.

. Pruebas del equipo

A continuación se describen las pruebas realizadas a la interfaz Ratuino Mobile, las primeras dos pruebas sirvieron para observar el funcionamiento del equipo en diferentes condiciones. La primera prueba se enfocó en la precisión con la que la interfaz puede registrar las respuestas en tiempo real y la segunda prueba se condujo para observar la estabilidad de desempeño de la interfaz mientras se usaba el dispositivo móvil para otras tareas. Estas pruebas de precisión y estabilidad fueron realizadas para determinar las capacidades y limitaciones de la interfaz. La última prueba consistió en usar la interfaz Ratuino Mobile como herramienta auxiliar en un curso de laboratorio para determinar su facilidad de uso y la estabilidad del equipo bajo uso continuo y en diferentes dispositivos.

.. **Precisión de registro**

La precisión de la tarjeta Arduino Uno para la investigación conductual ha sido determinada previamente (e.g, D'Ausilio, 2012). Sin embargo debido a que la interfaz empleada en este trabajo emplea comunicación serial a través de un *shield* Bluetooth, es posible que la precisión o la velocidad de transmisión sean alteradas. Por lo tanto se llevó a cabo una prueba de precisión. La prueba analizó el máximo número de respuestas por segundo registradas con precisión en una serie de 500 respuestas simuladas. Cada prueba fue realizada usando una tarjeta arduino autónoma programada para activar un relevador electromecánico (simulando una respuesta) y desactivarlo 500 veces sucesivas.

La duración de la respuesta simulada y el tiempo entre respuestas (TER) fueron cambiados paramétricamente desde 50 hasta 80 ms. Se realizaron 10 pruebas con cada combinación de duración y tiempo entre respuestas. Los cambios en el estado del relevador (las respuestas simuladas) fueron detectadas y registradas con la interfaz Ratuino Mobile en un Smartphone (Samsung Galaxy® Modelo S3 mini GT-18190 con Android 4.1 Jelly Bean). La Tabla 1 muestra los resultados de las pruebas. Se encontró que las 500 respuestas fueron detectadas con precisión cuando la duración de respuesta fue de por lo menos 70 ms con un TER de al menos 70 ms. Se detectaron una menor cantidad de respuestas cuando estas duraciones fueron disminuidas, por lo que se concluye que se pueden registrar de manera apropiada 7 respuestas por segundo.

Tabla 1. Media del número de respuestas simuladas detectadas durante 10 pruebas en cada combinación de duración de respuesta y tiempo entre respuesta. La desviación estándar se muestra entre paréntesis.

Duración de Respuesta (ms)	Tiempo entre respuestas (ms)			
	50	60	70	80
50	468.2(5.22)	470(5.16)	474.4(1.26)	474.2(0.78)
60	495.2(6.19)	490.8(10.79)	498(2.58)	498.4(2.45)
70	498.2(2.85)	498.8(2.52)	500(0)	500(0)
80	500(0)	500(0)	500(0)	500(0)

Esta resolución es suficiente para replicar una variedad de experimentos en condicionamiento operante. En los registros acumulativos mostrados por Ferster y Skinner (1957) una gráfica representaba la pendiente del registro asociada con 0.25, 0.5, 1 y 3 respuestas por segundo. Esta gráfica servía para facilitar el análisis de las tasas de respuesta a partir del registro acumulativo. Con palomas como sujetos, los programas de RV pueden producir tasas de 4 a 6 respuestas por segundo (Antonitis, 1977). Tasas tan altas de respuesta pueden ser registradas apropiadamente por esta interfaz. Sin embargo con programas de reforzamiento diferencial de tasa alta, la tasa de respuesta puede llegar a 15 respuestas por segundo (Skinner, 1957). En ese tipo de casos poco frecuentes, la interfaz Ratuino Mobile podría no ser la adecuada para el registro confiable de las respuestas.

.. **Estabilidad de desempeño**

La estabilidad de la aplicación a través de las situaciones cotidianas en un salón de clase es crucial para determinar su plausibilidad como herramienta didáctica. Si bien la estabilidad en el desempeño depende de diferentes recursos del dispositivo como la memoria RAM, el GPU, o el procesador, determinar el correcto funcionamiento de la aplicación en diferentes dispositivos puede auxiliar a generalizar la estabilidad del desempeño..

Para determinar la estabilidad de la aplicación, en un primer grupo de pruebas, se sometió a la aplicación en un dispositivo móvil (Sony Xperia E con Android 4.1 Jelly Bean) a una prueba utilizando aplicaciones comunes para los dispositivos móviles, la aplicación corrió tanto en primer como en segundo plano para cada prueba.

Se registró una llamada entrante, una llamada saliente, el cambio de aplicación, un mensaje entrante (whats app, Facebook, y SMS) y un mensaje saliente (whats app, Facebook y SMS), así como abrir una nueva aplicación que no requiere comunicación externa al dispositivo (contactos) y una que se comunique con internet (YouTube).

Las pruebas consistieron en emplear la aplicación Ratuino Mobile en primer o segundo plano y activar un micro *switch* conectado a la tarjeta Arduino manualmente para simular respuestas durante el uso de las aplicaciones. Se encontró que las llamadas entrantes y salientes generaron una respuesta en el registro al iniciar y al finalizar la llamada, sin embargo la aplicación funcionó normalmente mientras duró la llamada. No se encontró ningún otro problema de registro con respecto al resto de las aplicaciones, aunque algunas veces al regresar la aplicación a primer plano, en una posición de pantalla diferente a la inicial (de horizontal a vertical o viceversa), hacía que los datos en pantalla no se mostraran por un tiempo o la aplicación se detuviera. A partir de estas pruebas pudo concluirse que para

mejorar la estabilidad del sistema es necesario deshabilitar la opción de llamadas entrantes y evitar cambiar de aplicación o ejecutar nuevas aplicaciones mientras se está realizando una sesión experimental.

.. Uso en el salón de clases

... Método

... Participantes

Participaron 12 estudiantes de 5º semestre de la licenciatura en psicología. Los estudiantes formaron parte de un grupo perteneciente a la materia Aprendizaje Motivación y Cognición I impartida en la Facultad de Psicología, UNAM. Los participantes se dividieron en equipos de 4 participantes y a cada equipo se le asignó como sujeto experimental una rata Wistar macho de aproximadamente un año de edad. Las ratas fueron obtenidas del bioterio de la Facultad de Psicología de la UNAM y se mantuvieron en la colonia del Laboratorio de Condicionamiento Operante en un ciclo de 12 horas de luz y 12 horas de oscuridad. Las ratas tuvieron acceso libre al agua durante todo el desarrollo de las prácticas, a excepción del tiempo dentro de la cámara de condicionamiento operante, y se les limitó el acceso a la comida para mantenerlas al 80% de su peso ad libitum.

... **Aparatos**

Cada equipo de 4 participantes construyó su propia interfaz de control, utilizando como cámara de condicionamiento operante una jaula para roedores de diversos modelos con dimensiones de 30 cm de largo por 20 cm de ancho y 20 cm de altura, aproximadamente. La separación entre cada barrote fue menor de 2 centímetros. Dentro de cada jaula se colocó una palanca tipo “*joystick*”. El desplazamiento de la palanca hacia abajo cerraba un micro interruptor. La base sobre la cual se apoyó la palanca fue de 8 cm de largo. La palanca tenía un largo de 9 cm y se colocó a un centímetro del piso de la cámara de condicionamiento operante. A un costado de la palanca se colocó una charola de plástico en la que se dejaron caer bolitas de comida de 25 mg. Las bolitas de comida se fabricaron remoldeando comida para ratas. En la Figura 3 se muestra la fotografía de una de las cajas empleadas por estudiantes.

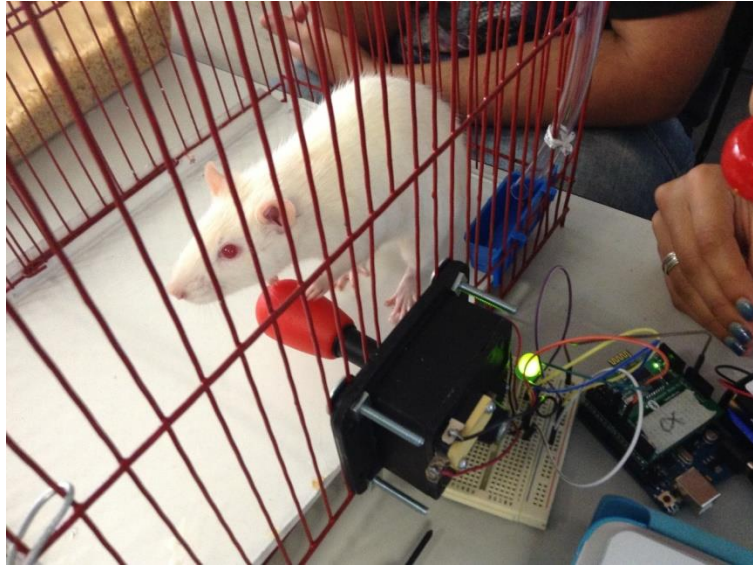


Figura 3. Fotografía de la jaula adecuada para su funcionamiento como cámara de condicionamiento operante. Conectado a la palanca puede observarse el circuito que compone la parte física de la interfaz

Se empleó, además de la palanca, un *buzzer* de 3,7 kHz, de 3 a 20 V CD, con tono de pulso lento de 90 dB para señalar la entrega de cada bolita de comida. Las bolitas de comida fueron entregadas manualmente por los estudiantes al escuchar el tono. Se usó un LED “gigante” de 10 mm, color verde difuso el cuál sirvió para señalar el correcto funcionamiento de la interfaz.

Los dispositivos de la cámara de condicionamiento operante se conectaron a una tarjeta Arduino UNO R3 previamente acoplada al shield Bluetooth. El cableado de todos los instrumentos que conformaron la parte física de la interfaz se realizó a través de cables jumper macho-macho, por medio de una placa de prototipos (*protoboard*) de 1 bloque y 2 tiras, con ensamble a presión y 400 perforaciones. Las conexiones se realizaron de acuerdo con diagrama electrónico que se muestra en la Figura 4, y el diagrama físico que se muestra en la Figura 5.

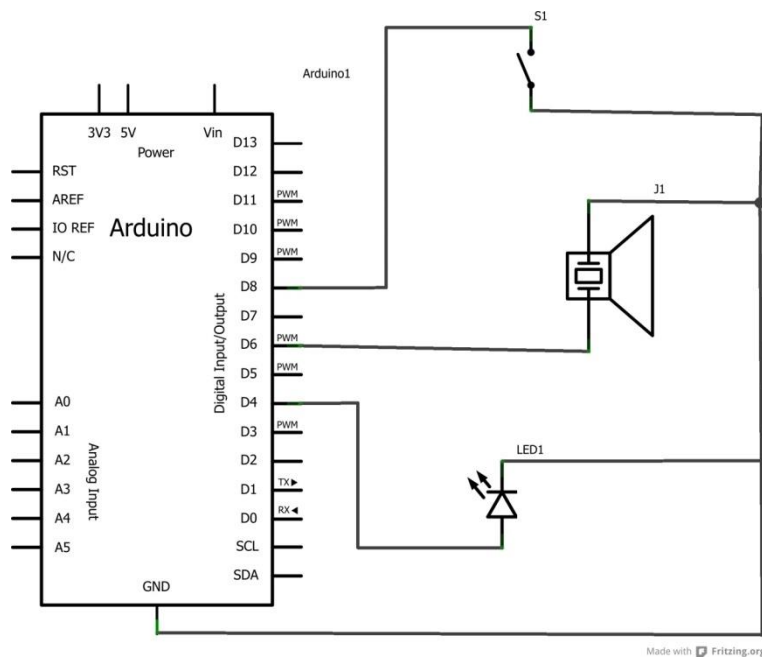


Figura 4. Diagrama electrónico de los componentes de la parte física de la interfaz.

Se emplearon como fuente de alimentación 4 pilas AA las cuales entregaron un total de 6 V CD en conjunto. Éstas se guardaron en un porta pilas que se conectó a la tarjeta Arduino con un adaptador para conector macho de 2.1 mm con centro positivo.

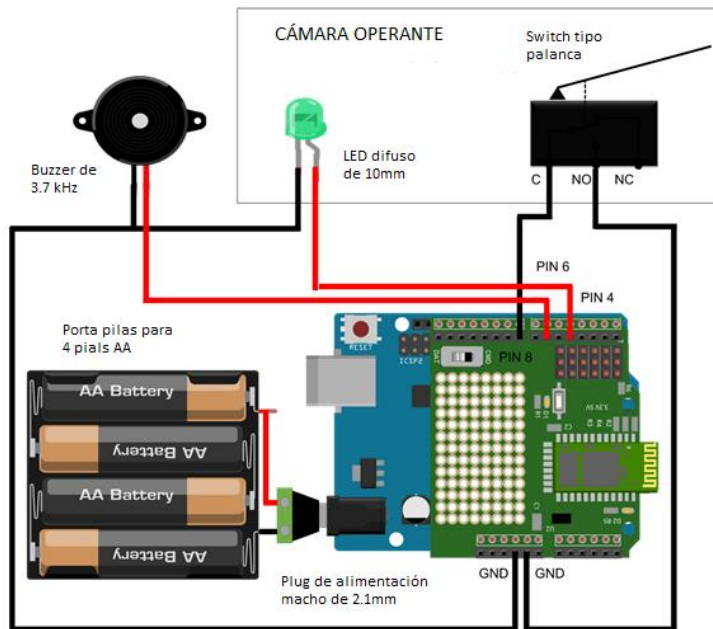


Figura 5. Diagrama físico de las conexiones entre la tarjeta Arduino con el Shield Bluetooth, y los dispositivos en la cámara de condicionamiento operante. En el diagrama se observa el LED que opera para informar sobre el correcto funcionamiento de la interfaz, la palanca para registrar la conducta del sujeto, un Buzzer para señalar cuando debía entregarse una bolita de comida manualmente. Los inputs y outputs se encuentran conectados al pin digital correspondiente y a la tierra en el Shield Bluetooth. Se emplearon 4 pilas AA para alimentar la interfaz.

... Procedimiento

Acorde con el temario de la materia Aprendizaje, Motivación y Cognición I, la parte del curso de laboratorio consistió de una clase de 2 horas de duración a la semana, durante las 16 semanas de duración del semestre. Durante este periodo se les enseñó a los estudiantes a armar la cámara de condicionamiento operante, a utilizar la interfaz Arduino-Bluetooth-Android para generar programas de reforzamiento simples y a realizar análisis de datos a través del programa descrito por Pérez-Herrera y Escobar (2013) disponible en la siguiente dirección:

<https://mega.co.nz/#!c5dQTDyZ!myYhKjvzyv1ViQkTYz4eT59j0kZmpo0UR0S6jbfj1DA>

En la Tabla 2 se muestra el cronograma de actividades realizadas durante el curso de laboratorio.

Tabla 2. Cronograma de actividades a través de los 16 días de clase.

Cronograma de actividades	
Número de sesión (día de clase)	Actividad
Día 1	Curso introductorio a la historia de las interfaces de control experimental, así como de la forma en la que funcionan las interfaces modernas y en específico la interfaz Ratuino
Día 2	Elaboración de la cámara experimental
Día 3	Sesiones de Moldeamiento
Día 4	Sesiones de Moldeamiento
Día 5	Sesiones de programas de RF1. RF2, RF5, RF7, RF10.
Día 6	Sesiones de programas de RF1. RF2, RF5, RF7, RF10.
Día 7	Sesiones de programas de RF10, RF7, RF5, RF2, RF1.
Día 8	Sesiones de programas de RF10, RF7, RF5, RF2, RF1.
Día 9	Sesiones de programas de RV y Razón Progresiva.
Día 10	Sesiones de programas de IF 1 s, IF 10 s, IF 20 s, IF 30 s, IF 45 s, IF 60 s.
Día 11	Sesiones de programas de IF 1 s, IF 10 s, IF 20 s, IF 30 s, IF 45 s, IF 60 s.
Día 12	Sesiones de programas de IF 60 s, IF 45 s, IF 30 s, IF 20 s, IF 10 s, IF 1 s.
Día 13	Sesiones de programas de IF 60 s, IF 45 s, IF 30 s, IF 20 s, IF 10 s, IF 1 s.
Día 14	Sesiones de programas de IV.
Día 15	Explicación del proceso de análisis de datos mediante el conjunto interconectado de hardware y software libre Ratuino.
Día 16	Entrega de trabajos Finales.

En la primer sesión se explicó brevemente a los estudiantes la historia de las cámaras operantes como herramientas de registro conductual para la conducción de experimentos en análisis de la conducta, así como también se explicó la manera en que se registran los sucesos experimentales tanto por las interfaces comerciales como por la interfaz que se utilizó durante el curso.

La segunda sesión fue dedicada a la elaboración de las cámaras de condicionamiento operante por parte de los alumnos. Se les explicó a los estudiantes como debían realizar el cableado, luego se modeló la construcción de la interfaz y se les entregó el equipo necesario para realizar la acción ellos mismos. Se proporcionó retroalimentación a cada equipo, hasta que todos los equipos lograran construir apropiadamente su cámara de condicionamiento operante (véase Miltenberger, 2008 para una descripción del entrenamiento de habilidades conductuales).

La tercera y cuarta sesiones fueron destinadas al moldeamiento de la conducta blanco de la rata (presionar la palanca) por parte de los estudiantes.

Una vez moldeada la conducta de palanqueo se expuso a los sujetos experimentales en diferentes programas de reforzamiento, inicialmente fueron programas de razón (RF 1, RF 2, RF 5, RF 7, RF 10, RV 5), posteriormente programas de intervalo fijo (IF 1 s, IF 10 s, IF 20 s, IF 30 s, IF 45 s, IF 60 s).

Una vez que se registraron los datos, se instruyó a los alumnos sobre cómo emplear el *conjunto interconectado de Hardware y Software libre Ratuino* (Pérez-Herrera & Escobar, 2013), para realizar el análisis de los datos de acuerdo a las hojas de datos proporcionadas por la aplicación.

. **Resultados**

Para evaluar el valor percibido, la efectividad y la facilidad de uso de la interfaz se les entregó un cuestionario a los estudiantes después de terminado el curso, dicho cuestionario constó de 6 preguntas y fue diseñado específicamente para evaluar la interfaz.

Los estudiantes calificaron cada elemento utilizando una escala tipo Likert de 1 (totalmente en desacuerdo) a 5 (totalmente de acuerdo). El cuestionario se envió a los participantes por correo electrónico y se esperó su respuesta. Únicamente 10 de los 12 participantes regresaron el cuestionario. La Tabla 3 muestra los ítems, la media, la desviación estándar y el rango para cada elemento.

Tabla 3. Media, desviación estándar y rango para los reactivos tipo Likert contestados por los alumnos sobre el uso de la interfaz Ratuino Mobile en el aula.

ítem	Pregunta	Media	Desviación Estándar	Rango
1	Usar ratas en el curso de laboratorio fue importante para mejorar el aprendizaje de los temas vistos durante clase	4,6	0,67	3-5
2	Usar programas de reforzamiento fue importante para mejorar el aprendizaje de los temas vistos durante clase	4,8	0,40	4-5
3	Trabajar con equipo electrónico y ensamblarlo yo mismo fue importante para mejorar el aprendizaje de los temas vistos en clase	4,6	0,67	3-5
4	Me gustó trabajar con equipo electrónico y ratas en el curso de laboratorio	4	1,38	1-5
5	Cursos de laboratorio con equipo electrónico y ratas son mejores que otros cursos basados en exposiciones orales de los alumnos	4,7	0,90	2-5
6	Fue difícil usar equipo electrónico en combinación con tablets y Smartphone	2,9	1,58	1-5

Los ítems 1 y 2 evaluaron la importancia percibida de los cursos de laboratorio en los que se emplean ratas como sujetos y se practica presentando programas de reforzamiento en cámaras de condicionamiento operante. En respuesta a estos ítems todos los participantes reaccionaron positivamente al curso de laboratorio. En respuesta el ítem 3, los estudiantes reportaron que el uso de equipo electrónico es importante para mejorar el aprendizaje durante la clase ($M=4.6$, $SD=0.67$, rango 3 -5). En respuesta al ítem 4, 80% de los estudiantes

reportaron que trabajar con equipo electrónico fue una experiencia positiva ($M=4$, $SD=1,38$, rango1 -5). Nueve estudiantes reportaron que trabajar con equipo electrónico y ratas fue mejor que otras técnicas, por ejemplo presentaciones orales de los alumnos usadas comúnmente en cursos similares ($M=4.7$, $SD=0,90$, rango2 -5). Finalmente, 50% de los estudiantes reportaron que no fue difícil trabajar con la interfaz ($M=2,9$, $SD=1,58$, rango1 -5).

Se calculó el número de sesiones en las cuales se encontraron problemas en el desempeño del equipo en relación con las sesiones totales. Cada equipo de alumnos contó con un dispositivo móvil diferente por lo que cada equipo tuvo un desempeño diferente con la aplicación. La Tabla 4 muestra la cantidad de sesiones totales producidas por el dispositivo, así como el número de sesiones en las cuales el dispositivo tuvo algún problema de desempeño y el porcentaje de sesiones exitosas llevadas a cabo con cada dispositivo.

Tabla 4. Porcentaje de sesiones concluidas exitosamente por dispositivo

Dispositivo	Sistema Operativo	Bluetooth	Procesador	RAM	Sesiones Totales	Sesiones con Fallas	Porcentaje de Sesiones exitosas
Tablet Galaxy Samsung 10.1	4.0.4 ICE CREAM SANDWICH	V3.0	Cortex A9 dual-core 1GHz	1GB	52	2	96%
Celular Xperia U	4.1 JELLYBEAN	V2.1	Cortex A9 dual-core 1GHz	512MB	54	17	69%
Celular Samsung Galaxy III mini	4.1 JELLYBEAN	V4.0	Cortex A9 dual-core 1GHz	1GB	46	7	85%
TODOS LOS DISPOSITIVOS					152	26	83%

Como puede observarse el dispositivo con las especificaciones más bajas, (512MB RAM, Bluetooth 2.1, Procesador 1GHz) fue el dispositivo con mayor número de sesiones fallidas, mientras que en el dispositivo con las especificaciones más altas (1GB RAM, Bluetooth 3.0, Procesador 1 GHz) ocurrieron muy pocas sesiones con problemas en el desempeño. Hay que notar también que el dispositivo con mejor desempeño era una tableta y no un celular como lo eran los otros dos dispositivos.

En general de las 152 sesiones llevadas a cabo como parte de las pruebas del equipo, el 83% de ellas ocurrió con éxito. Es importante notar que la mayoría de las sesiones fallidas ocurrieron con un mismo dispositivo.

Con base en los pasos descritos a los alumnos para utilizar la interfaz Arduino – Android se elaboró un manual de usuario para facilitar el uso del equipo en estudios o en cursos de laboratorio futuros. Este manual muestra una serie de diagramas e imágenes que se espera faciliten el uso del equipo por usuarios sin experiencia con equipo de control experimental. Este manual se muestra en el Apéndice C.

. **Discusión**

El objetivo del presente trabajo fue describir el diseño de una nueva interfaz para dispositivos móviles Android basada en la tarjeta Arduino (Ratuino Mobile) para realizar experimentos en condicionamiento operante, así como evaluar su estabilidad para registrar eventos experimentales dadas las condiciones de uso continuo que se producen en un salón de clase. Su bajo costo, su precisión de registro, su estabilidad en cuestiones de desempeño, y el reporte de los alumnos concerniente a la experiencia son parámetros a tomar en cuenta al momento de decidir su viabilidad como herramienta de registro conductual en el salón de clases.

La interfaz Ratuino Mobile descrita en el presente trabajo, es una herramienta de bajo costo basada en código abierto, por lo cual las partes necesarias para su construcción se pueden adaptar a las necesidades de la investigación. La interfaz usada específicamente para el presente trabajo constó de: un *shield* Bluetooth, un arduino, un led de 10mm, un Buzzer de 3.7kHz, una tarjeta de prototipos, cables para las conexiones, una joystick tipo maquina, un micro *switch* tipo palanca, una jaula habitación para aves o roedores y la aplicación. La Tabla 5 muestra el costo total y desglosado para la construcción de la interfaz.

Tabla 5. Costo desglosado y total por cámara de condicionamiento operante para prácticas de laboratorio

Materiales	Precio
Shield Bluetooth	450
Arduino	365
Led 10mm	3
Buzzer 3.7 kHz	29
Protoboard	49
Cable	5
Joystick	30
Switch	5
Jaula	150
Aplicación	0
Total	1086

Nota. Las cifras están en pesos mexicanos

Como puede observarse, el costo total para la construcción de una cámara de condicionamiento operante con interfaz es alrededor de \$1,086.00 pesos mexicanos, la interfaz necesita de un celular android para funcionar por lo que el costo aumenta en caso de no tener el dispositivo móvil.

Existen diversas compañías que comercializan interfaces de registro conductual y cámaras de condicionamiento operante. Una de ellas es Lafayette Instrument, esta empresa vende diversos aparatos de registro, así como diversos modelos de cámara para diversas especies. Con el propósito de comparar el costo del equipo comercial con el equipo de bajo costo descrito en el presente trabajo, se consideró el precio del sistema Complete Learning System (modelo 84025), el cual incluye: cámara estándar de condicionamiento operante, una

palanca omnidireccional, dos palancas estándar, dos luces, un dispensador de pellets, un dispensador de líquidos y la consola de control 81335C con el cableado apropiado. El precio por este equipo es de 3,243 USD (al momento de la consulta), lo cual equivale a \$42,436 pesos mexicanos (al momento de la consulta).

Otra empresa que proporciona equipos de control experimental es MED Associates Inc, la cotización de una cámara experimental de esta empresa que contiene componentes similares a la descrita en el párrafo anterior muestra un costo de 7,739 USD (al momento de la consulta), lo cual equivale a \$ 102,791 pesos mexicanos (al momento de consulta).

La Tabla 6 muestra una comparación entre la interfaz Ratuino y estas dos opciones de registro conductual.

Tabla 6. Comparación entre interfaces de control experimental comerciales y la interfaz Ratuino Mobile

Registro	Cámara operante	Interfaz de control	Portabilidad	Precio en pesos mexicanos
Lafayette trato directo	SI (acrílico y metal)	Completa e incluida	NO	42,436
Med-Associates	SI (acrílico y metal)	Completa e incluida	NO	102,791
Ratuino construida ad hoc	SI- Jaula	Completa	SI	1,086 + dispositivo móvil

Como puede observarse, la interfaz Ratuino Mobile es la opción de menor costo aún con el precio del celular pues se pueden conseguir equipos con las características necesarias para emplear la interfaz a partir de los \$1,299 pesos mexicanos (al momento de la consulta). Si bien la cámara de condicionamiento operante tiene una calidad notoriamente inferior a las que manufacturan estas compañías, la funcionalidad de estas para la conducción de

experimentos durante este proyecto mostró ser apropiada, por lo que cubre con el propósito de permitir la demostración de fenómenos conductuales en cursos de laboratorio.

A partir de las pruebas realizadas en el presente trabajo, puede sugerirse que la interfaz Ratuino Mobile podría ser una herramienta viable para la enseñanza del análisis de la conducta. Las clases se condujeron sin contratiempos y los alumnos pudieron observar los efectos de la mayoría de los programas de reforzamiento simple. En nuestras pruebas, sin embargo las sesiones tan cortas de cada programa y la cantidad amplia de programas abarcados en el curso no permitieron observar todos los patrones de conducta obtenidos con cada programa. Dedicar más sesiones a cada programa podría auxiliar a mostrar con más detalle los patrones característicos de los programas de reforzamiento.

El costo y portabilidad de la interfaz la vuelven una herramienta idónea para la conducción de prácticas de laboratorio en el salón de clase. Si bien los materiales empleados en este proyecto fueron de baja calidad en comparación con los materiales usados en las interfaces comerciales, cabe destacar que esta interfaz puede emplearse en conjunto con las interfaces comerciales para dotarlas de portabilidad debido a que es una alternativa de código abierto y las interfaces comerciales funcionan con inputs y outputs digitales fácilmente programables desde la tarjeta Arduino. Además esta herramienta puede emplearse con diversos componentes por lo que una inversión mayor en las prácticas puede mejorar notablemente la calidad de los componentes utilizados.

La precisión de registro es adecuada para replicar la mayoría de los experimentos en condicionamiento operante, por lo que la interfaz Ratuino Mobile puede emplearse para mostrar cómo se realizaron experimentos clásicos, así como para replicar una variedad de fenómeno conductuales pudiendo abarcar una gran cantidad de temas en distintas prácticas.

Sin embargo debido a las demoras provocadas por la conversión de información serial y la tasa de transferencia de datos con el Bluetooth, no se recomienda su uso como herramienta para investigación básica en el laboratorio. Para tal propósito puede emplearse la versión para computadora de la interfaz (Ratuino-PC) descrita por Pérez-Herrera y Escobar (2012).

Como se mencionó anteriormente la estabilidad en el desempeño depende ampliamente de las capacidades del dispositivo móvil que se use, tras realizar diversas pruebas en diferentes dispositivos, enumeramos las siguientes características de sistema como recomendadas para el óptimo desempeño:

- Sistema Operativo: Android Jellybean 4.1 o Mayor
- Bluetooth 3.0 o mayor.
- Memoria RAM de 1GB o mayor
- Memoria flash interna o externa del dispositivo referenciada como memoria SD (por defecto todas las memorias internas de los dispositivos Android hacen referencia a esta como memoria SD).

Se pueden conseguir dispositivos móviles que cubren estas características desde \$1,299 pesos mexicanos (al momento de la consulta).

También se recomienda emplear la aplicación en primer plano y cerrar todas las aplicaciones que se encuentren abiertas antes de iniciar la aplicación, así mismo recomendamos que al emplear la aplicación el dispositivo no tenga acceso a ningún tipo de red (ni Wi-Fi, ni de datos), debido a que los mensajes o llamadas entrantes pueden alterar el registro o abrir una aplicación en segundo plano.

Durante las pruebas realizadas a la aplicación se encontraron un número de errores en tiempo de ejecución, que afectan el desempeño de la aplicación. Este tipo de acontecimientos

son comunes en programación. Pese a que hemos dado solución a algunos de ellos durante el transcurso de las pruebas, aún pueden existir otros errores no resueltos, para mayor información sobre problemas conocidos con la aplicación véase el Apéndice D.

Existe la posibilidad de que existan problemas de desempeño de los cuales no estemos informados, para esto la aplicación tiene un solucionador de problemas diseñado por *Google* el cual permite al usuario mandar un informe de errores para dar solución a los problemas.

El reporte de la experiencia de los alumnos que participaron en las pruebas muestra que los alumnos describieron que trabajar con la interfaz fue “agradable” y reportaron como una experiencia “más agradable” el trabajar con ratas y la interfaz Ratuino que utilizar presentaciones orales como herramientas didácticas. Se puede concluir que al menos, el uso de la interfaz como herramienta didáctica parece ser agradable a la población estudiantil y podría fomentar el interés por las clases relacionadas con el análisis de la conducta. Cabe señalar que debido a la variación del reporte con respecto a la facilidad de uso y a la falta de comparaciones con otras herramientas didácticas, es difícil determinar qué tan fácil le resultó a los alumnos el emplear la interfaz en el salón de clase. Debido a la gran cantidad de sesiones ocurridas sin contratiempos, sin embargo, es posible sugerir que su construcción y uso sostenido es viable dentro de salones de clase con estudiantes que no han tenido experiencia previa con equipo electrónico o con lenguajes de programación.

Para finalizar, futuros estudios pueden enfocarse en optimizar el método, a manera de que se estandaricen los fenómenos a estudiar, el equipo a utilizar y el cronograma de actividades, con lo cual se podría establecer un programa introductorio de prácticas dentro del aula. Estos programas podrían ayudar a que los alumnos de los primeros semestres de la

licenciatura en psicología o incluso de nivel preparatoria puedan acercarse a la manera de investigar los fenómenos conductuales básicos en el análisis de la conducta. En conclusión la Interfaz Ratuino Mobile podría emplearse como herramienta didáctica para la enseñanza del análisis de la conducta. El empleo de estas interfaces podría permitir a los alumnos desarrollar las habilidades conductuales para realizar investigación básica en análisis de la conducta, así como acercarlos a la manera en que se genera el conocimiento de los fenómenos psicológicos.

Referencias

- Antonitis, J. (1977, October 20). Lottery. *Bangor Daily News*. p. 19.
- Cabello, F., Barnes-Holmes, D., O'Hora, D., & Stewart, I. (2002). Using visual basic in the experimental analysis of human behavior: A brief introduction. *Experimental Analysis of Human Behavior Bulletin*, *20*, 18-21.
- Canto, R., Bufalari, I., & D'Ausilio, A. (2011). A convenient and accurate parallel Input/Output USB device for E-Prime. *Behavior Research Methods*, *43*, 292-296. doi.10.3758/s13428-010-0022-3
- Colotla, V. A. & Ribes, I. E. (1981). Behavior analysis in Latin America: a historical review. *Spanish-Language Psychology*, *1*, 121-136
- D'Ausilio, A. (2012). Arduino: A low-cost multipurpose lab equipment. *Behavior Research Methods*, *44*, 305-313. doi.10.3758/s13428-011-0163-z
- Dinsmoor, J. A. (1990). Academic roots: Columbia University, 1943-1951. *Journal of the Experimental Analysis of Behavior*, *54*, 129-149.
- Dixon, M. R., & MacLin, O. H. (2003). *Visual Basic for behavioral psychologists*. Reno, NV: Context.
- Escobar, R., & Lattal, K. A. (2010). Interfaz de bajo costo usando un puerto paralelo y visual basic. *Revista Mexicana de Análisis de la Conducta*, *36*, 7-21.
- Escobar, R., & Pérez-Herrera, C. A. (2013a, Mayo). *Low-cost USB interface with Arduino and Visual Basic for experimental control*, trabajo presentado en ABAI (Association for Behavior Analysis International) 39th annual convention, Minneapolis, Minnesota, USA.

- Escobar, R., & Pérez-Herrera, C. A. (2013b, Noviembre). *Arma tu equipo de control de bajo costo para conducir experimentos en condicionamiento operante*, trabajo presentado en IV Seminario Internacional sobre Comportamiento y Aplicaciones (SINCA), Hermosillo, Sonora, Mexico.
- Escobar, R., & Pérez-Herrera, C. A. (2013c, Noviembre). *Equipo de control de bajo costo basado en la plataforma Arduino para conducir experimentos en Condicionamiento Operante*, trabajo presentado en XXIII Congreso Mexicano De Análisis De La Conducta, Cuernavaca, Morelos, México.
- Escobar, R., & Pérez-Herrera, C. A. (en prensa). Palanca retráctil de bajo costo para interfaces con puerto paralelo y Visual Basic. *Revista Mexicana de Análisis de la Conducta*.
- Escobar, R., Hernández-Ruiz, M., Santillán, N., & Pérez-Herrera, C. A. (2012). Diseño simplificado de una interfaz de bajo costo usando un puerto paralelo y Visual Basic. *Revista Mexicana de Análisis de la Conducta*, 38, 72-88.
- Fester, C. B., y Skinner, B. F. (1957). *Schedules of Reinforcement*. New York: Prentice- Hall.
- Fleshler, M., & Hoffman, H. S. (1962). A progresión for generating variable-interval schedules. *Journal of Experimental Analysis of Behavior*, 5, 529-530.
- Frick, F. C., Schoenfeld, W. N., & Keller, F. S. (1948) Apparatus designed for introductory psychology at Columbia College. *The American Journal of Psychology*, 61, 409-414.
- Goodman, D., & Maskell, R. (1985). A parallel breakout box for I/O monitoring. *Behavioral Research Methods, Instruments, & Computers*, 17, 403-405.
- Guttman, N., & Estes, W. K. (1949). A modified apparatus for the study of operant behavior in the rat. *The Journal of General Psychology*, 41, 297-301.

- Hoffman, A. M., Song, J., & Tuttle, E. M. (2007). ELOPTA: A novel microcontroller-based operant device. *Behavior Research Methods*, *39*, 776-782.
- Iversen, I. H. (1992). Skinner's early research: From reflexology to operant conditioning. *American Psychologist*, *47*, 1318-1328.
- Keller F.S., & Schoenfeld W.N. (1949). The psychology curriculum at Columbia College. *American Psychologist*. *4*, 165–172.
- Lattal, K. A. (2008). JEAB at 50: Coevolution of research and technology. *Journal of the Experimental Analysis of Behavior*, *89*, 129-135.
- Lopes Miranda, R., & Dias Cirino, S. (2010). Os primeiros anos dos laboratórios de análise do comportamento no Brasil. *Psychologia Latina*, *1*, 79-87.
- Martínez, S. H. (2006). Treinta años de la revista mexicana de análisis de la conducta: un reto a la supervivencia. *Avances en Psicología Latinoamericana*, *24*, 105-125.
- Matos, M. A. (1996). Contingências para a análise comportamental no Brasil. *Psicologia: Teoria e Pesquisa*, *12*, 107-111.
- Matos, M. A. (1998). Carolina Bori: a psicologia brasileira como missão. *Psicologia USP*, *9*, 67-70.
- Miltenberger, R. G. (2008). *Behavior modification: Principles and procedures*. Pacific Grove, CA: Thomson/Wadsworth.
- Palya, W. L. (1988). An introduction to the Walter/Palya Controller and ECBASIC. *Behavior Research Methods, Instruments, & Computers*, *20*, 81-87.
- Pérez Herrera C. A., & Escobar R. (2012, Noviembre). *Interfaz USB de bajo costo para el control experimental: RATUINO*, trabajo presentado en XXII Congreso mexicano de análisis de la conducta, Guanajuato, Guanajuato.

- Pérez-Herrera, C. A. & Escobar, R. (2013, Noviembre). *Conjunto interconectado de hardware y software para laboratorio Ratuino*, trabajo presentado en XXIII Congreso Mexicano De Análisis De La Conducta, Cuernavaca, Morelos, México.
- Ribes, E., Fernández, C., Rueda, M., Talento, M., & López, F. (1980). *Enseñanza y ejercicio de la investigación de la psicología un modelo integral*. México: Trillas.
- Santillán, N., & Escobar, R. (2013, Octubre). *Response variability during continuous reinforcement, intermittent reinforcement and extinction*. Poster presentado en el 7th International Conference de la Association for Behavior Analysis International. Mérida, México.
- Skinner, B. F. (1938). *The behavior of organisms*. New York: Appleton-Century-Crofts.
- Skinner, B. F. (1956). A case history in scientific method. *American Psychologist*, 5, 221-233.
- Skinner, B. F. (1957). The experimental analysis of behavior. *American Scientist*, 45, 343-371.
- Skinner, B. F. (1979) *The Shaping of a Behaviorist: (Particulars of My Life, Part 2)*, 1979. New York: Random House Inc.
- Stewart, N. (2006). A PC parallel port button box provides millisecond response time accuracy under Linux. *Behavior Research Methods*, 38, 170-173
- Todorov, J. C. (2006). Behavior Analysis in Brazil. *Avances en Psicología Latinoamericana*, 24, 29-36.
- Varnon, C., & Abramson, C. I. (2014, Mayo.). *Stimulating research in comparative psychology with the affordable propeller experiment controller*. Poster presentado en

40th annual convention de la the Association for Behavior Analysis International.
Chicago, Illinois.

Walter, D. E., & Palya, W. L. (1984). An inexpensive experiment controller for stand-alone applications or distributed processing networks. *Behavior Research Methods, Instruments, & Computers*, *16*, 125-134.

Apéndice A

. Código del microcontrolador Atmega328.

```
#include <SoftwareSerial.h>
#define BT_TX 0
#define BT_RX 1

int Houselight = 4;
int Valve = 6;
int Lever = 8;

void setup() {
  Serial.begin(9600);
  pinMode(Houselight, OUTPUT);
  pinMode(Valve, OUTPUT);
  pinMode(Lever, INPUT);
  digitalWrite(Houselight, LOW);
  digitalWrite (Valve, LOW);
  digitalWrite (Lever, HIGH);
}
void loop() {
  if (Serial.available()>0) {
    char Reinforcer=Serial.read();
    switch (Reinforcer) {
      case 'S':
        digitalWrite (Houselight, HIGH);
        break;

      case 'R':
        digitalWrite (Valve, HIGH);
        digitalWrite (Houselight, LOW);
        delay(500);
        digitalWrite (Valve, LOW);
        digitalWrite (Houselight, HIGH);
        break;

      case 'E':
        digitalWrite (Valve, LOW);
        digitalWrite (Houselight, LOW);
        delay(1000);
        break;

    }
  }
  Serial.print(digitalRead(Lever));
  delay (50);
}
```

Apéndice B

- . Código del programa Ratuino-Mobile.
- .. Código Main_Activity.Java

```
package com.Ratuino.ratuinomobile;

import android.app.ActionBar;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.text.format.Time;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.WindowManager;
import android.view.inputmethod.EditorInfo;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Calendar;
import java.util.Random;

public class MainActivity extends Activity {

    //Esto es del Ratuino

    String A_enviar;

    public void F_Enviar(){

        String message=A_enviar;
```

```

        sendMessage(message);
    }
    TextView arduino;

    //Esto es del primer layout
    EditText V_Subject;
    EditText V_Session;
    EditText V_Mtime;
    EditText V_MRReinforces;
    EditText V_ScheduleValue;
    RadioButton RB_FI;
    RadioButton RB_FR;
    RadioButton RB_VI;
    RadioButton RB_VR;
    EditText ittext;
    CheckBox itbox;

    String Subject;
    String Session;
    int Max_Time;
    int Max_Reinforcers;
    int Schedule_Value;
    int Schedule_Tipe;

    //Esto es del segudno layout

    TextView TXT_Subject;
    TextView TXT_Session;
    TextView TXT_Schedule;
    Button Finalizar;
    Button Inicio;
    Button Reforzar;
    String Schedule_Tipe_X;
    TextView TXT_Respuestaslocales;
    TextView TXT_Respuestasglobales;
    TextView TXT_Tiempolocal;
    TextView TXT_Tiempoglobal;
    TextView TXT_Reforzadores;

    // pa intervalos variables

    public int Iterator_X = 0;

    public int Valor_Variable = 0;
    public int v = 0;
    public int n= 0;
    public int order= 0;
    public int Iteracion_Variable= 0;
    public int[]rd;
    public Double []vi;

    public int Iteraciones_X = 0;
    public boolean Iteraciones_State_X = false;
    public int Sumaxser = 0;

```

```

// Debugging
private static final String TAG = "BluetoothChat";
private static final boolean D = true;

// Message types sent from the BluetoothChatService Handler
public static final int MESSAGE_STATE_CHANGE = 1;
public static final int MESSAGE_READ = 2;
public static final int MESSAGE_WRITE = 3;
public static final int MESSAGE_DEVICE_NAME = 4;
public static final int MESSAGE_TOAST = 5;

// Key names received from the BluetoothChatService Handler
public static final String DEVICE_NAME = "device_name";
public static final String TOAST = "toast";

// Intent request codes
private static final int REQUEST_CONNECT_DEVICE_SECURE = 1;
private static final int REQUEST_CONNECT_DEVICE_INSECURE = 2;
private static final int REQUEST_ENABLE_BT = 3;

// Layout Views
private ListView mConversationView;
private EditText mOutEditText;
private Button mSendButton;

// Name of the connected device
private String mConnectedDeviceName = null;
// Array adapter for the conversation thread
private ArrayAdapter<String> mConversationArrayAdapter;
// String buffer for outgoing messages
private StringBuffer mOutStringBuffer;
// Local Bluetooth adapter
private BluetoothAdapter mBluetoothAdapter = null;
// Member object for the chat services
private BluetoothChatService mChatService = null;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    this.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_
T_STATE_ALWAYS_HIDDEN);
    rd=new int[1];
    vi=new Double[1];
    rd[0]=5;
    vi[0]=5.0;
    V_Subject=(EditText) findViewById(R.id.EDTXT_Subject);
    V_Session=(EditText) findViewById(R.id.EDTXT_Session);
    V_Mtime=(EditText) findViewById(R.id.EDTXT_MTime);

```

```

        V_MReinforces=(EditText)
        findViewById(R.id.EDTXT_MReinforcers);
        V_ScheduleValue=(EditText)
        findViewById(R.id.EDTXT_Schedulevalue);
        RB_FI= (RadioButton) findViewById(R.id.Rad_FI);
        RB_FR= (RadioButton) findViewById(R.id.Rad_FR);
        RB_VI= (RadioButton) findViewById(R.id.Rad_VI);
        RB_VR= (RadioButton) findViewById(R.id.Rad_VR);
        itttext=(EditText) findViewById(R.id.itttext);
        itbox=(CheckBox) findViewById(R.id.itbox);

        itbox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
        public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
            if(Iteraciones_State_X==true) {
                Iteraciones_State_X=false;
            }else{
                Iteraciones_State_X=true;
            }
        }
    });

    // Get local Bluetooth adapter
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // If the adapter is null, then Bluetooth is not supported
    if (mBluetoothAdapter == null) {
        Toast.makeText(this, "Bluetooth is not available",
Toast.LENGTH_LONG).show();
        finish();
        return;
    }

}

@Override
public void onStart() {
    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");

    // If BT is not on, request that it be enabled.
    // setupChat() will then be called during onActivityResult
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        // Otherwise, setup the chat session
    } else {
        if (mChatService == null) setupChat();
    }
}

```

```

    }
}

@Override
public synchronized void onResume() {
    super.onResume();
    if(D) Log.e(TAG, "+ ON RESUME +");

    // Performing this check in onResume() covers the case in
    which BT was
    // not enabled during onStart(), so we were paused to enable
    it...
    // onResume() will be called when ACTION_REQUEST_ENABLE
    activity returns.
    if (mChatService != null) {
        // Only if the state is STATE_NONE, do we know that we
        haven't started already
        if (mChatService.getState() ==
BluetoothChatService.STATE_NONE) {
            // Start the Bluetooth chat services
            mChatService.start();
        }
    }
}

private void setupChat() {
    Log.d(TAG, "setupChat()");

    // Initialize the array adapter for the conversation thread
    mConversationArrayAdapter = new ArrayAdapter<String>(this,
R.layout.message);
    mConversationView = (ListView) findViewById(R.id.in);
    mConversationView.setAdapter(mConversationArrayAdapter);

    // Initialize the compose field with a listener for the return
key
    mOutEditText = (EditText) findViewById(R.id.edit_text_out);
    mOutEditText.setOnEditorActionListener(mWriteListener);

    // Initialize the send button with a listener that for click
events
    mSendButton = (Button) findViewById(R.id.button_send);
    mSendButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            // Send a message using content of the edit text
            widget
            TextView view = (TextView)
findViewById(R.id.edit_text_out);
            String message = "def";
            sendMessage(message);
        }
    });
}
}

```

```

        // Initialize the BluetoothChatService to perform bluetooth
connections
        mChatService = new BluetoothChatService(this, mHandler);

        // Initialize the buffer for outgoing messages
        mOutStringBuffer = new StringBuffer("");
    }

    @Override
    public synchronized void onPause() {
        super.onPause();
        if (D) Log.e(TAG, "-- ON PAUSE --");
    }

    @Override
    public void onStop() {
        super.onStop();
        if (D) Log.e(TAG, "-- ON STOP --");
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        // Stop the Bluetooth chat services
        if (mChatService != null) mChatService.stop();
        if (D) Log.e(TAG, "--- ON DESTROY ---");
    }

    private void ensureDiscoverable() {
        if (D) Log.d(TAG, "ensure discoverable");
        if (mBluetoothAdapter.getScanMode() !=
            BluetoothAdapter.SCAN_MODE_CONNECTABLE_DISCOVERABLE) {
            Intent discoverableIntent = new
            Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
            discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
            startActivity(discoverableIntent);
        }
    }

    /**
     * Sends a message.
     * @param message A string of text to send.
     */
    private void sendMessage(String message) {
        // Check that we're actually connected before trying anything
        if (mChatService.getState() !=
            BluetoothChatService.STATE_CONNECTED) {
            Toast.makeText(this, R.string.not_connected,
            Toast.LENGTH_SHORT).show();
            return;
        }

        // Check that there's actually something to send
        if (message.length() > 0) {

```

```

// Get the message bytes and tell the BluetoothChatService
to write    byte[] send = message.getBytes();
            mChatService.write(send);

// Reset out string buffer to zero and clear the edit text
field      mOutStringBuffer.setLength(0);
            mOutEditText.setText(mOutStringBuffer);
        }
    }

// The action listener for the EditText widget, to listen for the
return key
    private TextView.OnEditorActionListener mWriteListener =
        new TextView.OnEditorActionListener() {
            public boolean onEditorAction(TextView view, int
actionId, KeyEvent event) {
                // If the action is a key-up event on the return
key, send the message
                if (actionId == EditorInfo.IME_NULL &&
event.getAction() == KeyEvent.ACTION_UP) {
                    String message = view.getText().toString();
                    sendMessage(message);
                }
                if(D) Log.i(TAG, "END onEditorAction");
                return true;
            }
        };

    private final void setStatus(int resId) {
        final ActionBar actionBar = getActionBar();
        actionBar.setSubtitle(resId);
    }

    private final void setStatus(CharSequence subTitle) {
        final ActionBar actionBar = getActionBar();
        actionBar.setSubtitle(subTitle);
    }

// The Handler that gets information back from the
BluetoothChatService
    private final Handler mHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {

            switch (msg.what) {
                case MESSAGE_STATE_CHANGE:
                    if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " +
msg.arg1);

                    switch (msg.arg1) {
                        case BluetoothChatService.STATE_CONNECTED:

setStatus(getString(R.string.title_connected_to,
mConnectedDeviceName));

                            mConversationArrayAdapter.clear();
                            break;

```



```

        case BluetoothChatService.STATE_CONNECTING:
            setStatus(R.string.title_connecting);
            break;
        case BluetoothChatService.STATE_LISTEN:
        case BluetoothChatService.STATE_NONE:
            setStatus(R.string.title_not_connected);
            break;
    }
    break;
case MESSAGE_WRITE:
    byte[] writeBuf = (byte[]) msg.obj;
    // construct a string from the buffer
    String writeMessage = new String(writeBuf);
    mConversationArrayAdapter.add("Me: " +
writeMessage);
    break;
case MESSAGE_READ:
    byte[] readBuf = (byte[]) msg.obj;
    // construct a string from the valid bytes in the
buffer
    String readMessage = new String(readBuf, 0,
msg.arg1);

    mConversationArrayAdapter.add(mConnectedDeviceName+": " +
readMessage);

        arduino.setText(readMessage);
        if(Active){

TXT_Tiempolocal.setText(String.valueOf(Time_of_Period_in_seconds() ));
TXT_Tiempoglobal.setText(String.valueOf(Time_in_seconds()));
        TXT_Respuestaslocales.setText(String.valueOf(
Response_Counter));
        TXT_Respuestasglobales.setText(String.valueOf(
Total_Responses));
        TXT_Reforzadores.setText(String.valueOf(
Total_Reinforcers));}
    break;
case MESSAGE_DEVICE_NAME:
    // save the connected device's name
    mConnectedDeviceName =
msg.getData().getString(DEVICE_NAME);
    Toast.makeText(getApplicationContext(), "Connected
to "
        + mConnectedDeviceName,
Toast.LENGTH_SHORT).show();
    break;
case MESSAGE_TOAST:
    Toast.makeText(getApplicationContext(),
msg.getData().getString(TOAST),
        Toast.LENGTH_SHORT).show();
    break;
    }
}
};

```

```

    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
    if(D) Log.d(TAG, "onActivityResult " + resultCode);
    setContentView(R.layout.ratuino);

    arduino=(TextView) findViewById(R.id.arduino);

    TXT_Schedule=(TextView) findViewById(R.id.TXT_Schedule);
    TXT_Session=(TextView) findViewById(R.id.TXT_Session);
    TXT_Subject=(TextView) findViewById(R.id.TXT_Subject);
    Finalizar=(Button) findViewById(R.id.Finalizar);
    Inicio=(Button) findViewById(R.id.Inicio);
    Reforzar=(Button) findViewById(R.id.Reforzar);
    TXT_Reforzadores=(TextView)
findViewById(R.id.TXT_Reforzadores);
    TXT_Respuestasglobales=(TextView)
findViewById(R.id.TXT_Respuestasglobales);
    TXT_Respuestaslocales=(TextView)
findViewById(R.id.TXT_Respuestaslocales);

    TXT_Tiempoglobal=(TextView)
findViewById(R.id.TXT_Tiempoglobal);
    TXT_Tiempolocal=(TextView) findViewById(R.id.TXT_Tiempolocal);

    Finalizar.setVisibility(View.GONE);
    Reforzar.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            sendMessage("R");
            Escribir_archivo(String.valueOf(Time_in_miliseconds())
+ ".555"+"\r\n");
            Total_Reinforcers = Total_Reinforcers + 1;
            Response_Counter = 0;
            Period_Start = System.currentTimeMillis();
        }
    });

    Inicio.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            Inicio.setEnabled(false);
            Inicio.setText("Sesion en curso");
            someMethod();

        }
    });
    Finalizar.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });
}

```

```

        TXT_Session.setText(Session);
        TXT_Subject.setText(Subject);
        switch (Schedule_Tipe){
            case 1:
                TXT_Schedule.setText(getString(R.string.IF)+"
"+String.valueOf(Schedule_Value));
                break;

            case 2:
                TXT_Schedule.setText(getString(R.string.RF)+"
"+String.valueOf(Schedule_Value));
                break;

            case 3:
                TXT_Schedule.setText(getString(R.string.IV)+"
"+String.valueOf(Schedule_Value));
                break;

            case 4:
                TXT_Schedule.setText(getString(R.string.RV)+"
"+String.valueOf(Schedule_Value));
                break;

            default:
                TXT_Schedule.setText("NO Seleccioando");
                break;
        }

```

```

switch (requestCode) {
    case REQUEST_CONNECT_DEVICE_SECURE:
        // When DeviceListActivity returns with a device to
connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, true);
        }
        break;
    case REQUEST_CONNECT_DEVICE_INSECURE:
        // When DeviceListActivity returns with a device to
connect
        if (resultCode == Activity.RESULT_OK) {
            connectDevice(data, false);
        }
        break;
    case REQUEST_ENABLE_BT:
        // When the request to enable Bluetooth returns
        if (resultCode == Activity.RESULT_OK) {
            // Bluetooth is now enabled, so set up a chat
session
            setupChat();
        } else {

```

```

        // User did not enable Bluetooth or an error
        occurred

        Log.d(TAG, "BT not enabled");
        Toast.makeText(this,
R.string.bt_not_enabled_leaving, Toast.LENGTH_SHORT).show();
        finish();
    }
}

private void connectDevice(Intent data, boolean secure) {
    // Get the device MAC address
    String address =
data.getExtras().getString(DeviceList.EXTRA_DEVICE_ADDRESS);
    // Get the BluetoothDevice object
    BluetoothDevice device =
mBluetoothAdapter.getRemoteDevice(address);
    // Attempt to connect to the device
    mChatService.connect(device, secure);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.option_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Intent serverIntent = null;
    switch (item.getItemId()) {
        case R.id.secure_connect_scan:
            // Launch the DeviceListActivity to see devices and do
            scan

            Subject=V_Subject.getText().toString();
            Session=V_Session.getText().toString();

            Max_Time=Integer.valueOf(V_Mtime.getText().toString());
            Max_Reinforcers=
            Integer.valueOf(V_MRinforcers.getText().toString());
            Schedule_Value=
            Integer.valueOf(V_ScheduleValue.getText().toString());

            if (Iteraciones_State_X==true){
                Iteraciones_X =
                Integer.valueOf(ittext.getText().toString());
            }

            if (RB_FI.isChecked()){
                Schedule_Tipe=1;
                Schedule_Tipe_X=RB_FI.getText().toString();
            }else
            if (RB_FR.isChecked()){

```

```

        Schedule_Tipe=2;
        Schedule_Tipe_X=RB_FR.getText().toString();
    }else
    if (RB_VI.isChecked()){
        Schedule_Tipe=3;
        Schedule_Tipe_X=RB_VI.getText().toString();
    }else
    if (RB_VR.isChecked()){
        Schedule_Tipe=4;
        Schedule_Tipe_X=RB_VR.getText().toString();
    }

    serverIntent = new Intent(this, Devicelist.class);
    startActivityForResult(serverIntent,
REQUEST_CONNECT_DEVICE_SECURE);
    return true;
    case R.id.insecure_connect_scan:
        // Launch the DeviceListActivity to see devices and do
scan

        Subject=V_Subject.getText().toString();
        Session=V_Session.getText().toString();

Max_Time=Integer.valueOf(V_Mtime.getText().toString());
        Max_Reinforcers=
Integer.valueOf(V_MReinforces.getText().toString());
        Schedule_Value=
Integer.valueOf(V_ScheduleValue.getText().toString());
        if (Iteraciones_State_X==true){
            Iteraciones_X =
Integer.valueOf(ittext.getText().toString());
        }

        if (RB_FI.isChecked()){
            Schedule_Tipe=1;
        }else
        if (RB_FR.isChecked()){
            Schedule_Tipe=2;
        }else
        if (RB_VI.isChecked()){
            Schedule_Tipe=3;
            Schedule_Tipe_X=RB_VI.getText().toString();
        }else
        if (RB_VR.isChecked()){
            Schedule_Tipe=4;
            Schedule_Tipe_X=RB_VR.getText().toString();
        }

    serverIntent = new Intent(this, Devicelist.class);
    startActivityForResult(serverIntent,
REQUEST_CONNECT_DEVICE_INSECURE);
    return true;
    case R.id.discoverable:
        // Ensure this device is discoverable by others
ensureDiscoverable();
    return true;
}

```

```

        return false;
    }

    //esto es el loop infinito en un hilo

    long MTimemillis;
    long TimeSLRmillis;
    long TimeSLRsecs;
    long Time_start = 0;
    int Total_Reinforcers= 0;
    int Response_Counter = 0;
    int Total_Responses = 0;
    long Period_Start = 0;
    long Period_End = 0;
    long Time_End = 0;

    int Previous_Response = 0;
    int Actual_Response = 0;
    boolean Active= false;

    public int Data_Handling() {

        try {
            Actual_Response =
Integer.valueOf(arduino.getText().toString());

            if (Actual_Response != Previous_Response &&
Actual_Response != 1) {
                Escribir_archivo(String.valueOf(Time_in_miliseconds())
+ ".100"+"\r\n");
                Response_Counter = Response_Counter + 1;

                Total_Responses = Total_Responses + 1;
            }else
            if (Actual_Response != Previous_Response &&
Actual_Response != 0) {
                Escribir_archivo(String.valueOf(Time_in_miliseconds())
+ ".200"+"\r\n");
            }
            Schedule_of_Reinforcement();
            Previous_Response = Actual_Response;
        }

        catch (Exception e){

        }

        return (Total_Responses);
    }

    public int Reinforce() {
        sendMessage("R");
        Escribir_archivo(String.valueOf(Time_in_miliseconds()) +
".500"+"\r\n");
    }

```

```

        Total_Reinforcers = Total_Reinforcers + 1;
        return 0;
    }

    public void Schedule_of_Reinforcement(){

        //Razon Fija
        if (Schedule_Tipe == 2) {
            if (Response_Counter == Schedule_Value){
                Reinforce();
                Response_Counter = 0;
                Period_Start = System.currentTimeMillis();
            }
        } else
        //Intervalo Fijo
        if (Schedule_Tipe == 1){
            if (Time_of_the_Period() >= (Schedule_Value * 1000) &&
Actual_Response != Previous_Response && Actual_Response != 1){
                Reinforce();
                Response_Counter = 0;
                Period_Start = System.currentTimeMillis();
            }
        } else
        //Razon Variable
        if (Schedule_Tipe==4){
            if (Response_Counter >= rd[Valor_Variable] &&
Actual_Response != 1 && Response_Counter >= 1) {
                Reinforce();
                Response_Counter = 0;
                Period_Start = System.currentTimeMillis();
                Valor_Variable = Valor_Variable + 1;
            }
        } else
        //Intervalo Variable
        if(Schedule_Tipe==3){
            if (Time_of_the_Period() >= rd[Valor_Variable] &&
Actual_Response != 1 && Actual_Response != Previous_Response){
                Reinforce();
                Response_Counter = 0;
                Period_Start = System.currentTimeMillis();
                Valor_Variable = Valor_Variable + 1;
            }
        }

    }

    public long Start_Chronometers() {
        Time_start = System.currentTimeMillis();
        Period_Start = System.currentTimeMillis();
        return (Time_start);
    }

    public long Time_of_the_Period() {

```

```

        Period_End = System.currentTimeMillis();
        return (Period_End - Period_Start);
    }

    public long Time_of_Period_in_seconds() {
        Period_End = System.currentTimeMillis();
        return ((Period_End - Period_Start) / 1000);
    }

    public long Time_in_miliseconds(){
        Time_End = System.currentTimeMillis();
        return (Time_End - Time_start);
    }

    public long Time_in_seconds() {
        Time_End = System.currentTimeMillis();
        return ((Time_End - Time_start) / 1000);
    }

    String Recopilador;
    String filename;
    FileOutputStream fos;

    public void Initialize_File_Writing() {

        Recopilador =Subject+"."+Session;

        filename= Recopilador  +".txt";

        File carpeta1 = new
File(Environment.getExternalStorageDirectory().toString()+"/ratuino");

        carpeta1.mkdirs();
        String carpeta2 = carpeta1.toString();
        File file = new File(carpeta2, filename);

        final Time Hoy=new Time(Time.getCurrentTimezone());
        Hoy.setToNow();

        String Primera_linea;

        Primera_linea=java.text.DateFormat.getDateInstance().format(Calendar.g
etInstance().getTime());
        Primera_linea=Primera_linea+"\r\n"+"Subject: " +
Subject+"\r\n";
        Primera_linea= Primera_linea+ "Session: " + Session+"\r\n";
        Primera_linea=Primera_linea+"Schedule: " + Schedule_Tipe_X + "
" + String.valueOf(Schedule_Value)+"\r\n"+"r\n";
        Primera_linea=Primera_linea+"Events:"+"\r\n"+"r\n";
        Primera_linea=Primera_linea+"100 = Response-OFF"+"r\n";
        Primera_linea=Primera_linea+"200 = Response-ON
(RESPONSE) "+"r\n";
        Primera_linea=Primera_linea+"500 = Reinforcer
delivery"+"r\n";

```



```

Primera_linea=Primera_linea+"555 = Independant
reinforcer"+ "\r\n"+ "\r\n";
Primera_linea=Primera_linea+"List of events: "+ "\r\n"+ "\r\n";

```

```

byte[] data_to_copy = Primera_linea.getBytes();
try {
    fos = new FileOutputStream(file);
    fos.write(data_to_copy);

```

```

} catch (FileNotFoundException e) {

```

```

} catch (IOException e) {

```

```

}

```

```

}

```

```

byte[] data_towrite;
public void Escribir_archivo(String informacion){
    data_towrite=informacion.getBytes();
    try {
        fos.write(data_towrite);

```

```

} catch (FileNotFoundException e) {

```

```

} catch (IOException e) {

```

```

}

```

```

}

```

```

public void End_File_Writing() {
    String Ultima_linea;
    Ultima_linea= "\r\n"+ "\r\n"+"Total Responses: " +
String.valueOf(Total_Responses)+"\r\n";
    Ultima_linea=Ultima_linea+"Total Reinforcers delivered: " +
String.valueOf(Total_Reinforcers)+"\r\n";
    Ultima_linea=Ultima_linea+"Time elapsed: " +
String.valueOf(Time_in_seconds()+"\r\n"+ "\r\n";
    Ultima_linea=Ultima_linea+"END OF THE SESSION";
    data_towrite=Ultima_linea.getBytes();
    try{
        fos.write(data_towrite);
        fos.flush();
        fos.close();
    } catch (FileNotFoundException e) {

```

```

    } catch (IOException e) {
    }
}

public void someMethod(){
    // ...
    // initializing and starting a new local Thread object
    Thread myTread = new Thread() {
        // setting the behavior we want from the Thread
        @Override
        public void run() {

            iniciar_ratuino();

        }
    };
    myTread.start();
    // ...
}
public void iniciar_ratuino() {

    Active=true;
    sendMessage("S");
    Start_Chronometers();
    Initialize_File_Writing();

    while (Total_Reinforcers < Max_Reinforcers){

        Generar_Intervalo();

        Data_Handling();

    }

    End_File_Writing();
    sendMessage("E");
    Active=false;
    Message msg = new Message();
    msg.obj = 1;
    puente.sendMessage(msg);

}

private Handler puente = new Handler() {

```

```

@Override
public void handleMessage(Message msg) {

    Reforzar.setEnabled(false);
    Reforzar.setVisibility(View.GONE);

    TXT_Tiempolocal.setText(String.valueOf(Time_of_Period_in_seconds() ));

    TXT_Tiempoglobal.setText(String.valueOf(Time_in_seconds()));
    TXT_Respuestaslocales.setText(String.valueOf(
Response_Counter));
    TXT_Respuestasglobales.setText(String.valueOf(
Total_Responses));
    TXT_Reforzadores.setText(String.valueOf(
Total_Reinforcers));
    Finalizar.setVisibility(View.VISIBLE);
    }
};

// definir intervalos variables

public void Generar_Intervalo() {

    if (Valor_Variable == n){
        Valor_Variable = 0;
    }

    if (Valor_Variable == 0){
        //Intervalo Variable
        if (Schedule_Type == 3){
            v = Schedule_Value * 1000;
        }
        //Razon Variable
        if (Schedule_Type == 4){
            v = Schedule_Value;
        }
        if (Iteraciones_State_X == false){
            n = (int)Math.round(Max_Reinforcers / 3);
        }
        if (Iteraciones_State_X == true) {
            n = Iteraciones_X;
        }

        rd=new int[n];
        vi=new Double[n];

        for (int i=1; i<=rd.length; i++){

            if (i==n){
                vi[i-1] = v * (1 + StrictMath.log(n));
            }else{
                vi[i-1] = v * (1 + (StrictMath.log(n)) + (n - i) *
(StrictMath.log(n - i)) - (n - i + 1) * StrictMath.log(n - i + 1));
            }
            Random r = new Random();

```

```

        order = r.nextInt(n-1 - 0 + 1) + 0;
        if (rd[order] == 0){
            rd[order] = (int)Math.round(vi[i-1]);
        }
        else{
            i--;
        }
    }

    for (int a = 0;a< rd.length;a++){
        Sumaxser = Sumaxser + rd[a];
    }

    if (Sumaxser != (v * n)) {
        rd[0] = rd[0] + ((v * n) - Sumaxser);
    }
    Sumaxser = 0;
}

}
}

```

.. Código BluetoothCharService.Java

```
package com.Ratuino.ratuinomobile;

/**
 * Created by Investigación on 6/08/13.
 */
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothServerSocket;
import android.bluetooth.BluetoothSocket;
import android.content.Context;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

public class BluetoothChatService {
    // Debugging
    private static final String TAG = "BluetoothChatService";
    private static final boolean D = true;

    // Name for the SDP record when creating server socket
    private static final String NAME_SECURE = "BluetoothChatSecure";
    private static final String NAME_INSECURE =
"BluetoothChatInsecure";

    // Unique UUID for this application
    private static final UUID MY_UUID_SECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    private static final UUID MY_UUID_INSECURE =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    // Member fields
    private final BluetoothAdapter mAdapterer;
    private final Handler mHandler;
    private AcceptThread mSecureAcceptThread;
    private AcceptThread mInsecureAcceptThread;
    private ConnectThread mConnectThread;
    private ConnectedThread mConnectedThread;
    private int mState;

    // Constants that indicate the current connection state
    public static final int STATE_NONE = 0; // we're doing
nothing
    public static final int STATE_LISTEN = 1; // now listening for
incoming connections
    public static final int STATE_CONNECTING = 2; // now initiating an
outgoing connection
```

```

    public static final int STATE_CONNECTED = 3; // now connected to
a remote device

    /**
     * Constructor. Prepares a new BluetoothChat session.
     * @param context The UI Activity Context
     * @param handler A Handler to send messages back to the UI
Activity
    */
    public BluetoothChatService(Context context, Handler handler) {
        mAdapter = BluetoothAdapter.getDefaultAdapter();
        mState = STATE_NONE;
        mHandler = handler;
    }

    /**
     * Set the current state of the chat connection
     * @param state An integer defining the current connection state
    */
    private synchronized void setState(int state) {
        if (D) Log.d(TAG, "setState() " + mState + " -> " + state);
        mState = state;

        // Give the new state to the Handler so the UI Activity can
update
        mHandler.obtainMessage(MainActivity.MESSAGE_STATE_CHANGE,
state, -1).sendToTarget();
    }

    /**
     * Return the current connection state. */
    public synchronized int getState() {
        return mState;
    }

    /**
     * Start the chat service. Specifically start AcceptThread to
begin a
     * session in listening (server) mode. Called by the Activity
onResume() */
    public synchronized void start() {
        if (D) Log.d(TAG, "start");

        // Cancel any thread attempting to make a connection
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

        // Cancel any thread currently running a connection
        if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

        setState(STATE_LISTEN);

        // Start the thread to listen on a BluetoothServerSocket
        if (mSecureAcceptThread == null) {
            mSecureAcceptThread = new AcceptThread(true);
            mSecureAcceptThread.start();

```

```

    }
    if (mInsecureAcceptThread == null) {
        mInsecureAcceptThread = new AcceptThread(false);
        mInsecureAcceptThread.start();
    }
}

/**
 * Start the ConnectThread to initiate a connection to a remote
 device.
 * @param device The BluetoothDevice to connect
 * @param secure Socket Security type - Secure (true) , Insecure
 (false)
 */
public synchronized void connect(BluetoothDevice device, boolean
secure) {
    if (D) Log.d(TAG, "connect to: " + device);

    // Cancel any thread attempting to make a connection
    if (mState == STATE_CONNECTING) {
        if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}
    }

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

    // Start the thread to connect with the given device
    mConnectThread = new ConnectThread(device, secure);
    mConnectThread.start();
    setState(STATE_CONNECTING);
}

/**
 * Start the ConnectedThread to begin managing a Bluetooth
 connection
 * @param socket The BluetoothSocket on which the connection was
 made
 * @param device The BluetoothDevice that has been connected
 */
public synchronized void connected(BluetoothSocket socket,
BluetoothDevice
device, final String socketType) {
    if (D) Log.d(TAG, "connected, Socket Type:" + socketType);

    // Cancel the thread that completed the connection
    if (mConnectThread != null) {mConnectThread.cancel();
mConnectThread = null;}

    // Cancel any thread currently running a connection
    if (mConnectedThread != null) {mConnectedThread.cancel();
mConnectedThread = null;}

    // Cancel the accept thread because we only want to connect to
 one device
    if (mSecureAcceptThread != null) {

```

```

        mSecureAcceptThread.cancel();
        mSecureAcceptThread = null;
    }
    if (mInsecureAcceptThread != null) {
        mInsecureAcceptThread.cancel();
        mInsecureAcceptThread = null;
    }

    // Start the thread to manage the connection and perform
    transmissions
    mConnectedThread = new ConnectedThread(socket, socketType);
    mConnectedThread.start();

    // Send the name of the connected device back to the UI
    Activity
    Message msg =
    mHandler.obtainMessage(MainActivity.MESSAGE_DEVICE_NAME);
    Bundle bundle = new Bundle();
    bundle.putString(MainActivity.DEVICE_NAME, device.getName());
    msg.setData(bundle);
    mHandler.sendMessage(msg);

    setState(STATE_CONNECTED);
}

/**
 * Stop all threads
 */
public synchronized void stop() {
    if (D) Log.d(TAG, "stop");

    if (mConnectThread != null) {
        mConnectThread.cancel();
        mConnectThread = null;
    }

    if (mConnectedThread != null) {
        mConnectedThread.cancel();
        mConnectedThread = null;
    }

    if (mSecureAcceptThread != null) {
        mSecureAcceptThread.cancel();
        mSecureAcceptThread = null;
    }

    if (mInsecureAcceptThread != null) {
        mInsecureAcceptThread.cancel();
        mInsecureAcceptThread = null;
    }
    setState(STATE_NONE);
}

/**
 * Write to the ConnectedThread in an unsynchronized manner
 * @param out The bytes to write
 * @see ConnectedThread#write(byte[])

```



```

    */
    public void write(byte[] out) {
        // Create temporary object
        ConnectedThread r;
        // Synchronize a copy of the ConnectedThread
        synchronized (this) {
            if (mState != STATE_CONNECTED) return;
            r = mConnectedThread;
        }
        // Perform the write unsynchronized
        r.write(out);
    }

    /**
     * Indicate that the connection attempt failed and notify the UI
     Activity.
     */
    private void connectionFailed() {
        // Send a failure message back to the Activity
        Message msg =
mHandler.obtainMessage(MainActivity.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(MainActivity.TOAST, "Unable to connect
device");
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        // Start the service over to restart listening mode
        BluetoothChatService.this.start();
    }

    /**
     * Indicate that the connection was lost and notify the UI
     Activity.
     */
    private void connectionLost() {
        // Send a failure message back to the Activity
        Message msg =
mHandler.obtainMessage(MainActivity.MESSAGE_TOAST);
        Bundle bundle = new Bundle();
        bundle.putString(MainActivity.TOAST, "Device connection was
lost");
        msg.setData(bundle);
        mHandler.sendMessage(msg);

        // Start the service over to restart listening mode
        BluetoothChatService.this.start();
    }

    /**
     * This thread runs while listening for incoming connections. It
     behaves
     * like a server-side client. It runs until a connection is
     accepted
     * (or until cancelled).
     */
    private class AcceptThread extends Thread {

```

```

// The local server socket
private final BluetoothServerSocket mmServerSocket;
private String mSocketType;

public AcceptThread(boolean secure) {
    BluetoothServerSocket tmp = null;
    mSocketType = secure ? "Secure":"Insecure";

    // Create a new listening server socket
    try {
        if (secure) {
            tmp =
mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE,
MY_UUID_SECURE);
        } else {
            tmp =
mAdapter.listenUsingInsecureRfcommWithServiceRecord(
NAME_INSECURE, MY_UUID_INSECURE);
        }
    } catch (IOException e) {
        Log.e(TAG, "Socket Type: " + mSocketType + "listen()
failed", e);
    }
    mmServerSocket = tmp;
}

public void run() {
    if (D) Log.d(TAG, "Socket Type: " + mSocketType +
"BEGIN mAcceptThread" + this);
    setName("AcceptThread" + mSocketType);

    BluetoothSocket socket = null;

    // Listen to the server socket if we're not connected
    while (mState != STATE_CONNECTED) {
        try {
            // This is a blocking call and will only return on
a
            // successful connection or an exception
            socket = mmServerSocket.accept();
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType +
"accept() failed", e);
            break;
        }

        // If a connection was accepted
        if (socket != null) {
            synchronized (BluetoothChatService.this) {
                switch (mState) {
                    case STATE_LISTEN:
                    case STATE_CONNECTING:
                        // Situation normal. Start the
connected thread.
                        connected(socket,
socket.getRemoteDevice(),
mSocketType);
                }
            }
        }
    }
}

```

```

        break;
    case STATE_NONE:
    case STATE_CONNECTED:
        // Either not ready or already
connected. Terminate new socket.
        try {
            socket.close();
        } catch (IOException e) {
            Log.e(TAG, "Could not close
unwanted socket", e);
        }
        break;
    }
}
}
}
}
if (D) Log.i(TAG, "END mAcceptThread, socket Type: " +
mSocketType);
}

public void cancel() {
    if (D) Log.d(TAG, "Socket Type" + mSocketType + "cancel "
+ this);
    try {
        mmServerSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "Socket Type" + mSocketType + "close() of
server failed", e);
    }
}
}

/**
 * This thread runs while attempting to make an outgoing
connection
 * with a device. It runs straight through; the connection either
 * succeeds or fails.
 */
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    private String mSocketType;

    public ConnectThread(BluetoothDevice device, boolean secure) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            if (secure) {
                tmp = device.createRfcommSocketToServiceRecord(
                    MY_UUID_SECURE);
            } else {

```

```

        tmp =
device.createInsecureRfcommSocketToServiceRecord(
        MY_UUID_INSECURE);
    }
    } catch (IOException e) {
        Log.e(TAG, "Socket Type: " + mSocketType + "create()
failed", e);
    }
    mmSocket = tmp;
}

public void run() {
    Log.i(TAG, "BEGIN mConnectThread SocketType:" +
mSocketType);
    setName("ConnectThread" + mSocketType);

    // Always cancel discovery because it will slow down a
connection
    mAdapter.cancelDiscovery();

    // Make a connection to the BluetoothSocket
try {
        // This is a blocking call and will only return on a
        // successful connection or an exception
        mmSocket.connect();
    } catch (IOException e) {
        // Close the socket
        try {
            mmSocket.close();
        } catch (IOException e2) {
            Log.e(TAG, "unable to close() " + mSocketType +
                " socket during connection failure", e2);
        }
        connectionFailed();
        return;
    }

    // Reset the ConnectThread because we're done
synchronized (BluetoothChatService.this) {
        mConnectThread = null;
    }

    // Start the connected thread
    connected(mmSocket, mmDevice, mSocketType);
}

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "close() of connect " + mSocketType + "
socket failed", e);
    }
}

}

/**

```

```

* This thread runs during a connection with a remote device.
* It handles all incoming and outgoing transmissions.
*/
private class ConnectedThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    public ConnectedThread(BluetoothSocket socket, String
socketType) {
        Log.d(TAG, "create ConnectedThread: " + socketType);
        mmSocket = socket;
        InputStream tmpIn = null;
        OutputStream tmpOut = null;

        // Get the BluetoothSocket input and output streams
        try {
            tmpIn = socket.getInputStream();
            tmpOut = socket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, "temp sockets not created", e);
        }

        mmInStream = tmpIn;
        mmOutStream = tmpOut;
    }

    public void run() {
        Log.i(TAG, "BEGIN mConnectedThread");
        byte[] buffer = new byte[1024];
        int bytes;

        // Keep listening to the InputStream while connected
        while (true) {
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);

                // Send the obtained bytes to the UI Activity
                mHandler.obtainMessage(MainActivity.MESSAGE_READ,
bytes, -1, buffer)
                    .sendToTarget();
            } catch (IOException e) {
                Log.e(TAG, "disconnected", e);
                connectionLost();
                // Start the service over to restart listening
mode

                BluetoothChatService.this.start();
                break;
            }
        }
    }
}

/**
 * Write to the connected OutStream.
 * @param buffer The bytes to write
 */

```

```

public void write(byte[] buffer) {
    try {
        mmOutputStream.write(buffer);

        // Share the sent message back to the UI Activity
        mHandler.obtainMessage(MainActivity.MESSAGE_WRITE, -1,
-1, buffer)
            .sendToTarget();
    } catch (IOException e) {
        Log.e(TAG, "Exception during write", e);
    }
}

public void cancel() {
    try {
        mmSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "close() of connect socket failed", e);
    }
}
}
}

```

.. Código Devicelist.Java

```
package com.Ratuino.ratuinomobile;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

public class Devicelist extends Activity {

    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.devicelist);

        // Set result CANCELED in case the user backs out
        setResult(Activity.RESULT_CANCELED);

        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });
    }
}
```

```

    }
    });

    // Initialize array adapters. One for already paired devices
and
    // one for newly discovered devices
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
    mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

    // Find and set up the ListView for paired devices
    ListView pairedListView = (ListView)
findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);

    // Find and set up the ListView for newly discovered devices
    ListView newDevicesListView = (ListView)
findViewById(R.id.new_devices);
    newDevicesListView.setAdapter(mNewDevicesArrayAdapter);

newDevicesListView.setOnItemClickListener(mDeviceClickListener);

    // Register for broadcasts when a device is discovered
    IntentFilter filter = new
IntentFilter(BluetoothDevice.ACTION_FOUND);
    this.registerReceiver(mReceiver, filter);

    // Register for broadcasts when discovery has finished
    filter = new
IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
    this.registerReceiver(mReceiver, filter);

    // Get the local Bluetooth adapter
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

    // Get a set of currently paired devices
    Set<BluetoothDevice> pairedDevices =
mBtAdapter.getBondedDevices();

    // If there are paired devices, add each one to the
ArrayAdapter
    if (pairedDevices.size() > 0) {

findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n"
+ device.getAddress());
        }
    } else {
        String noDevices =
getResources().getText(R.string.none_paired).toString();
        mPairedDevicesArrayAdapter.add(noDevices);
    }
}
}

```



```

@Override
protected void onDestroy() {
    super.onDestroy();

    // Make sure we're not doing discovery anymore
    if (mBtAdapter != null) {
        mBtAdapter.cancelDiscovery();
    }

    // Unregister broadcast listeners
    this.unregisterReceiver(mReceiver);
}

/**
 * Start device discover with the BluetoothAdapter
 */
private void doDiscovery() {
    if (D) Log.d(TAG, "doDiscovery()");

    // Indicate scanning in the title
    setProgressBarIndeterminateVisibility(true);
    setTitle(R.string.scanning);

    // Turn on sub-title for new devices

findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

    // If we're already discovering, stop it
    if (mBtAdapter.isDiscovering()) {
        mBtAdapter.cancelDiscovery();
    }

    // Request discover from BluetoothAdapter
    mBtAdapter.startDiscovery();
}

// The on-click listener for all devices in the ListViews
private AdapterView.OnItemClickListener mDeviceClickListener = new
AdapterView.OnItemClickListener() {
    public void onItemClick(AdapterView<?> av, View v, int arg2,
long arg3) {
        // Cancel discovery because it's costly and we're about to
connect
        mBtAdapter.cancelDiscovery();

        // Get the device MAC address, which is the last 17 chars
in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Create the result Intent and include the MAC address
        Intent intent = new Intent();
        intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

        // Set result and finish this Activity
        setResult(Activity.RESULT_OK, intent);
        finish();
    }
}

```

```

    }
};

// The BroadcastReceiver that listens for discovered devices and
// changes the title when discovery is finished
private final BroadcastReceiver mReceiver = new
BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        // When discovery finds a device
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            // Get the BluetoothDevice object from the Intent
            BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            // If it's already paired, skip it, because it's been
            listed already
            if (device.getBondState() !=
BluetoothDevice.BOND_BONDED) {
                mNewDevicesArrayAdapter.add(device.getName() +
"\n" + device.getAddress());
            }
            // When discovery is finished, change the Activity
            title
        } else if
(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            setProgressBarIndeterminateVisibility(false);
            setTitle(R.string.select_device);
            if (mNewDevicesArrayAdapter.getCount() == 0) {
                String noDevices =
getResources().getText(R.string.none_found).toString();
                mNewDevicesArrayAdapter.add(noDevices);
            }
        }
    }
};
}

```

.. AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.Ratuino.ratuinomobile"
    android:versionCode="3"
    android:versionName="1.2" >

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="16" />

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.Ratuino.ratuinomobile.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.Ratuino.ratuinomobile.Devicelist"
            android:label="@string/title_activity_devicelist" >
        </activity>
    </application>

</manifest>
```

.. Ratuino.xml

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

<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <ScrollView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/scrollView">

    <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

      <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="Small Text"
        android:id="@+id/textView2"
        android:visibility="invisible"/>

      <HorizontalScrollView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:id="@+id/horizontalScrollView"
        android:layout_gravity="center">

        <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"
          android:gravity="center"
          android:baselineAligned="false">

          <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Subject"
            android:id="@+id/textView"/>

          <TextView
            android:layout_width="20dp"
            android:layout_height="wrap_content"
```

```

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="M"
    android:id="@+id/textView17"
    android:visibility="invisible"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="SUBJECT"
    android:id="@+id/TXT_Subject"
    android:textColor="#000000"
    android:background="#dec4ff"/>

    <TextView
        android:layout_width="40dp"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="M"
    android:id="@+id/textView18"
    android:visibility="invisible"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Session"
    android:id="@+id/textView4"/>

    <TextView
        android:layout_width="20dp"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="M"
    android:id="@+id/textView19"
    android:visibility="invisible"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="SESSION"
    android:id="@+id/TXT_Session"
    android:textColor="#000000"
    android:background="#dec4ff"/>

    </LinearLayout>
</HorizontalScrollView>

<TextView

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="Small Text"
        android:id="@+id/textView3"
        android:visibility="invisible"/>

    <HorizontalScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/horizontalScrollView6"
        android:layout_gravity="center">

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Schedule"
                android:id="@+id/textView5"/>

            <TextView
                android:layout_width="20dp"
                android:layout_height="wrap_content"

                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="M"
                android:id="@+id/textView20"
                android:visibility="invisible"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

                android:textAppearance="?android:attr/textAppearanceLarge"
                android:text="SCHEDULE"
                android:id="@+id/TXT_Schedule"
                android:textColor="#000000"
                android:background="#dec4ff"
                android:autoText="false"/>

        </LinearLayout>
    </HorizontalScrollView>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="Small Text"
        android:id="@+id/textView6"
        android:visibility="invisible"/>

```

```

<HorizontalScrollView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/horizontalScrollView7"
    android:layout_gravity="center">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="DATA"
            android:id="@+id/tcttt"/>

        <TextView
            android:layout_width="20dp"
            android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="M"
            android:id="@+id/textView21"
            android:visibility="invisible"/>

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="DATA"
            android:id="@+id/arduino"
            android:textColor="#000000"
            android:background="#8bff8d"/>
    </LinearLayout>
</HorizontalScrollView>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceSmall"
    android:text="Small Text"
    android:id="@+id/textView7"
    android:visibility="invisible"/>

<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="START SESSION"
    android:id="@+id/Inicio"
    android:background="#8adfe7"
    android:textColor="#ed0b10"
    android:layout_gravity="center"/>

```

```

<TextView
    android:layout_width="wrap_content"
    android:layout_height="35dp"

android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/textView9"
    android:visibility="invisible"/>

<HorizontalScrollView
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:id="@+id/horizontalScrollView3"
    android:layout_gravity="center">

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/textView8"
        android:singleLine="false"
        android:lines="2"
        android:minLines="2"
        android:text="@string/feres"/>

    <TextView
        android:layout_width="20dp"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="M"
        android:id="@+id/textView22"
        android:visibility="invisible"/>

    <TextView
        android:layout_width="120dp"
        android:layout_height="80dp"

android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="0000"
        android:id="@+id/TXT_Respuestaslocales"
        android:layout_gravity="center"
        android:background="#ffbbf"
        android:gravity="center"
        android:textSize="40dp"
        android:textStyle="bold|italic"
    />

    <TextView
        android:layout_width="80dp"

```



```

        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="MOMOMO"
        android:id="@+id/textView29"
        android:visibility="invisible"/>

<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/textView10"
        android:singleLine="false"
        android:lines="2"
        android:minLines="2"
        android:text="@string/TSLRS"/>

<TextView
        android:layout_width="20dp"
        android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="M"
        android:id="@+id/textView23"
        android:visibility="invisible"/>

<TextView
        android:layout_width="120dp"
        android:layout_height="80dp"

android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="0000"
        android:id="@+id/TXT_Tiempolocal"
        android:layout_gravity="center"
        android:background="#affffc"
        android:gravity="center"
        android:textSize="40dp"
        android:textStyle="bold|italic"/>

</LinearLayout>
</HorizontalScrollView>

<TextView
        android:layout_width="wrap_content"
        android:layout_height="50dp"

android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView12"
        android:visibility="invisible"/>

<HorizontalScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/horizontalScrollView4"
        android:layout_gravity="center">

```

```

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:id="@+id/linearLayout">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/textView11"
        android:singleLine="false"
        android:lines="2"
        android:minLines="2"
        android:text="@string/TRS"/>

    <TextView
        android:layout_width="20dp"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="M"
        android:id="@+id/textView24"
        android:visibility="invisible"/>

    <TextView
        android:layout_width="180dp"
        android:layout_height="120dp"

        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="0000"
        android:id="@+id/TXT_Respuestasglobales"
        android:layout_gravity="center"
        android:background="#fffbbf"
        android:gravity="center"
        android:textSize="50dp"
        android:textStyle="bold|italic"/>

    <TextView
        android:layout_width="100dp"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceMedium"
        android:text="MOMOMO"
        android:id="@+id/textView28"
        android:visibility="invisible"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceMedium"
        android:id="@+id/textView13"
        android:singleLine="false"
        android:lines="2"

```

```

        android:minLines="2"
        android:text="@string/TTSS"/>

<TextView
    android:layout_width="20dp"
    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="M"
    android:id="@+id/textView25"
    android:visibility="invisible"/>

<TextView
    android:layout_width="180dp"
    android:layout_height="120dp"

android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="0000"
    android:id="@+id/TXT_Tiempoglobal"
    android:layout_gravity="center"
    android:background="#affffc"
    android:gravity="center"
    android:textSize="45dp"
    android:textStyle="bold|italic"/>

<TextView
    android:layout_width="100dp"
    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="MOMOMO"
    android:id="@+id/textView27"
    android:visibility="invisible"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:id="@+id/textView16"
    android:singleLine="false"
    android:lines="2"
    android:minLines="2"
    android:text="@string/TRRS"/>

<TextView
    android:layout_width="20dp"
    android:layout_height="wrap_content"

android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="M"
    android:id="@+id/textView26"
    android:visibility="invisible"/>

<TextView
    android:layout_width="180dp"
    android:layout_height="120dp"

```

```

        android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="0000"
            android:id="@+id/TXT_Reforzadores"
            android:layout_gravity="center"
            android:background="#aefb3"
            android:gravity="center"
            android:textSize="45dp"
            android:textStyle="bold|italic"/>

    </LinearLayout>
</HorizontalScrollView>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="50dp"

    android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView14"
        android:visibility="invisible"/>

<Button
    android:layout_width="220dp"
    android:layout_height="150dp"
    android:text="REINFORCE"
    android:id="@+id/Reforzar"
    android:textStyle="bold|italic"
    android:textSize="40dp"
    android:background="#26ff35"
    android:gravity="center"
    android:layout_gravity="center"
    android:visibility="visible"/>

<Button
    android:layout_width="220dp"
    android:layout_height="150dp"
    android:text="END SESSION"
    android:id="@+id/Finalizar"
    android:textStyle="bold|italic"
    android:textSize="40dp"
    android:background="#ff1d0c"
    android:gravity="center"
    android:layout_gravity="center"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="50dp"

    android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/textView15"
        android:visibility="invisible"/>
</LinearLayout>
</ScrollView>

</LinearLayout>

```

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >

    <ScrollView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/scrollView">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="30dp"

                android:textAppearance="?android:attr/textAppearanceMedium"
                android:text="Medium Text"
                android:id="@+id/textView12"
                android:visibility="invisible"/>

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

                android:textAppearance="?android:attr/textAppearanceLarge"
                android:text="SESSION DATA"
                android:id="@+id/textView3"/>

            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">

                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Subject:    "
                    android:id="@+id/textView"/>

                <EditText
                    android:layout_width="200dp"
                    android:layout_height="wrap_content"
                    android:id="@+id/EDTXT_Subject"
                    android:singleLine="true"/>
            </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
```

```

        android:layout_height="fill_parent"
        android:id="@+id/linearLayout">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Session:    "
        android:id="@+id/textView2"/>

    <EditText
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:id="@+id/EDTXT_Session"
        android:singleLine="true"/>
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="30dp"

    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/textView11"
    android:visibility="invisible"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="END SESSION"
    android:id="@+id/textView4"/>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout2">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Time:    "
        android:id="@+id/textView5"/>

    <EditText
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:id="@+id/EDTXT_MTime"
        android:inputType="number"/>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout3">

    <TextView

```

```

        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Reinforcers:   "
        android:id="@+id/textView6"/>

    <EditText
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:id="@+id/EDTXT_MReinforcers"
        android:inputType="number"/>
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="30dp"

    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/textView10"
    android:visibility="invisible"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="REINFORCEMENT DELIVERY"
    android:id="@+id/textView7"/>

<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/IF"
        android:id="@+id/Rad_FI"/>

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RF"
        android:id="@+id/Rad_FR"/>

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/IV"
        android:id="@+id/Rad_VI"/>

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RV"
        android:id="@+id/Rad_VR"/>
</RadioGroup>

```

```

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/linearLayout5">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Value:    "
        android:id="@+id/textView9"/>

    <EditText
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:id="@+id/EDTXT_Schedulevalue"
        android:inputType="number"/>
</LinearLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="30dp"

    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/textView8"
    android:visibility="invisible"/>

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Iterations (variable programs)"
    android:id="@+id/itbox"/>

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/itttext"/>

<CheckBox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Delay"
    android:id="@+id/demorabox"/>

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    >
    <EditText android:id="@+id/edit_text_out"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_gravity="bottom"
        android:visibility="invisible"/>
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"

```



```
        android:layout_height="wrap_content"
        android:text="@string/send"
        android:visibility="invisible"/>

</LinearLayout>

<ListView android:id="@+id/in"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:stackFromBottom="true"
    android:transcriptMode="alwaysScroll"
    android:layout_weight="1"
    android:visibility="invisible"/>
</LinearLayout>
</ScrollView>

</LinearLayout>
```

.. **Devicelist.xml**

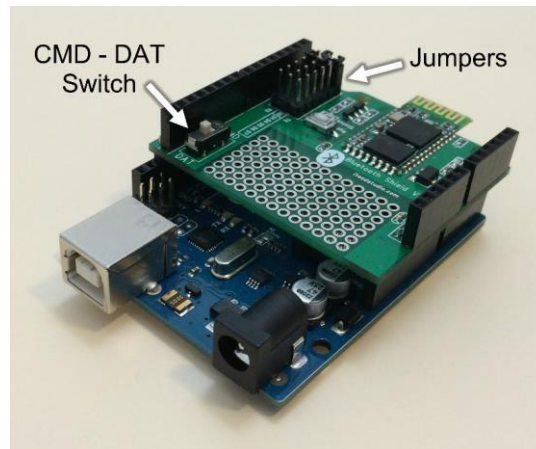
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
    />
    <TextView android:id="@+id/title_new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
    />
</LinearLayout>
```

Apéndice C

• **MANUAL DEL USUARIO**

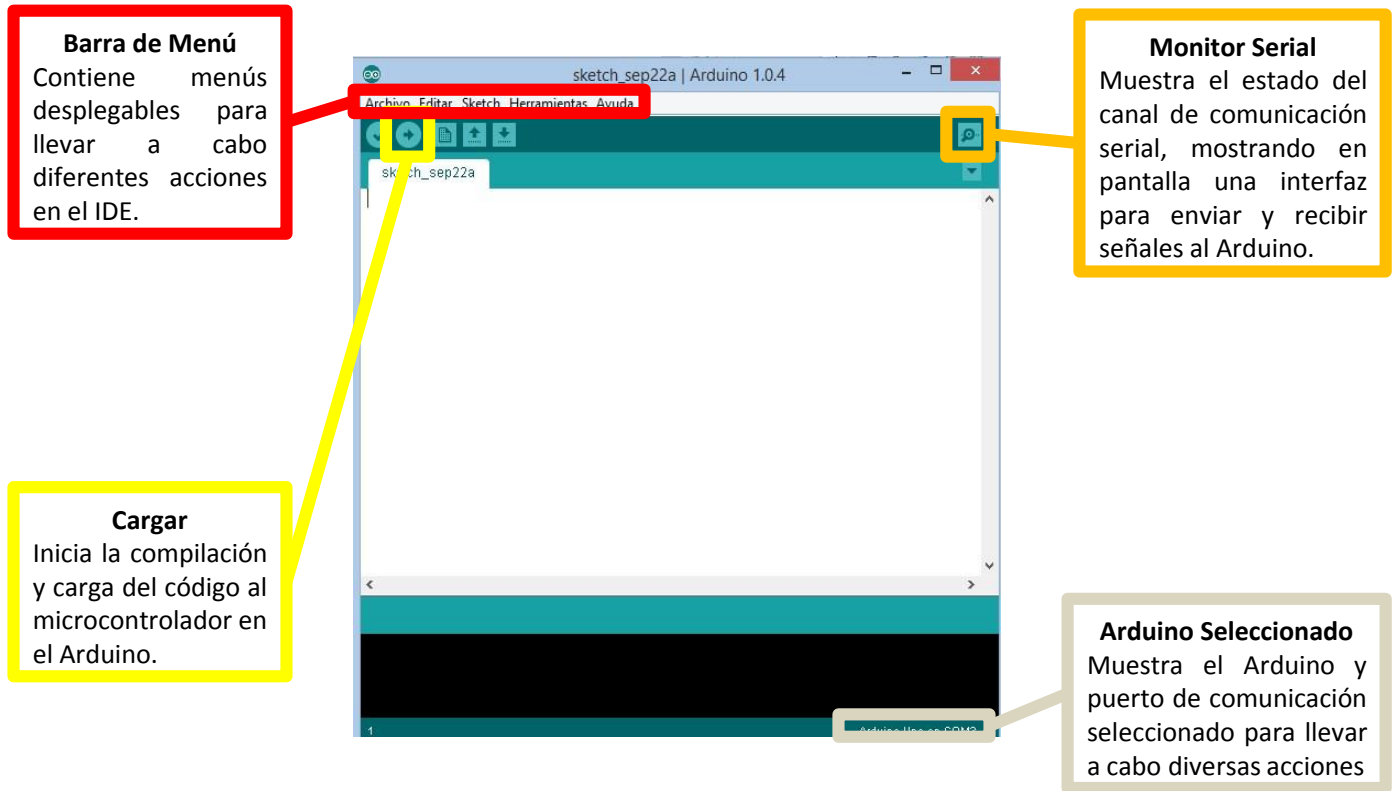
Interfaz de control operante Arduino-Android Ratuino
Mobile

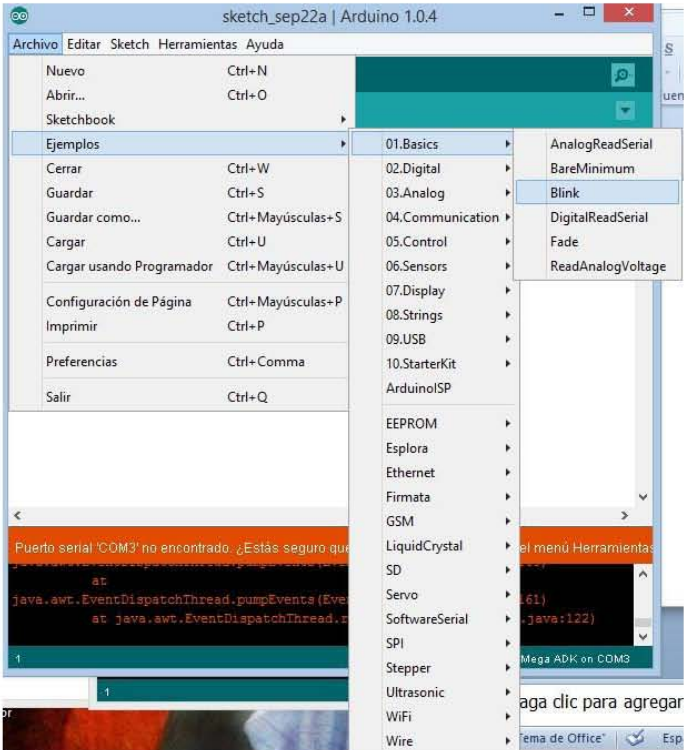
v. 1.4



Antes de iniciar a utilizar la interfaz, se debe conectar el Shield Bluetooth de manera correcta a la interfaz, para esto se debe colocar el jumper D0 en la columna RX y el jumper D1 en la columna TX, posteriormente se debe colocar el switch de la esquina inferior izquierda en CMD para reconfigurar el Shield Bluetooth. La configuración del Shield y la reprogramación del Arduino se realizan mediante el IDE de Arduino que se puede descargar de la siguiente dirección: <http://arduino.cc/en/Main/Software>

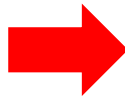
Interfaz gráfica del IDE de Arduino





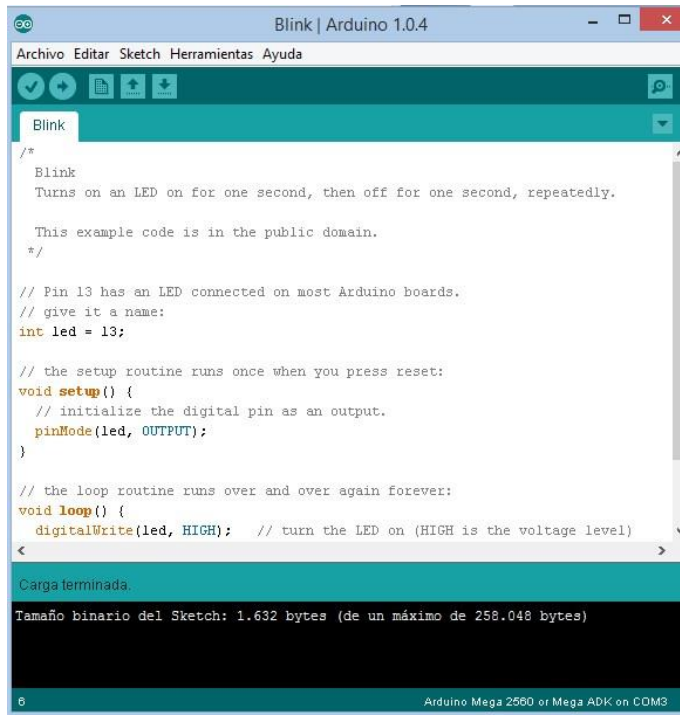
Dentro del programa de Arduino, se despliega el menú de Archivo, y se selecciona la opción de ejemplos. Posteriormente se selecciona 01.Basics y finalmente se carga el programa de Blink.

```
Arduino IDE - Blink | Arduino 1.0.4
Archivo Editar Sketch Herramientas Ayuda
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
}
```



```
Arduino IDE - Blink | Arduino 1.0.4
Archivo Editar Sketch Herramientas Ayuda
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
}
Compilando el Sketch...
Arduino Mega 2560 or Mega ADK on COM3
```

Una vez abierto el programa se conecta el Arduino a la computadora, y presionando el botón de cargar, se reprograma el microcontrolador con este programa. Al presionar le botón de carga una barra en la esquina inferior derecha muestra el avance en la compilación y carga del programa al microcontrolador.



The screenshot shows the Arduino IDE window titled "Blink | Arduino 1.0.4". The menu bar includes "Archivo", "Editar", "Sketch", "Herramientas", and "Ayuda". The toolbar contains icons for file operations and a help icon. The main editor area displays the following code:

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
}
```

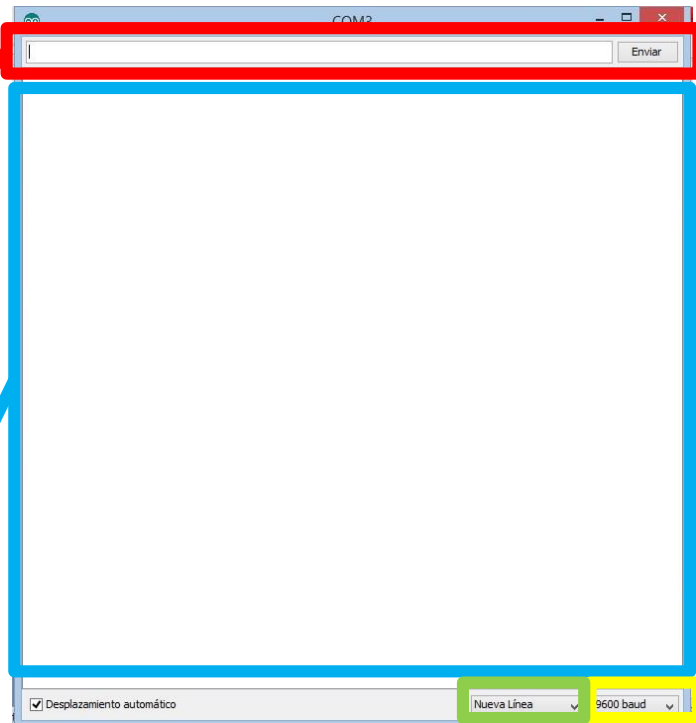
Below the code editor, the serial monitor shows the output "Carga terminada." and "Tamaño binario del Sketch: 1.632 bytes (de un máximo de 258.048 bytes)". The status bar at the bottom indicates "6" and "Arduino Mega 2560 or Mega ADK on COM3".

Una vez que se indica que la carga ha concluido exitosamente, debe abrirse el monitor serial para reconfigurar el Shield bluetooth.

Interfaz de comunicación Serial

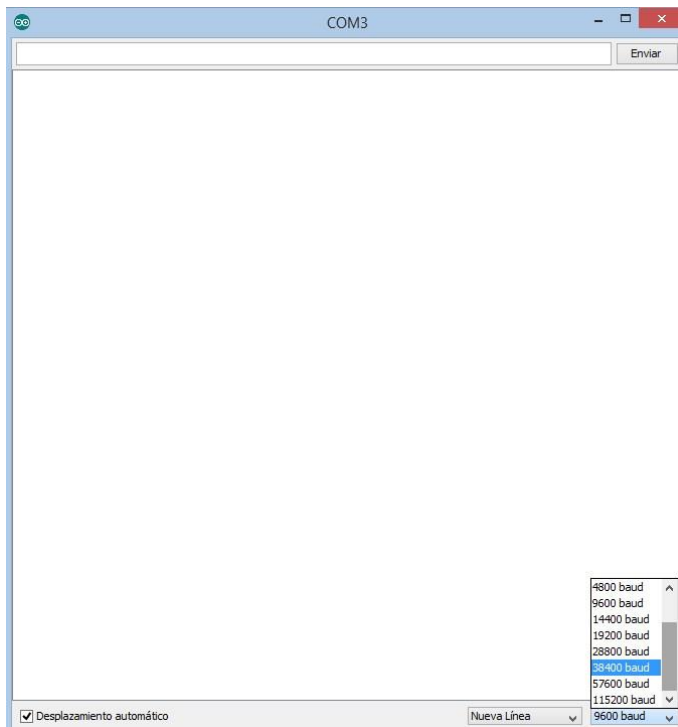
Menú de comunicación saliente
Permite enviar información al Arduino mediante el teclado de la computadora.

Registro de comunicación
Muestra en pantalla toda la comunicación entre el Arduino y la computadora tanto saliente como entrante.



Modo de escritura
Indica el modo de escritura al enviar un mensaje al Arduino.

Baud Rate
Indica la velocidad de comunicación entre el Arduino y la computadora.



Una vez que se inicia el monitor de comunicación serial, debe cambiarse la tasa de transferencia con el menú desplegable en la esquina inferior derecha. Debe elegirse como nuevo valor “38400 baud”

Dentro del menú de comunicación saliente deben escribirse los siguientes comandos para reconfigurar el Shield Bluetooth:

Se escribe AT y se presiona *enter*.

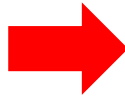
Si aparece el mensaje OK, es posible introducir los comandos para modificar el nombre del dispositivo y la contraseña.

Se escribe AT+NAME? y *enter* para solicitar el nombre actual del dispositivo. Se escribe AT+NAME= =<nuevo nombre> y *enter* para renombrar el dispositivo (e.g. AT+NAME=BETA para cambiar el nombre del dispositivo a BETA).

Se escribe AT+PSWD =<nuevo password> y *enter* para cambiar el password de nuestro dispositivo (e.g. AT+PSWD =0602 cambia el password del dispositivo por 0602).

Al finalizar de reconfigurar el Shield Bluetooth, se cambia el switch en la tarjeta a la posición DATA para poder emplearlo en conjunto con el dispositivo Android.

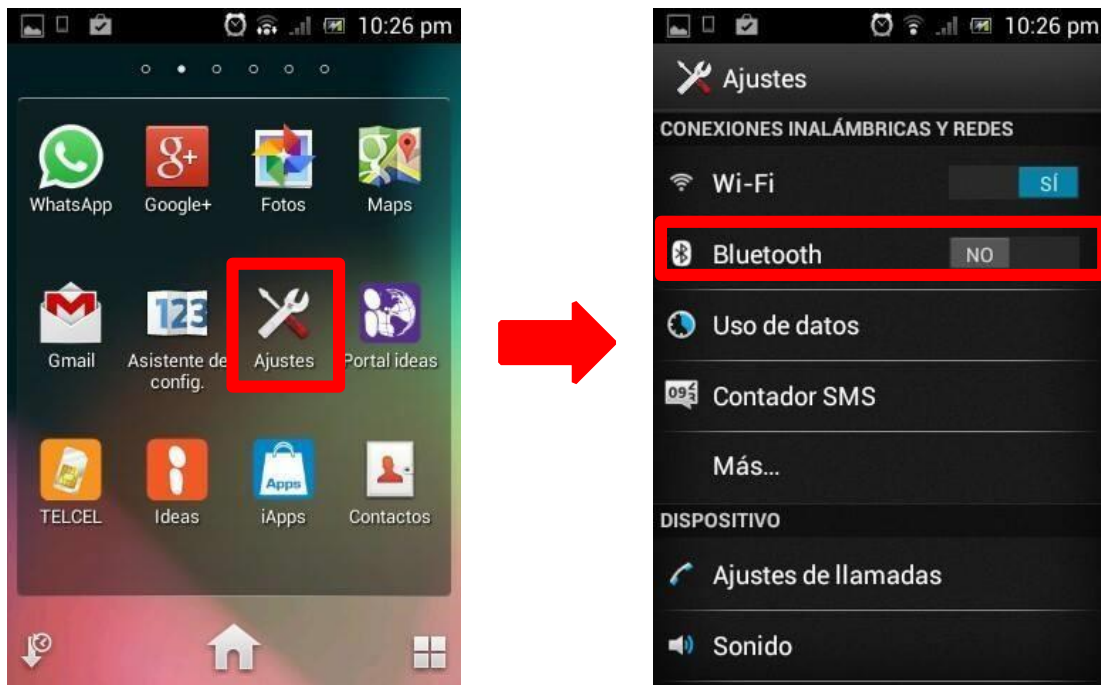
```
Arduino IDE | Blink | Arduino 1.0.4
Archivo Editar Sketch Herramientas Ayuda
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
}
```



```
Arduino IDE | Blink | Arduino 1.0.4
Archivo Editar Sketch Herramientas Ayuda
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
}
```

Compilando el Sketch.

Una vez que se configura el Shield Bluetooth, debe abrirse el archivo con el código para la comunicación con el dispositivo móvil que puede descargarse gratuitamente de la siguiente dirección: http://analisisdelaconducta.net/?page_id=126 y se carga al Arduino usando el botón *upload*.



En el dispositivo Android, se selecciona el ícono de ajustes y se selecciona la opción de comunicación bluetooth para abrir los dispositivos enlazados.



Dentro de la pantalla de dispositivos enlazados, se selecciona la opción de buscar dispositivos y se elige el dispositivo que se acaba de renombrar. En este caso BETA.

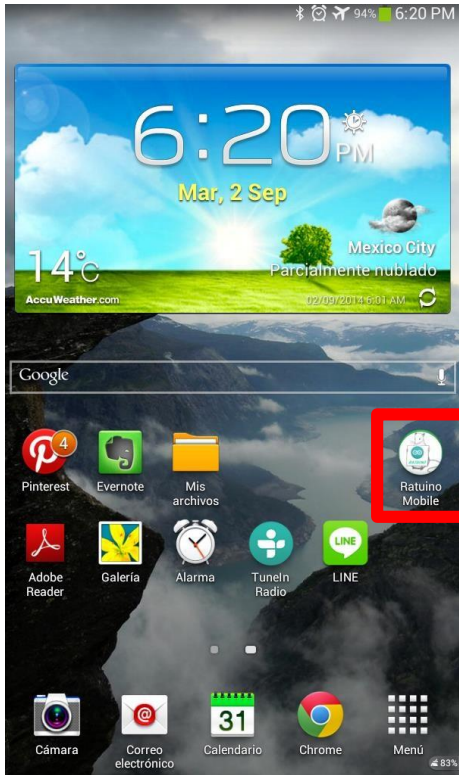
Una vez elegido el dispositivo, se ingresa el password y se observa que ahora los dispositivos se encuentran enlazados.



Para descargar la aplicación Ratuino Mobile, se selecciona el ícono de la *Play Store* y en el buscador ingresamos la palabra Ratuino.



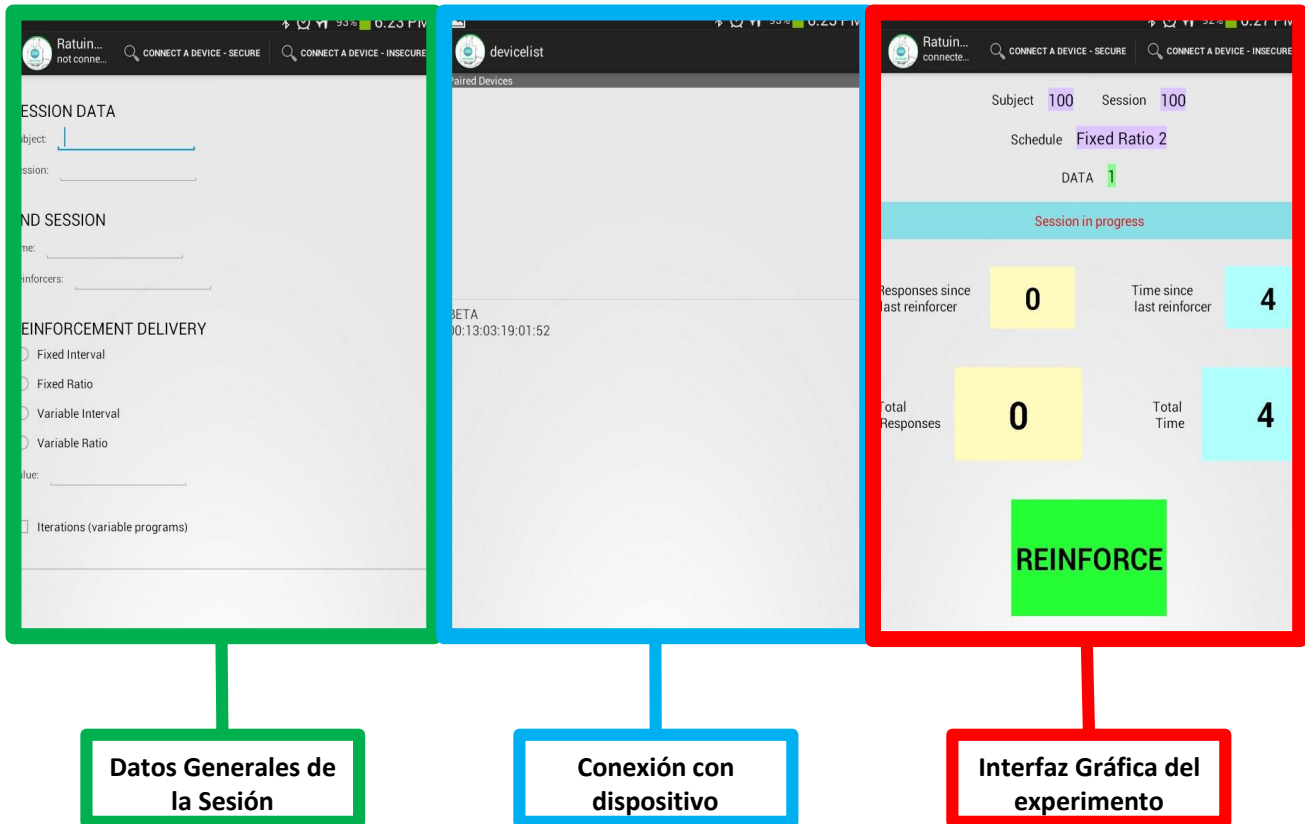
La aplicación señala como autor LCO UNAM y no tiene ningún costo, una vez que se selecciona debe instalarse en el equipo.

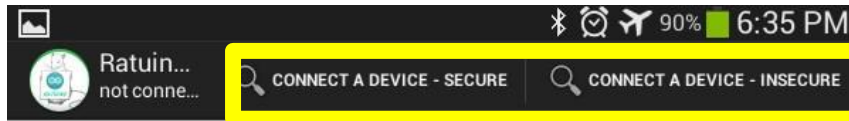


La aplicación puede colocarse en el escritorio del dispositivo Android para un acceso rápido. Antes de abrir la aplicación se deberán activar las funciones de bluetooth del dispositivo android para que funcione correctamente.

La aplicación cuenta con 3 pantallas de navegación que cubren diferentes funciones.

Pantallas de la Aplicación Ratuino Mobile





4.-Menú de
Conexión

1.-Datos Generales
de la Sesión

SESSION DATA

Subject:

Session:

2.-Criterios de
Finalización

END SESSION

Time:

Reinforcers:

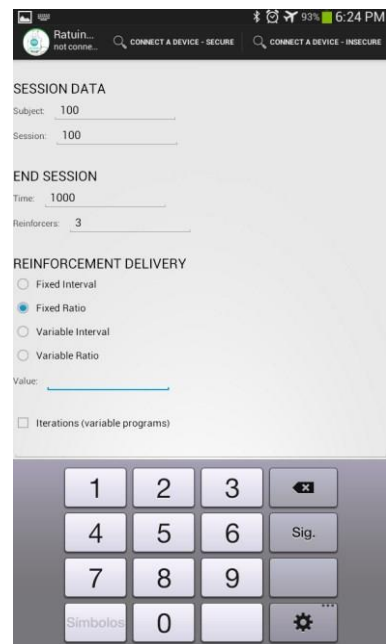
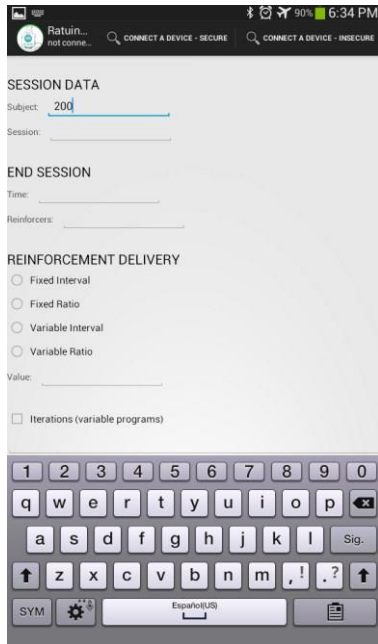
3.-Criterios de
Reforzamiento

REINFORCEMENT DELIVERY

- Fixed Interval
- Fixed Ratio
- Variable Interval
- Variable Ratio

Value:

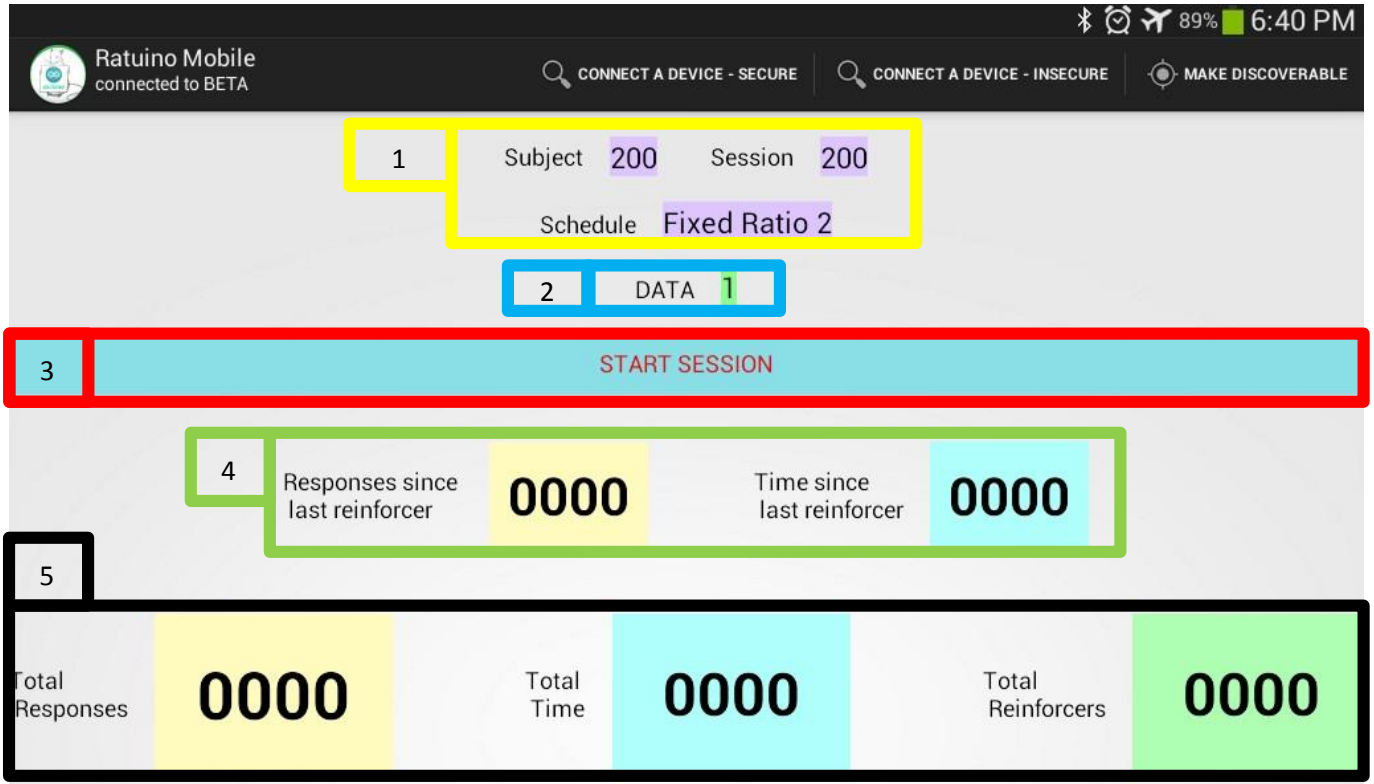
Iterations (variable programs)



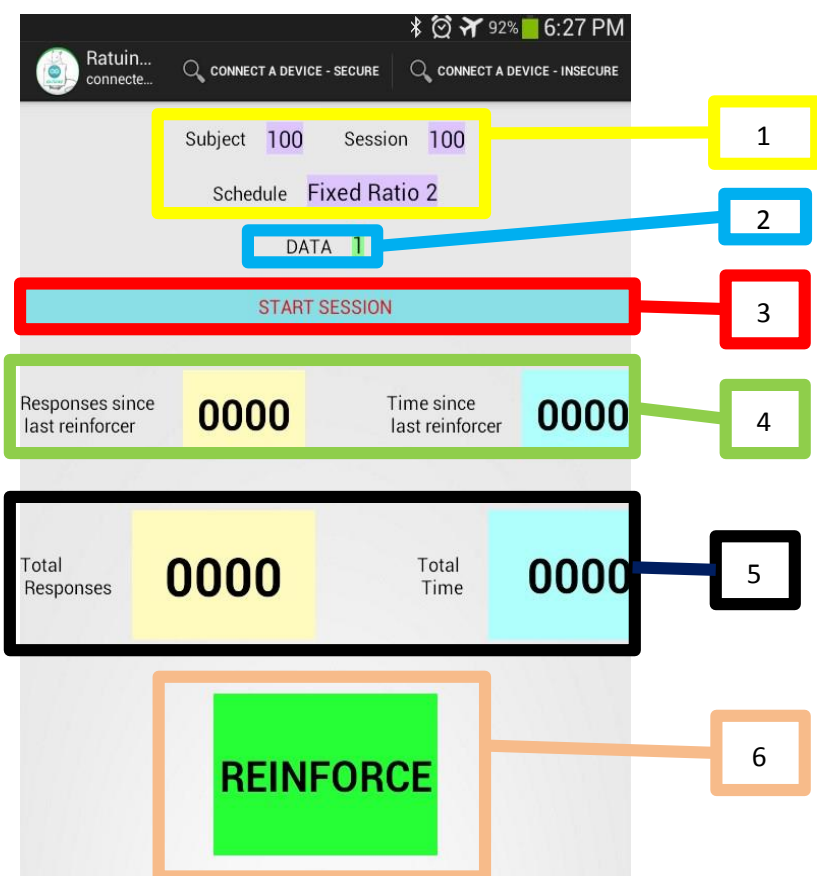
La primer pantalla permite llenar un formulario con los datos generales de la sesión (sujeto y sesión), los criterios de finalización de la sesión (tiempo y reforzadores máximos), además de que permite seleccionar uno de los 4 programas de reforzamiento simples y brinda la oportunidad de especificar el número de iteraciones para programas variables de acuerdo con la sucesión de Fleshler y Hoffman.



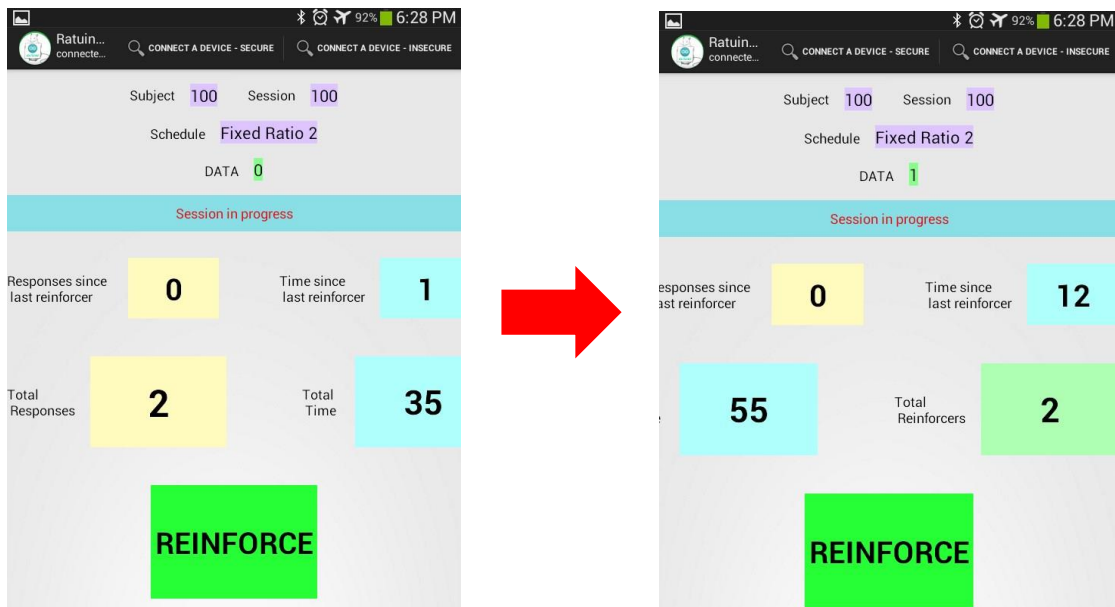
Una vez llenado el formulario, se selecciona la opción del menú de conexión, la cual muestra la selección del dispositivo con el que se puede iniciar la comunicación serial. Es importante seleccionar el dispositivo previamente enlazado.



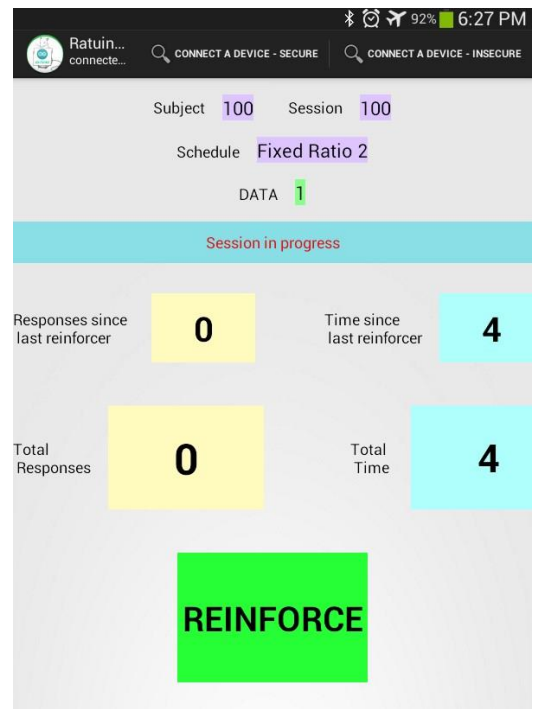
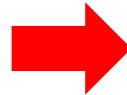
1. Datos generales de la sesión
2. Datos de la comunicación Serial
3. Botón de Inicio de Sesión
4. Datos desde el último reforzador
5. Datos desde el inicio de la sesión



1. Datos generales de la sesión
2. Datos de la comunicación Serial
3. Botón de Inicio de Sesión
4. Datos desde el último reforzador
5. Datos desde el inicio de la sesión
6. Botón de entrega de reforzador independiente a la conducta.



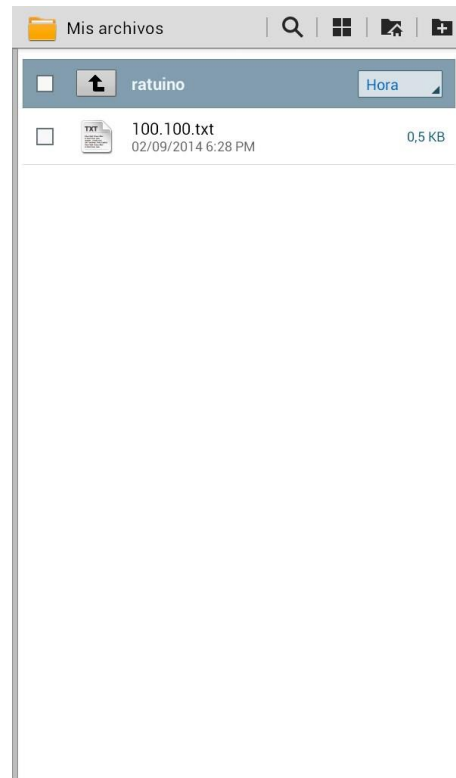
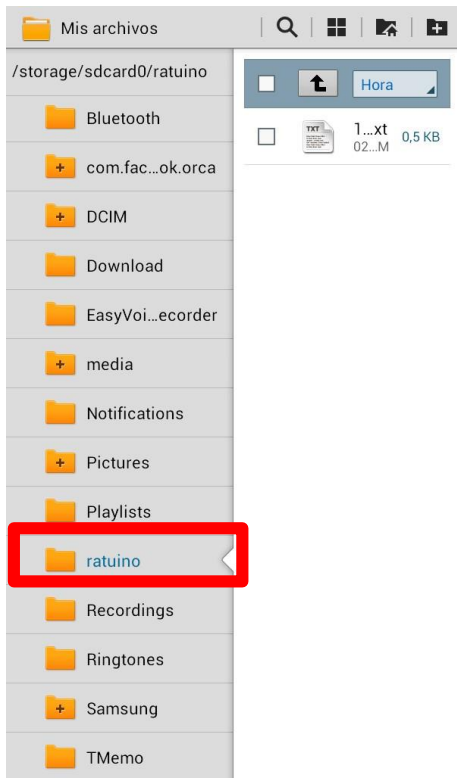
Dependiendo de la resolución del dispositivo la cantidad de información mostrada en pantalla puede ser mayor o menor que la que se muestra en la imagen. Sin embargo, el usuario puede desplazarse deslizando la pantalla para acceder a la información completa.



Una vez que la información mostrada en el espacio de los datos de comunicación serial se establezca, el usuario puede iniciar la sesión presionando el botón de inicio de sesión. Esto cambiará el texto en pantalla para reflejar en tiempo real el desarrollo de la sesión.

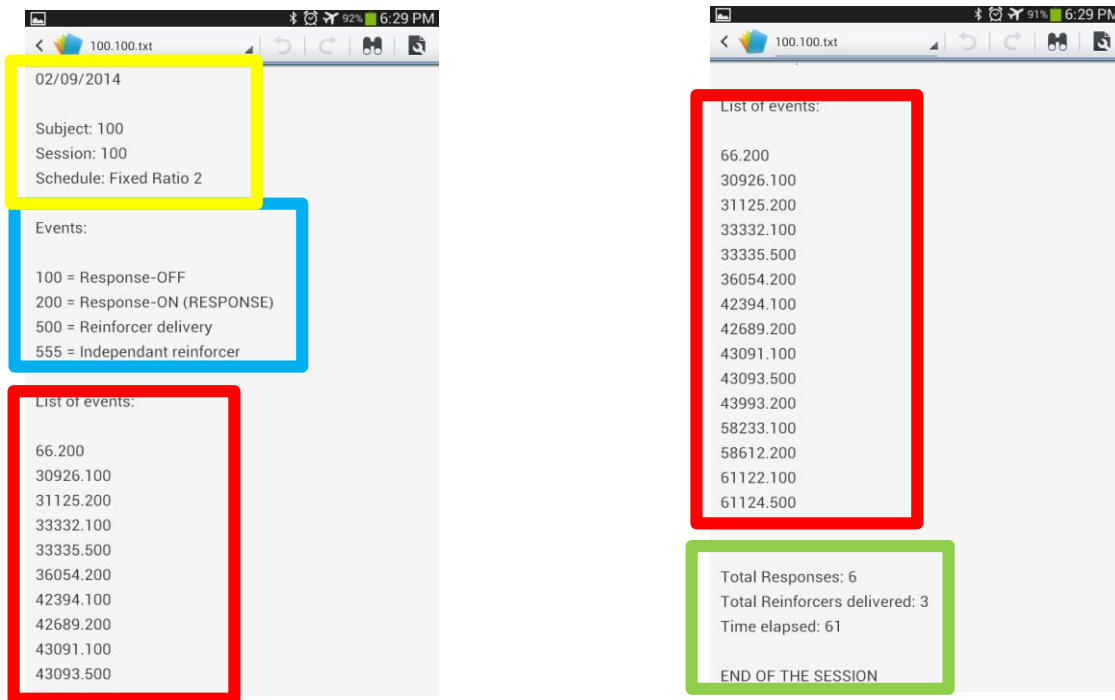


Al terminar la sesión, el botón de reforzamiento independiente a la conducta cambiará a un botón de finalización de la sesión. Este botón debe presionarse para cerrar correctamente la aplicación si se desea iniciar otra sesión.



La aplicación Ratuino Mobile creará una carpeta en la memoria interna del dispositivo en donde almacenará la información de todas las sesiones en archivos con extensión .txt para un posterior análisis.

Los archivos generados por la aplicación se dividen en 4 segmentos principales



Datos Generales de la sesión (señalados con amarillo)

Acotaciones sobre los eventos experimentales (señalados en azul)

Eventos experimentales con precisión de milisegundos (señalados en rojo)

Resumen de la sesión (señalados en verde)

Apéndice D

. Ratuino Mobile App: errores conocidos

La siguiente es una lista de los errores en tiempo de ejecución que pueden llevar a un mal registro o al cierre de la aplicación. Se proporciona también información sobre cómo evitar este problema.

Condición previa	Posible error en el programa	Método de prevención
Iniciar la aplicación sin tener el Bluetooth encendido la aplicación abrirá un cuadro que permite iniciar el Bluetooth. Sin embargo al regresar a primer plano la aplicación deja de funcionar.	Cierre completo del programa, al mismo tiempo activa el Bluetooth en el dispositivo.	Encender el Bluetooth antes de ejecutar la aplicación.
Pasar al menú de elección de dispositivo cuando falta introducir alguno de los datos necesarios para correr la aplicación.	La aplicación permite seleccionar el dispositivo para iniciar comunicación serial Sin embargo al pasar a la pantalla de la sesión la aplicación se reiniciará.	Rellenar todos los campos necesarios en el formulario antes de empezar una sesión.
Seleccionar un dispositivo que no se encuentre previamente enlazado para iniciar la comunicación serial.	Cierre parcial del programa. Dependiendo de si el dispositivo puede enlazarse o no el programa puede reiniciarse.	Establecer comunicación serial únicamente con dispositivos Bluetooth previamente enlazados.

Condición previa	Posible error en el programa	Método de prevención
Seleccionar un dispositivo Bluetooth que no contenga el código apropiado para emplear la aplicación	La sesión empieza a trabajar de manera normal, sin embargo, no podrá registrar o emitir los comandos necesarios para el cierre de sesión o del archivo de escritura.	Establecer comunicación serial únicamente con dispositivos que contengan el código y configuración apropiada para el uso de la interfaz.
Cambiar la orientación del dispositivo.	Dependiendo del dispositivo la aplicación puede cerrarse, detenerse momentáneamente o puede no pasar nada en absoluto.	Bloquear el cambio de orientación de dispositivo antes de iniciar la aplicación.
No introducir los criterios necesarios para determinar el número de intervalos (iteraciones) según la sucesión de Fleshler y Hoffman.	El programa utilizará como número de iteraciones el número máximo de reforzadores dividido entre tres.	Especificar los criterios necesarios para determinar los intervalos según la sucesión de Fleshler y Hoffman.
Intentar terminar la sesión si el número de reforzadores máximos no ha sido alcanzado.	La aplicación continuará corriendo aún si el tiempo de la sesión ha sobrepasado los valores máximos especificados.	No existe método de prevención. Es necesario cerrar la aplicación desde Android.
Cambiar la aplicación de primer a segundo plano, de segundo plano a primer plano o cambiar de aplicación.	Dependiendo del dispositivo la aplicación puede reiniciarse o cerrarse.	No cambiar de aplicación durante la ejecución del programa.
Recibir una llamada durante el transcurso de una sesión.	Se registrará una respuesta al momento de aceptar la llamada entrante.	Poner el dispositivo en modo “avión” para no recibir llamadas durante la sesión.

Condición previa	Posible error en el programa	Método de prevención
En la línea de datos provenientes de la comunicación serial empiezan a aparecer números compuestos por más de un dígito (e.g. 11, 111, 11111, 10, 01).	El dispositivo empieza a registrar respuestas que no están ocurriendo en la cámara operante.	Modificar el código de arduino, aumentando la demora al final del código en periodos de 10ms hasta estabilizar la comunicación serial.
Terminar una llamada durante el transcurso de una sesión.	Dependiendo del dispositivo la aplicación puede reiniciarse, o puede no suceder nada.	Poner el dispositivo en modo avión para no recibir llamadas durante la sesión.

